Vizijski sustav za klasifikaciju objekata temeljen na YOLOv8 modelu

Stepinac, Anamarija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Anamarija Stepinac

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Anamarija Stepinac

Zagreb, 2024.

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

Zahvaljujem se mentoru, izv. prof. dr. sc. Tomislavu Stipančiću, na pruženoj pomoći i podršci tijekom studiranja i izrade diplomskog rada.

Zahvaljujem se svojoj obitelji, posebno roditeljima, na bezuvjetnoj ljubavi i podršci u svemu što radim. Bez njih, sve ovo ne bi bilo moguće.

Zahvaljujem se i svojim prijateljima na vjeri u mene i što su uljepšali moje studentske dane.

Hvala vam svima što ste me pratili kroz diplomski studij i pomogli mi da ga uspješno završim.

Anamarija Stepinac

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 01**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Anamarija Stepinac** JMBAG: 0035215608

Naslov rada na hrvatskom jeziku: **Vizijski sustav za klasifikaciju objekata temeljen na YOLOv8 modelu**

Naslov rada na engleskom jeziku: **A vision system for object classification based on the YOLOv8 model**

Opis zadatka:

Modeli strojnog vida omogućuju računalima da prepoznaju značajke na slikama te ih koriste kao gradivne elemente kod složenijih primjena u sklopu stvarne okoline. U tu se svrhu često koriste različite metode umjetne inteligencije temeljene na pred treniranim modelima s pomoću kojih vizijski sustavi vrše akviziciju informacija, te uče ili identificiraju objekte, pojave ili ljude.
YOLO je model temeljen na konvolucijskoj neuronskoj mreži koji se koristi za otkrivanje različitih objekata na slici u ovisnosti o skupu podataka za trening. YOLO model čak može identificirati više objekata unutar iste slike te ih klasificirati u različite kategorije.
U radu je potrebno izraditi cjelovito softversko rješenje za klasifikaciju različitih objekata u sklopu stvarne primjene temeljeno na YOLO v8 modelu za prepoznavanje, koje uključuje:

- bazu označenih slika pripremljenih za trening modela, te
- model za pronalaženje i praćenje barem dvije klase objekata u pokretu bez obzira na trenutnu orijentaciju te perspektivu gledanja.

Dobiveno softversko rješenje je potrebno eksperimentalno evaluirati uključivši različite objekte te dati kritički osvrt na rad aplikacije.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
16. studenoga 2023.	18. siječnja 2024.	22. – 26 . siječnja 2024.
Zadatak zadao:		Predsjednik Povjerenstva:
Izv. prof. dr. sc. Tomislav Stipančić		Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Računalni vid je područje umjetne inteligencije, odnosno dubokog učenja, koje računalnim sustavima daje mogućnost vida. Za primjenu računalnog vida su razvijeni algoritmi za izvršavanje zadataka na slikama i videima, poput prepoznavanja objekata, klasifikacije i praćenja objekata. U radu je iznesena teorijska osnova potrebna za razumijevanje algoritama računalnog vida te je potkrijepljena primjerom u kojem je korišten algoritam YOLOv8. Algoritam je treniran i provjeren na označenom skupu podataka, a zatim testiran za prepoznavanje, klasifikaciju i praćenje na neviđenom primjeru.

Ključne riječi: umjetna inteligencija, duboko učenje, računalni vid, prepoznavanje objekata, klasifikacija, praćenje objekata, YOLOv8

SUMMARY

Computer vision is an area of artificial intelligence, or rather deep learning, that gives computer systems the ability to see. Computer vision algorithms have been developed for performing tasks on images and videos, such as object recognition, classification and object tracking. The paper presents the theoretical basis necessary for understanding computer vision algorithms and is supported by an example in which the YOLOv8 algorithm was used. The algorithm was trained and validated on a labelled dataset and then tested for recognition, classification, and tracking on an unseen example.

Key words: artificial intelligence, deep learning, computer vision, object detection, classification, object tracking, YOLOv8

1. UVOD

Danas je umjetna inteligencija jedna od najaktualnijih tema tehnologije i šire. Zbog naglog razvitka, ponekad je teško razumjeti kako taj cijeli proces funkcionira. Umjetna inteligencija seže još iz sredine prošlog stoljeća, no kao i sve druge tehnologije, imala je eksponencijalan rast zadnjih godina. Iako je razvoj bio krajnje opsežan, puno modela i algoritama u primjeni potječu od iste osnovne strukture.

Računalni vid dio je umjetne inteligencije koji podrazumijeva stvaranje mogućnosti vida kod računala. Za to je potrebno oblikovati cijeli proces – od dobivanja slikovnog prikaza do obrade i konačno dobivanja konteksta. Način na koji se to ostvaraju je razvojem umjetnih neuronskih mreža koje su stvorene po uzoru na one biološke. Premda još uvijek nije istraženo kako vid funkcionira kod ljudi, taj proces nije moguće na isti taj način replicirati ni putem računala. Isto tako, to je razlog zbog kojeg postoji mnogo algoritama koji rade na različite načine, pokušavajući što više se približiti ljudskoj interpretaciji. U tom nastojanju, danas postoje brojni algoritmi i modeli koji postižu izvanredne rezultate, čak bolje od ljudi u mnogim slučajevima.

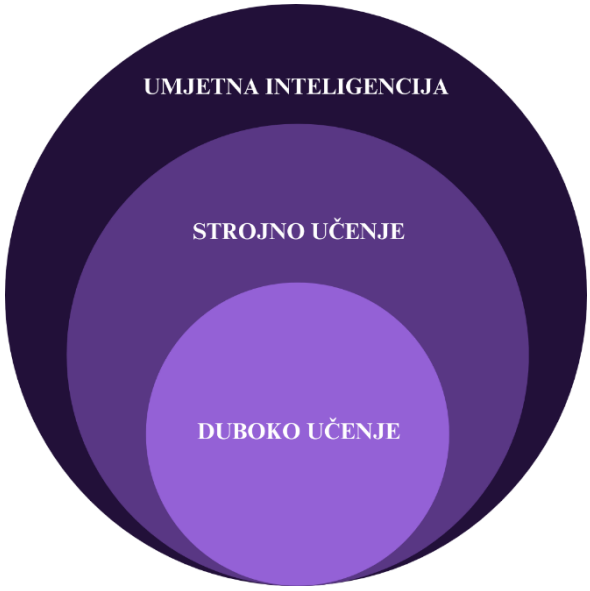
Cilj rada je prikazati važne strukture za razvoj i primjenu u računalnom vidu te kroz praktični dio prikazati proces učenja i primjene modela YOLOv8, temeljenog na neuronskim mrežama.

2. UMJETNA INTELIGENCIJA

2.1. Definicija i vrste

Umjetna inteligencija ili UI (eng. *AI – artificial intelligence*) je mehanizam za uključivanje ljudske inteligencije u strojeve kroz skup pravila, odnosno algoritam. [1] Cilj je istrenirati stroj da oponaša ljudski mozak i njegove sposobnosti. Postoji mnogo vrsta UI koji nastoje ispuniti taj cilj, slika 1.

UI se fokusira na 3 vještine: učenje, razlikovanje i samoispravljanje.



