UNIVERZITET CRNE GORE
ELEKTROTEHNIČKI FAKULTET

ANDREJ CVIJETIĆ

PROCJENA BRZINE VOZILA ZASNOVANA NA DUBOKOM UČENJU PRIMJENOM YOLO DETEKTORA I KONVOLUCIONIH NEURALNIH MREŽA

– MASTER RAD –

PODGORICA, 2024. GODINE

PODACI I INFORMACIJE O MAGISTRANDU

Ime i prezime: **Andrej Cvijetić**
Datum i mjesto rođenja: **8. mart 2000. godine, Cetinje**
Prethodno završene studije:

Elektrotehnički fakultet, osnovne akademske studije, studijski program: Elektronika, telekomunikacije i računari, 2021. godine

INFORMACIJE O MASTER RADU

Naziv master studija: **Master studije, studijski program Računari**

Naslov rada: **Procjena brzine vozila zasnovana na dubokom učenju primjenom YOLO detektora i konvolucionih neuralnih mreža**

Fakultet na kojem je rad odbranjen: **Elektrotehnički fakultet Podgorica**

UDK, OCJENA I ODBRANA MASTER RADA

Datum prijave master rada: **18.09.2023.**
Datum sjednice Vijeća na kojoj je prihvaćena tema: **16.11.2023.**

Mentor: **Prof. dr Slobodan Đukanović**

Komisija za ocjenu rada:

1. **Prof. dr Vesna Popović - Bugarin**, ETF Podgorica - predsjednica
2. **Prof. dr Slobodan Đukanović**, ETF Podgorica - mentor
3. **Prof. dr Nikola Žarić**, ETF Podgorica - član

Komisija za odbranu rada:

1. **Prof. dr Vesna Popović - Bugarin**, ETF Podgorica - predsjednica
2. **Prof. dr Slobodan Đukanović**, ETF Podgorica - mentor
3. **Prof. dr Nikola Žarić**, ETF Podgorica - član

Datum odbrane: **11.11.2024.**

Ime i prezime autora: Andrej Cvijetić

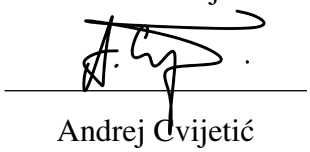
Etička izjava

U skladu sa članom 22 Zakona o akademskom integritetu i članom 18 Pravila studiranja na postdiplomskim studijama, pod krivičnom i materijalnom odgovornošću, izjavljujem da je magistarski rad pod naslovom:

„Procjena brzine vozila zasnovana na dubokom učenju primjenom YOLO detektora i konvolucionih neuralnih mreža"

moje originalno djelo.

Podnosilac izjave:



Andrej Cvijetić

Podgorica, 20. septembar 2024.

Sadržaj

Sažetak

Ovaj rad prezentuje sistem za procjenu brzine vozila baziran isključivo na algoritmima dubokog učenja na osnovu vizuelnih podataka dobijenih iz jedne video kamere. Predloženi sistem precizno i pouzdano estimira brzinu vozila koristeći YOLO algoritam za detekciju i praćenje vozila, i jednodimenzionalnu konvolucionu neuralnu mrežu (1D-CNN) za procjenu brzine. YOLO algoritam na svom izlazu daje granične okvire oko detektovanih objekata na slici, odnosno u našem slučaju vozila čiju brzinu želimo estimirati. Kao ulaz u naš 1D-CNN model za procjenu brzine, uvodimo novu karakteristiku koja odražava promjenu površine graničnog okvira oko vozila. Ova karakteristika, dobija se računanjem površine graničnih okvira, kroz frejmove, kako vozilo prilazi kameri. Oblik krive promjene ostaje gotovo isti za sva vozila, s razlikama uslovljenim vrijednošću posmatrane brzine vozila. Predložena metoda je obučena i testirana na VS13 skupu podataka. Eksperimentalni rezultati pokazuju da metoda veoma precizno i pouzdano estimira brzinu vozila s prosječnom greškom od 2,70 km/h, pri čemu vozilo sa najboljim performansama pokazuje prosječnu grešku od samo 1,38 km/h. Predložena metoda ističe svoju robusnost kao ključnu prednost, eliminišući potrebu za prethodnim poznavanjem dimenzija u stvarnom svijetu, kao što su veličina vozila, širina puta, udaljenost kamere i ugao kamere u odnosu na put itd.

Ključne riječi: procjena brzine vozila, kompjuterska vizija, duboko učenje, jednodimenzione konvolucione neuralne mreže

Abstract

This paper presents a vehicle speed estimation system based solely on deep learning algorithms using visual data obtained from a single video camera. The proposed system accurately and reliably estimates vehicle speed by utilizing the YOLO algorithm for vehicle detection and tracking, and a one-dimensional convolutional neural network (1D-CNN) for speed estimation. The YOLO algorithm outputs bounding boxes around detected objects in the image, in our case vehicles, whose speed we aim to estimate. As an input to our 1D-CNN speed estimation model, we introduce a novel feature that reflects the change in the area of the bounding box around the vehicle. This feature is derived by calculating the bounding box area across frames as the vehicle approaches the camera. The shape of the curve representing the change remains almost the same for all vehicles, with differences conditioned by the observed vehicle speed. The proposed method was trained and tested on the VS13 dataset. Experimental results show that the method estimates vehicle speed with high accuracy and reliability, with an average error of 2.70 km/h, with the best-performing vehicle showing an average error of only 1.38 km/h. The proposed method highlights its robustness as a key advantage, eliminating the need for prior knowledge of real-world dimensions such as vehicle size, road width, camera distance, camera angle relative to the road, etc.

Keywords: vehicle speed estimation, computer vision, deep learning, one-dimensional convolutional neural networks

Glava 1

Uvod i dosadašnja istraživanja u oblasti

Sveprisutna urbanizacija i dinamični rast broja vozila na putevima za posljedicu imaju sve veće opterećenje putne infrastrukture, čineći efikasno upravljanje saobraćajem sve težim zadatkom. To dovodi do problema kao što su gužve, saobraćajne nesreće i zagađenje vaduha, koji značajno utiču na naš svakodnevni život. Digitalizacija, odnosno integracija najnovijih tehnologija u saobraćajni sistem postala je imperativ za poboljšanje bezbjednosti, efikasnosti i održivosti saobraćaja.

Inteligentni transportni sistemi (eng. *Intelligent Transport Systems* - ITS) stoje na čelu ove evolucije, nudeći inovativna rješenja za optimizaciju funkcionisanja saobraćaja i oblikujući budućnost njenog razvoja. Pod pojmom ITS podrazumijevamo sva, uslovno rečeno, pametna rješenja (aplikacije, uređaje, sisteme itd.) bazirana na telekomunikacionim, informacionim i kompjuterskim tehnologijama, dizajnirana i razvijena u cilju poboljšanja upravljanja, održavanja, monitoringa, kontrole i sigurnosti svih vrsta saobraćaja. ITS obuhvata širok spektar primjena, od upravljanja saobraćajem i komunikacije vozila sa infrastrukturom do naprednih sistema pomoći vozačima i autonomnih vozila, oslanjajući se na najnovije tehnologije senzora, bežičnih komunikacija, kontrolnih sistema, obrade podataka, vještačke inteligencije itd. [1, 2]. Po svemu sudeći, ovi sistemi postaju ključni faktor u transformaciji načina na koji shvatamo, organizujemo i optimizujemo saobraćaj.

Fokus ovog istraživanja i rada je na preciznom mjerenju brzine vozila, važnom faktoru koji utiče na fluidnost saobraćaja, sigurnost i ukupnu funkcionalnost puteva. Precizno mjerenje brzine u realnom vremenu ključno je za sprovođenje propisanih ograničenja, identifikaciju saobraćajnih gužvi i smanjenje saobraćajnih nesreća povezanih sa prekoračenjima brzine. U blizini kamera za mjerenje brzine, primjetno je smanjenje broja saobraćajnih nezgoda i vozila koja prekoračuju brzinu, sa potencijalnim smanjenjem od čak

25% i 35%, respektivno. Takođe, što je veća prisutnost kamera i strožija primjena propisa, to se očekuje veće smanjenje učestalosti saobraćajnih nesreća [3].

Prekoračenje brzine sa sobom nosi određene posljedice po prestupnike koje mogu uključivati kazne za prekoračenje brzine, oduzimanje vozačke dozvole, ali i teže, poput zatvorskih kazni. Zbog toga uređaji, odnosno sistemi za mjerenje brzine moraju biti veoma precizni. Najčešće se maksimalna dozvoljena greška prilikom mjerenja brzine definiše kao:

$$\text{Greška} \leq \begin{cases} \pm X & \text{ako je } L \leq 100 \text{ km/h} \\ < X\% & \text{ako je } L > 100 \text{ km/h} \end{cases}$$

gdje je L tačna brzina vozila, a X može uzeti vrijednost između jedan i tri, u skladu sa propisima države [4]. Zbog toga, trenutno, najčešće korišteni i najpouzdaniji metodi za mjerenje brzine su ili visokoprecizni i skupi senzori za mjerenje brzine na daljinu, kao što su oni koji princip rada temelje na principu Doplerovog efekta (radari) [5] ili laserske svjetlosti (LiDAR)[6, 7], ili, u nekim slučajevima, posebni senzori postavljeni u parovima ispod površine puta, kao što su detektorske petlje (eng. *Loop detectors*) koje koriste induktivnost ili piezoelektricitet [8–10]. Visoka cijena ovih uređaja i neprekidan rast obima i složenosti saobraćaja govore nam da je potrebno pribjeći nekom ekonomičnijem, ali jednako preciznom rješenju.

Posljednjih godina, svjedoci smo značajnih pomaka u oblasti kompjuterske vizije koji su omogućili potencijalni prelazak sa tradicionalnih sistema na one koji predlažu korišćenje vizuelnih i audio podataka kao osnovni metod za određivanje brzine vozila. Jedan od glavnih izazova u korišćenju pristupa baziranog na viziji je da video senzori, koji transformišu 3D svijet u 2D format, mogu biti neprecizni, a tačnost opada kako se udaljenost između kamere i vozila povećava. Bez obzira na ova ograničenja, prednosti korišćenja video kamera su značajne. One uključuju ekonomičnost, mogućnost korišćenja već postojećih saobraćajnih kamera, upotrebu dobijenog video materijala i u druge svrhe u cilju očuvanja sigurnosti saobraćaja itd. Pomaci na polju kompjuterske vizije uopšte, zaslužni su za veliki broj radova i značajan napredak u ovoj oblasti [4].

Najgrublja podjela dosadašnjih metoda baziranih na viziji može se napraviti na osnovu izbora vizuelne tehnologije na kojoj autori temelje svoja istraživanja, odnosno na stereo sisteme (dvije kamere) i monokularne sisteme (jedna kamera). S obzirom na to da su većina danas instaliranih saobraćajnih kamera monokularne i stacionarne, usmjerila je interesovanje i istraživanje u ovom radu upravo na ove sisteme.

U ovom radu, način klasifikacije metoda procjene brzine zasnovanih na viziji usvojen je iz preglednih radova [4, 11] jer na sveobuhvatan i temeljan način daju analizu postojećih tehnika. Međutim, iako su određeni kriterijumi precizno definisani i korišćene tehnike se mogu jasno razdvojiti, najčešće dolazi do preplitanja u korišćenim tehnikama stoga se različiti pristupi i metode ne mogu precizno klasifikovati.

Najveći broj postojećih, i u ovom radu analiziranih, pristupa zahtijeva preciznu kalibraciju kamere tj. određivanje unutrašnjih i spoljašnjih parametara kamere. Spoljašnji parametri se odnose na rotaciju i translaciju kamere u odnosu na realni koordinatni sistem odnosno na orijentaciju i poziciju kamere u stvarnom svijetu. Unutrašnji parametri predstavljaju odnos između pikselnih koordinata i koordinata kamere i definisani su optičkim, geometrijskim i digitalnim karakteristikama same kamere. Preciznim dobijanjem parametara, moguće je konvertovati koordinatni sistem kamere u globalni koordinatni sistem i dobiti realne dimenzije. Zatim se stvarno pomjeranje vozila, odnosno pređeni put, može izračunati na osnovu udaljenosti u pikselima. Ovi pristupi obično uključuju detekciju i praćenje skupova karakterističnih obilježja vozila ili vozila u cjelosti, a procjena brzine se vrši upoređivanjem trajektorija praćenih obilježja ili vozila kroz frejmove sa poznatim stvarnim dimenzijama.

Najčešće praćene karakteristike su registarske tablice [12–15] ili, u nekim slučajevima, čak i uglovi vozila [14]. Dobijena trajektorija, odnosno vektori pomjeraja, ne nalaze se na površini puta već "lebde" iznad ravni puta. Uzeta pretpostavka, da trajektorija i površina puta leže na paralelnim ravnima čija je međusobna udaljenost približno ista za sva vozila, neće uvijek davati dobre rezultate. Uzimajući u obzir nekoliko realnih faktora kao što su dinamično i nepravolinijsko kretanje vozila na putu, veća vozila poput kamiona ili autobusa, kao i različite pozicije karakterističnih obilježja na vozilu, smanjuje se tačnost ovakvog pristupa. Pored poznavanja stvarnih dimenzija, ove metode zahtijevaju i preciznu kalibraciju kamere.

U radovima [16, 17] brzina se određuje na osnovu detekcije i praćenja cijelog vozila pomoću metoda koje uključuju oduzimanje statičke pozadine. Autori rada [16] predlažu metodu koja koristi dijagonalni heksadecimalni obrazac (eng. *diagonal hexadecimal pattern*) za oduzimanje statičke pozadine i praćenje cijelog vozila kroz frejmove, uz mogućnost praćenja više vozila na video snimku. Jedinstveni identifikator objekta na frejmu sada su ivice slike, odnosno ivice izdvojenog objekta. Brzina se računa na osnovu pozicije objekta u svakom frejmu.

Autori radova [18–20] detektovanje i praćenje vozila vrše pomoću naprednih metoda, odnosno algoritama dubokog učenja kao što su R-CNN[1], YOLO[2] i SSD[3]. Procjena brzine vozila se takođe bazira na trajektorijama odnosno putanjama praćenih vozila i tačnost zavisi od precizne kalibracije kamere. Metoda predstavljena u radu [21] zahtijeva poznavanje svega nekoliko unutrašnjih i spoljašnjih parametara kamere. Koristeći se definicijom čvrstog ugla (eng. *solid angle*) određuje se pomjeraj vozila a zatim i brzina vozila.

Sa druge strane neki autori svoje pristupe baziraju na referentnoj kalibraciji (eng. *refrence calibration*). Koristeći znanje o stvarnim dimenzijama ili poziciji određenih objekata na snimku kao što je dužina vozila, širina puta ili jedne trake, tablice vozila ili oznaka na putu, na indirektan način se vrši transformacija iz pikselnih u stvarne koordinate, a samim tim računa i pomjeranje vozila. Primjer referentne kalibracije dat je u radu [22], gdje se kalibracija vrši koristeći informaciju o dužini isprekidanih bijelih linija i širini trake, kao i virtuelnih linija koje se postavljaju normalno na pravac puta. Brzina vozila se estimira na osnovu vremena prolaska vozila kroz dvije virtuelne linije. Metoda predložena u radu [23] zahtijeva poznavanje širine puta kao i dužinu dijela puta koji se nalazi u vidnom polju kamere. Koristeći tehnike oduzimanja pozadine i postavljanja regija od interesa vrši se detekcija i praćenje vozila, a zatim određuje vektor pomjeraja u pikselima po frejmovima. Kako bi se ovo konvertovalo u stvarne dimenzije (npr. km/h) koriste se dvije transformacione funkcije koje pretvaraju piksele u metre odnosno frejmove u sekunde.

Radovi [24, 25] predstavljaju pravo osvježenje jer eliminišu potrebu za kalibracijom kamere. Autori rada [24] svoju metodu baziraju na upotrebi trodimenzionalnih konvolucionih neuralnih mreža, pri čemu dobijaju prosječnu brzinu vozila na snimku. Autori rada [25] dobijaju informaciju o pravcu kretanja vozila bez izdvajanja i praćenja karakterističnih obilježja vozila. Najprije ručno označavajući početnu poziciju vozila na snimku i postavljajući regiju od interesa na putu prate razliku između frejmova unutar regije od interesa na osnovu koje se dobijaju projekcioni histogrami koji sadrže informaciju o kretanju vozila. Zatim se sve moguće brzine testiraju jedna po jedna, i ekstremna vrijednost test funkcije se bira za odgovarajuću brzinu. Predložena metoda daje zadovoljavajuće rezultate međutim, zahtijeva ručno podešavanje početnih parametara.

Vidimo da većina predloženih sistema često zahtijeva posebnu kalibraciju i instalaciju, što ih čini veoma nerobusnim, odnosno neprilagodljivim raznim očekivanim promjenama na saobraćajnicama. One sa sobom nose različite nedostatke koji mogu ometati njihovu

[1] R-CNN: Region-based Convolutional Neural Network
[2] YOLO: You Only Look Once
[3] SSD: Single Shot Multibox Detector

efikasnost i primjenjivost u stvarnim situacijama. Zavisnost od pažljivo prikupljenih kalibracionih podataka predstavlja značajno ograničenje, budući da tačnost sistema postaje uslovljena sveobuhvatnošću i reprezentativnošću tih podataka. Kalibrisani sistemi imaju značajno ograničene u sposobnosti generalizacije, teško se prilagođavajući situacijama van njihovog kalibrisanog okruženja. Sam proces kalibracije iziskuje značajne resurse i vrijeme, uz zahtjev za čestim ažuriranjima i održavanjem kako bi se suzbilo odstupanje sistema tokom vremena. Kompleksnost kalibrisanih sistema, uz njihovu zavisnost od određenih konfiguracija hardvera, može ograničiti prilagodljivost i povećati operativne troškove. Iako kalibracija može poboljšati tačnost u kontrolisanim uslovima, navedeni nedostaci ističu važnost razvoja metoda procjene brzine koje prioritet stavljaju na prilagodljivosti, generalizaciji i otpornosti u različitim stvarnim situacijama.

Pristup predložen u ovom radu ima za cilj da pokaže da je primjenom dubokog učenja i konvolucionih neuralnih mreža, samo na osnovu vizuelnih podataka dobijenih iz jednog senzora (kamere), moguće razviti efikasnu i preciznu metodu procjene brzine vozila nezavisnu od stvarnih dimenzija posmatranih objekata i njihovih promjena. Predloženi pristup sastoji se iz dvije cjeline: algoritma za detekciju i praćenje vozila na video snimku i jednodimenzione konvolucione mreže za estimaciju brzine vozila.

Rad je organizovan na sljedeći način. U poglavlju 2 obrađuje se oblast detekcije objekata. Porede se i analiziraju ključni algoritmi i metode koji su oblikovali ovu oblast, s posebnim naglaskom na detektore zasnovane na dubokom učenju. U okviru ovog poglavlja predstavljen je YOLO algoritam. Kako su obje cjeline predloženog sistema, detektor objekata i model za estimaciju brzine, zasnovane na konvolucionim neuralnim mrežama, neophodno je opisati njihove ključne koncepte i principe funkcionisanja. To je učinjeno u poglavlju 3. Predložena metoda za procjenu brzine vozila prezentovana je u poglavlju 4. Ovdje će biti predstavljen način funkcionisanja svakog dijela metode, njihova međusobna povezanost, kao i uloga koju svaki dio ima u cjelokupnom sistemu. Pored toga, u ovom poglavlju detaljno je analizirana i arhitektura predloženog modela za estimaciju brzine, a opisan je i skup podataka koji je korišćen za treniranje mreže i validaciju njenih rezultata. U okviru poglavlja 5 demonstriran je proces treniranja mreže i način postavke eksperimenta. Takođe, opisana je mjera kvaliteta modela i izloženi su rezultati istraživanja. Osvrt na predloženu metodu i dobijene rezultate, kao i pravci za potencijalna buduća istraživanja, dati su u zaključku.

Glava 2

Detekcija objekata

Detekcija objekata predstavlja jedan od važnijih zadataka u oblasti kompjuterske vizije, fokusiran na identifikaciji, lokalizaciji i klasifikaciji objekata određene klase u digitalnim slikama i video zapisima. Cilj detekcije objekata je razvoj efikasnih modela i tehnika koje pružaju odgovor na pitanje: *Koji objekti su prisutni na slici i gdje se oni tačno nalaze?*

Kao jedan od osnovnih problema kompjuterske vizije, detekcija objekata pruža dragocjene informacije za semantičko razumijevanje slika i video zapisa i danas nalazi brojne primjene u realnom svijetu kao što su analiza ljudskog ponašanja, prepoznavanje lica, autonomna vožnja, robotska vizija, video nadzor itd. [26].

Problem detekcije objekata podrazumijeva identifikaciju tačnih lokacija objekata na slici, odnosno lokalizaciju i utvrđivanje kojoj kategoriji ili klasi svaki od tih objekata pripada, odnosno klasifikaciju.

Različiti zadaci detekcije imaju različite ciljeve i ograničenja, što rezultira varijabilnom složenošću. Dvije najvažnije metrike za evaluaciju performansi detektora objekata su tačnost (uključujući tačnost klasifikacije i lokalizacije) i brzina. Pored uobičajenih izazova u kompjuterskoj viziji, kao što su različiti uglovi gledanja, osvjetljenje i varijacije unutar iste klase, detekcija objekata suočava se i sa specifičnim poteškoćama poput rotacije i promjene skale objekata (naročito kod manjih objekata), precizne lokalizacije, identifikacije gustih ili djelimično zaklonjenih objekata, kao i potrebe za ubrzanjem samog procesa detekcije.

Iako su temelji detekcije objekata uspostavljeni još devedesetih godina prošlog vijeka, oblast doživljava procvat 2010-ih godina kada tradicionalne detektore sa ručno pravljenim karakteristikama mijenjaju metode zasnovane na neuralnim mrežama, odnosno dubokom

učenju.

Danas se detektori objekata skoro u potpunosti oslanjaju na napredne algoritme dubokog učenja, posebno na konvolucione neuralne mreže, koje su trenirane na velikim skupovima podataka kako bi prepoznale specifične karakteristike objekata. Ove mreže su izuzetno precizne u identifikaciji različitih objekata, bez obzira na varijacije u osvjetljenju, položaju ili perspektivi.