Slika 1 Hijerarhija umjetne inteligencije, strojnog učenja i dubokog učenja

Strojno učenje (eng. *ML – machine learning*) je proces koji omogućava računalu da uči automatski na temelju svojih iskustava i prema tome se poboljšava, bez eksplicitnog programiranja. To je podskup umjetne inteligencije te se fokusira na razvoj programa za samostalni dohvat podataka. Glavni cilj strojnog učenja je omogućiti računalu da samostalno uči kroz iskustvo bez ikakve ljudske intervencije ili pomoći. Strojno učenje se dijeli u 3 kategorije: nadzirano, nenadzirano i podržano učenje. Nadzirano učenje je kada računalo koristi prethodno označene podatke. Dan mu je točan rezultat s kojim uspoređuje svoje rješenje te se prema tome podešava. Taj proces se ponavlja dok nije zadovoljena točnost modela na podacima za treniranje, a zatim se može koristiti i na drugim primjerima. Kod nenadziranog učenja, podaci nisu označeni – računalo mora samostalno uočiti uzorke i strukture i na temelju toga doći do zaključka. Kombinacijom ove dvije kategorije javlja se i dodatna kategorija polunadzirano učenje, gdje se za trening koriste označeni i neoznačeni podaci. Treća kategorija

učenja je podržano te ono daje najveću slobodu računalu. Koristi metodu „pokušaj i promašaj“ (eng. *trial and error*) za postizanje najboljih rezultata.

Duboko učenje (eng. *DL – deep learning*) je podskup strojnog učenja koji koristi razne vrste neuronskih mreža za oponašanje ljudskog mozga. Algoritmi koje koristi se zasnivaju na prepoznavanju uzoraka u podacima i sukladno tome ih klasificiraju. Primjenjuje se na velike skupove podataka i mehanizam predviđanja provode računala. Naziva se „dubokim“ jer koristi višestruke slojeve u mreži.

2.1.1. Biološka i umjetna neuronska mreža

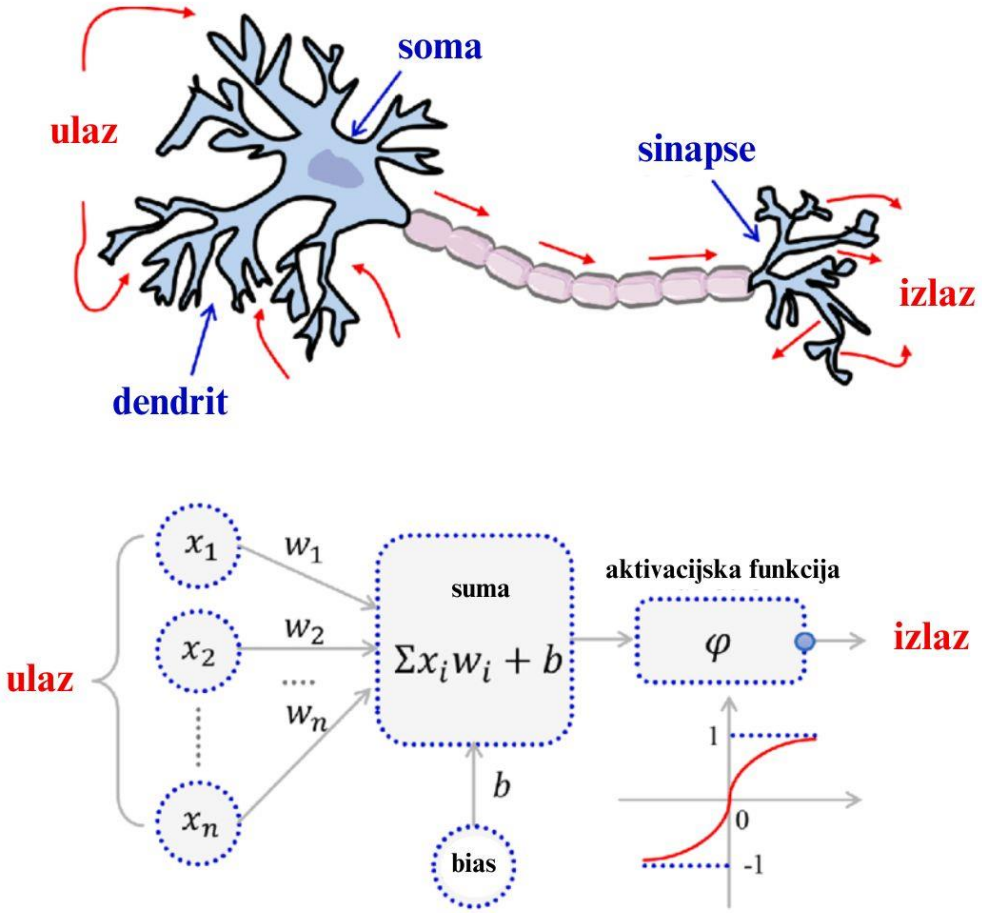
Umjetne neuronske mreže su stvorene po uzoru na biološke, slika 2. Biološke neuronske mreže tvore pojedini međusobno povezani neuroni ili živčane stanice. Neuroni su osnovne jedinice žičanog sustava i električno su nabijene. Imaju 3 dijela, a to su dendriti, soma i akson. [2] Dendriti su razgranati nastavci neurona koji prihvaćaju živčane impulse, odnosno prikupljaju informacije. Dobiveni živčani impulsi se prosljeđuju staničnom tijelu neurona koji se naziva soma te je ona zadužena za procesiranje informacija dobivenih putem dendrita. Na somu se nastavlja završni dio neurona zvani akson, na čijem početku se stvara akcijski potencijal. Akcijski potencijal je zapravo vrijednost električnog potencijala između unutarnjeg i vanjskog dijela neurona, i ako impuls uzrokuje da neuron s potencijalom mirovanja prijeđe tu vrijednost, akson neurona se aktivira i prenosi impuls na susjedni ili susjedne neurone. Komunikaciju između neurona omogućuje sinapsa, kemijska ili električna veza koja prenosi informacije s jednog neurona na drugi.

U umjetnoj neuronskoj mreži, postoje dijelovi koji simuliraju navedene funkcije bioloških neurona, tablica 1. Dendrite oponaša ulaz, somu čine čvorovi, a izlaz je akson. Sinapse su zamijenjene vezama s težinama, a težine ukazuju na to koliko pojedini neuron ima utjecaj na sljedeći. Akcijski potencijal je zamijenjen aktivacijskom funkcijom. Cijela neuronska mreža se ponaša kao vid kod ljudi, gdje je slika na ulazu u mrežu poput slike koja se vidi očima, put od očiju do centra za vid u mozgu su skriveni slojevi, a nakon analize na izlazu mreže se dobiva kontekst poput konteksta koji ljudi stvaraju u mozgu.[3]

Tablica 1 Usporedba dijelova biološkog i umjetnog neurona

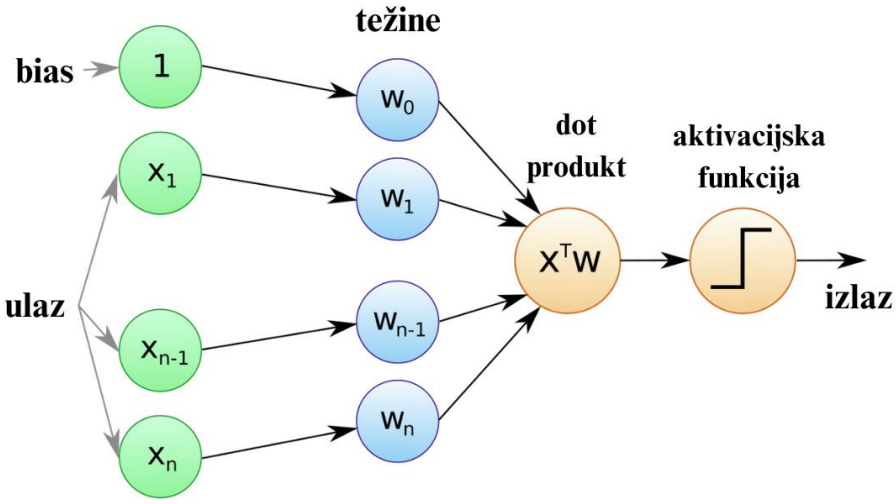
BIOLOŠKI NEURON	UMJETNI NEURON
dendriti	ulaz
soma	čvor
akson	izlaz
sinapsa	veze s težinama

Čvorovi su raspoređeni u slojeve, a izlaz jednog sloja je ulaz u sljedeći sloj. Iznos ulazne vrijednosti čvora se računa kao suma umnožaka vrijednosti ulaza, odnosno dot produkt, i težine odgovarajuće veze. Na dobivenu vrijednost se primjenjuje neka od aktivacijskih funkcija i daje izlaznu vrijednost. U umjetnim neuronskim mrežama se pojavljuje i pristranost (eng. *bias*), realna konstanta koja se zbraja u sumu umnožaka ulaza i težina kako bi se poboljšalo učenje. Ovaj proces se ponavlja u svim skrivenim slojevima dok se ne dostigne izlazni sloj.



Slika 2 Usporedba biološkog i umjetnog neurona

Najjednostavnija neuronska mreža je ona s jednim neuronom, odnosno bez skrivenog sloja, a naziva se perceptron te je prikazana na slici 3. Koristi se za klasifikaciju linearno separabilnih podataka što je čini vrlo ograničenom, stoga se rijetko koristi. Mreže s jednim ili više skrivenih slojeva se mogu primijeniti na kompleksnije i nelinearne probleme te se zbog toga češće primjenjuju. [4]



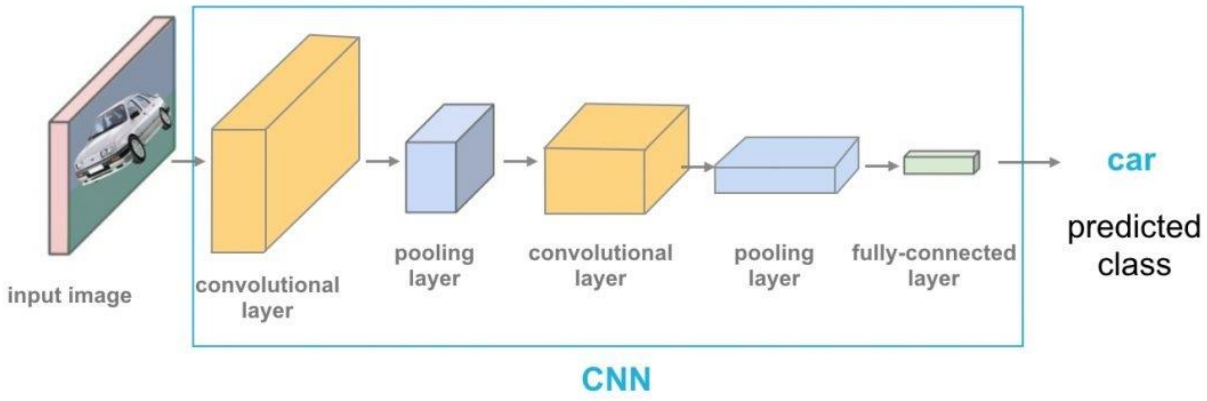
Slika 3 Perceptron

2.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (eng. *convolutional neural networks*), skraćeno CNN ili ConvNets, su posebna vrsta neuronskih mreža namijenjenih za rad s dvodimenzionalnim slikovnim podacima, no mogu se koristiti i za rad s jednodimenzionalnim i trodimenzionalnim podacima. Ime su dobile po konvolucijskom sloju koji provodi konvoluciju. [5] Arhitektura mreže i njeni dijelovi su objašnjeni u sljedećim potpoglavljima.

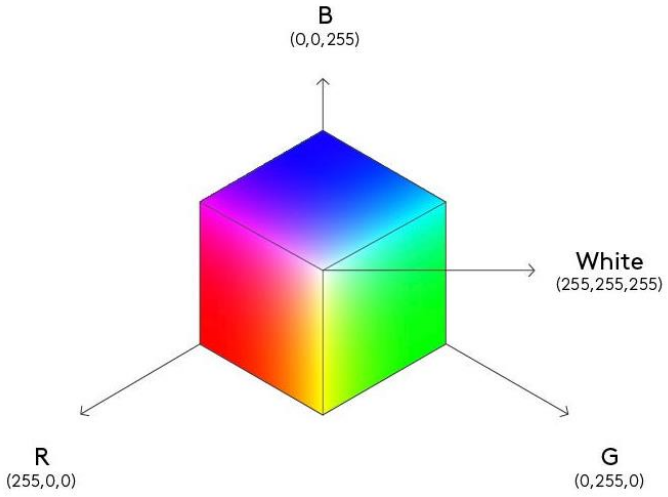
2.2.1. Arhitektura

Arhitektura općenite konvolucijske neuronske mreže sadržava ulazni i izlazni sloj, konvolucijske slojeve, aktivacijske funkcije, slojeve sažimanja te potpuno povezane slojeve. Premda se ova vrsta neuronskih mreža najčešće koristi za računalni vid, odnosno obradu slikovnih podataka, za ulaz se koristi slika. Na slici 4 je prikazana klasična arhitektura CNN mreže sa svim navedenim dijelovima.



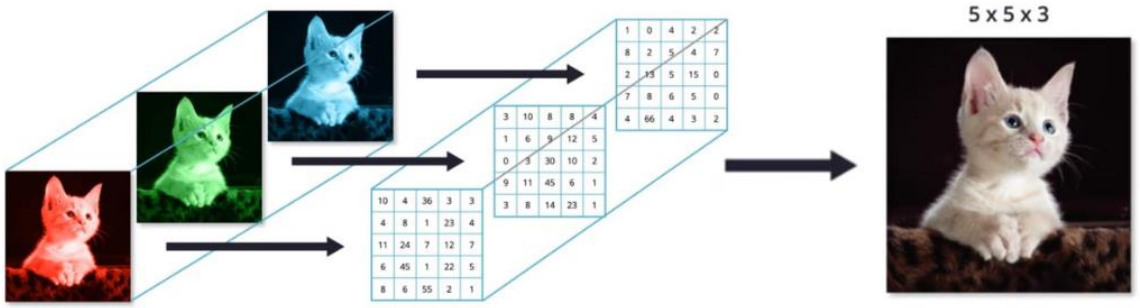
Slika 4 Arhitektura CNN mreže

Slika je u računalu prikazana kao skup najmanjih gradivnih elementa slike piksela, koji određuju širinu i visinu slike te se prikazuju kao rezolucija, npr. 1920x1080. Što je rezolucija veća, to je slika detaljnija. Boje se prikazuju kao dubina slika, odnosno kanali (eng. *channels*), ovisno o formatu u kojem se slika prikazuje. Najkorišteniji format prikaza slike je RGB (*red-green-blue*), slika 5, aditivni model boja kod kojeg se kombinacijom osnovnih boja, odnosno crvene, zelene i plave, dobivaju sve ostale boje. [6]