2.1 Skupovi podataka i metrika

U cilju evaluacije performansi različitih detektora objekata koristi se nekoliko skupova podataka koji su vremenom postali standard u ovoj oblasti. Ovi skupovi podataka kreirani su za potrebe raznih eminentnih takmičenja pokrenutih u cilju uključivanja što većeg broja istraživača, što je doprinijelo popularizaciji oblasti ali i značajnim naprecima u istoj.

Neki od najpoznatijih su Pascal VOC [27, 28], MS-COCO [29], ILSVRC [30] i Open images [31].

U ranim istraživanjima detekcije objekata nije postojala univerzalno prihvaćena metrika za evaluaciju tačnosti. Međutim, posljednjih godina postignut je konsenzus, te se najčešće koristi prosječna preciznost (eng. *Average precision* - AP) kao standardna metrika za ocjenjivanje performansi detektora objekata. Prosječna preciznost se računa za svaku klasu objekata posebno i zasnovana je na metrikama preciznosti i odziva. Preciznost je odnos između tačnih pozitivnih detekcija i ukupnog broja pozitivnih detekcija (tačni pozitivni + lažni pozitivni), dok odziv predstavlja odnos između tačnih pozitivnih detekcija i ukupnog broja stvarnih pozitivnih slučajeva (tačni pozitivni + lažni negativni).

Između preciznosti i odziva često postoji kompromis. Na primjer, povećanje broja detektovanih objekata (veći odziv) može dovesti do većeg broja lažnih pozitivnih slučajeva (niža preciznost). Kako bi se ovaj kompromis uzeo u obzir, AP metrika koristi krivu preciznost-odziv, koja prikazuje odnos između preciznosti i odziva za različite pragove povjerenja. Ova metrika omogućava uravnoteženu procjenu preciznosti i odziva kroz analizu površine ispod krive preciznost-odziv [32]. Srednja prosječna preciznost (eng. *Mean average precision* - mAP) koja se računa kao prosjek za sve klase objekata, obično se na kraju uzima kao mjera kvaliteta detektora.

Uspješnost detektora objekata ogleda se i u tome koliko precizno uspijeva da lokalizuje detektovane objekte. Preciznost lokalizacije mjeri se određivanjem presjeka nad unijom (eng. *Intersection over union* - IoU) koja mjeri stepen preklapanja između detektovanog

graničnog okvira oko objekta (eng. *bounding box*) i stvarnog, odnosno unaprijed precizno postavljenog graničnog okvira (eng. *ground truth bounding box*). Računa se kao odnos površine koja se nalazi u presjeku detektovanog i stvarnog graničnog okvira i površine unije ovih graničnih okvira. Ukoliko je ovaj odnos veći od nekog predefinisanog praga poklapanja koji je između 0.5 i 0.95, smatra se da je objekat uspješno detektovan [26].

2.2 Detektori zasnovani na dubokom učenju

Istraživanje u oblasti detekcije objekata može se podijeliti na dva vremenska, odnosno istorijska perioda: period tradicionalne detekcije objekata i period detekcije objekata zasnovane na dubokom učenju.

Detektori zasnovani na dubokom učenju donijeli su revoluciju u ovu oblast, postavši neizostavan standard u savremenim primjenama. Ovi napredni modeli, zahvaljujući svojoj sposobnosti da uče složene karakteristike iz podataka, nude izuzetnu preciznost i efikasnost u detekciji objekata.

Do 2014. godine oblast detekcije objekata bilježi stagnaciju sa veoma malim napretcima i varijantama već postojećih tradicionalnih modela. Ova stagnacija uslovljenja je nizom faktora. Generisanje kandidatskih graničnih okvira najčešće tehnikom pomjerajućeg prozora pokazalo se kao redundantno i veoma neefikasno. Tradicionalni detektori koriste ručno dizajnirane deskriptore niskog nivoa, kao što je histogram orijentisanih gradijenata (eng. *Histogram of Oriented Gradients* - HOG), koji su ograničeni u svojoj sposobnosti da uhvate složene vizuelne obrasce. Često su specifično dizajnirani i namijenjeni za određene tipove objekata i teško ih je prilagoditi novim kategorijama objekata bez modifikacije i dodatnog inženjeringa karakteristika, što ih čini neskalabilnim. Kombinacija ovih deskriptora sa plitkim arhitekturama modela nije uspjela da premosti semantički jaz, što je rezultiralo niskom tačnošću i slabijom sposobnošću prepoznavanja varijacija objekata.

Ponovnom popularizacijom neuralnih mreža, konkretno konvolucionih neuralnih mreža [33], počinje se razmišljati u pravcu primjene ove tehnologije na detekciju objekata. Pojava R-CNN (eng. *Region-based convolutional neural network*) detektora 2014. godine označava pravu prekretnicu, uspostavljajući primat dubokog učenja kao nadalje dominantne tehnologije u detekciji objekata. Ovaj napredak je riješio mnoge nedostatke prethodnih metoda, omogućavajući efikasnije i preciznije generisanje graničnih okvira i prevazilaženje semantičkog jaza, što je dovelo do značajnog poboljšanja performansi na različitim poljima primjene kompjuterske vizije.

Osnovu svakog modernog detektora objekata čini specijalno dizajnirana konvoluciona neuralna mreža. Primarni zadatak ove mreže je ekstrakcija značajnih karakteristika iz ulaznih slika, prije nego što se te karakteristike proslijede daljim koracima detekcije, kao što je faza lokalizacije [34, 35].

Postoji nekoliko arhitektura koje se smatraju standardima. Ove arhitekture su postavile temelje za mnoge savremene modele i tehnike, pružajući efikasne i pouzdane okvire za širok spektar primjena. Među najpopularnijim arhitekturama izdvajaju se:

1. ***AlexNet***

   AlexNet [33] predstavlja model konvolucione neuralne mreže specijalizovan za klasifikaciju slika, koji je pobijedio na *ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012* takmičenju. AlexNet je postigao znatno bolju tačnost u odnosu na tadašnje modele. Sastoji se od ukupno osam slojeva sa težinskim parametrima: pet konvolucionih i tri potpuno povezana sloja. Poslednji sloj povezan je sa *softmax* klasifikatorom koji ima *N* izlaza (*N* predstavlja broj klasa). AlexNet koristi različite konvolucione prozore za ekstrakciju karakteristika iz slika, *dropout* za regularizaciju i ReLU aktivacionu funkciju za bržu konvergenciju. Pojavom ove arhitekture revitalizovana je upotreba konvolucionih neuralnih mreža, koje su ubrzo postale glavna tehnika kojoj se pribjegava u različitim problemima na polju kompjuterske vizije.

2. ***VGG***

   VGGNet [36], razvijena 2014. godine, predstavlja duboku konvolucionu neuralnu mrežu karakterističnu po upotrebi malih 3x3 prozora u svim konvolucionim slojevima mreže, čime se omogućava dublja arhitektura mreže. Ovakav pristup značajno smanjuje broj parametara, što rezultira bržom konvergencijom. VGGNet se sastoji od serije konvolucionih slojeva, obično od 8 do 16, nakon kojih slijede tri potpuno povezana sloja koja su dalje praćena *softmax* klasifikatorom. U ovoj arhitekturi svaki od slojeva praćen je *ReLu* aktivacionom funkcijom. Ova duboka arhitektura, sa rasponom od 11 do 19 slojeva, pokazala je superiorne performanse, nadmašivši pobjednika takmičenja ILSVRC 2014, GoogLeNet, te je ubrzo postala široko prihvaćena osnova za mnoge modele klasifikacije i detekcije objekata [35].

3. ***GoogLeNet/Inception***

   GoogLeNet [37] predložena je u cilju rješavanja problema prevelikih računarskih zahtjeva i uopšte primjenjivosti konvolucionih mreža na stvarne probleme. Arhitektura ove, 22 sloja duboke, mreže bazirana je na korišćenju *Inception* modula koji kombinuju prozore različitih veličina na istom nivou, smanjujući broj parametara i računsku složenost. GoogLeNet je postigao impresivnu preciznost od 93,3% na ImageNet datasetu, nadmašivši mnoge svoje savremenike u brzini. Kasnija poboljšanja *Inception* modula dodatno su unaprijedila performanse ove arhitekture [35].

4. ***ResNets*** (Residual Networks)

   ResNets [38] predstavlja arhitekturu koja je uvela značajnu inovaciju u treniranju vrlo dubokih modela. Kako mreža postaje dublja, odnosno kako broj slojeva mreže raste tako performanse mreže počinju da stagniraju ili opadaju. Ključna ideja koja stoji iza ove arhitekture je uvođenje takozvanog rezidualnog učenja (*eng. Residual learning*) koje korišćenjem kratkih putanja (*eng. Skip connections*) omogućava preskakanje jednog ili više slojeva unutar mreže. Umjesto da svaki sloj direktno uči potpunu transformaciju ulaznih podataka, rezidualni blokovi se fokusiraju na učenje razlike između ulaza i izlaza. Ovaj pristup značajno pomaže u izbjegavanju degradacije performansi, što omogućava efikasnije treniranje dubokih mreža. To se postiže tako što se ulaz prosljeđuje direktno na izlaz, gdje se zatim sabira s naučenom transformacijom.

Detektore objekata zasnovane na dubokom učenju možemo podijeliti u dvije grupe:

- Dvofazni detektori
- Jednofazni detektori

2.3 Dvofazni detektori

Ovi detektori koriste dvostepeni pristup, što i objašnjava njihov naziv. Prvi korak, poznat kao generisanje predloga regija (eng. *Region proposal*), podrazumijeva identifikaciju i izdvajanje potencijalnih oblasti na slici koje bi mogle sadržati objekte od interesa. Ovaj korak je od suštinske važnosti jer omogućava detektoru da se fokusira isključivo na relevantne djelove slike, čime se značajno smanjuje kompleksnost cjelokupnog procesa obrade.

Nakon definisanja predloženih regija, prelazi se na drugi korak, u kojem se te regije detaljno analiziraju. Ovaj korak uključuje klasifikaciju, tj. određivanje tačne vrste objekta unutar svake predložene regije, kao i precizno lociranje objekta kroz prilagođavanje graničnih okvira. Time se postiže ne samo ispravno prepoznavanje objekta, već i tačno određivanje njegovog položaja na slici.

U ovoj sekciji osvrnućemo se na najvažnije algoritme i detektore koji implementiraju ovaj dvostepeni pristup.

R-CNN (eng. *Region-based convolutional neural network*), dvofazni detektor predložen 2014. godine [39], predstavlja algoritam za detekciju objekata kojim je prvi put pokazano da upotreba konvolucionih neuralnih mreža može dovesti do drastično boljih performansi u detekciji objekata. Postiže povećanje srednje prosječne preciznosti za oko 30% u odnosu na prethodno najbolje postignute rezultate na Pascal VOC07 skupu podataka [27], dostigavši mAP od 58,5% [26].

Na početku, ulazna slika prolazi kroz modul za predlaganje regiona, koji generiše 2000 kandidata za objekte. Ovaj modul identifikuje djelove slike sa većom vjerovatnoćom pronalaženja objekata koristeći algoritam selektivne pretrage (eng. *Selective Search*) [40]. Predloženi regioni, odnosno okviri, se zatim dimenziono prilagođavaju i propuštaju kroz konvolucionu mrežu koja za svaki predloženi region izvlači vektor karakteristika dužine 4096.

Autori koriste fiksnu veličinu ulaza od 227×227 piksela za mrežu. Kako su objekti na različitim slikama različitih veličina i proporcija, predloženi regioni takođe variraju u dimenzijama. Bez obzira na veličinu ili proporcije kandidatskih okvira, svi se prilagođavaju operacijama rezanja (eng. *crop*) ili preoblikovanja (eng. *warp*) tako da odgovaraju potrebnoj veličini od 227×227 piskela. Kao osnov arhitekture mreže, autori rada koriste AlexNet [35].

Dobijeni vektori karakteristika se dalje prosljeđuju već treniranim, klasno-specifičnim SVM-ovima (eng. *Support Vector Machine*). Ocijenjeni regioni se zatim koriguju koristeći regresiju graničnih okvira i filtriraju koristeći pohlepni NMS (eng. *Non-maximum suppression*) algoritam kako bi se dobili konačni granični okviri.

Iako je R-CNN postigao značajan napredak u poređenju sa tradicionalnim detektorima, pokazao se kao veoma memorijski i vremenski zahtjevan. Ponavljajuće računske operacije na velikom broju preklapajućih predloženih regiona (preko 2000 predloženih okvira na slici) dovode do izuzetno sporog vremena detekcije od 14 sekundi po slici uz korišćenje GPU-a [26].

R-CNN modeli imaju još jedan nedostatak: prilagođavanje dimenzija regiona prije konvolucione mreže može negativno uticati na preciznost detektora, jer rezanjem regiona može se izostaviti dio objekta, dok preoblikovanje može prouzrokovati nepoželjna geometrijska izobličenja.

Budući da samo potpuno povezani slojevi mreže zahtijevaju ulaz fiksne veličine, SPP-Net [41] prevazilazi ovaj problem dodavanjem sloja prostorne piramidalne agregacije (eng. *Spatial Pyramid Pooling Layer*) nakon posljednjeg konvolucionog sloja. SPP sloj vrši agregaciju informacija i proizvodi izlaze fiksne dužine, koji se zatim prosljeđuju potpuno povezanim slojevima. Na taj način, obavlja se agregacija informacija na dubljem nivou mrežne hijerarhije, između konvolucionih i potpuno povezanih slojeva, čime se izbjegava potreba za rezanjem ili preoblikovanjem na samom početku.

Takođe, karakteristike se računaju samo jednom za cijelu sliku čime se uklanja redudantno računanje karakteristika za svaki predloženi region i značajno ubrzava proces detekcije. SPP-net postiže srednju prosječnu preciznost (mAP) od 59,2% [26] na VOC07 skupu podataka [27].

Iako je SPP-net postigao značajna poboljšanja u pogledu tačnosti i efikasnosti u odnosu na R-CNN, i dalje ima određene nedostatke. SPP-net koristi skoro isti višefazni pristup kao R-CNN, koji uključuje ekstrakciju karakteristika, fino podešavanje mreže, obuku SVM modela i prilagođavanje graničnih okvira. Zbog toga je i dalje potrebna dodatna memorija za skladištenje podataka. Fast R-CNN [42] nadmašuje R-CNN i SPP-Net u efikasnosti, smanjujući kako memorijske zahtjeve, tako i vrijeme izvršavanja. Na Pascal VOC07 datasetu, Fast R-CNN je povećao mAP sa 58.5% (R-CNN) na 70.0%, uz brzinu detekcije koja je preko 200 puta veća u poređenju sa *R-CNN*-om i 10 puta veća u odnosu na *SPP-Net* [26].

Kao i SPP-Net, Fast R-CNN obrađuje cijelu sliku zajedno sa skupom predloga regiona. Proces počinje prolaskom cijele slike kroz nekoliko konvolucionih slojeva i slojeva agregacije kako bi se generisala konvoluciona mapa karakteristika. Za svaki predlog regiona, sloj za agregaciju regije od interesa (eng. *Region of interest pooling layer* - RoI) izvlači vektor karakteristika fiksne dužine iz ove mape.

Ovi vektori karakteristika zatim se šalju u niz potpuno povezanih slojeva, nakon čega se dijele na dva izlazna sloja. Prvi sloj predstavlja *softmax* klasifikator za N klasa objekata, uključujući i klasu za pozadinu, dok drugi sloj daje četiri realne vrijednosti za svaku od N klasa, koje predstavljaju koordinate graničnih okvira.

Fast R-CNN vrši optimizaciju svih parametara na kraju procesa pomoću više-zadatne

funkcije cijene kojom se istovremeno obuhvata klasifikacija i regresija graničnih okvira. Ovakav pristup prevazilazi ograničenje *SPP-Net*-a, koje se ogleda u nemogućnosti ažuriranja konvolucionih slojeva prije SPP sloja, odnosno eliminišu dodatni memorijski zahtjevi koji su bili posljedica višefaznog treniranja. Sa ovim unapređenjima, *Fast R-CNN* postiže veću tačnost, posebno kod dubljih mreža, dok zadržava visoku računsku efikasnost.

Prethodno pomenuti algoritmi koriste algoritam selektivne pretrage za generisanje prijedloga regiona. Međutim, selektivna pretraga je spor i vremenski zahtjevan postupak koji usporava rad mreže i prepoznat je kao usko grlo cijelog procesa.

Faster R-CNN [43, 44] uspijeva prevazići ovu prepreku uvodeći *Region Proposal Network* (RPN) za generisanje prijedloga regiona.

RPN koristi slojeve konvolucione neuralne mreže za ekstrakciju mape karakteristika. Dodavanjem dodatnih konvolucionih slojeva na ovu mapu karakteristika formira se sama RPN. Faster R-CNN je prvi detektor dubokog učenja koji radi gotovo u realnom vremenu, postigavši impresivne rezultate. Na MS-COCO datasetu, Faster R-CNN je ostvario mAP od 42.7%, dok je na Pascal VOC07 datasetu postigao mAP od 73.2% uz brzinu od 17 frejmova po sekundi (fps).

Prethodno navedeni detektori koriste mape karakteristika samo sa posljednjeg nivoa konvolucione neuralne mreže. Međutim, konvoluciona neuralna mreža prirodno formira piramidu mapa karakteristika različitih nivoa tokom propagacije unaprijed. Feature Pyramid Network (FPN) [45] koristi informacije sa više slojeva konvolucione neuralne mreže kako bi poboljšala performanse. Budući da različiti nivoi odgovaraju različitim skalama i veličinama objekata na slici, korišćenje višeslojnih mapa karakteristika omogućava precizniju detekciju objekata različitih veličina. Korišćenjem FPN-a u osnovnoj verziji Faster R-CNN sistema, postižu se rezultati detekcije koji su u rangu sa najboljim rezultatima na MS-COCO skupu podataka, i to bez ikakvih dodatnih optimizacija (COCO mAP@.5=59.1%).

2.4 Jednofazni detektori

Jednofazni detektori razvijali su se paralelno sa dvofaznim, nudeći alternativni pristup detekciji objekata. Iako su oba pristupa usmjerena ka identifikaciji i lokalizaciji objekata unutar slike ili video zapisa, razlikuju se po strukturi, kompleksnosti i performansama u različitim uslovima primjene.

Prethodno predstavljeni dvofazni detektori sastoje se iz više modula koji zahtijevaju pojedinačno treniranje. Obično koriste pristup koji započinje grubom detekcijom objekata odnosno generisanjem regiona od interesa gdje se objekti mogu potencijalno nalaziti. Zatim se prelazi na finiju obradu gdje se objekti precizno lokalizuju i klasifikuju. Ovakav pristup omogućava postizanje visoke tačnosti i pouzdanosti, što ga čini posebno pogodnim za zadatke u kojima je preciznost presudna. Međutim, ova preciznost dolazi uz određene nedostatke, poput povećane računske složenosti i smanjene brzine obrade. Zbog toga, dvofazni detektori često nisu optimalan izbor za zadatke koji zahtijevaju obradu u realnom vremenu ili implementaciju na uređajima sa ograničenim hardverskim resursima.

Sa druge strane, jednofazni detektori pojednostavljuju proces integrišući čitav proces detekcije u jedan jedinstveni korak. Ovi detektori direktno predviđaju klase i koordinate graničnih okvira oko objekata iz cjelokupne slike, bez potrebe za predlaganjem regiona. Ovaj pristup značajno smanjuje računsko opterećenje i omogućava bržu obradu podataka, što ih čini izuzetno pogodnim za primjene u realnom vremenu kao što su autonomna vozila, video nadzor itd. Takođe, njihova efikasnost omogućava i implementaciju na uređajima sa ograničenim resursima, poput pametnih telefona i ugrađenih sistema. Ipak, uprkos njihovoj brzini i efikasnosti, jednofazni detektori često se suočavaju sa izazovima kada je u pitanju detekcija manjih i gusto raspoređenih objekata, gdje njihova preciznost može biti znatno smanjena u poređenju sa dvofaznim detektorima.

Jedan od prvih, a pokazaće se kasnije i jedan od najvažnijih jednofaznih detektora je You Only Look Once (YOLO) [46]. Tokom godina, detektor je prošao kroz nekoliko verzija, od originalnog YOLOv1 do YOLOv8, pri čemu je svaka sljedeća donijela direktno unapređenje u odnosu na prethodnu. Pošto je u ovom istraživanju korišćen upravo ovaj algoritam, fokus će biti na najznačajnijim predstavnicima YOLO porodice algoritama, sa kratkim osvrtom i na druge značajne detektore.

YOLO algoritam objedinjuje odvojene komponente detekcije objekata u jednu neuralnu mrežu gdje se predikcija svih graničnih okvira odvija istovremeno. YOLO dijeli ulaznu sliku u $S \times S$ ćelija gdje se za svaku ćeliju predviđa B graničnih okvira, kao i kojoj

klasi pripada. Ovo je predstavljeno nizom od C vrijednosti, gdje svaka vrijednost predstavlja vjerovatnoću pripadnosti određenoj klasi. Svaki od graničnih okvira predstavljen je sa pet ključnih parametara. Ovi parametri su koordinate centra graničnog okvira u odnosu na ćeliju (x,y), njegova visina i širina (h,w) u odnosu na cijelu sliku kao i pouzdanost koja izražava koliko je model siguran da okvir sadrži objekat. Samim tim, izlaz YOLO detektora je tenzor dimenzija $S \times S \times (B \times 5 + C)$. Uklanjanje preklapajućih okvira YOLO detektor rješava primjenom NMS algoritma [32].

Arhitekturu YOLO detektora čine 24 konvoluciona sloja praćena sa dva potpuno povezana sloja. Na početku autori treniraju prvih 20 slojeva na ImageNet skupu podataka [30]. Zatim se mreži dodaju posljednja 4 sloja sa nasumično inicijalizovanim težinskim parametrima. Nakon toga se vrši dotreniranje jedinstvene mreže na PASCAL VOC skupovima podataka.

Funkcija cijene koja se optimizuje kod YOLO detektora je:

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
&+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - p_i^t(c))^2.
\end{aligned} \tag{2.1}$$

U izrazu 2.1, faktor skaliranja $\lambda_{\text{coord}} = 5$ daje veću težinu predikcijama graničnih okvira, dok faktor $\lambda_{\text{noobj}} = 0.5$ smanjuje važnost okvira koji ne sadrže objekte. Oznaka 1_{obj}^{i} signalizira prisustvo objekta u i-toj ćeliji, dok 1_{obj}^{ij} označava da li se j-ti granični okvir odnosi na tu specifičnu detekciju.

YOLO postiže impresivnih 45 fps uz srednju prosječnu preciznost od $63,4\%$ na VOC07 skupu podataka. Važno je napomenuti da je u okviru YOLOv1 predstavljena i pojednostavljena arhitektura sa 9 konvolucionih slojeva, nazvana Fast YOLO, koja dostiže brzinu od 155 fps sa mAP od $52,7\%$.

Single Shot MultiBox Detector (SSD) [47] je drugi važan jednofazni detektor. SSD je

uspio nadmašiti YOLOv1 detektor u pogledu performansi, postigavši brzinu od 59 fps uz mAP od 74.3%. Arhitektura mreže temelji se na VGG-16, uz dodatne pomoćne slojeve radi poboljšanja performansi. Metoda usidrenih okvira (eng. *Anchor boxes*) uvedena je kako bi se efikasno riješio izazov detekcije objekata različitih veličina, s kojim se suočava YOLO. Iako je SSD detektor ostvario značajna poboljšanja u brzini i preciznosti, imao je poteškoća s detekcijom malih objekata. Ovaj problem je kasnije riješen korišćenjem naprednijih osnovnih arhitektura, poput ResNet-a, i drugih manjih optimizacija.

RetinaNet detektor [48] uvodi novu funkciju troška pod nazivom *focal loss* koja se definiše kao:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log p_t \tag{2.2}$$

gdje parametar γ mora zadovoljiti uslov $\gamma \geq 0$. Ova nova funkcija troška dobija se dodavanjem faktora $(1-p_t)^{\gamma}$ na krosentropiju i uvodi se kako bi se riješio problem neravnomjerne zastupljenosti klasa. Naime, većina ćelija koje detektor generiše predstavljaće pozadinu, odnosno dodjeljuje im se klasa sa vrijednošću nula. Ovaj problem neravnomjerne zastupljenosti dovodi do toga da su pozitivni primjeri, odnosno stvarni objekti, značajno manje prisutni u procesu treniranja, što značajno otežava ovaj proces. Arhitektura RetinaNet detektora temelji se na kombinaciji ResNet-a i FPN-a.

YOLOv2 [49] predstavlja značajno unapređenje prve verzije, donoseći poboljšanja na više nivoa. Potpuno povezani slojevi su u potpunosti uklonjeni, a arhitektura se sada oslanja isključivo na konvolucione slojeve. U osnovi ovog detektora nalazi se Darknet-19 mreža koja sadrži 19 konvolucionih i 5 slojeva agregacije. Budući da ne koristi potpuno povezane slojeve, ulazni podaci mogu biti različitih veličina. Kako bi bio robustan na različite veličine ulaznih slika, model je treniran tako da se rezolucija ulaza mijenja svakih deset serija. Dodatno, primijenjena su i poboljšanja u vidu normalizacije po seriji (eng. *batch normalization*) i metode usidrenih okvira. Uz sva ova poboljšanja, YOLOv2 je postigao srednju prosječnu preciznost od 78,6% na PASCAL VOC2007 skupu podataka, u poređenju sa 63,4% koje je ostvario YOLOv1.

YOLOv3 [50] dodatno produbljuje arhitekturu mreže koristeći Darknet-53 sa 53 konvoluciona sloja. Arhitekturi je dodat i SPP sloj. Model koristi višeklasnu klasifikaciju s nezavisnim logističkim klasifikatorima kako bi se bolje nosio sa složenim skupovima podataka koji sadrže preklapajuće oznake. Umjesto standardne *softmax* funkcije, primjenjuje se binarna krosentropija tokom treniranja klasifikatora, omogućavajući dodjeljivanje više oznaka jednom objektu. Kada je predstavljen YOLOv3, referentni skup podataka za evaluaciju performansi detektora postao je MS-COCO. Od tada se sve verzije YOLO al-

goritma ocjenjuju na ovom skupu podataka. YOLOv3 postiže AP od 36.2%.

YOLOv4 [51] predložen je 2020. godine. Poboljšanja koja ovaj algoritam donosi uključuju modifikovanu Darknet-53 arhitekturu sa *Mish* aktivacionom funkcijom, napredne tehnike treniranja poput *mosaic* augmentacije, *DropBlock* regularizacije i CIoU funkcije troška. Takođe, genetski algoritmi su korišćeni za fino podešavanje hiperparametara. YOLOv4 postiže AP od 43,5% na MS-COCO skupu podataka.

YOLOv5 [52] predložen je svega nekoliko mjeseci nakon YOLOv4. Ovaj algoritam koristi mnoge koncepte koji su predloženi u sklopu YOLOv4 algoritma. Među ključnim inovacijama je *AutoAnchor* algoritam koji automatski provjerava i prilagođava usidrene okvire ukoliko nisu optimalno usklađeni sa karakteristikama skupa podataka ili postavkama treniranja. U osnovi arhitekture mreže ovog algoritma stoji unaprijeđena verzija CSPDarknet53 mreže koja započinje sa tzv. *stem* slojem. *Stem* sloj je konvolucioni sloj sa velikim prozorom koji smanjuje memorijske i računske zahtjeve. YOLOv5 detektor nudi nekoliko verzija, odnosno modela, koji se razlikuju po širini i dubini konvolucionih slojeva, prilagođavajući se specifičnim potrebama aplikacija i hardverskim ograničenjima.

YOLOv6 [53] i YOLOv7 [54] predloženi su 2022. godine. YOLOv6 uvodi inovacije u vidu EfficientRep arhitekture i *VariFocal* funkcije troška za klasifikaciju, pri čemu postiže AP od 52.5% na MS-COCO skupu podataka [32]. YOLOv7 predložen je od strane iste istraživačke grupe koja je razvila YOLOv4 i donosi poboljšanja u brzini i tačnosti. Posebno se ističe uvođenjem *Efficient Layer Aggregation Network* (E-ELAN) arhitekture koja omogućava modelu da efikasnije uči i konvergira. YOLOv7 uspijeva nadmašiti svoje prethodnike postigavši AP od 56.8% na MS-COCO skupu podataka, čime se ističe kao jedan od najpreciznijih modela za detekciju objekata u realnom vremenu [32].

YOLOv8 [55] predložen je 2023. godine. Kao i YOLOv5, ovaj algoritam nudi nekoliko različitih modela, prilagođenih različitim potrebama i hardverskim ograničenjima. YOLOv8 omogućava rješavanje više zadataka iz oblasti kompjuterske vizije, kao što su detekcija objekata, segmentacija, procjena poze, praćenje i klasifikacija. Ova svestranost čini ga moćnim alatom za širok spektar primjena u kompjuterskoj viziji. YOLOv8 zadržava sličnu arhitekturu kao YOLOv5, međutim, metoda usidrenih okvira je izbačena iz ovog modela.

Detaljan pregled svih verzija YOLO detektora može se naći u radu [32].

Glava 3

Konvolucione neuralne mreže

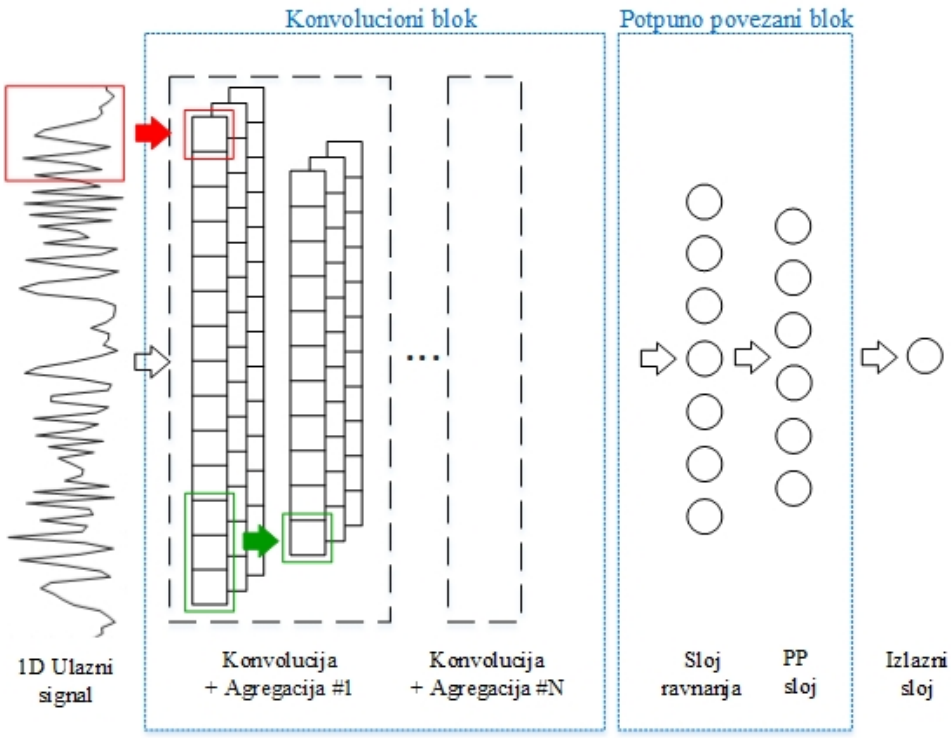
U posljednjoj deceniji, konvolucione neuralne mreže su, po svemu sudeći, kamen temeljac modernih dostignuća u oblasti vještačke inteligencije, a posebno u domenu kompjuterske vizije (eng. *computer vision*). Ove napredne neuralne mreže predstavljaju neospornu prekretnicu u pristupu analizi vizuelnih podataka, zahvaljujući svojoj sposobnosti prepoznavanja i interpretacije kompleksnih vizuelnih obrazaca. Konvolucione neuralne mreže predstavljaju ključni i primarni instrument u rješavanju problema kao što su klasifikacija slika, detekcija objekata i segmentacija slika.

U osnovi konvolucionih neuralnih mreža nalaze se konvolucioni slojevi koji se sastoje od seta prozora ili kernela kojima se vrše operacije konvolucije nad ulaznim podacima ekstrahujući relevantne podatke. Konvolucioni slojevi raspoređuju neurone u tri dimenzije: širinu, visinu i dubinu, koja predstavlja broj kanala u početnom sloju, a u kasnijim slojevima broj prozora koji se primjenjuje. Kod jednodimenzionalnih konvolucionih neuralnih mreža (eng. *one-dimensional convolutional neural network* - 1D-CNN), kako radimo sa 1D signalima, visina je jedan, pa se ova dimenzija eliminiše.

Na slici 3.1 prikazana je jedna tipična arhitektura 1D-CNN, podijeljena na dva glavna bloka: konvolucioni blok i potpuno povezani blok. Ulazni signal, predstavljen u talasnom obliku, prolazi kroz niz konvolucionih slojeva unutar konvolucionog bloka, gdje se primjenjuju 1D konvolucije, aktivacije i agregacije. Ovi slojevi obrađuju signal i izvlače relevantne karakteristike. Nakon toga, obrađeni podaci ulaze u potpuno povezani blok, koji sadrži sloj ravnanja i jedan ili više potpuno povezanih ili gustih slojeva, što na kraju vodi do izlaznog sloja.

Ovo poglavlje ima za cilj da istraži konvolucione neuralne mreže, razmatrajući njihovu arhitekturu i mehanizme učenja. Fokus će biti na 1D-CNN, budući da su upravo

one korišćene u ovom istraživanju. Sve rečeno primjenjivo je i odnosi se također i na dvodimenzionalne. Priča o konvolucionim neuralnim mrežama biće podijeljena u nekoliko sekcija.



Slika 3.1: Tipična arhitektura 1D konvolucione neuralne mreže. PP je potpuno povezani sloj.

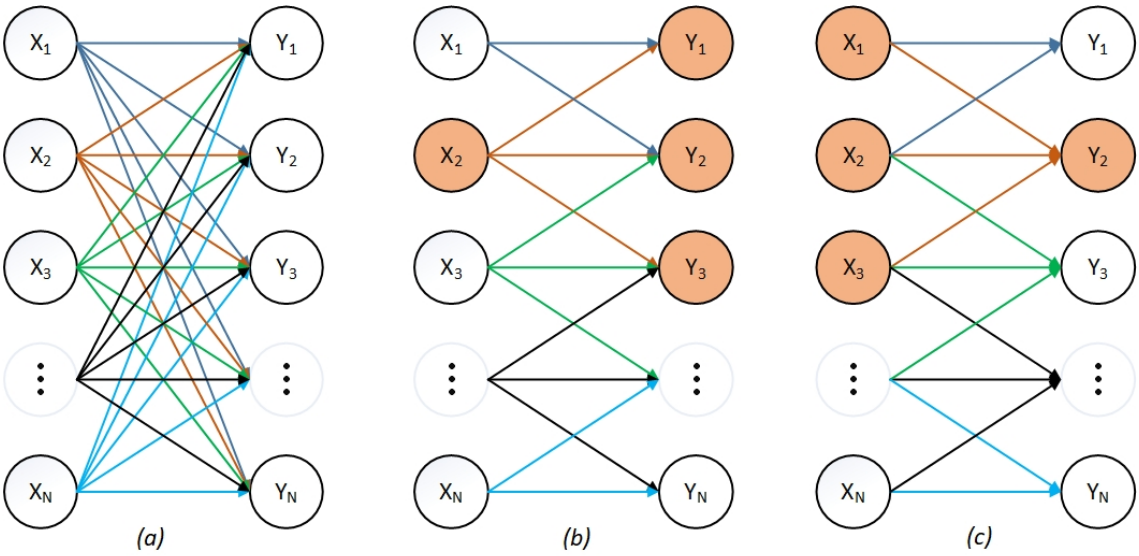
3.1 Motivacija

Konvolucione neuralne mreže nude nam nekoliko značajnih prednosti u odnosu na klasične, potpuno povezane neuralne mreže (eng. *fully connected neural networks*). Funkcionisanje potpuno povezanih neuralnih mreža zasniva se na množenju matrica. Ulaz sloja množi se matricom parametara gdje se svaka interakcija između ulazne i izlazne jedinice opisuje posebnim parametrom, odnosno težinskim koeficijentom. To znači da svaka izlazna jedinica sloja interaguje sa svakom ulaznom jedinicom.

Kod konvolucionih neuralnih mreža svaki neuron nije više povezan sa svim neuronima iz prethodnog sloja već samo sa malim brojem neurona iz prethodnog sloja (slika 3.2). Ova karakteristika konvolucionih neuralnih mreža naziva se lokalna (eng. *local*) ili rijetka

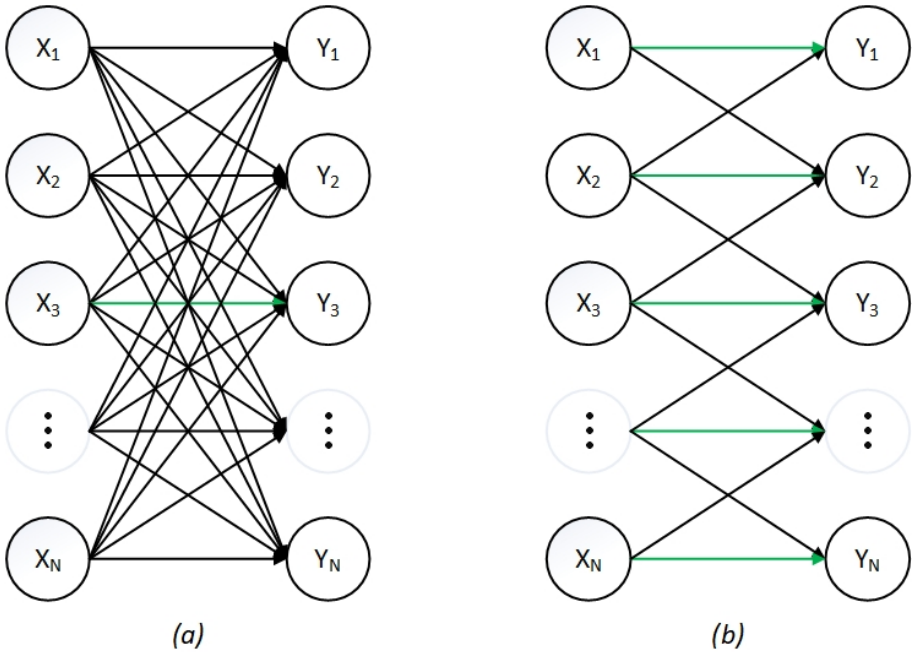
(eng. *sparse*) konekcija, u zavisnosti od literature. To se postiže postavljanjem prozora (kernela) koji je dimenzionalno manji od ulaza.

Kako bismo bolje približili ovaj koncept, nakratko ćemo se osvrnuti na 2D konvolucione mreže. Ukoliko govorimo o primjeni mreže nad slikom, možemo reći da potpuno povezani slojevi uče globalne obrasce tj. obrasce koji povezuju sve piksele unutar slike. Za razliku od njih, konvolucioni slojevi uče da prepoznaju lokalne obrasce kroz male 2D prozore. Kao rezultat, broj težinskih koeficijenata u mreži je znatno manji, što je veoma efikasno sa stanovišta memorijske potrošnje. Pored toga, operacija konvolucije je računski manje zahtjevna od matričnog množenja.



Slika 3.2: Rijetka konekcija: *(a)* Potpuno povezana neuralna mreža: svaka izlazna jedinica sloja interaguje sa svakom ulaznom jedinicom. *(b)* Rijetka konekcija, pogled odozdo: Ulaz X_2 povezan je samo sa podskupom izlaznih jedinica Y. *(c)* Rijetka konekcija, pogled odozgo: Ističe se jedna izlazna jedinica Y_2 kao i ulazne jedinice koje utiču na nju. Ove ulazne jedinice predstavljaju receptivno polje (eng. *receptive filed*) izlaza Y_2.

Dijeljenje parametara (eng. *Weight sharing)* još jedan je nezaobilazni koncept konvolucionih neuralnih mreža. Kod klasičnih neuralnih mreža svaki od elemanta matrice parametara koristi se tačno jednom prilikom računanja izlaza. Množi se sa jednim elementom ulaznog sloja i nigdje više se ne koristi, što znači da za svaki neuron koristimo različite težinske parametre. Kod konvolucionih mreža, sa druge strane, dijeljenje težinskih parametara odnosi se na postupak korišćenja istog skupa težina (prozora) na različitim prostornim lokacijama ulaznih podataka. Umjesto učenja odvojenih parametara za



Slika 3.3: Dijeljenje parametara: Zelene strelice prikazuju upotrebu određenog težinskog koeficijenta u dva modela. *(a)* Jedna zelena strelica ukazuje na upotrebu elementa težinske matrice u potpuno povezanom modelu. Parametar se koristi samo jednom - nema dijeljenja parametara. *(b)* U konvolucionoj mreži, centralni element prozora veličine 3 koristi se na svim ulazima, kao posljedica dijeljenja parametara.

svaku lokaciju, mreža koristi jedan prozor da skenira ili konvoluira čitav ulaz. Koncept dijeljenja parametara prikazan je na slici 3.3.

Predstavljeni koncept daje konvolucionim mrežama dvije vrlo značajne osobine koje opravdavaju njihovu primjenu, naročito u problemima kompjuterske vizije. Obrasci koje konvolucione neuralne mreže uče su translaciono invarijantni. Nakon što mreža nauči odgovarajući obrazac u nekom dijelu slike može ga prepoznati ma gdje god se on nalazio na slici. Potpuno povezana mreža bi morala ponovo da uči taj obrazac ukoliko se on pojavi na nekom drugom dijelu slike. Zbog toga je konvolucionim mrežama potrebno znatno manje podataka za treniranje, jer imaju veću moć generalizacije.

Takođe, konvolucione mreže imaju sposobnost učenja prostornih hijerarhija obrazaca. U početnim slojevima, obično se detektuju jednostavne karakteristike, poput ivica, boja i tekstura. Kako se ide dublje u mrežu, slojevi nadograđuju karakteristike koje su izdvojene prethodnim slojevima, hvatajući sve složenije, apstraktne i kompleksnije reprezentacije. Na primjer, dok početni slojevi mogu prepoznati obris oka ili nosa, dublji slojevi mogu prepoznati čitave strukture lica ili čak specifične izraze. Slojevi agregacije, često ubačeni

između konvolucionih slojeva, dodatno doprinose hijerarhijskom učenju smanjenjem prostornih dimenzija, čineći mrežu postepeno invarijantnom prema tačnim prostornim lokacijama karakteristika. Ovo hijerarhijsko i prostorno učenje daje konvolucionim neuralnim mrežama posebnu efikasnost u zadacima poput prepoznavanja slika, gde je razumijevanje prostornih hijerarhija i odnosa između karakteristika ključno za tačnu klasifikaciju.

3.2 Algoritam propagacije unaprijed

Važno je objasniti operaciju koja je u prethodnom dijelu poglavlja istaknuta kao srž konvolucionih neuralnih mreža - konvoluciju. U kontekstu dubokog učenja, operacija i sam termin "konvolucija" ne prati strogu definiciju tradicionalne obrade signala, već se više podudara, sa konvoluciji veoma sličnom operacijom, kros korelacijom. Operacija konvolucije u kontekstu neuralnih mreža, podrazumijeva pomjeranje prozora preko ulaznih podataka konvolucionog sloja, pri čemu se vrši operacija množenja koeficijenata prozora i preklapajućih elemenata ulaza, a zatim se ti proizvodi sabiraju. Iako se naziva "konvolucija", ova operacija je zapravo sličnija kros korelaciji jer ne dolazi do rotiranja prozora prije primjene [56]. Međutim, kako se termin konvolucija odomaćio u literaturi i oblasti dubokog učenja, koristićemo ovaj jedinstveni termin za obje operacije.