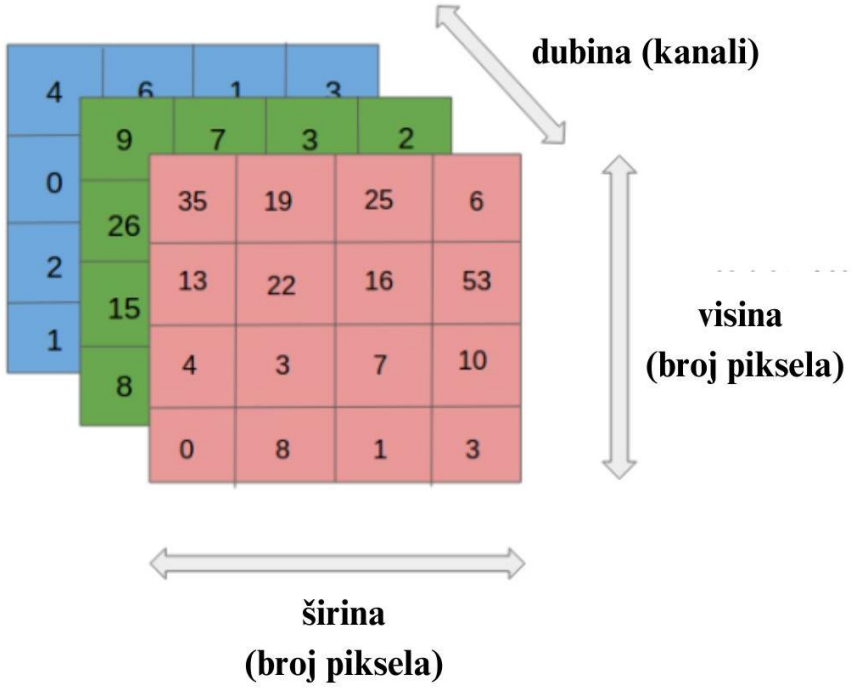
Slika 5 RGB kocka

Svaki kanal je matrica piksela u kojoj se nalaze iznosi vrijednosti pojedinih piksela za boju tog kanala, a njihovim preklapanjem se dobiva slika u punom spektru boja, slika 6.



Slika 6 Preklapanje kanala za dobivanje slike u punom spektru boja

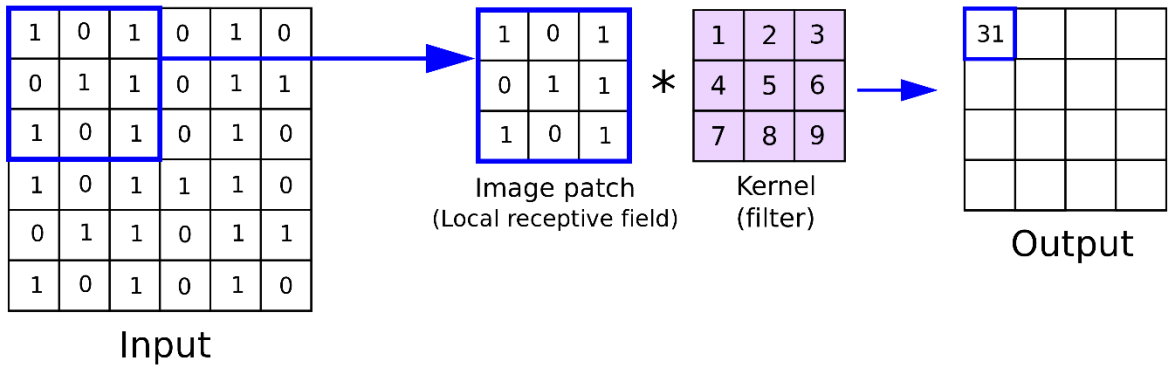
Pojedina boja se prikazuje kao skup vrijednosti ovih triju, npr. (28, 47, 223), slika 7, gdje se njihove vrijednosti kreću između 0 i 255. Crno-bijele ili jednobojne slike imaju samo jedan sloj za dubinu, dok slike u više boja imaju 3 sloja.



Slika 7 Prikaz širine, visine i dubine slike

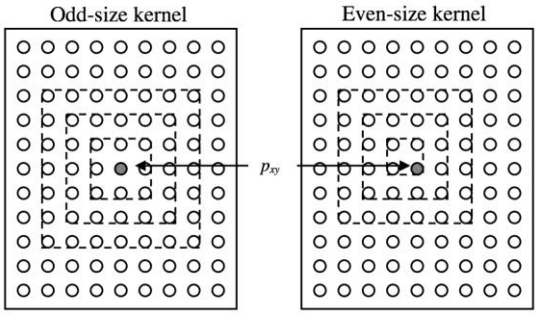
2.2.2. Konvolucijski sloj

Konvolucija je linearna matematička operacija u kojoj se težine množe s ulaznim podacima po elementima. Težine i ulazni podaci su zapravo matrice. Ulazni podaci su cijela slika koja je prikazana matricom veličine rezolucije (broj piksela po širini x broj piksela po visini), no za konvoluciju se koristi samo dio te matrice koji odgovara veličini matrice težina, slika 8. [7]



Slika 8 Konvolucija

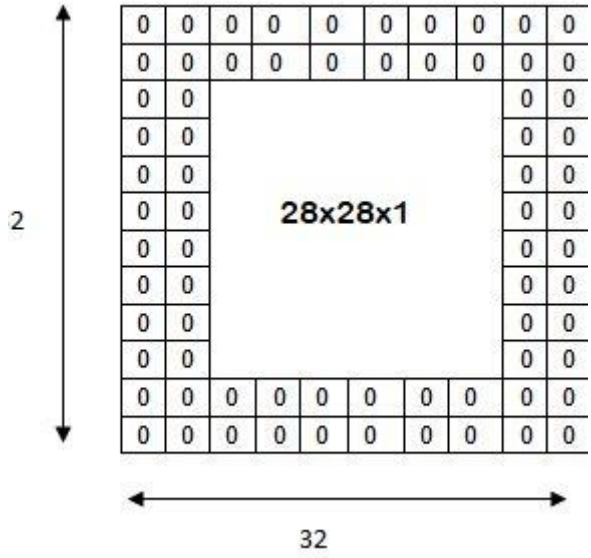
Težine se nalaze u kvadratnoj matrici koja se naziva filtar ili kernel. Veličina kernela je proizvoljna, ali se odabire neparan broj kako bi piksel nad kojim se vrši operacija ostao centralan, npr. 5x5, slika 9.



Slika 9 Prikaz centralnog piksela kod neparnog kernela i nepostojanje istog kod parnog kernela

Nakon konvolucije nad jednim pikselom, kernel se premješta na idući i na taj način se „šeće“ po slici dok ne izvrši konvoluciju nad svakim pikselom.

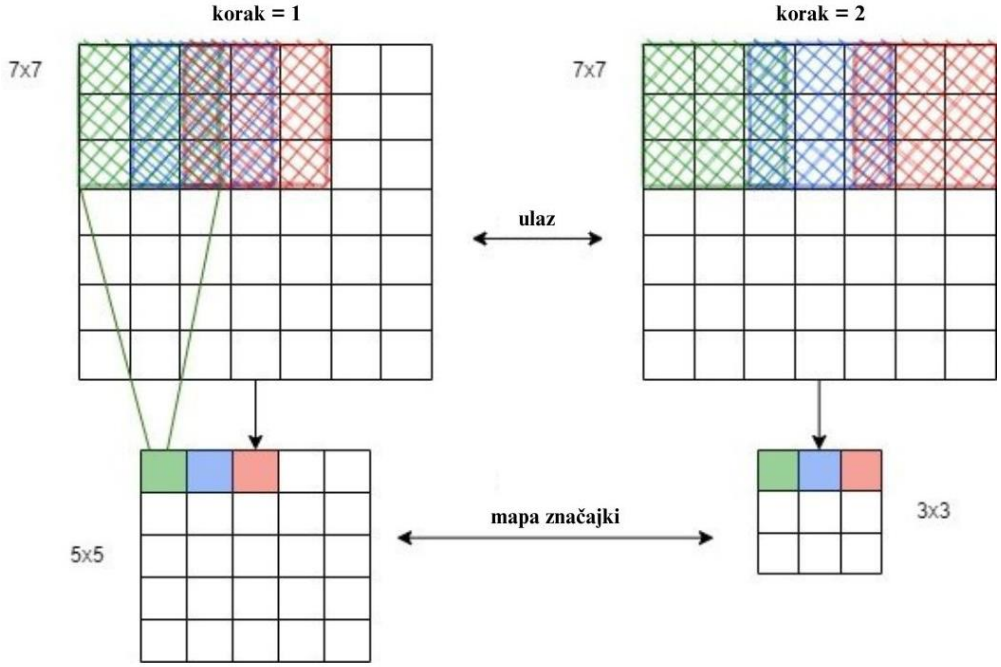
Kako bi se konvolucija mogla izvršiti i nad rubnim pikselima, koristi se popunjavanje nulama (eng. *zero padding*), slika 10. To su pikseli koji se dodaju poput okvira s vrijednošću nula kako ne bi utjecali na rezultat, a dodaje ih se onoliko koliko je potrebno da se kernel može centralno pozicionirati na rubni piksel.