Kros korelacija se često koristi kao mjera sličnosti između dva signala, što je logično jer se prilikom treniranja mreže filtri uče da prepoznaju različite karakteristike u signalu [56–58].

Radi jednostavnosti, u ovoj i narednim sekcijama, kao pokazni primjer funkcionisanja 1D-CNN koristićemo mrežu sa jednim skrivenim slojem, uz osvrt na opšti slučaj višeslojne mreže.

S tim rečeno, za ulazni signal x dužine N, dat kao:

$$x = [x(0), x(1), \ldots, x(N-1)]^T \tag{3.1}$$

i k-ti prozor w_k dužine M dat kao:

$$w_k^1 = \left[w_k^1(0), w_k^1(1), \ldots, w_k^1(M-1)\right]^T, k = 1, 2 \ldots, K \tag{3.2}$$

k-ti izlaz konvolucionog sloja definiše se kao:

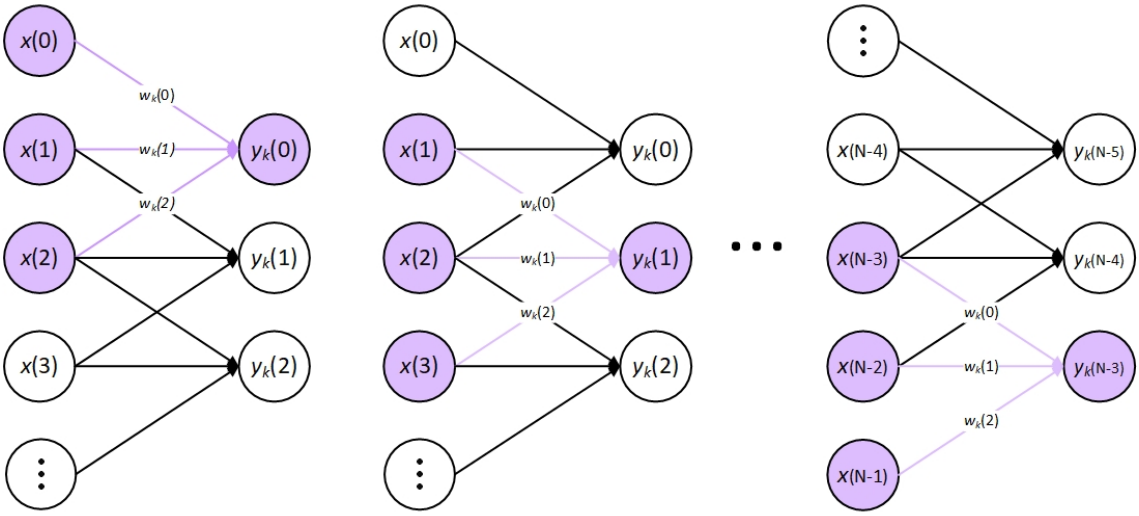
$$y_k^1(i) = \sum_{m=0}^{M-1} w_k^1(m)x(i+m), i = 0,1\ldots,I-1 \tag{3.3}$$

gdje I predstavlja dužinu izlaza konvolucionog sloja.

Dimenzije filtra su najčešće tri ili pet, pa će za vrijednost $M = 3$ jednačina 3.3 u razloženom obliku biti:

$$y_k^1(i) = w_k^1(0)x(i) + w_k^1(1)x(i+1) + w_k^1(2)x(i+2) \tag{3.4}$$

Prolazak konvolucionog prozora w_k kroz ulazni signal x prikazan je na slici 3.4.



Slika 3.4: Prikaz konvolucije sa prozorom širine 3 u 1D-CNN, gdje svaki izlazni čvor y zavisi od tri uzastopna ulazna čvora x. Strelice prikazuju veze između ulaznih i izlaznih jedinica, dok $w_k(0)$, $w_k(1)$ i $w_k(2)$ predstavljaju težinske koeficijente prozora koji se primjenjuju na svaku grupu ulaza.

Na dimenzije izlaznog signala može uticati nekoliko faktora: dubina konvolucionog sloja (broj filtara kojim se ulaz konvoluira), veličina filtra, dopunjavanje nulama (eng. *zero padding*), pomjeraj (eng. *stride*) kao i slojevi agregacije (eng. *pooling layers*) koji, u konvolucionom bloku, dolaze nakon konvolucionih slojeva i o kojima će više biti riječi u sekciji 3.4.

Dimenzije k-tog izlaza konvolucionog sloja računaju se kao $I = N - M + 1$, što znači da će, u našem slučaju, posljednji element signala y_k biti definisan za $i = N - 3$. Kako se

nad ulaznim signalom x primjenjuje K različitih prozora istih dimenzija, izlaz konvolucionog sloja sastoji se od K nizova, svaki sa po I elemenata. Ukupan broj elemenata izlaza stoga iznosi $K(N-M+1)$.

Očigledno je da se prostorne dimenzije ulaznog signala smanjuju nakon svakog prolaska kroz konvolucione slojeve. Stopa ovog smanjenja može biti dramatična ukoliko koristimo prozore većih dimenzija, pa to može ograničiti broj konvolucionih slojeva koji se mogu uključiti u mrežu. Ukoliko uzmemo u obzir pomjeraj i slojeve agregacije koji se često mogu naći u arhitekturi mreže i koji također utiču na smanjenje dimenzija izlaznog signala, vrlo brzo možemo doći do takvih dimenzija signala da nam je svako dalje dodavanje konvolucionih slojeva besmisleno pa čak i nemoguće.

Dopunjavanje ulaznog signala nulama, kao hiperparametar mreže, omogućava nam kontrolu dimenzija izlaznog signala. Koristi se ukoliko želimo da nam rezultat konvolucije bude istih dimenzija kao i ulazni signal, i time ne ograničimo organizaciju i strukturu narednog sloja. U tom slučaju ulazni signal je potrebno dopuniti sa ukupno $M-1$ nula. Ukoliko nam je broj elemenata filtra neparan, dopunjava se sa $(M-1)/2$ nula simetrično sa obje strane. Tada će ukupan broj elemenata signala y_k biti $I = N-M+1+P$ gdje P predstavlja ukupan broj dodatih nula.

Ukoliko je ulazni signal sporo promjenjiv i ima dovoljan broj odbiraka, konvolucija tj. vrijednosti $y_k(i)$ ne moraju se računati za svaki vremenski trenutak i. Uvođenjem pomjeraja, koji je također još jedan hiperparametar mreže, definišemo u kojim trenucima i će se ulaz konvoluirati odnosno na koji način će se prozor pomjerati. Ovo u suštini predstavlja smanjenje broja uzoraka (eng. *downsampling*), a sam pomjeraj mjeru smanjenja. Ukoliko svaku sljedeću konvoluciju, nakon početne koja je u trenutku $i = 0$, računamo za trenutak $i+1$, kao što je predstavljeno na slici 3.4, vrijednost pomjeraja je jedan. Međutim, ukoliko preskočimo jedan trenutak i svaku narednu konvoluciju, nakon početne, računamo za $i+2$, vrijednost pomjeraja je dva itd. Uvođenje hiperparametra pomjeraja rezultiraće smanjenjem broja uzoraka signala y_k. Dužina signala računaće se kao $I = \frac{N-M+P}{S} + 1$, gdje S predstavlja uvedeni pomjeraj. Naravno, moramo paziti prilikom odabira pomjeraja tako da on zadovoljava uslov $1 \leq S \leq (N-M+P)$ i da je vrijednost $N-M+P$ cjelobrojni umnožak broja S. Ukoliko nijesu zadovoljeni ovi uslovi, I će biti necjelobrojna vrijednost i ovakva postavka hiperparametara je nevažeća. Biblioteka za konvolucione neuralne mreže bi mogla javiti grešku ali i odraditi dopunjavanje nulama ili skraćivanje ulaznog signala kako bi se prilagodila, što je također nepovoljan rezultat jer gubimo kontrolu nad izlaznim podacima sloja čineći ih nepredvidivim.

Ukupan broj težinskih parametara nakon prvog konvolucionog sloja mreže biće direktno proporcionalan veličini filtra i broju filtara u tom sloju, odnosno iznosiće MK. Bitno je napomenuti da će ovo važiti samo ukoliko je ulaz mreže jednodimenzionalni signal. U opštem slučaju, potrebno je uzeti u obzir i dubinu prethodnog sloja. Dubina može predstavljati broj filtara u prethodnom sloju, ali i broj kanala ulaznih podataka prvog sloja. Ako dubinu prethodnog sloja označimo kao K_p, onda broj parametara konvolucionog sloja računamo kao MKK_p. Ovo predstavlja značajnu memorijsku uštedu u odnosu na potpuno povezane mreže gdje je broj parametara sloja proporcionalan dimenzijama ulaznih podataka. Kada uzmemo u obzir da možemo raditi sa slikama velike rezolucije gdje jedna slika može imati više miliona piksela i da većina mreža ima više skrivenih slojeva, vidimo da se broj težinskih parametara kod potpuno povezanih mreža veoma brzo povećava i da ovo predstavlja njihovo značajno ograničenje.

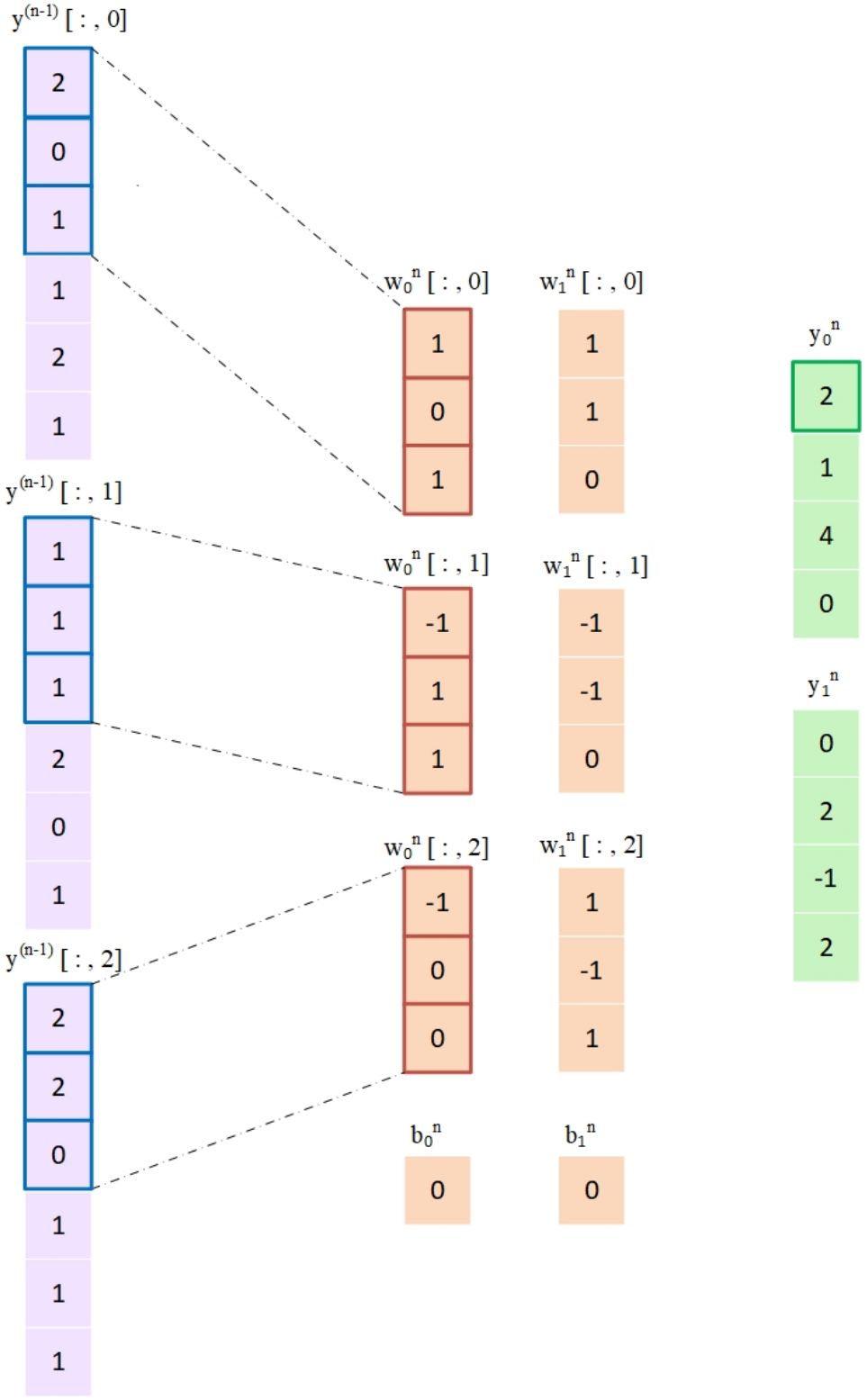
Konvolucionim slojevima se, kao i kod potpuno povezanih slojeva, može dodati *bias*. Ista vrijednost *bias*-a b_k dodaje se svim elementima $y_k(i)$, što će rezultirati povećanjem ukupnog broja parametara konvolucionog sloja za K. Pa će ukupan broj parametara konvolucionog sloja, u opštem slučaju, biti $K(MK_p+1)$. Vrijednosti *bias*-a b_k za različito k ne moraju biti iste.

Jednačina 3.3 sada postaje:

$$y_k^1(i) = \sum_{m=0}^{M-1} w_k^1(m)x(i+m) + b_k^1. \tag{3.5}$$

Konvolucioni sloj je tipično praćen aktivacionom funkcijom i slojem agregacije o kojima će više riječi biti u sekcijama 3.3 i 3.4, resprektivno. Niz ovih operacija konvolucija-aktivacija-agregacija, u jednoj konvolucionoj neuralnoj mreži, zajedno čini konvolucioni blok (slika 3.1).

Napomena: Izraz 3.5 važiće ukoliko je ulaz konvolucionog sloja x jednodimenzionalni signal, odnosno važiće samo za prvi konvolucioni sloj mreže. Kao što smo već napomenuli u prethodnom dijelu teksta, svaki naredni sloj sa sobom nosi određenu dubinu direktno proporcionalnu broju prozora prethodnog sloja. Nakon prvog sloja, ulazne podatke narednih slojeva posmatramo kao K_p nizova y_k, odnosno kao matrice nizova dimenzija $(\frac{N_p-M_p+P_p}{S_p}+1)\times K_p$, pod pretpostavkom da konvolucione slojeve ne prate agregacioni slojevi. Indeks p ukazuje na to da govorimo o dimenzijama i hiperparametrima prethodnog sloja. Naravno, i konvolucioni prozori moraju pratiti višekanalnost ulaznih podataka, pa se ulazni podaci konvoluiraju, u opštem slučaju, sa K filtara veličine $M \times K_p$.



Slika 3.5: Slika prikazuje princip funkcionisanja konvolucionih slojeva u opštem slučaju. Vrijednosti hiperparametara su: $P_p = 0$, $S_p = 1$, $K_p = 3$, $K = 2$ i $M = 3$.

Iz rečenog slijedi da se izlaz n-tog konvolucionog sloja, pod pretpostavkom da nije izvršena agregacija i da je vrijednosti pomjeraja jedan, računa kao:

$$y_k^n(i) = \sum_{l=0}^{K_p-1} \sum_{m=0}^{M-1} w_k(m,l) y^{n-1}(i+m,l) + b_k^n \tag{3.6}$$

što je i prikazano na slici 3.5.

Dakle, izlaz konvolucionog bloka čini K vektora koji se prosljeđuju potpuno povezanom bloku, gdje K sada predstavlja broj prozora posljednjeg konvolucionog sloja. Ukoliko su u posljednjem konvolucionom sloju, vrijednosti hiperparametara dopunjavanja nulama i pomjeraja iznosile nula i jedan, i ako poslednji sloj nije praćen agregacionim slojem, dimenzije ovih vektora iznosiće $N-M+1$, gdje N predstavlja broj elemenata ulaza posljednjeg sloja a M dimenzije prozora w_k posljednjeg konvolucionog sloja.

Sa druge strane, potpuno povezani blok sastoji se od sloja ravnanja (eng. *flattening layer*), jednog ili više potpuno povezanih slojeva i izlaznog sloja mreže (Slika 3.1). Sloj ravnanja ima za cilj povezati K proslijeđenih vektora u jedinstveni vektor. Ovo omogućava dalju obradu podataka u potpuno povezanim slojevima, gde je svaki neuron povezan sa svim neuronima u narednom sloju. Ova složena međusobna veza omogućava mreži da nauči kompleksne obrasce i strukture u podacima.

Na kraju, izlazni sloj, uz upotrebu odgovarajućih aktivacionih funkcija generiše krajnje predikcije na osnovu naučenih karakteristika.

Prisustvo potpuno povezanih slojeva značajno će uticati na ukupan broj težinskih koeficijenata mreže. Konkretno, prvi potpuno povezani sloj nakon konvolucionog bloka povećava ukupan broj težinskih parametara za $K(N-M+1)d_1+d_1$, gdje d_1 predstavlja broj neurona u tom sloju. Smanjenjem dimenzija izlaza konvolucionih slojeva, smanjiće se i broj karakteristika koje se prosljeđuju potpuno povezanim slojevima, čime se može kontrolisati rast broja parametara mreže.

3.3 Aktivacione funkcije

Svaki sloj mreže preuzima izlaz prethodnog sloja kao ulaz, obrađuje ga i zatim prosljeđuje dalje kroz mrežu. U višeslojnoj neuralnoj mreži, nelinearne aktivacione funkcije služe kao veza između slojeva. U suštini, transformišu linearnu kombinaciju neurona sloja

unoseći nelinearnosti u prenosnu funkciju mreže. Aktivacione funkcije su defakto sastavni dio svake konvolucione neuralne mreže i nalaze se nakon svakog sloja učenja (slojevi sa težinskim parametrima koji se prilagođavaju tokom treninga mreže). Za zadati ulaz definišu izlaz:

$$f(y_k^1(i)) = f(\sum_{m=0}^{M-1} w_k^1(m)x(i+m) + b_k^1) \tag{3.7}$$

gdje f predstavlja korišćenu nelinearnu aktivacionu funkciju.

Pokazuje se da bi u krajnjem, bez korišćenja aktivacionih funkcija, izlaz mreže predstavljao linearnu kombinaciju ulaza, a što je i očigledno iz jednačine 3.3. To znači da skriveni slojevi ne bi imali veliki uticaj i sposobnost učenja takve mreže je ograničena. Uvođenjem nelinearnosti u mrežu, značajno se povećava njena moć učenja kompleksnih karakteristika a samim tim omogućava i rješavanje složenijih problema.

Najčešće korišćene aktivacione funkcije u dubokim neuralnim mrežama, uključujući i konvolucione mreže, su:

1. **Sigmoid**

   Sigmoid aktivaciona funkcija, poznata i pod nazivom logistička funkcija preslikava, odnosno ograničava proslijeđeni ulaz u opseg $[0,1]$ pa svoje primjene često nalazi u problemima binarne klasifikacije. Njen matematički oblik je:

   $$f(x) = \frac{1}{1+e^{-x}} \tag{3.8}$$

   i prikazana je na slici 3.6. Ova aktivaciona funkcija sklona je zasićenju, fenomenu gdje izlaz funkcije postaje relativno konstantan za ekstremne ulazne vrijednosti. Drugim riječima, za veoma velike ili veoma male ulazne vrijednosti, izlaz sigmoid funkcije ostaje blizu vrijednosti 1 ili 0, pa promjene u ulazu rezultiraju gotovo zanemarljivim promjenama u izlazu. Zasićenje uzrokuje pojavu problema nestajućeg gradijenta. Kada gradijent postane veoma mali, težinski parametri se veoma sporo ili uopšte ne ažuriraju, što može ozbiljno usporiti ili čak zaustaviti proces treniranja neuralne mreže [59].

2. **Hiperbolički tangens**

   Hiperbolički tangens, za razliku od sigmoida, ulaz preslikava u opseg $[-1,1]$. Međutim, problem nestajućeg gradijenta odlikuje i ovaj tip aktivacione funkcije. Njen matematički oblik je:

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.9}$$

i prikazana je na slici 3.6 .

3. **ReLU**

   ReLU (eng. *Rectified Linear Unit*) najkorišćenija je aktivaciona funkcija u konvolucionim neuralnim mrežama. Njen matematički oblik je:

$$f(x) = max(0,x) = \begin{cases} \text{x}, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.10}$$

   i prikazana je na slici 3.6 . ReLU uvodi nelinearnost tako što za negativne ulazne vrijednosti daje izlaz nula, dok za pozitivne ulazne vrijednosti zadržava linearnost. Ova karakteristika omogućava brže konvergiranje tokom procesa treniranja tako što u velikoj mjeri umanjuje problem nestajućeg gradijenta koji se javlja kod prethodno pomenutih aktivacionih funkcija. Dodatno, ReLU je računski efikasnija u poređenju sa ostalim aktivacionim funkcijama zahvaljujući svojoj djelimično linearnoj prirodi, što omogućava bržu i lakšu obradu.

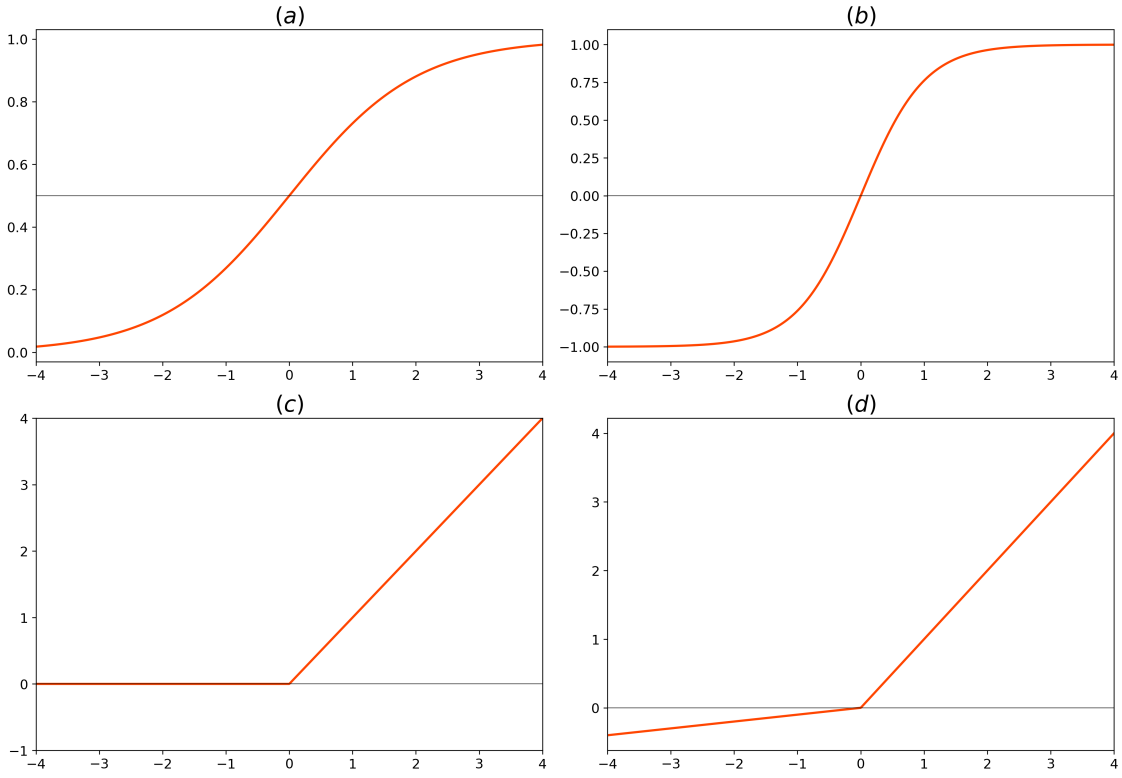
   Međutim, ReLU nije bez svojih ograničenja, budući da se suočava sa problemom "umirućeg ReLU-a"(eng. *Dying ReLU problem*), gdje neuroni postaju neaktivni za negativne ulaze, što potencijalno dovodi do, uslovno rečeno, neaktivnih puteva u mreži [60].

4. **Leaky ReLU**

   Leaky ReLU predstavlja modifikaciju ReLU funkcije i prikazana je na slici 3.6. Za razliku od ReLU funkcije, koja za negativne ulazne vrednosti daje izlaz nula, Leaky ReLU negativne ulaze preslikava u neku malu vrijednost čime se umnogome rješava problem "umirućeg ReLu". Njen matematički oblik je:

$$f(x) = \begin{cases} \text{x}, & x > 0 \\ \text{kx}, & x \leq 0 \end{cases} \tag{3.11}$$

   gdje je k obično veoma mala vrijednost.



Slika 3.6: Aktivacione funkcije: *(a)* Sigmoid *(b)* Hiperbolički tangens *(c)* ReLU *(d)* LeakyReLU

3.4 Slojevi agregacije

Često se u arhitekturi konvolucionih neuralnih mreža mogu naći periodično umetnuti slojevi agregacije (eng. *Pooling layers)* između uzastopnih konvolucionih slojeva. Rezultat dodavanja slojeva agregacije u mrežu je postupno smanjenje prostornih veličina ulaznih podataka čime se umanjuje računska složenost mreže ali i kontroliše pojava preprilagođavanja mreže, koja je detaljnije objašnjena u sekciji 3.8. Samim tim, dodavanje slojeva agregacije uticaće i na smanjenje ukupnog broja težinskih parametara mreže, konkretno parametara potpuno povezanih slojeva koji su direktno proporcionalni broju ulaznih podataka. Kao i u prethodnom dijelu pomenuti pomjeraj, predstavlja još jedan mehanizam za redukciju kompleksnosti mreže, omogućavajući efikasniju obradu podataka.

Sloj agregacije djeluje nezavisno na svakoj dubini ulaza pri čemu ne utiče na istu (slika 3.7). Kao što je već rečeno, broj primijenjenih prozora jedini je faktor koji kontroliše dubinu izlaza sloja.

Ovi slojevi nemaju težinske parametre već funkcionišu tako što prolaze kroz ulazne podatke zamjenjujući grupe podataka (pravougaono susjedstvo) sa jednom vrijednošću koja predstavlja neki rezime, odnosno statistiku grupe. Na primjer, maksimum agregacioni sloj (eng. *max pooling*), koji se izdvaja kao najpopularniji, zamijeniće pravougaono susjedstvo sa vrijednošću najvećeg člana grupe (slika 3.7). Druge popularne funkcije agregacije uključuju prosječnu vrijednost pravougaonog susjedstva, L_2 normu pravougaonog susjedstva itd.

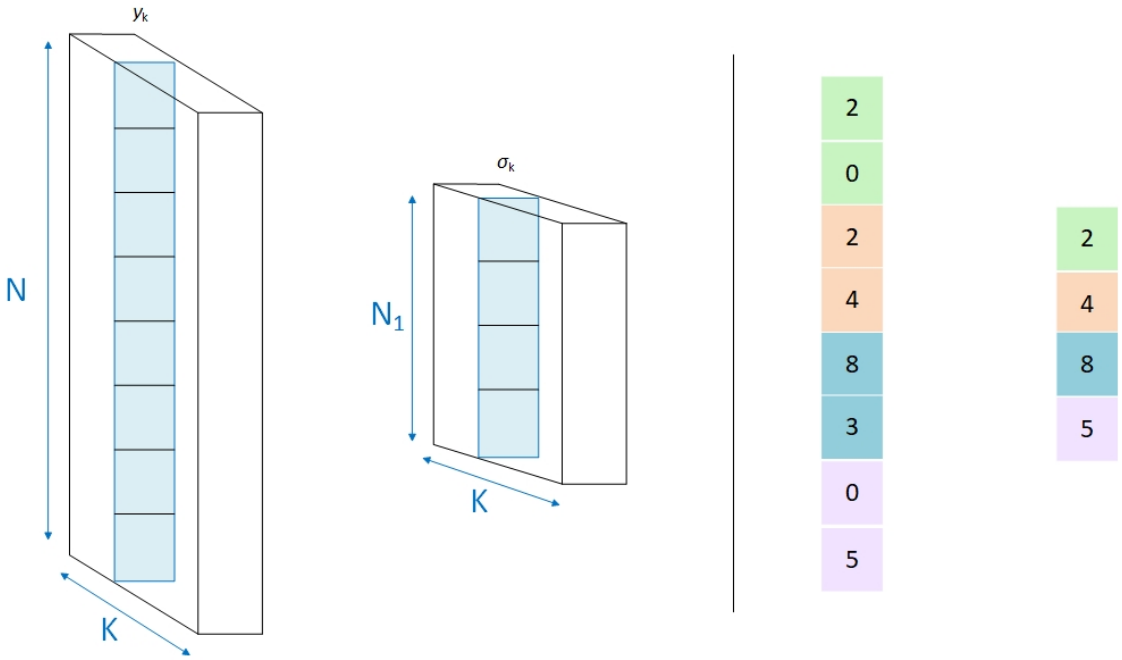
Svaki agregacioni sloj ima dva parametra koji ga definišu, veličinu prozora, odnosno pravougaonog susjedstva i pomjeraj koji određuje korak kojim se prozor za uzimanje podataka pomjera duž ulaza. Ova dva parametra će direktno uticati na dimenzije izlaznog signala agregacionog sloja.

Matematički, maksimum agregacioni sloj, za vrijednost pomjeraja jedan, definiše se kao:

$$\sigma_k(i) = max\{f(y_k(i)), f(y_k(i+1)), \ldots, f(y_k(i+L-1))\}. \tag{3.12}$$

Za dužinu ulaznog signala y_k označenu kao N, veličinu prozora L i vrijednost pomjeraja S, dužinu signala σ_k računamo kao $(N-L)/S+1$.

Umetnuti maksimum agregacioni sloj uvešće određeni nivo invarijatnosti na male lokalne translacije u signalu. Invarijatnost na translacije znači da ako pomjerimo ulazne



Slika 3.7: Primjer: *(Lijevo)* smanjenja dimenzija ulaznog signala gdje N i N_1 predstavljaju prostorne dimenzije, a K broj kanala. Agregacija se vrši po svakom kanalu. *(Desno)* funkcionisanja maksimum agregacionog sloja sa veličinom prozora 2 i istim tolikim pomjerajem. Sloj odbacuje 50% vrijednosti.

podatke za neku malu vrijednost većina vrijednosti izlaza sloja agregacije ostaje nepromijenjena, što je prikazano sa slici 3.7.

Ovo je veoma korisna osobina ukoliko rješavamo neki problem klasifikacionog tipa gdje nam je daleko važnije da li je neka karakteristika prisutna nego njena tačna lokacija u signalu. Umetanjem maksimum agregacionog sloja nakon svakog ili svakih nekoliko konvolucionih slojeva moguće je postići translacionu invarijantnost na većem nivou.

Ipak, primjena maksimum agregacionog sloja donosi određene nedostatke. Prvo, očigledno je da može biti vrlo destruktivno: primjena prozora veličine 2 sa korakom 2 rezultira izlazom koji je dva puta manji od ulaza, odbacujući 50% ulaznih vrijednosti (slika 3.7). Osim toga, postoji niz situacija u kojima invarijantnost nije poželjna osobina. Na primjer, kod semantičke segmentacije (zadatka klasifikacije svakog piksela na slici prema objektu kojem taj piksel pripada) jasno je da, ukoliko se ulazna slika pomjeri za jedan piksel udesno, očekivano je da se i rezultat pomjeri za jedan piksel u istom smjeru. U ovom kontekstu, traži se ekvivarijantnost, odnosno da male promjene ulaza rezultiraju proporcionalnim promjenama izlaza, a ne invarijantnost.

Za razliku od aktivacionih funkcija, koje su neophodan dio svake mreže i ključne su za njen pravilan rad, agregacioni slojevi više predstavljaju dodatnu mogućnost koja se koristi u skladu sa konkretnim zadatkom koji želimo riješiti. Obično ih želimo u situacijama kada se suočavamo sa zadacima klasifikacije, zahvaljujući njihovoj sposobnosti da u velikoj mjeri umanje važnost tačnih lokacija specifičnih karakteristika, kao i u regresionim zadacima, gdje pomažu u kontroli pretjeranog prilagođavanja mreže.

3.5 Funkcija troška

Iz dosada rečenog, jasno je da se duboko učenje temelji na uspostavljanju veza između ulaznih podataka, kao što su signali ili slike, i ciljnih rezultata, određene klase ili vrijednosti u zavisnosti od problema koji rješavamo. Duboke neuralne mreže ostvaruju ovo mapiranje ulaza u ciljeve kroz seriju, možemo slobodno reći, jednostavnih transformacija podataka, čije se podešavanje vrši kroz proces treniranja neuralne mreže.

Funkcionalnost sloja, odnosno način na koji sloj obrađuje ulazne podatke, određena je njegovim težinama, koji su u suštini skupovi numeričkih vrijednosti. Tehnički gledano, može se reći da je transformacija koju sloj vrši parametrizovana njegovim težinama. U ovom kontekstu, proces učenja podrazumijeva identifikaciju odgovarajućeg skupa vrijednosti za težinske parametre svih slojeva unutar mreže, kako bi se omogućilo što preciznije mapiranje ulaznih podataka u željeni cilj.

Međutim, kao što smo do sada vidjeli, duboka neuralna mreža može imati milione takvih parametara, što potragu za optimalnim vrijednostima čini izuzetno složenim zadatkom, posebno imajući u vidu da promjena jednog parametra utiče na ponašanje svih ostalih.

Da bi se nešto kontrolisalo, nužno je da to nešto bude mjerljivo. Za kontrolu izlaza neuralne mreže, ključno je mjerenje razlike između dobijenog izlaza i očekivanog rezultata, odnosno numerički predstaviti grešku koju mreža pravi. Mjerenje greške obavlja funkcija troška (eng. *Loss function*). Funkcija troška upoređuje izlaz mreže sa stvarnim ciljem (tačnim izlazom) i računa razliku, odnosno ocjenjuje uspješnost mreže na konkretnom primjeru. Veoma je važno da odabir funkcije troška odgovara tipu problema koji rješavamo.

Kod regresionih problema često se pribjegava korišćenju srednje kvadratne greške (eng. *mean squared error - MSE*) kao funkcije troška. Ona se definiše kao:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2 \tag{3.13}$$

i predstavlja kvadrat razlike između izlaza mreže y_i i očekivanih rezultata t_i, uzet u prosjeku preko skupa podataka odnosno uzoraka N koji je proslijeđen mreži. Ukoliko znamo da ulaz mreže sadrži veliki broj izuzetaka (eng. *outlier*), odnosno podataka koji znatno odudaraju od ostalih podataka u skupu, MSE nije najbolje rješenje. Pošto je greška kvadratna, a mreža teži prilagođavanju većini podataka, greške uzrokovane izuzecima biće neproporcionalno kažnjene. Ovo može dovesti do toga da model postane prekomjerno fokusiran na smanjenje tih izuzetaka umjesto na učenje generalnog trenda podataka.

U takvim slučajevima, MSE može rezultirati modelom koji nije optimalno prilagođen stvarnoj raspodjeli podataka, već je previše prilagođen smanjenju efekata izuzetaka. Ovo može ograničiti sposobnost modela da pravilno generalizuje jer se ključni aspekti skupa podataka mogu zanemariti ili pogrešno interpretirati. Zbog toga se u navedenom slučaju pribjegava srednjoj apsolutnoj grešci (eng. *mean absolute error - MAE*) kao funkciji troška mreže koja pruža ravnomjerniju "kaznu" za greške. Ona se definiše kao:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - t_i| \tag{3.14}$$

Alternativno, može se koristiti i Huberov gubitak (eng. *The Huber loss*) koji predstavlja njihovu kombinaciju. Huberov gubitak je kvadratna funkcija kada je greška manja od praga δ (tipično 1) a linearna kada je veća [61].

3.6 Treniranje neuralnih mreža

Strategija u dubokom učenju jeste korišćenje ove ocjene kao povratne informacije za finu korekciju vrijednosti težinskih parametara, usmjeravajući ih u pravcu koji smanjuje grešku za analizirani primjer. Za ovakvo prilagođavanje zadužen je optimizator, koji implementira algoritam poznat kao algoritam propagacije unazad (eng. *backpropagation*), koji predstavlja srž dubokog učenja. U ovoj sekciji biće detaljnije objašnjen rad ovog algoritma.

Skup podataka kojim treniramo jednodimenzonalne neuralne mreže najčešće čini veliki broj signala. Rijetko se svi ovi signali prosljeđuju mreži istovremeno, a razlozi za to su višestruki. Prvenstveno, obrada cjelokupnog skupa podataka u jednom prolazu zahtijevala bi ogromne količine memorije, što često prevazilazi kapacitete dostupnog hardvera.

Podjelom skupa podataka na manje serije (eng. *batch*) od po nekoliko desetina signala, ne samo da se smanjuje opterećenje na memoriju, već se i omogućava brža konvergencija modela kroz češća ažuriranja težina, što dodatno doprinosi efikasnosti treniranja. Treniranje na manjim serijama može poboljšati sposobnost modela da generalizuje. Kada se model ažurira na osnovu manjih serija, izložen je širem spektru "mini" distribucija podataka, što može pomoći u sprečavanju preprilagođavanja mreže na trening skupu podataka. Takođe, korišćenjem manjih serija unosi se stohastičnost u proces treniranja, što može pomoći modelu da izbjegne zaglavljivanje u lokalnim minimumima funkcije troška. Ova stohastičnost proizilazi iz slučajnog odabira serija iz cjelokupnog skupa podataka.

Proces treniranja neuralne mreže može se definisati kroz sljedeća četiri koraka:

1. Izvlačenje serije (eng. *batch*) uzoraka za treniranje, x, i odgovarajućih ciljeva t. Ovo podrazumijeva odabir podskupa podataka koje će model koristiti u jednom koraku treniranja. U praksi, ovo se obično radi nasumično da bi se izbjegli obrasci koji mogu uticati na proces učenja.

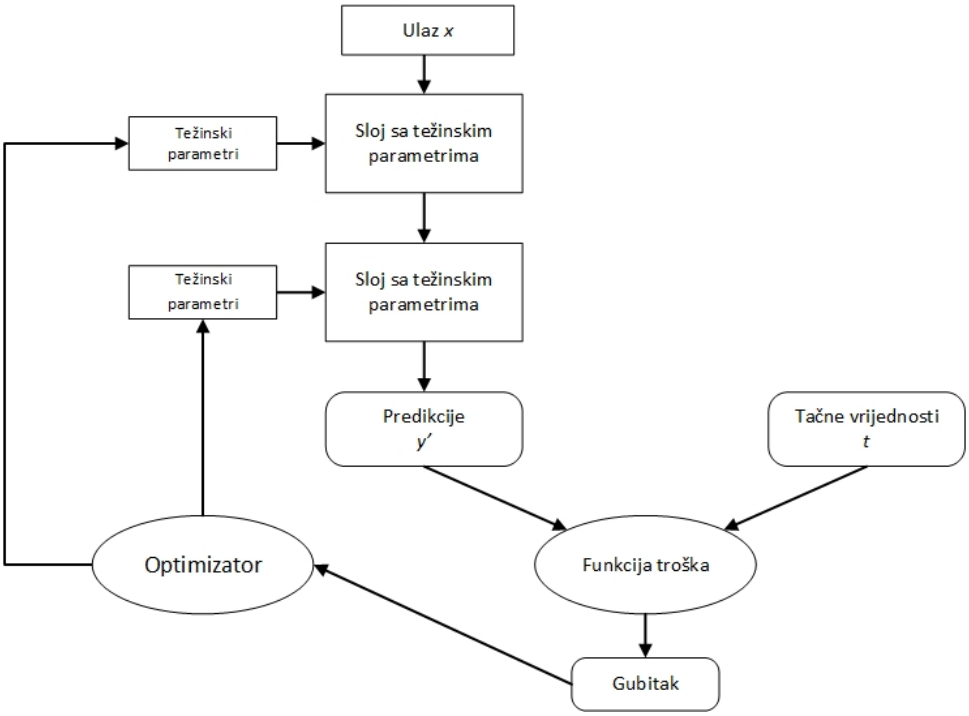
2. Pokretanje modela nad podacima x (korak poznat kao algoritam propagacije unaprijed) kako bismo dobili predikcije modela $y^{'}$. U ovom koraku, model obrađuje izvučenu seriju koristeći trenutne težinske parametre da generiše predikcije. Ovaj proces uključuje sve matematičke operacije definisane u modelu, kao što su konvolucije, aktivacije, itd.

3. Računanje greške modela na seriji, odnosno mjere nesklada između $y^{'}$ i t. Funkcija troška kvantifikuje koliko su predikcije modela različite od stvarnih ciljnih vrijednosti. Ova mjera gubitka zatim se koristi za prilagođavanje težinskih koeficijenata.

4. Ažuriranje svih težinskih koeficijenata mreže na način koji blago smanjuje grešku na ovoj seriji. Ovaj korak, poznat kao algoritam propagacije unazad, uključuje izračunavanje gradijenata funkcije troška u odnosu na sve parametre modela, a zatim ažuriranje tih težina u suprotnom smjeru od gradijenata kako bi se smanjio ukupni gubitak. Ovo se postiže pomoću optimizatora.

Proces treniranja neuralne mreže prikazan je na slici 3.8.

Koncept iteracije i epohe takođe je bitan za razumijevanje procesa treniranja. Iteracija predstavlja jedan prolazak kroz seriju, gdje model ažurira težinske koeficijente na osnovu podataka iz te serije. Na primjer, ako imamo 1000 signala u trening skupu podataka i veličinu serije 100, biće potrebno 10 iteracija da se prođe kroz cjelokupan skup podataka.



Slika 3.8: Dijagram procesa treniranja neuralne mreže sa dva skrivena sloja.

Epoha, sa druge strane, predstavlja prolazak kroz cjelokupan skup podataka, što u ovom slučaju uključuje svih 10 iteracija. Naravno, kada se cijeli skup podataka proslijedi mreži u jednom koraku, pojam iteracija i epoha se poklapaju.

Na početku procesa treniranja, težinama mreže dodjeljuju se nasumične vrijednosti, čime se mreža svodi na niz slučajnih transformacija. Očekivano, rezultati njenog rada daleko su od idealnih, što se ogleda u visokom nivou gubitka.

Međutim, procesiranjem svakog novog primjera, težine se blago koriguju u pravom smjeru, čime se postepeno smanjuje gubitak. Ova iterativna petlja obuke, ponavljana dovoljan broj puta preko hiljada primjera, vodi ka optimizaciji vrijednosti težina koje minimiziraju funkciju gubitka. Mreža sa minimalnim gubitkom jeste ona čiji su izlazi maksimalno usklađeni sa ciljevima, odnosno obučena mreža.

Svaki od ovih koraka se ponavlja kroz definisani broj epoha, odnosno prolazaka kroz kompletan trening set, kako bi se model postepeno poboljšavao i fino prilagođavao za zadati zadatak.

3.7 Optimizacija

Optimizacioni algoritmi zaduženi su za traženje težinskih koeficijenata takvih da funkcija troška ima najmanju moguću vrijednost. Gradijenti spust (eng. *Gradient descent*) izdvaja se kao daleko najpopularniji algoritam optimizacije čija osnovna ideja leži u praćenju gradijenta i postupnom ažuriranju težinskih parametara.

Gradijentni spust je metoda koja minimizuje funkciju troška $J(\theta)$, tako što ažurira parametre θ u suprotnom pravcu od gradijenta funkcije troška $\nabla_\theta J(\theta)$. Ovo se matematički može predstaviti kao:

$$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta)$$

Ovdje η predstavlja stopu učenja koja određuje veličinu koraka kojim vršimo ažuriranje težinskih koeficijenata. Stopa učenja je hiperparametar mreže koji je poželjno mijenjati za vrijeme treninga.

Postoje tri varijante gradijentnog spusta, koje se razlikuju isključivo po količini korišćenih podataka za računanje gradijenta funkcije troška. Mini-serijski gradijentni spust (eng. *mini-batch gradient descent*), koji ažurira parametre na osnovu nasumično odabranih manjih podskupova podataka, daleko je najzastupljeniji, a razlozi za to su detaljno objašnjeni u sekciji 3.6.