Slika 10 Popunjavanje nulama

Kernel se može smatrati i detektorom značajki jer uočava uzorke koji se ponavljaju, tvoreći značajke na slici te kao izlaz daju mapu značajki (eng. *feature maps*). Kerneli za otkrivanje pojedinih značajki razlikuju. Postoje kerneli za zamućivanje, za izoštravanje, za otkrivanje rubova i za mnoge druge značajke slike.

Osim veličine kernela, na veličini mape značajke utječe i korak (eng. *stride*). [8] Korak je također proizvoljan parametar i određuje za koliko će se piksela pomaknuti kernel, slika .



Slika 11 Razlika u mapi značajki kod različitih vrijednosti koraka

2.2.3. Aktivacijska funkcija

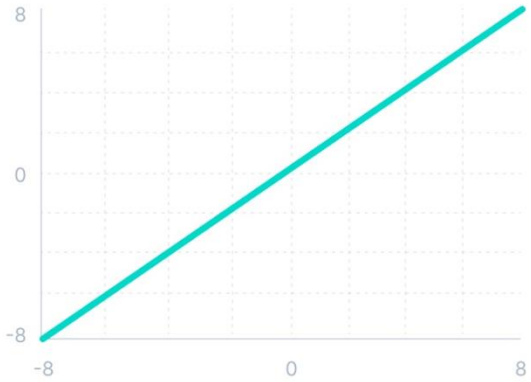
Aktivacijska funkcija određuje hoće li se neuron aktivirati ili neće računajući sumu umnožaka težina i ulaza i dodajući pristranost. [9] Ako ta vrijednost prelazi prag (eng. *treshold*) aktivacijske funkcije, neuron će se aktivirati.

Neuronske mreže su same po sebi linearni regresijski modeli, stoga mogu rješavati samo linearne probleme. Iz tog razloga se za aktivacijsku funkciju uzima neka od nelinearnih funkcija, kako bi unijela nelinearnost u model i mogla rješavati kompleksnije probleme.

Najčešće korištene aktivacijske funkcije su linearna, sigmoidna, tanh, ReLU i Softmax funkcija.

Linearna funkcija je zapravo obični pravac prikazan jednadžbom (1) i slikom 12. Bez obzira na broj slojeva, rezultira linearnom funkcijom te se iz tog razloga koristi samo za izlazni sloj.

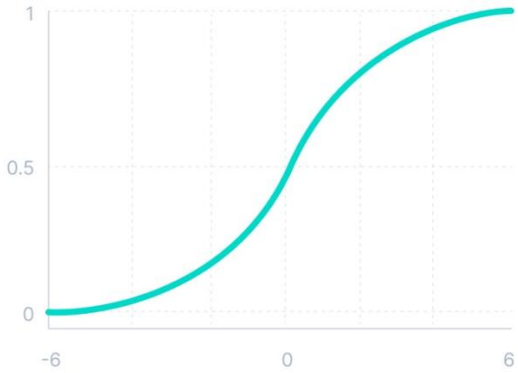
$$f(x) = x \tag{1}$$



Slika 12 Linearna funkcija

Sigmoidna funkcija je prikazana jednadžbom (2), nelinearna je i vrlo strma, slika 13. Vrijednost ove funkcije se kreće između 0 i 1. Iz tog razloga se koristi u mrežama gdje je potrebno predvidjeti pripadnost klasi jer se vjerojatnost pripadnosti isto kreće u rangu između 0 i 1.

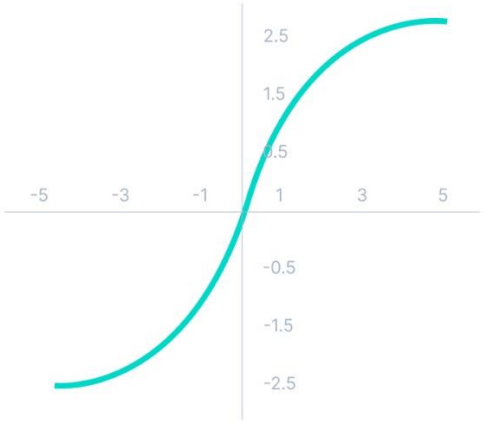
$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$



Slika 13 Sigmoidna funkcija

Tanh funkcija je funkcija hiperbolnog tangensa, prikazana jednadžbom (3) i slikom 14. Oblikom odgovara sigmoidnoj funkciji, no vrijednosti koje poprima su između -1 i 1, odnosno pomaknute su u odnosu na vrijednosti koje poprima sigmoidna funkcija.

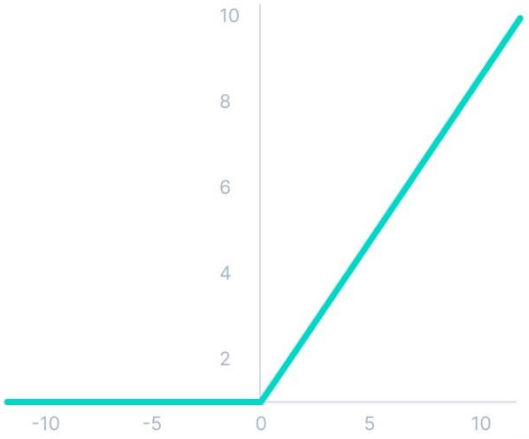
$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$



Slika 14 Tanh funkcija

ReLu (eng. *rectified linear unit*) je najčešća aktivacijska funkcija, oblika prikazanog na slici prema jednadžbi (4) i slici 15, te izgledom podsjeća na linearnu. Koristi se jer unosi nelinearnost, a brža je od sigmoidne i funkcije hiperbolnog tangensa. Specifično za ovu funkciju je da/Brža je iz razloga što ne aktivira sve neurone istovremeno; oni će biti deaktivirani samo u slučaju kada im vrijednost padne ispod 0.

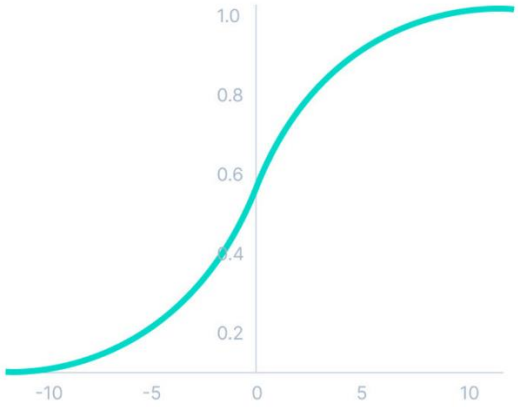
$$f(x) = max(0, x) \tag{4}$$



Slika 15 ReLU funkcija

Softmax funkcija je vrsta sigmoidne funkcije koja se koristi za klasificiranje u problemima više klasa jer je opisana kao kombinacija više sigmoidnih funkcija. Rang funkcije se kreće u vrijednostima između 0 i 1, kao i vrijednosti vjerojatnosti pripadnosti klasa. Prikazana je jednadžbom (5) i slikom 16. Na izlazu daju vjerojatnosti pripadnosti klasi.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{5}$$



Slika 16 Softmax funkcija

Nepisano pravilo kod aktivacijskih funkcija je da se počinje s upotrebom ReLU funkcije i nakon toga primijeni neka druga ako navedena ne daje zadovoljavajuće rezultate. Smjernice za primjenu aktivacijskih funkcija su:

- ReLU bi se trebala koristiti samo u skrivenim slojevima.
- Sigmoidna funkcija i funkcija hiperbolnog tangensa se ne bi trebale koristiti u skrivenim slojevima jer su podložne stvaranju problema tijekom treninga mreže.

Ovisno o problemu koji se rješava, preporučene aktivacijske funkcije za izlazni sloj su:

Regresija – linearna;

Binarna klasifikacija – sigmoidna;

Višeklasna klasifikacija – softmax;

Klasifikacija s više oznaka – sigmoidna.

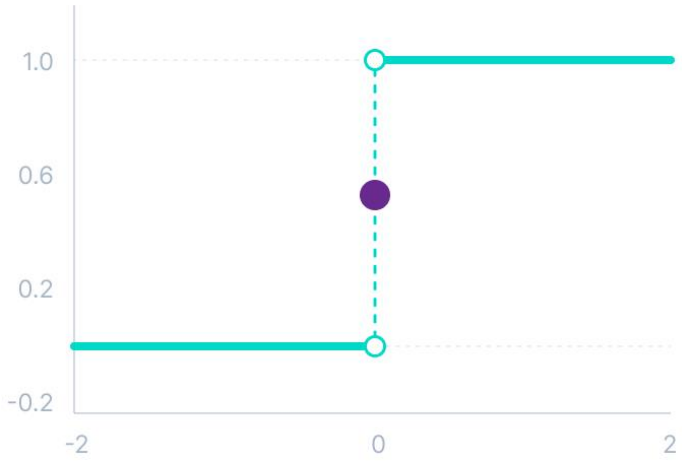
Aktivacijska funkcija tipična za skrivene slojeve ovisi o arhitekturi neuronske mreže:

Konvolucijska neuronska mreža – ReLU;

Neuronska mreža s povratnom vezom – hiperbolni tangens ili sigmoidna.

Postoji još jedna aktiacijska funkcija koja se koristi samo kod perceptrona, a to je Heavisideova step funkcija, prikazana jednadžbom (6) i slikom 17.

$$f(x) = \begin{cases} 0 \; za \; x \; < 0 \\ 1 \; za \; x \; \geq 0 \end{cases} \tag{6}$$



Slika 17 Heavisideova step funkcija

Perceptron je neuronska mreža koja se koriti kao binarni klasifikator, obzirom da može poprimiti samo dvije vrijednosti: 0 ili 1. Postoji mogućnost i korištenja signum funkcije koja daje vrijednosti -1 i 1.

2.2.4. Sloj sažimanja

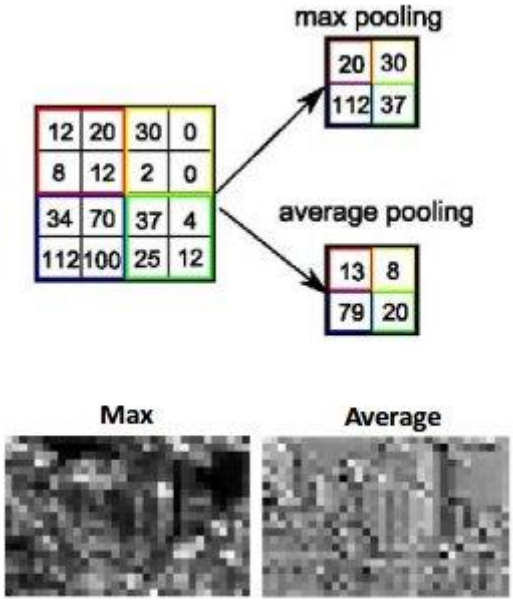
Sažimanje (eng. *pooling*) je proces koji se provodi tako da dvodimenzionalni kvadratni filtar ili kernel prelazi po svakom kanalu mape značajki i sažima značajke na dijelu koji kernel pokriva. Svrha sloja sažimanja je smanjenje dimenzija mape značajki, ujedno smanjujući i broj

parametara za učenje i broj potrebnih računalnih operacija koje izvršava mreža. [10] Uobičajeno je da se nalazi nakon konvolucijskog sloja kako bi se sažele značajke i kako bi daljnja analiza bila manje zahtjevna. To čini mrežu robusnijom na promjene u položaju značajki u ulaznim podacima, odnosno slici.

Parametri procesa sažimanja su veličina kernela (može biti i paran i neparan broj jer nije bitno da postoji centralni piksel) i korak za koji se kernel pomiče.

Najčešće se koriste 2 vrste sažimanja:

- Sažimanje maksimalnih vrijednosti (eng. *max pooling*) – na dijelu slike koji prekriva kernel nalazi maksimalnu vrijednost, slika ;
- Prosječno sažimanje (eng. *average pooling*) – na dijelu slike koji prekriva kernel nalazi prosječnu vrijednost, slika .



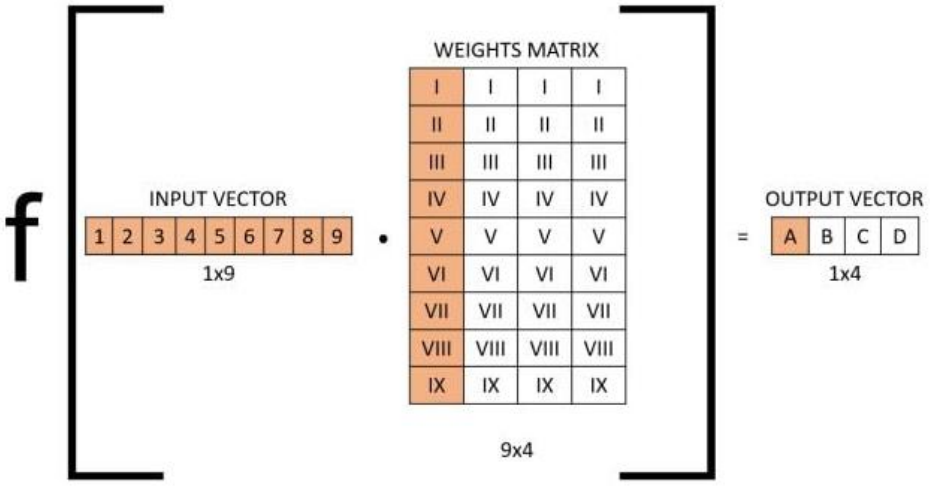
Slika 18 Sažimanje maksimalnih vrijednosti i prosječno sažimanje te usporedba rezultata tih sažimanja

U konvolucijskim neuronskim mrežama se koristi sažimanje maksimalnih vrijednosti.