Kako je oblast dubokog učenja napredovala, razvijen je veliki broj optimizacionih algoritama, pri čemu su mnogi od njih zapravo poboljšane i prilagođene verzije gradijentnog spusta.

Moment ažuriranje [62] uvedeno je s ciljem ubrzanja konvergencije i smanjenja rizika od zaglavljivanja u lokalnim minimumima. Tradicionalni gradijentni spust ažurira parametre isključivo na osnovu trenutnog gradijenta i stope učenja. Kada su ove vrijednosti male, optimizacija se odvija veoma sporo.

Moment ažuriranje pridaje veliku važnost prethodnim gradijentima, akumulirajući ih u promenljivoj m. Time se ublažava negativan uticaj malih trenutnih gradijenata na ažuriranje težinskih koeficijenata, što značajno pomaže mreži da brže i stabilnije konvergira ka globalnom minimumu. Ažuriranje težinskih koeficijenata vrši se na sledeći način:

1. $m \leftarrow \beta m - \eta \nabla_\theta J(\theta)$
2. $\theta \leftarrow \theta + m$

Ovdje β predstavlja hiperparametar koji smanjuje intenzitet uticaja prethodnog gradijenta i obično se postavlja na vrijednost 0.9.

Adagrad [63] je optimizacioni algoritam koji uvodi adaptivnu stopu učenja. Prethodno smo vidjeli da se svi parametri θ ažuriraju istovremeno, koristeći istu globalnu stopu učenja η, koja je konstantna za sve parametre. Takođe, ova stopa učenja ostaje konstantna za sve vremenske korake, osim ako nije implementirana neka strategija smanjenja stope učenja kako se približavamo minimumu.

Sa druge strane, u Adagrad algoritmu, stopa učenja se prilagođava individualno za svaki parametar θ_i i varira u svakom vremenskom koraku. Kvadrati gradijenata iz prethodnih iteracija akumuliraju se u vektoru s, gdje svaka komponenta s_i akumulira kvadrate gradijenata za parametar θ_i. Ova akumulacija osigurava da se parametrima sa većim gradijentima tokom treniranja postepeno smanjuje stopa učenja i obrnuto. Ažuriranje parametara vrši se na sledeći način:

1. $s \leftarrow s + \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$
2. $\theta \leftarrow \theta - \eta \nabla_\theta J(\theta) \oslash \sqrt{\mathbf{s} + \varepsilon}$

gdje ε predstavlja malu vrijednost, obično 10^{-8}, i uvodi se kako ne bi došlo do dijeljenja sa nulom. Simboli $\otimes$ i $\oslash$ predstavljaju množenje i dijeljenje element po element, respektivno [61].

Međutim, neizbježna akumulacija kvadratnih gradijenata dovodi do drastičnog smanjenja stope učenja, što rezultira time da proces treniranja vremenom potpuno stane. Zbog toga se ovaj algoritam rijetko koristi u treniranju neuralnih mreža.

RMSprop [64] je često korišćeni optimizacioni algoritam koji prevazilazi ograničenja Adagrad algortima. Uvođenjem faktora opadanja (eng. *decay factor*) β, čija je vrijednost manja od 1, a najčešće 0.9, kontroliše se rast akumuliranih kvadrata gradijenata na sljedeći način:

1. $s \leftarrow \beta s + (1-\beta) \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$
2. $\theta \leftarrow \theta - \eta \nabla_\theta J(\theta) \oslash \sqrt{\mathbf{s} + \varepsilon}$

Faktor opadanja β predstavlja hiperparametar mreže.

Adam [65] jedan je od najpopularnijih i najefikasnijih optimizacionih algoritama i definiše se na sledeći način:

1. $m \leftarrow \beta_1 m + (1-\beta_1) \nabla_\theta J(\theta)$
2. $s \leftarrow \beta_2 s + (1-\beta_2) \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$
3. $\hat{m} \leftarrow \frac{m}{1-\beta_1^t}$

4. $\hat{s} \leftarrow \frac{s}{1-\beta_2^t}$
5. $\theta \leftarrow \theta + \eta \hat{m} \oslash \sqrt{\hat{s} + \varepsilon}$

Iz ovih izraza jasno je da Adam kombinuje ključne ideje svojih prethodnika, moment ažuriranja i RMSprop-a, koristeći prednosti oba pristupa za stabilno i efikasno treniranje. Koraci 3 i 4 uvedeni su kako bi se riješio problem pristrasnosti u ranim fazama treniranja, budući da se m i s inicijalizuju na nulu. Vrijednosti hiperparametara β_1 i β_2 najčešće su postavljene na 0.9 i 0.999, respektivno.

3.8 Regularizacija

Tokom treniranja, model se ponekad može previše prilagoditi specifičnim uzorcima i karakteristikama iz trening skupa. Ova pojava, poznata kao pretreniranost ili preprilagođavanje (eng. *overfitting*), dovodi do toga da model gubi sposobnost generalizacije i postiže loše rezultate na novim podacima koji nisu dio trening skupa. Pretreniranost može nastati iz više razloga, poput malog i nedovoljno raznovrsnog skupa podataka, prekomjerno složenog ili dubokog modela, predugog procesa treniranja, prisustva šuma u podacima itd.

Korićenjem tehnika regularizacije sprječava se pretreniranost i povećava moć generalizacije modela. Kao najpopularnije izdvajaju se L_1, L_2 i *Dropout* regularizacija.

L_1 i L_2 uvode kaznu za velike vrijednosti težinskih koeficijenata na način što modifikuju funkciju troška kao:

$$J(\theta) = J(\theta) + \lambda \|\theta\|_l^l$$

gdje $J(\theta)$ predstavlja funkciju troška, λ stepen regularizacije a l parametar koji može uzeti vrijednost 1 ili 2 u zavisnosti od tipa regularizacije.

Dropout regularizacija [66] predstavlja izuzetno efikasnu tehniku koja funkcioniše tako što nasumično isključuje (postavlja na nulu) određeni procenat neurona u sloju tokom svake iteracije treniranja. Procenat isključenih neurona predstavlja hiperparametar koji se obično postavlja između 10% i 50%. Neuroni se isključuju samo tokom procesa obuke, ali ne i tokom validacije i testiranja.

Glava 4

Procjena brzine vozila

Cilj ovog istraživanja je razviti preciznu i robusnu metodu za procjenu brzine vozila na video snimcima, koja se u potpunosti oslanja na duboko učenje. Predložena metoda omogućava procjenu brzine vozila koristeći isključivo monokularne kamere, koje su široko zastupljene u savremenim saobraćajnim sistemima. Ključna karakteristika ove metode je njena sposobnost da funkcioniše bez potrebe za prethodnom kalibracijom kamere ili specifičnim prilagođavanjem različitim okruženjima, čime se eliminišu potencijalne prepreke u njenoj primjeni.

Prilikom razvijanja ove metode, posebna pažnja posvećena je odabiru i razvijanju računski optimalnih modela, poput YOLO detektora i jednostavne CNN arhitekture, kako bi se omogućila njena implementacija i na sistemima sa ograničenim resursima, čime se dodatno povećava njena dostupnost i primjenjivost. Očekuje se da ovaj pristup doprinese razvoju efikasnijih, pouzdanijih i skalabilnijih sistema za nadzor saobraćaja koji mogu raditi u stvarnim uslovima i realnom vremenu, bez potrebe za dodatnim podešavanjima.

U ovom poglavlju detaljno će biti analizirani svi aspekti predložene metode.

4.1 Predložena metoda

Metoda predložena u ovom radu testirana je i evaluirana na VS13 skupu podataka koji je opisan u sekciji 4.2. Svaki zapis u skupu podataka prikazuje jedno vozilo koje prolazi pored kamere konstantnom brzinom.

Metodu želimo u potpunosti bazirati na algoritmima dubokog učenja. Stoga ćemo procjenu brzine, kako je već napomenuto, posmatrati kao regresioni problem i u tu svrhu koristiti jednodimenzione konvolucione neuralne mreže.

Međutim, iz originalnog skupa podataka potrebno je pronaći način kako da se odaberu i ekstrahuju karakteristike koje mogu predstavljati ulaz naše jednodimenzionalne mreže, pritom vodeći računa o očuvanju robusnosti, odnosno nezavisnosti naše metode od stvarnih dimenzija, ostalih objekata na videu i svih drugih spoljašnjih faktora. Stoga, držeći se početnog stanovišta da metoda u potpunosti treba biti bazirana na dubokom učenju, kao pomoćno sredstvo za izvlačenje relevantnih karakteristika koristili smo algoritam detekcije objekata.

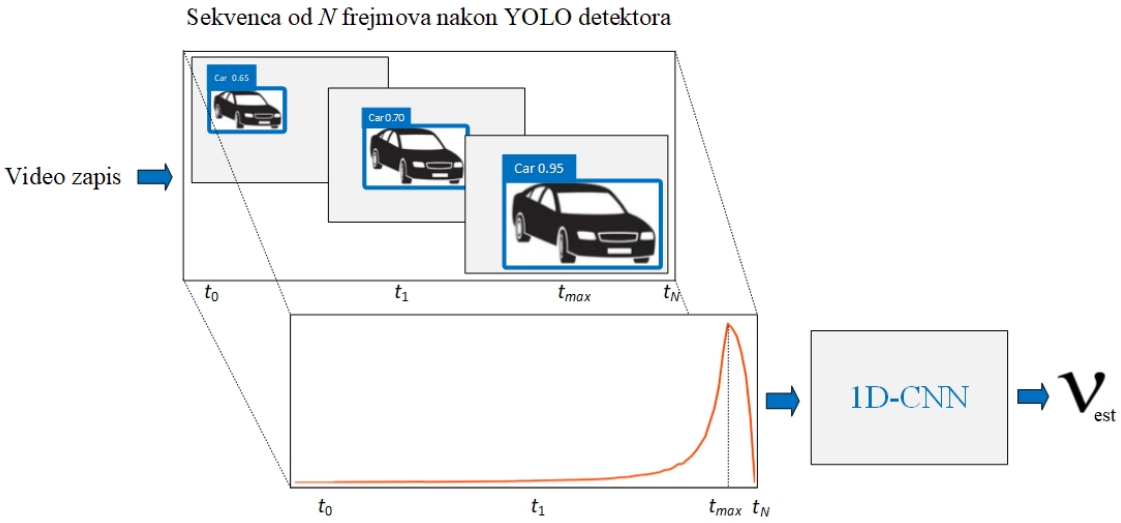
Samim tim, možemo reći da se metoda predložena u ovom radu sastoji iz dva ključna bloka: algoritma za detekciju i praćenje vozila na video snimku i jednodimenzione konvolucione mreže za estimaciju brzine vozila.

Kao algoritam detekcije objekata odabran je YOLO algoritam koji se izdvaja kao jedan od najnaprednijih modela za detekciju i praćenje objekata na video snimcima. YOLO detektor će najprije biti fino podešen i obučen tako da odgovara našem problemu - detekciji vozila. Početni skup podataka će se, video po video, propuštati kroz detektor. Rezultat ovog procesa biće frejmovi ulaznih video podataka koji sadrže granične okvire oko detektovanih vozila kao i tenzori koji uključuju informacije o samim graničnim okvirima. Informacije u tenzorima obuhvataju koordinate centra graničnog okvira, njegovu širinu i visinu, klasu detektovanog vozila, kao i vjerovatnoću tačnosti detekcije.

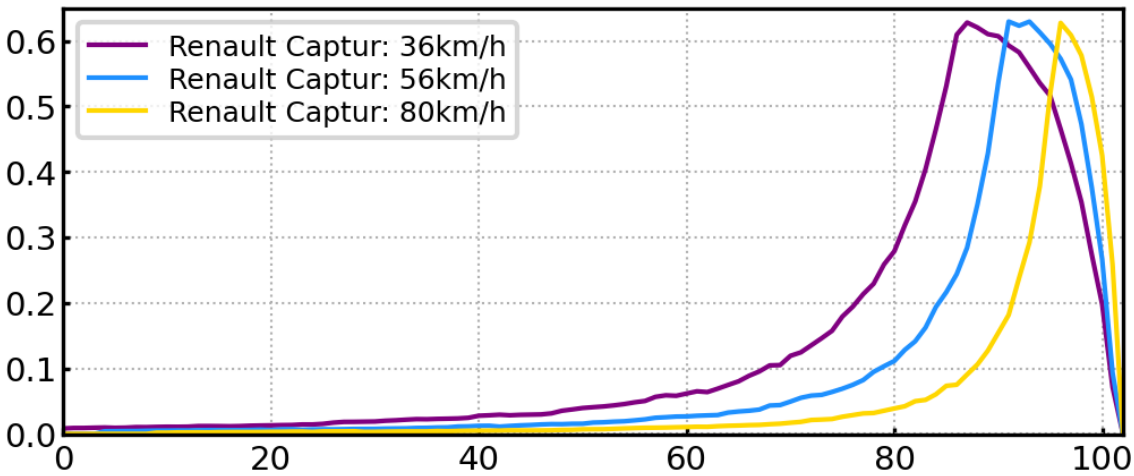
Na osnovu dobijenih informacija, vrlo lako se može izračunati površina pojedinačnih graničnih okvira u kvadratnim pikselima. Površina graničnih okvira se postepeno povećava kroz frejmove video snimka kako vozilo prilazi kameri, dostižući maksimum kada vozilo dođe do desne ivice kadra. Od tog trenutka nadalje, površina se smanjuje, padajući na nulu kada vozilo izađe iz vidnog polja kamere. Ovaj obrazac promjene je uniforman za sva vozila iz našeg skupa podataka, s tim da brzina kretanja vozila diktira oblik i širinu krive promjene površine kao što je predstavljeno na slici 4.2.

Podaci nad kojima je treniran i evaluiran blok za procjenu brzine, odnosno jednodimenziona konvoluciona mreža su upravo krive promjene površina graničnih okvira kroz frejmove svakog video snimka iz skupa podataka. Dužina krive jednaka je broju frejmova između prve i poslednje detekcije vozila na snimku.

Na slici 4.1 prikazan je princip rada predložene metode.



Slika 4.1: Princip rada predložene metode za procjenu brzine. Tri frejma odgovaraju vremenskim trenucima: t_0 (početak detekcije), t_1 (sredina posmatranog vremenskog intervala) i $t_{\max}$ (pozicija maksimalne vrijednosti krive, kada vozilo počinje da izlazi iz vidnog polja kamere).



Slika 4.2: Izgled normalizovanih krivih promjene površine graničnih okvira za tri različite brzine vozila. Izvršena normalizacija opisana je u sekciji 4.3.

4.2 Skup podataka

Uspjeh algoritama dubokog učenja zavisi od brojnih faktora, a jedan od ključnih je kvalitet skupa podataka na kojima se algoritmi treniraju. Algoritmi koji su obučeni na kvalitetnom skupu podataka pokazaće bolje performanse.

Dobar skup podataka nije samo pitanje kvantiteta, već i kvaliteta. Pored toga što je poželjno da bude obiman, skup podataka mora biti sveobuhvatan, uravnotežen, nepristrasan i pažljivo pripremljen, pokrivajući sve relevantne varijacije i scenarije koji mogu nastati u stvarnom svijetu.

Dodatno, kako bi se omogućilo objektivno poređenje različitih algoritama, neophodno je osigurati jednake uslove u kojima se algoritmi evaluiraju. Ovo uključuje upotrebu standardizovanih i referentnih skupova podataka, koji omogućavaju pouzdano mjerenje i upoređivanje performansi, što olakšava identifikaciju prednosti i nedostataka svakog modela.

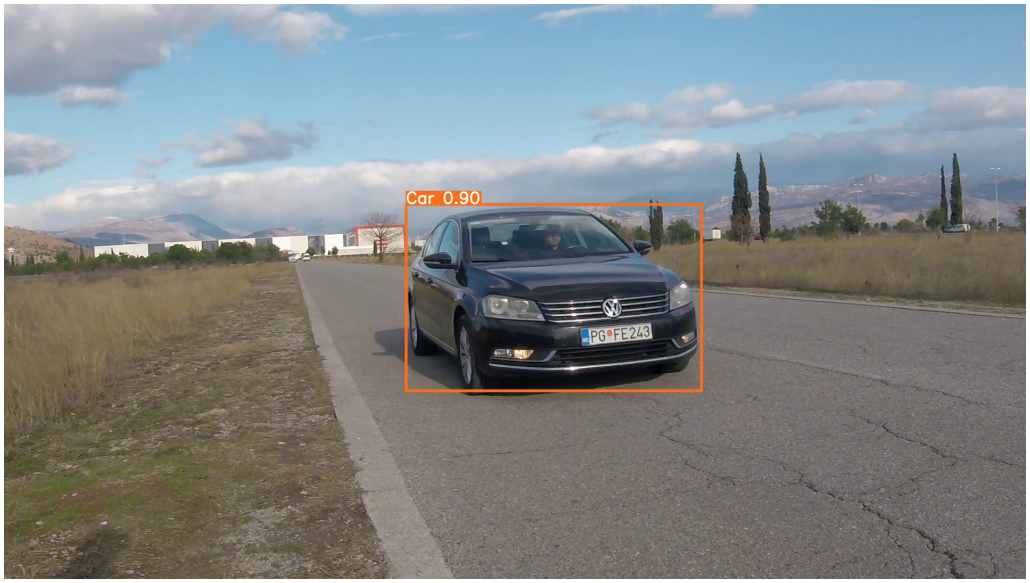
Iako performanse algoritama dubokog učenja zavise od mnogih faktora, čak i najnapredniji modeli su samo onoliko dobri koliko i podaci na kojima su obučeni.

Uprkos tome što je procjena brzine vozila izuzetno aktuelna istraživačka oblast, broj dostupnih skupova podataka razvijenih za ovu svrhu je veoma mali [4]. Jedan od najznačajnijih skupova podataka je AI City Challenge[1] koji predstavlja dio godišnjeg takmičenja koji je pokrenula kompanija NVIDIA u cilju podsticanja inovacija u razvoju algoritama za analizu video zapisa u saobraćajnom okruženju. Ovaj skup podataka služi kao osnova za istraživače i inženjere koji rade na rješenjima za inteligentno upravljanje saobraćajem, detekciju saobraćajnih incidenata, prepoznavanje vozila, kao i procjenu brzine vozila. Svake godine se struktura skupa podataka mijenja.

BrnoCompSpeed [67] je skup podataka koji sadrži 18 video zapisa u Full HD rezoluciji i 50 fps, svaki u trajanju od približno jednog sata. Tačna brzina vozila u ovim video zapisima, kojih ima nešto više od 20.000, mjerena je uz pomoć LiDAR tehnologije.

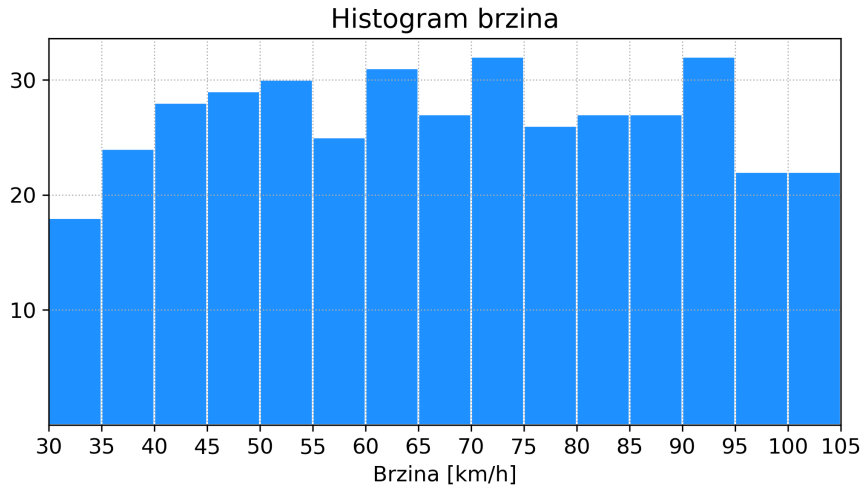
Skup podataka na kojem trenirana i evaluiriana metoda predložena u ovom radu je VS13 skup podataka [68]. VS13 skup podataka obuhvata 400 označenih audio-vizuelnih snimaka, zabilježenih jednom kamerom pri konstantnim brzinama vozila. Video zapisi su trajanja 10 sekundi, u Full HD rezoluciji i sa frekvencijom od 30 frejmova po sekundi. Skup podataka snimljen je pomoću GoPro Hero5 kamere koja je postavljena na udaljenosti od 0.5m od puta na visini od 1.2m.

[1] www.aicitychallenge.org



Slika 4.3: Izdvojeni frejm video zapisa iz VS13 skupa podataka nakon što je propušten kroz YOLO algoritam. Pored detekcije, algoritam vrši i klasifikaciju detektovanih objekata.

Svaki video zapis sadrži prolazak samo jednog vozila pored kamere kao što je prikazano na slici 4.3. Skup podataka sadrži ukupno 13 vozila i raznovrstan je u smislu proizvođača vozila, godine proizvodnje, tipa motora, snage i transmisije. Na svakom vozilu je aktiviran tempomat koji održava konstantnu brzinu tokom svakog video zapisa. Snimci su zabilježeni u urbanom okruženju, na putu bez drugih vozila u pozadini. Brzine vozila u ovom skupu podataka variraju od 30 km/h do 105 km/h.



Slika 4.4: Histogram brzina u VS13 skupu podataka (15 intervala jednake širine).

Tabela 4.1: Vozila i njima odgovarajuće brzine u VS13 skupu podataka.