2.2.5. Potpuno povezani slojevi

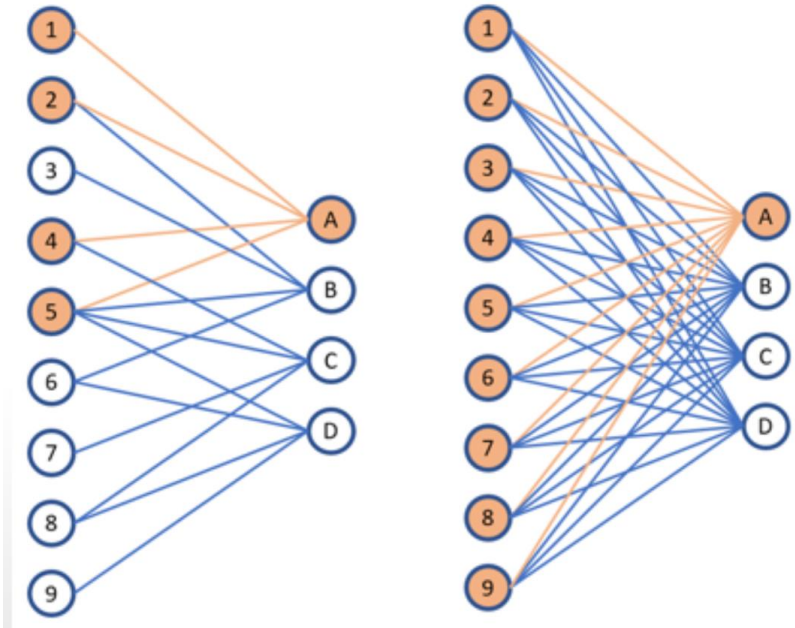
Potpuno povezani slojevi (eng. *fully connected layers*) su slojevi u kojima su svi ulazni neuroni povezani sa svim izlaznim neuronima i tvori potpuno povezanu neuronsku mrežu (eng. *FCNN – fully connected neural network*) koja je posljedni dio kovolucijske neuronske mreže. Prvi u nizu potpuno povezanih slojeva uzima mapu značajki dobivenu pomoću konvolucijskih slojeva

i slojeva sažimanja te provodi ravnanje (eng. *flattening*) dvodimenzionalne strukture u jednodimenzionalnu, odnosno pretvara dobivenu matricu u vektor, slika 19.



Slika 19 Ravnanje dvodimenzionalne strukture u jednodimenzionalnu

Uspoređujući s konvolucijskim slojem gdje na neurone sljedećeg sloja utječu samo oni neuroni na koje djeluje kernel, ovi slojevi se razlikuju po tome što svi neuroni jednog sloja utječu na sve neurone sljedećeg sloja, slika 20. Svaka veza između neurona 2 sloja ima težinu te prikazuje koliki utjecaj ima, slika , a one se podešavaju u procesu učenja svakim prolaskom kroz mrežu zvanim iteracija, kao i iznosi pristranosti.



Slika 20 Usporedba konvolucijskog (lijevo) i potpuno povezanog sloja (desno)

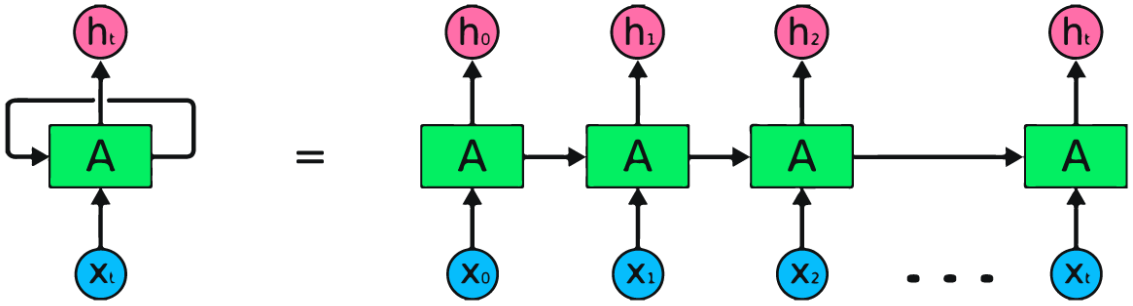
Zadaća ovog sloja je izvršiti klasifikaciju. Izlaz zadnjeg potpuno povezanog sloja je izlaz cijele konvolucijske mreže, a prikazan je u obliku vektora u kojem se vidi vjerojatnost pripadnosti klasi. Prema tome, veličina tog vektora je 1xn, gdje je n broj klasa koje mreža može prepoznati.

2.3. Ostale metode

Duboko učenje se, osim za potrebe računalnog vida, koristi i u mnogim drugim područjima. To mogu biti prepoznavanje govora, procesiranja prirodnog govora (eng. *NLP – natural language processing*), strojni prijevod i mnoge druge primjene. Najpoznatije i najprimjenjivanije metode dubokog učenja su navedene u sljedećih nekoliko potpoglavlja. [12]

2.3.1. Rekurentna neuronska mreža

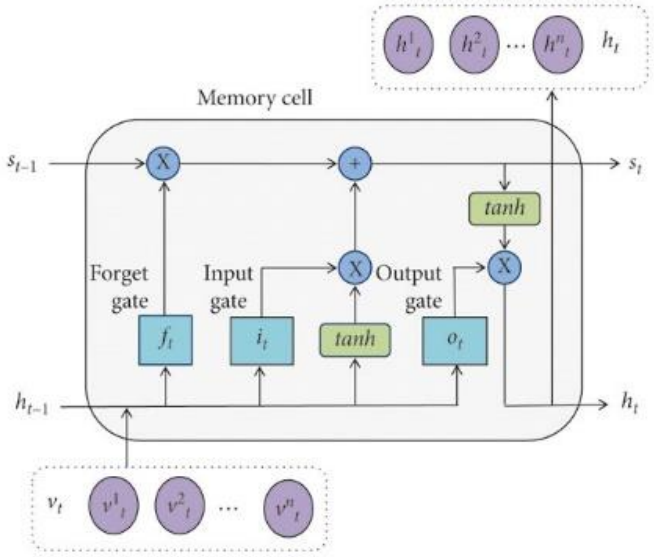
Rekurentne neuronske mreže (eng. *RNNs – recurrent neural networks*) su mreže u čijoj arhitekturi postoje veze koje tvore usmjerene krugove i radi sa sekvencijalnim podacima. [13] Od ostalih neuronskih mreža se razlikuju po tome što imaju „memoriju“, odnosno koriste podatke prijašnjih ulaza kako bi utjecali na trenutne podatke.



Slika 21 RNN mreža

2.3.2. Mreža dugog kratkoročnog pamćenja

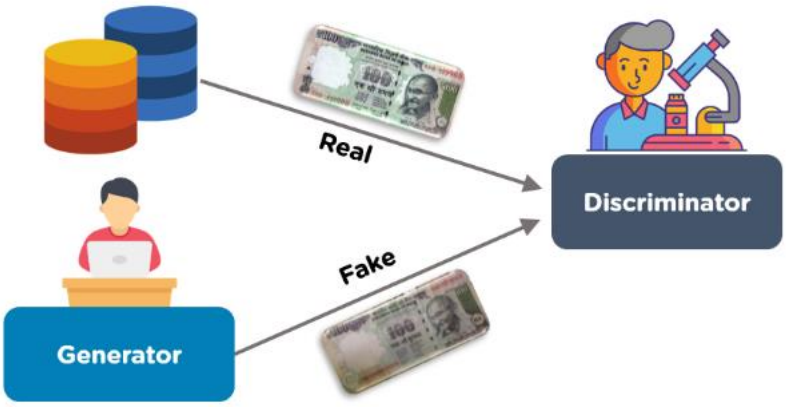
Mreže dugog kratkoročnog pamćenja (eng. *LSTMs – long short term memory networks*) su specijalne rekurentne neuronske mreže sposobne za učenje dugoročnih ovisnosti (eng. *long term dependencies*). [14] U ovoj mreži postoji LSTM ćelija koja služi kao dugoročna memorije koja kombinira „pamćenje“ i „zaboravljanje“ prethodnih informacija.