Vozilo	Brzine (km/h)
Citroen C4 Picasso 1.6 HDI (CitroenC4Picasso)	35, 38, 41, 44, 48, 51, 54, 57, 59, 63, 65, 68, 72, 74, 78, 80, 83, 85, 87, 92, 94, 96, 101
Kia Sportage 1.6 GDI (KiaSportage)	31, 33, 35, 38, 41, 44, 46, 48, 51, 53, 55, 58, 61, 63, 65, 68, 69, 72, 74, 77, 78, 80, 83, 85, 86, 89, 91, 93, 96, 98, 100, 103, 105
Mazda 3 Skyactive (Mazda3)	30, 33, 35, 38, 40, 43, 45, 47, 50, 52, 55, 57, 60, 62, 64, 67, 70, 72, 75, 79, 81, 84, 86, 88, 90, 92, 94, 96, 99, 101, 103, 105
Mercedes AMG 550 (MercedesAMG550)	30, 33, 35, 38, 40, 42, 45, 47, 50, 52, 55, 58, 60, 62, 65, 67, 70, 73, 75, 78, 80, 82, 85, 87, 90, 93, 95, 98, 100, 105
Mercedes GLA 200D (MercedesGLA)	30, 33, 36, 39, 41, 42, 45, 47, 48, 49, 52, 54, 55, 59, 61, 63, 65, 68, 70, 72, 75, 78, 81, 83, 85, 88, 90, 92, 93, 96, 100, 101, 103, 104
Nissan Qashqai 1.5 DCI (NissanQashqai)	35, 38, 40, 42, 45, 48, 50, 53, 55, 58, 60, 61, 64, 65, 68, 70, 73, 75, 78, 80, 82, 85, 88, 90, 93, 94, 96, 98, 102
Opel Insignia 2.0 CDTI (OpelInsignia)	31, 35, 38, 41, 44, 47, 50, 53, 55, 58, 61, 64, 66, 68, 70, 72, 73, 76, 78, 80, 83, 86, 89, 91, 94, 97, 100
Peugeot 208 1.4 HDI (Peugeot208)	30, 32, 34, 37, 40, 43, 45, 47, 50, 51, 54, 57, 60, 62, 64, 67, 68, 71, 73, 76, 77, 79, 82, 84, 87, 90, 92, 95, 96
Peugeot 3008 1.6 HDI (Peugeot3008)	40, 43, 45, 47, 50, 52, 54, 55, 56, 58, 60, 61, 63, 65, 67, 68, 70, 72, 74, 75, 78, 80, 83, 85, 87, 89, 90, 92, 95, 97, 100
Peugeot 307 2.0 HDI (Peugeot307)	30, 33, 35, 38, 40, 43, 45, 47, 48, 50, 53, 56, 59, 60, 63, 66, 69, 72, 73, 76, 79, 82, 85, 88, 91, 94, 97, 101, 103
Renault Captur 1.5 DCI (RenaultCaptur)	30, 33, 36, 38, 40, 41, 44, 46, 47, 48, 50, 52, 56, 58, 60, 63, 66, 68, 70, 72, 76, 78, 80, 83, 86, 88, 90, 92, 94, 97, 98, 100, 102
Renault Scenic 1.9 DCI (RenaultScenic)	30, 35, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 57, 60, 62, 64, 66, 68, 70, 71, 72, 74, 75, 77, 80, 82, 84, 86, 87, 90, 91, 94, 95, 98, 101
VW Passat B7 1.6 TDI (VWPassat)	30, 35, 39, 40, 42, 45, 47, 49, 50, 52, 54, 55, 57, 60, 61, 64, 65, 67, 70, 71, 72, 73, 75, 78, 80, 81, 82, 85, 88, 90, 91, 94, 96, 98, 100

Detaljan pregled svih vozila u skupu podataka i brzina na kojima su snimljeni njihovi prolasci dati su u tabeli 4.1, dok je histogram brzina prikazan na slici 4.4.

Međutim, za potrebe dubokog učenja, VS13 se smatra relativno malim skupom podataka, što predstavlja izazov u pogledu treniranja modela. Treniranje na malim skupovima često dovodi do problema kao što su nedovoljna istreniranost (podtreniranost) ili pretreniranost modela, što može negativno uticati na njegovu sposobnost generalizacije na novim podacima.

S obzirom na ograničenu veličinu skupa podataka, veoma je važno obezbijediti kvalitet podataka koji se koriste za treniranje modela. Pojava izuzetaka (eng. *outliers*) ili drugih nepravilnosti mogla bi imati ozbiljne posljedice po preciznost i stabilnost modela. Zbog toga se, u ovom istraživanju, primjenjuju tehnike pretprocesiranja podataka koje između ostalog uključuju uklanjanje izuzetaka i normalizaciju. Ove metode su primijenjene nad podacima dobijenim nakon YOLO detektora, o čemu će više biti riječi u narednoj sekciji.

4.3 Detekcija vozila

Da bi metoda procjene brzine bila primjenjiva, jedan od glavnih kriterijuma je brzina procjene. To nameće potrebu za odabirom efikasnog detektora objekata, kako bi se izbjeglo stvaranje uskog grla u sistemu.

U momentu kada je rađeno ovo istraživanje najnovija verzija YOLO algoritma bila je YOLOv7. Ipak, na osnovu rezultata iz rada [69] i specifičnih zahtjeva našeg problema, kao algoritam detekcije objekata odabrana je peta generacija YOLO algortima - YOLOv5. Ovaj algoritam poznat je po svojoj brzoj sposobnosti detekcije, visokoj tačnosti, prilagodljivosti različitim skupovima podataka i jednostavnom procesu obuke. YOLOv5 je projekat otvorenog koda koji održava kompanija Ultralytics[2], uz podršku brojnih saradnika koji kontinuirano doprinose njegovom unaprjeđenju. Vrlo je jednostavan za korišćenje, obuku i implementaciju. Ultralytics nudi i specijalizovane verzije YOLOv5 za mobilne uređaje, uključujući iOS i Android, što omogućava upotrebu ovog algoritma na različitim platformama [70].

YOLOv5 nudi nekoliko arhitektura modela koji se razlikuju po broju konvolucionih slojeva, što naravno direktno utiče na broj parametara neuralne mreže. Za naše potrebe, odabrali smo YOLOv5m (srednji model) kao optimalan balans između tačnosti i računarskih zahtjeva.

YOLOv5 je inicijalno treniran na MS-COCO skupu podataka. Zbog toga će biti dodatno prilagođen na specijalizovanom setu podataka za detekciju vozila, kako bi se model optimizovao za potrebe našeg zadatka. Set podataka korišćen za ovu dodatnu obuku detektora je [71].

Kao što je već pomenuto, izlazi YOLO detektora koriste se kao podaci nad kojima se trenira predložena jednodimenzionalna konvoluciona mreža. Ipak, treba imati na umu da detektor može ponekad napraviti grešku, što može dovesti do izostanka detekcije na određenom frejmu. Ovakve situacije su naročito uobičajene na početku snimka, kada je vozilo još uvijek na većoj udaljenosti od kamere, zbog čega je manja vjerovatnoća da će biti tačno detektovano što će dovesti do problema nedostajućih odbiraka na krivoj promjene površina graničnih okvira, što dalje može ozbiljno ugroziti sposobnost učenja 1D-CNN modela za procjenu brzine iz proslijeđenih karakteristika.

Zato će, kako bi se ovo pravovremeno spriječilo, na mjestima gdje je izostala detekcija biti upisana vrijednost dobijena na osnovu prethodne i sljedeće vrijednosti površine ko-

[2]www.ultralytics.com

rišćenjem linearne interpolacije. Linearna interpolacija predstavlja logičan izbor u ovom slučaju jer se ne očekuje značajna promjena u površini graničnih okvira između dva susjedna frejma, čime se obezbeđuje glatka tranzicija. Ovaj pristup održava kontinuitet podataka bez uvođenja nepravilnosti, odnosno naglih padova u vrijednostima, što je ključno za tačnost i stabilnost konačnih rezultata. Pojava izuzetaka u podacima kojih je i ovako malo mogla bi ozbiljno ugroziti performanse modela.

Prilikom pretprocesiranja podataka izvršena je i normalizacija, gdje su sve vrijednosti preskalirane tako da budu u rasponu od 0 do 1, dijeljenjem sa najvećom vrijednošću površine graničnog okvira iz cjelokupnog skupa podataka, što se i može vidjeti na slici 4.2. Ovaj korak je posebno važan jer su površine graničnih okvira računate u kvadratnim pikselima, što rezultira veoma velikim vrijednostima. Normalizacija poboljšava efikasnost učenja modela sprečavajući dominaciju pojedinih karakteristika uslijed njihovih velikih vrijednosti. Dodatno, normalizacija doprinosi bržem konvergiranju algoritama optimizacije, što na kraju rezultira boljim performansama modela.

4.4 Predložena arhitektura

Postoji više programskih okvira otvorenog koda koji značajno olakšavaju razvoj neuralnih mreža. U ovom istraživanju korišćen je TensorFlow [3]. Python je odabran kao programski jezik zbog niza prednosti koje ga čine pogodnim za rad sa neuralnim mrežama. Kao najkorišćeniji jezik u zajednici koja se bavi dubokim učenjem, Python ne samo da omogućava jednostavnu implementaciju i razvoj složenih modela, već zahvaljujući svojoj velikoj i aktivnoj zajednici pruža obilje podrške, dokumentacije i gotovih biblioteka, što značajno ubrzava i olakšava rad. Dodatno, s obzirom na to da je YOLO algoritam takođe realizovan u Python-u, korišćenje istog programskog jezika obezbjeđuje konzistentnost i pojednostavljuje integraciju različitih djelova predložene metode.

Arhitektura konvolucione neuralne mreže određena je eksperimentalnim putem. Sama arhitektura mreže je vrlo jednostavna. Čini je ukupno šest slojeva od kojih su četiri sa težinskim parametrima. Prva dva sloja mreže su identični 1D konvolucioni slojevi, svaki sa po 64 filtera. Veličina kernela, odnosno prozora kojim se vrši konvolucija je 10. Kako bismo uveli nelinearnost u model, nad izlazima oba sloja primijenjena je ReLU aktivaciona funkcija.

[3] www.tensorflow.org

Tabela 4.3: Pregled predložene arhitekture 1D konvolucione mreže.

Sloj	Oblik ulaza	Oblik izlaza	Broj parametara
1D Konvolucioni sloj	(None, 107, 1)	(None, 98, 64)	704
1D Konvolucioni sloj	(None, 98, 64)	(None, 89, 64)	41,024
Dropout	(None, 89, 64)	(None, 89, 64)	0
Sloj ravnanja	(None, 89, 64)	(None, 5696)	0
Potpuno povezani sloj	(None, 5696)	(None, 64)	364,608
Potpuno povezani sloj	(None, 64)	(None, 1)	65
Ukupan broj parametara: **406,401**			

Kako bismo spriječili pretreniranost modela, kao treći sloj mreže uveden je *Dropout* sloj sa stopom isključivanja od 40%, čime se poboljšava sposobnost generalizacije modela. Ovaj sloj praćen je slojem ravnjanja koji povezuje proslijeđene vektore u jedinstveni vektor koji se dalje prosljeđuje potpuno povezanim slojevima.

Peti sloj je potpuno povezani sloj sa 64 neurona i ReLU aktivacionom funkcijom. Na kraju, model se završava još jednim potpuno povezanim slojev sa jednim neuronom koji takođe koristi ReLu aktivacionu funkciju dajući predikciju na izlazu.

Za inicijalizaciju težinskih parametara koristi se Xavier odnosno Glorot inicijalizacija [72], koja osigurava da početne težine budu skalirane tako da se izbjegne problem eksplozije ili nestajanja gradijenata.

Arhitektura mreže razvijana je prateći određene polazne pretpostavke. S obzirom na to da na raspolaganju imamo relativno mali skup podataka, jednostavna arhitektura kao što je ova može biti izuzetno pogodna. U ovakvim slučajevima, složenije mreže sa mnogo slojeva i parametara često dovode do pretreniranosti, jer model "uči" previše detalja iz ograničenog broja uzoraka, što negativno utiče na njegovu sposobnost generalizacije. Ovo se pokazalo kao tačno. Svako naredno dodavanje slojeva se pokazalo veoma kontraproduktivnim, jer nije poboljšavalo performanse modela, već je samo povećavalo potrošnju resursa bez značajnog doprinosa u rezultatima.

Ova mreža, sa četiri sloja sa težinskim parametrima, predstavlja uravnotežen pristup koji pruža dovoljnu složenost, odnosno omogućava modelu da nauči relevantne karakteristike iz podataka, dok istovremeno minimizuje rizik od pretreniranosti. U ovom kontekstu, regularizacija u vidu *Dropout* sloja je posebno korisna, jer dodatno pomaže u smanjenju pretreniranosti, što je ključno kada se radi sa ovako malim setovima podataka.

Kratak pregled čitave arhitekture mreže može se vidjeti u tabeli 4.3. Ukupan broj parametara mreže je 406.401 što ovu arhitekturu čini relativno jednostavnom.

Glava 5

Postavka eksperimenta i rezultati

Iz VS13 skupa podataka, nakon što je propušten kroz YOLO detektor, izdvojeno je 400 karakteristika odnosno krivih promjene površina koje predstavljaju ulaz našeg konvolucionog modela za procjenu brzine.

Uslijed prisustva potpuno povezanih slojeva u našoj mreži, dužina, odnosno broj odbiraka ovih karakteristika mora biti ujednačen. Kako bi se ovo obezbijedilo potrebno je dodati još jedan segment pretprocesiranja podataka prije nego što se proslijede modelu.

U našem slučaju dužina karakteristika određena je na 107 odbiraka i ona predstavlja minimalni broj frejmova između prve i poslednje detekcije među svim video zapisima iz skupa podataka. Jasno je odmah da smo primijenili tehniku skraćivanja signala kako bismo sve karakteristike prilagodili dužini najkraćeg. Odlučeno je da se skraćivanje karakteristika izvrši sa početka. Ovakav pristup zaista ima smisla posebno u kontekstu rada sa detektorima objekata. U početku snimka kada je vozilo relativno daleko od kamere, posebno kod modela kao što je YOLO, može doći do povećanog broja grešaka u detekciji, o čemu je bilo riječi u prethodnoj sekciji. Skraćivanjem signala sa početka, eliminišu se potencijalno nepouzdani djelovi signala i zadržava se dio u kojem je model već stabilizovao svoje detekcije i pruža precizne rezultate.

Dakle, ovakvim pristupom, iako se odbacuje dio signala, ne gube se relevantne informacije već, naprotiv, poboljšava se kvalitet i pouzdanost podataka na kojima se dalje bazira treniranje modela. Drugim tehnikama kao što je umetanje nula ili drugih vrijednosti kako bi se izjednačila dužina signala, povećava se rizik od unosa šuma u podatke, što može negativno uticati na performanse modela.

Postavka eksperimenta u ovom istraživanju je sljedeća. Jedno vozilo (tj. izdvojene karakteristike koje odgovaraju tom vozilu) se odvaja sa strane i isključivo koristi za testi-

ranje, dok se preostalih dvanaest vozila koristi za treniranje i validaciju modela. Koristeći izdvojeno vozilo kao testno, mreža se trenira i testira 10 puta. Ova procedura se ponavlja za svako vozilo iz skupa podataka, tako da se na kraju svako vozilo koristi kao testno u jednom trenutku.

Ponavljanjem procesa treniranja i testiranja mreže 10 puta za svako vozilo, osiguravamo da model prođe kroz različite slučajne inicijalizacije težinskih parametara i varijacije u podjelama podataka na trening i validaciju. Ovaj pristup smanjuje uticaj ovih slučajnih faktora i sprječava donošenje zaključaka zasnovanih na potencijalno pristrasnim rezultatima iz samo jednog pokušaja. Kombinovanjem rezultata iz svih 10 ponavljanja dobija se prosječna ocjena performansi modela na određenom vozilu, koja je znatno stabilnija i reprezentativnija.

5.1 Treniranje

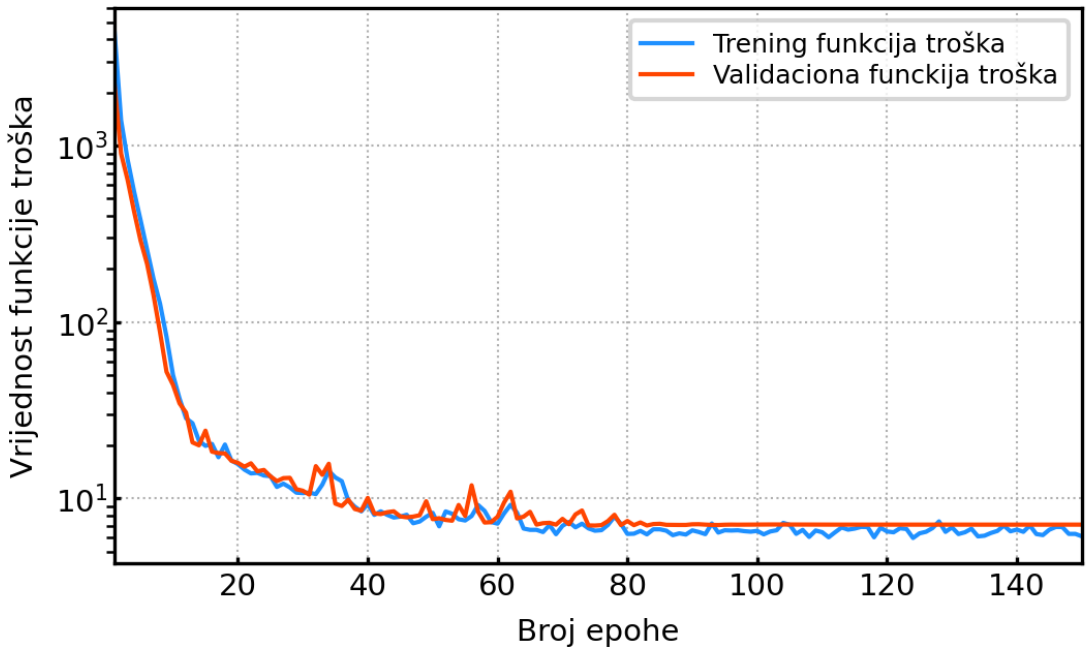
Samo treniranje mreže postavljeno je na sljedeći način. Podaci proslijeđeni mreži se najprije dijele na trening i validacioni skup. Ova podjela se vrši nakon nasumičnog miješanja podataka, što je dobra i česta praksa kod obučavanja neuralnih mreža, pri čemu 80% podataka ide u trening skup, a preostalih 20% u validacioni skup.

Model se trenira korišćenjem mini-serijskog gradijentnog spusta, pri čemu je veličina serije postavljena na 32. Kao optimizator odabran je *Adam* sa početnom stopom učenja od 10^{-3}.

Funkcija troška koja se koristi je srednja kvadratna greška (MSE), što je i najčešći izbor za regresione zadatke. Performanse modela prate se kroz metriku srednje apsolutne greške (MAE), a sam proces treniranja odvija se u 150 epoha.

Tokom treniranja mreže korišćena je i tehnika prilagođavanja stope učenja. Konkretno, praćena je vrijednost funkcije troška na validacionom skupu. Ukoliko model prestane da napreduje kroz 5 uzastopnih epoha, stopa učenja se smanjuje za faktor 0.3, odnosno na 30% od trenutne vrijednosti. Minimalna stopa učenja postavljena je na 10^{-7}. Ovaj pristup omogućava modelu da stabilnije konvergira eliminišući oscilacije što je posebno značajno u kasnijim fazama treniranja.

Vrijednosti funkcija troška tokom treniranja i validacije prikazane su na slici 5.1 kao funkcija broja epoha. Da bi se grafik mogao bolje čitati vrijednosti funkcija troška predstavljene su na logaritamskoj skali.



Slika 5.1: Funkcije troška jednog trening ciklusa za nasumično odabrano vozilo.

Posmatrajući grafik, jasno se uočava sposobnost modela da brzo konvergira, smanjujući vrijednost funkcije troška još u ranoj fazi treniranja. Tokom prvih 20 epoha dolazi do značajnog pada u vrijednostima funkcija troška, što ukazuje na efikasan proces učenja modela. Ovaj period karakteriše veoma agresivno smanjenje greške kako na trening, tako i na validacionom skupu podataka.

Nakon toga, funkcije troška se stabilizuju. Ova stabilizacija ukazuje na to da je model postigao ravnotežu između tačnosti na trening podacima i sposobnosti generalizacije na nepoznatim podacima, odnosno validacionom skupu. Validaciona funkcija troška ostaje bliska vrijednostima trening funkcije troška tokom cjelokupnog procesa treniranja, što je jasan pokazatelj da model generalizuje dobro na validacionom skupu podataka, ali i da nije došlo do pretreniranja mreže.

Posmatrajući funkcije troška, čini se da je model na početku treniranja pokazao bolje rezultate na validacionom skupu nego na trening skupu na kojem je obučavan i na osnovu kojeg su ažurirani težinski parametri. Objašnjenje ovog, ne tako rijetkog fenomena može se naći u sljedećim činjenicama. Greška na validacionom skupu podataka računa se samo na kraju svake epohe, dok se greška na trening skupu računa kontinuirano tokom svake epohe. Ova razlika u načinu izračunavanja može stvoriti privid da su rezultati na validacionom skupu bolji u ranim epohama, iako se greška na trening skupu ažurira kako

model obrađuje svaku seriju podataka. Takođe, regularizacione tehnike kao što je *Dropout* aktivne su samo tokom treniranja, ali ne i tokom validacije i testiranja. Tehnike regularizacije namjerno smanjuju tačnost na trening skupu u nadi da će poboljšati tačnost na validacionom skupu i sposobnost generalizacije modela [61].

5.2 Mjera kvaliteta

U problemima kao što je procjena brzine vozila, poželjno je, a ujedno i logično, da metrika kojom se ocjenjuje uspješnost sistema bude usklađena sa ciljnim veličinama, te da se izražava u istim jedinicama. Time se osigurava jasna i nedvosmislena interpretacija rezultata, dok se istovremeno omogućava intuitivnije sagledavanje preciznosti sistema u odnosu na stvarne vrijednosti, što je ključno za dalje unapređivanje njegovih performansi.

Zato se u ovom istraživanju kao mjera kvaliteta koristi korijen srednje kvadratne greške (eng. *Root Mean Squared Error* RMSE) definisana kao:

$$\mathrm{RMSE}_{\mathrm{glob}} = \sqrt{\frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (v_{i,j}^{est} - v_{j}^{true})^2} \tag{5.1}$$

gdje:

- $v_{i,j}^{\mathrm{est}}$ predstavlja estimiranu vrijednost za j-ti testni uzorak tokom i-tog mjerenja.
- v_{j}^{true} predstavlja stvarnu brzinu odnosno j-ti testni uzorak vozila za koju se RMSE računa.
- N je broj ponavljanja eksperimenta.
- M je broj testnih uzoraka.

Ovaj izraz predstavlja globalnu RMSE, koja se računa tako što se sve greške iz svih pojedinačnih ponavljanja eksperimenta kombinuju u jednu mjeru, čime se dobija jedinstvena vrijednost RMSE za svako vozilo. Globalna RMSE daje sveobuhvatan pregled ukupne greške modela na nivou pojedinačnog vozila.

S druge strane, alternativni pristup je prosječna RMSE, koja se računa kao prosjek RMSE vrijednosti dobijenih za svako pojedinačno mjerenje (ponavljanje eksperimenta). Ovaj pristup omogućava detaljniji uvid u performanse modela kroz različita ponavljanja i definisan je sljedećom formulom:

$$\text{RMSE}_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{M}\sum_{j=1}^{M}(v_{i,j}^{est} - v_{j}^{true})^2} \tag{5.2}$$

Dok globalna RMSE pruža jednu sveobuhvatnu vrijednost koja sažima performanse modela kroz sva ponavljanja i sve uzorke, prosječna RMSE nudi bolji uvid u varijacije performansi modela kroz različita ponavljanja.

5.3 Rezultati

Rezultati sprovedenih eksperimenata za svako vozilo prikazani su u tabelama 5.1 i 5.2. Kao što je prethodno navedeno, $\text{RMSE}_{\text{glob}}$ i RMSE_{avg} za svako vozilo računati su na osnovu 10 ponavljanja eksperimenta.

Rezultati globalne RMSE pokazuju da je prosječna greška modela u procjeni brzine kroz sva vozila 2.70 km/h, što ukazuje na zadovoljavajuću tačnost modela u cjelini. Međutim, analiza pojedinačnih vrijednosti RMSE otkriva određene razlike u performansama modela u zavisnosti od vozila.

Vozila poput Kia Sportage i Peugeot 3008 pokazuju relativno visoke vrijednosti RMSE (5.58 km/h i 4.01 km/h, respektivno), što sugeriše da model ima poteškoća u tačnoj procjeni brzine za ove modele vozila. Sa druge strane, vozila poput Mercedes GLA i Renault Scenic imaju niske vrijednosti RMSE (1.38 km/h i 1.82 km/h), što ukazuje na visoku preciznost modela za ta vozila.

Iako postoje razlike u performansama, teško je precizno utvrditi njihov uzrok. Valjan argument bi bio da bi primjena novije i naprednije verzije YOLO algoritma mogla značajno poboljšati tačnost detekcije, a time i performanse modela. Ipak, prethodno primijenjene tehnike pretprocesiranja u velikoj mjeri su smanjile mogućnost grešaka u detekciji, čime su rane detekcije gotovo u potpunosti eliminisane. Dodatno, YOLOv5, koji je korišćen u ovom istraživanju, važi za izuzetno precizan i široko prihvaćen algoritam. Takođe, sva vozila su snimana u gotovo identičnim uslovima, što znači da nema očiglednog razloga zbog kojeg bi algoritam bio precizan za jedno vozilo, a neprecizan za drugo. Vizuelnom inspekcijom nisu utvrđenje nepravilnosti i značajna odstupanja krivih promjene površina spornih vozila u odnosu na ostala.

Jedno je sigurno – skup podataka je mali za zadatke dubokog učenja. Ipak, primjenom različitih, prethodno opisanih tehnika, uspjeli smo se izboriti s ovim ograničenjem i u velikoj mjeri ublažiti njegov uticaj na performanse modela.

Ono što je zaista ohrabrujuće je da su vrijednosti globalne RMSE i prosječne RMSE gotovo identične, s razlikama vidljivim tek na drugoj ili trećoj decimali. Ovo ukazuje na to da nema značajnih varijacija u rezultatima između ponovljenih eksperimenata, što dodatno potvrđuju niske vrijednosti standardne devijacije u većini slučajeva. Greške modela su konzistentne kroz sva ponavljanja, što nam omogućava da zaključimo da je model stabilan i pouzdan.

Tabela 5.1: $\text{RMSE}_{\text{glob}}$ procijenjenih brzina.

Vozilo	$\text{RMSE}_{\text{glob}}$ [km/h]
Citroen C4 Picasso	2.26
Kia Sportage	5.58
Mazda 3 Skyactive	2.63
Mercedes AMG 550	2.96
Mercedes GLA	1.38
Nissan Qashqai	2.19
Opel Insignia	1.98
Peugeot 208	3.13
Peugeot 307	3.22
Peugeot 3008	4.01
Renault Captur	2.48
Renault Scenic	1.82
VW Passat B7	1.48
Srednja vrijednost	2.70

Rezultati su predstavljeni i u grafičkom obliku na slici 5.2, gdje plava linija predstavlja stvarne, izmjerene vrijednosti brzine vozila, dok crvena oblast predstavlja 95% interval pouzdanosti, koji predstavlja raspon vrijednosti unutar kojeg se, s vjerovatnoćom od 95%, nalazi stvarna vrijednost. Širina crvene oblasti direktno ukazuje na stepen sigurnosti modela – uži intervali signaliziraju veću sigurnost i konzistentnost u predikcijama, dok širi intervali ukazuju na veću varijabilnost i nesigurnost modela. U našem slučaju, intervali su uglavnom uski, što je jasan pokazatelj da je model stabilan i pruža pouzdane predikcije, što takođe potvrđuju rezultati iz tabele 5.2.

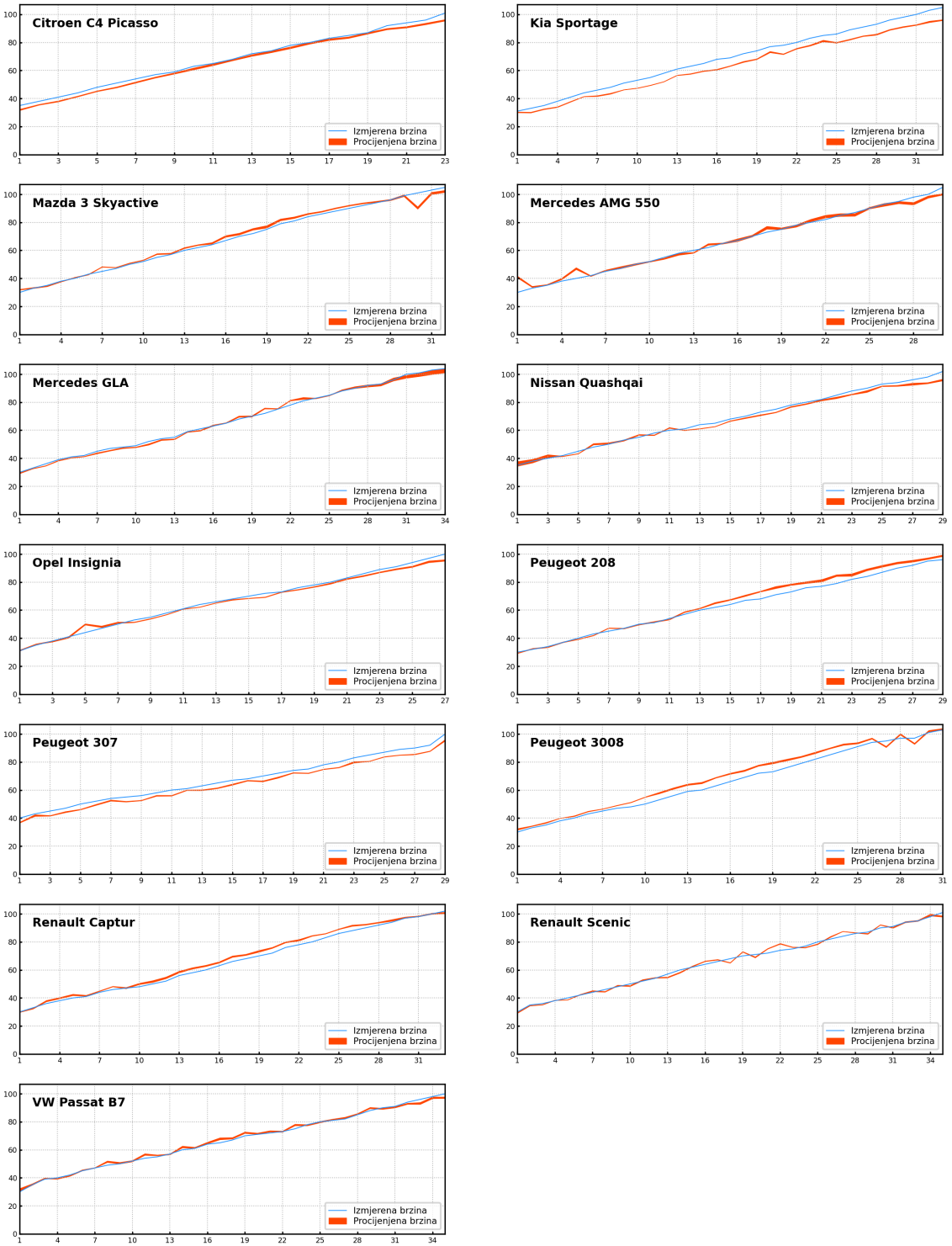
Posmatrajući grafike, može se zaključiti da model veoma dobro prati stvarne brzine. U većini slučajeva, procijenjena brzina (crvena linija) se skoro savršeno poklapa sa stvarnom

Tabela 5.2: Prosječna RMSE i standardna devijacija (STD) procjene brzine za sva vozila kroz 10 ponavljanja.

Vozilo	$RMSE_{avg}$ [km/h]	STD [km/h]
Citroen C4 Picasso	2.25	0.21
Kia Sportage	5.58	0.05
Mazda 3 Skyactive	2.63	0.06
Mercedes AMG 550	2.95	0.09
Mercedes GLA	1.37	0.10
Nissan Qashqai	2.18	0.16
Opel Insignia	1.98	0.11
Peugeot 208	3.12	0.27
Peugeot 307	3.22	0.13
Peugeot 3008	4.00	0.14
Renault Captur	2.47	0.14
Renault Scenic	1.82	0.04
VW Passat B7	1.47	0.19

brzinom (plava linija), što potvrđuje visoku preciznost modela.

Kia Sportage pokazuje najveća odstupanja od stvarnih brzina, pri čemu model konstantno podcjenjuje stvarne brzine, iako interval pouzdanosti ostaje relativno uzak, što ukazuje na umjerenu sigurnost modela u predikcijama. S druge strane, za vozila poput Mercedes GLA i Opel Insignia model veoma precizno prati stvarne brzine, sa minimalnim odstupanjima i također uskim intervalima pouzdanosti.



Slika 5.2: Interval 95% pouzdanosti za svako vozilo iz skupa podataka.

Zaključak

Dinamični rast obima saobraćaja već duže vrijeme nameće potrebu za pronalaženjem efikasnijih i pristupačnijih rješenja koja mogu zamijeniti tradicionalne, skupe radarske sisteme za merenje brzine vozila. U tom kontekstu, sistemi zasnovani na video podacima postaju sve relevantniji, jer koriste već postojeću infrastrukturu kamera postavljenih duž saobraćajnica, čime se značajno smanjuju troškovi implementacije i održavanja. Detekcija brzine na osnovu vizuelnih podataka predstavlja izuzetno atraktivnu i rastuću oblast, potpomognuta značajnim naprecima u oblasti kompjuterske vizije.

Ovo istraživanje rezultiralo je inovativnom metodom za procjenu brzine vozila koja se u potpunosti oslanja na algoritme dubokog učenja. Za razliku od mnogih postojećih rješenja opisanih u poglavlju 1, predložena metoda ne zahtijeva preciznu kalibraciju kamere niti poznavanje stvarnih dimenzija objekata ili scene, što samo po sebi predstavlja značajan iskorak. Korišćenjem specijalno dizajnirane jednodimenzionalne konvolucione mreže za obradu 1D signala izvučenih iz video snimaka pomoću detektora objekata, pokazano je da je moguće ostvariti preciznu i pouzdanu procjenu brzine vozila. Na osnovu postignutih rezultata, uz prosječnu grešku od 2.70 km/h, može se zaključiti da je istraživanje uspješno.

Medjutim, prostora za napredak svakako ima. Značajno ograničenje u ovom istraživanju bila je veličina VS13 skupa podataka. Iako je skup kvalitetan i pažljivo pripremljen, sa svojih 400 snimaka u trajanju od po 10 sekundi, predstavlja relativno mali skup u kontekstu dubokog učenja. Dalji rad na unapređenu predložene metode zasigurno bi uključivao prikupljane većeg ali i raznovrsnijeg skupa podataka za treniranje mreže.

Predložena metoda suočava se sa određenim ograničenjima. Svaki sistem baziran isključivo na viziji donekle je uslovljen i ograničen senzorima koje upotrebljava, odnosno kamerama. Tačnost i upotrebljivost podataka koje dobijamo opada kako se udaljenost kamere i posmatranog objekta povećava. Osim toga razni vremenski uslovi kao što su kiša i magla značajno utiču na pouzdanost i preciznost.

U ovom istraživanju korišćeni su snimci sa samo jednim vozilom u kadru, čime su stvoreni kontrolisani uslovi za procjenu brzine koji ne odražavaju dinamičnu prirodu saobraćaja, što se ovoj metodi može zamjeriti. U scenarijima sa više vozila, kao što je slučaj u realnom saobraćaju, bilo bi neophodno izvršiti modifikacije kako bi se precizno izdvojilo željeno vozilo. Ovo predstavlja pravac u kojem bi svakako trebalo ići.

Iako je preložena metoda robusna u smislu da ne zahtijeva poznavanje stvarnih dimenzija objekata, njen osnovni princip funkcionisanja je da se vozilo mora kretati prema kameri kako bismo imali promjenu površina graničnih okvira. Iako ovo ne predstavlja značajno ograničenje jer se kamere obično postavljaju tako da im je vidno polje paralelno u odnosu na pravac puta, interesantno bi bilo dalja istraživanja usmjeriti u pravcu estimacije brzine udaljenih vozila. U tom slučaju potrebno bi bilo definisati novu karakteristiku koja bi predstavljala ulaz našeg modela.

Literatura

[1] George Dimitrakopoulos, Lorna Uden, and Iraklis Varlamis. *The Future of Intelligent Transport Systems, 1st Edition*. 02 2020.

[2] Asier Perallos, Unai Hernandez-Jayo, Enrique Onieva, and Ignacio Julio García-Zuazola. *Intelligent Transport Systems: Technologies and Applications*. Wiley Publishing, 1st edition, 2015.

[3] Cecilia Wilson, Charlene Willis, Joan Hendrikz, Robyne Le Brocque, and Nicholas Bellamy. Speed cameras for the prevention of road traffic injuries and deaths. *Cochrane database of systematic reviews (Online)*, 11:CD004607, 10 2010.

[4] David Fernández-Llorca, Antonio Hernandez Martinez, and Ivan garcia daza. Vision-based vehicle speed estimation: A survey. *IET Intelligent Transport Systems*, 15, 05 2021.

[5] Lei Du, Qiao Sun, Changqing Cai, Jie Bai, Zhe Fan, and Yue Zhang. A vehicular mobile standard instrument for field verification of traffic speed meters based on dual-antenna doppler radar sensor. *Sensors*, 18(4):1099, 2018.

[6] Jiaxing Zhang, Wen Xiao, Benjamin Coifman, and Jon P Mills. Vehicle tracking and speed estimation from roadside lidar. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:5597–5608, 2020.

[7] Mahendra Mandava, Robert S Gammenthaler, and Steven F Hocker. Vehicle speed enforcement using absolute speed handheld lidar. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2018.

[8] Piotr Burnos, Janusz Gajda, Piotr Piwowar, Ryszard Sroka, Marek Stencel, and Tadeusz Zeglen. Measurements of road traffic parameters using inductive loops and piezoelectric sensors. 2007.

[9] Xiao-Yun Lu, Pravin Varaiya, Roberto Horowitz, Zhaomiao Guo, and Joe Palen. Estimating traffic speed with single inductive loop event data. *Transportation research record*, 2308(1):157–166, 2012.

[10] Pierre-Emmanuel Mazaré, Olli-Pekka Tossavainen, Alexandre Bayen, and D Work. Trade-offs between inductive loops and gps probe vehicles for travel time estimation: A mobile century case study. In *Transportation Research Board 91st Annual Meeting (TRB'12)*, volume 349, 2012.

[11] Jing Chen, Qichao Wang, Harry H Cheng, Weiming Peng, and Wenqiang Xu. A review of vision-based traffic semantic understanding in itss. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[12] Diogo Carbonera Luvizon, Bogdan Tomoyuki Nassu, and Rodrigo Minetto. A video-based system for vehicle speed measurement in urban roadways. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1393–1404, 2016.

[13] David Fernández Llorca, Carlota Salinas, Mario Jimenez, Ignacio Parra, AG Morcillo, Rubén Izquierdo, Javier Lorenzo, and MA Sotelo. Two-camera based accurate vehicle speed measurement using average speed at a fixed point. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2533–2538. IEEE, 2016.

[14] Huan-Sheng Song, Sheng-Nan Lu, Xiang Ma, Yuan Yang, Xue-Qin Liu, and Peng Zhang. Vehicle behavior analysis using target motion trajectories. *IEEE Transactions on Vehicular Technology*, 63(8):3580–3591, 2014.

[15] Wencheng Wu, Vladimir Kozitsky, Martin E Hoover, Robert Loce, and DM Todd Jackson. Vehicle speed estimation using a monocular camera. In *Video Surveillance and Transportation Imaging Applications 2015*, volume 9407, pages 17–30. SPIE, 2015.

[16] D Jeyabharathi and D Dejey. Vehicle tracking and speed measurement system (vtsm) based on novel feature descriptor: Diagonal hexadecimal pattern (dhp). *Journal of Visual Communication and Image Representation*, 40:816–830, 2016.

[17] Genyuan Cheng, Yubin Guo, Xiaochun Cheng, Dongliang Wang, and Jiandong Zhao. Real-time detection of vehicle speed based on video image. In *2020 12th*

International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), pages 313–317. IEEE, 2020.

[18] Jakub Sochor, Roman Juránek, and Adam Herout. Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement. *Computer Vision and Image Understanding*, 161:87–98, 2017.

[19] Chenghuan Liu, Du Q Huynh, Yuchao Sun, Mark Reynolds, and Steve Atkinson. A vision-based pipeline for vehicle counting, speed estimation, and classification. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7547–7560, 2020.

[20] D Bell, W Xiao, and P James. Accurate vehicle speed estimation from monocular camera footage. In *XXIV ISPRS Congress*. Newcastle University, 2020.

[21] Elnaz Vakili, Maryam Shoaran, and Mohammad R Sarmadi. Single–camera vehicle speed measurement using the geometry of the imaging system. *Multimedia Tools and Applications*, 79:19307–19327, 2020.

[22] Minh-Triet Tran, Tung Dinh-Duy, Thanh-Dat Truong, Vinh Ton-That, Thanh-Nhon Do, Quoc-An Luong, Thanh-An Nguyen, Vinh-Tiep Nguyen, and Minh N Do. Traffic flow analysis with multiple adaptive vehicle detectors and velocity estimation with landmark-based scanlines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 100–107, 2018.

[23] Ali Tourani, Asadollah Shahbahrami, Alireza Akoushideh, Saeed Khazaee, and Ching Y Suen. Motion-based vehicle speed measurement for intelligent transportation systems. *International Journal of Image, Graphics and Signal Processing*, 11(4):42–54, 2019.

[24] Huanan Dong, Ming Wen, and Zhouwang Yang. Vehicle speed estimation based on 3d convnets and non-local blocks. *Future Internet*, 11(6):123, 2019.

[25] Shengnan Lu, Yuping Wang, and Huansheng Song. A high accurate vehicle speed estimation method. *Soft Computing*, 24:1283–1291, 2020.

[26] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.

[27] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[28] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136, 2015.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[31] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.

[32] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716, 2023.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[34] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, cnns and vision transformers: A review. *IEEE Access*, 11:35479–35516, 2023.

[35] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal,

Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, 126:103514, 2022.

[36] K Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[39] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[40] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104:154–171, 2013.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[42] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[45] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[47] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[48] T Lin. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[49] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[50] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. cite arxiv:1804.02767Comment: Tech Report.

[51] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[52] Glenn Jocher. Ultralytics yolov5, 2020.

[53] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.

[54] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[55] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics. 2023.

[56] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[57] Ljubiša Stanković and Danilo Mandic. Convolutional neural networks demystified: A matched filtering perspective-based tutorial. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[58] Shengxi Li, Xinyi Zhao, Ljubisa Stankovic, and Danilo Mandic. Demystifying cnns for images by matched filters. *arXiv preprint arXiv:2210.08521*, 2022.

[59] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022.

[60] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[61] A Géron. Hands-on machine learning with scikit-learn, keras & tensorflow farnham. *Canada: O'Reilly*, 2019.

[62] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[63] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[64] Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.

[65] P Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

[66] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arxiv 2012. *arXiv preprint arXiv:1207.0580*, 2012.

[67] Jakub Sochor, Roman Juránek, Jakub Špaňhel, Lukáš Maršík, Adam Širokỳ, Adam Herout, and Pavel Zemčík. Comprehensive data set for automatic single camera visual speed measurement. *IEEE Transactions on Intelligent Transportation Systems*, 20(5):1633–1643, 2018.

[68] Slobodan Djukanović, Nikola Bulatović, and Ivana Čavor. A dataset for audio-video based vehicle speed estimation. In *2022 30th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2022.

[69] Oluwaseyi Ezekiel Olorunshola, Martins Ekata Irhebhude, and Abraham Eseoghene Evwiekpaefe. A comparative study of yolov5 and yolov7 object detection algorithms. *Journal of Computing and Social Informatics*, 2(1):1–12, 2023.

[70] Marko Horvat and Gordan Gledec. A comparative study of yolov5 models performance for image localization and classification. In *33rd Central European Conference on Information and Intelligent Systems (CECIIS)*, volume 349, 2022.

[71] Jacob Solawetz. Vehicles-openimages dataset. `https://universe.roboflow.com/roboflow-gw7yv/vehicles-openimages`, June 2022.

[72] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.