Slika 22 LSTM mreža

2.3.3. Generativna suparnička mreža

Generativne suparničke mreže (eng. *GANs – generative adversarial networks*) su vrsta mreža koja se koristi za danas sve popularniju generativnu umjetnu inteligenciju. Rade na način da uče iz postojećih trening podataka i na temelju toga stvaraju nove koji im sliče. [15] U arhitekturi mreže se nalaze 2 modela: generator i diskriminator. Generator je zaslužan za stvaranje novih instanci koje diskriminator uspoređuje sa stvarnima, slika 23. Diskriminator uči uspoređujući stvarne i generirane primjere, a generator uči prema tome koliko je ili nije zavarao diskriminatora.



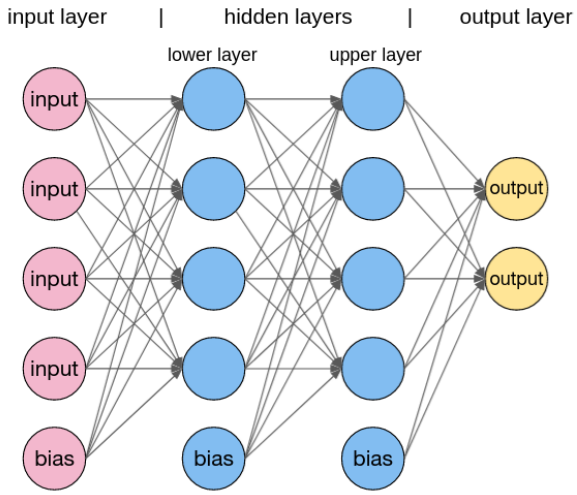
Slika 23 GAN mreža

2.3.4. Mreža radijalne baze

Mreže radijalne baze (eng. *RBFNs – radial basis function networks*) su neuronske mreže od 3 sloja (ulazni, skriveni i izlazni) koje kao aktivacijsku funkciju koriste radijalne bazne funkcije. Radijalne bazne funkcije su one čiji iznos ovisi o udaljenosti instance od fiksne točke. [16]

2.3.5. Višeslojni perceptron

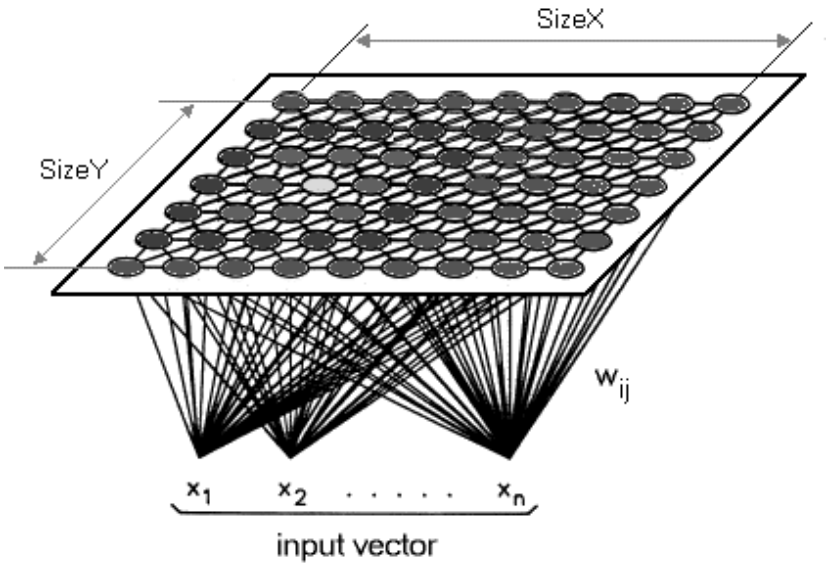
Višeslojni perceptron (eng. *MLP – multilayer perceptron*) je vrsta potpuno povezana neuronske mreže, podijeljene u najmanje 3 sloja (ulazni, izlazni i najmanje 1 skriveni). Razlika u odnosu na perceptron je ta da ima skriveni sloj, odnosno slojeve, slika 24, što implicitno znači da se može koristiti i za nelinearne probleme. Važno je napomenuti da za ulaz uzima jednodimenzionalne podatke, točnije vektore. [17]



Slika 24 MLP mreža

2.3.6. Samoorganizirajuća mapa značajki

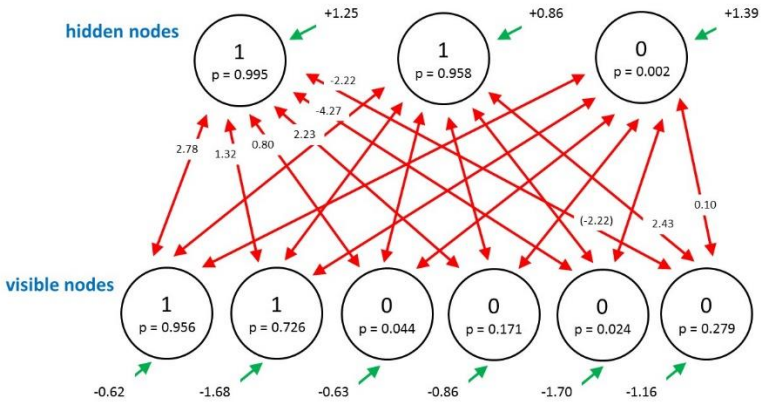
Samoorganizirajuća mapa značajki (eng. *SOM – self-organizing map*) je vrsta neuronske mreže koji ima samo ulazni i izlazni sloj, koji je zapravo mapa značajki, slika 25. Koristi se za pretvaranje višedimenzionalnih mapa podataka u mape nižih dimenzije metodama klasteriranja i mapiranja. [18]



Slika 25 SOM mreža

2.3.7. Ograničeni Boltzmannov stroj

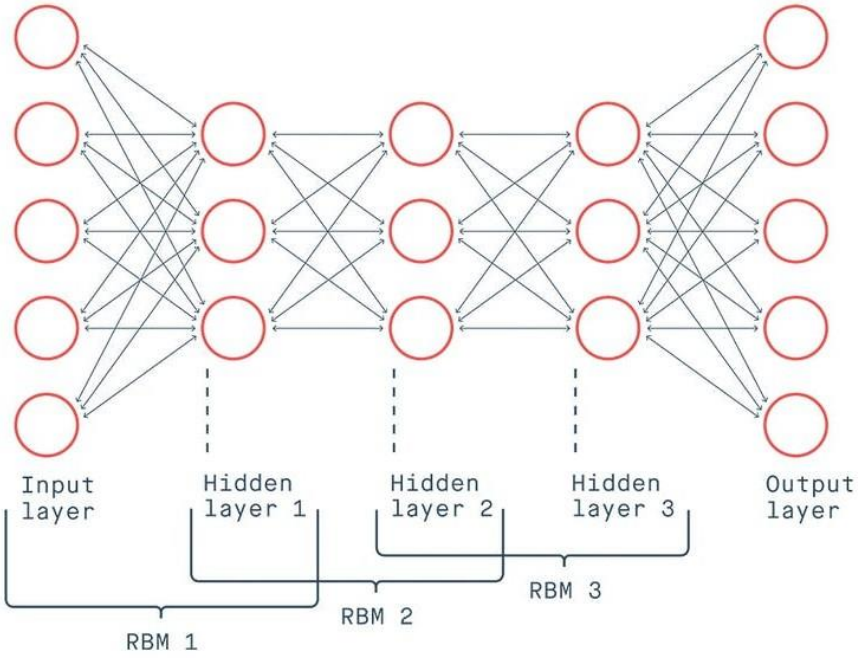
Ograničeni Boltzmannov stroj (eng. *RBM – restricted Boltzmann machine*) je generativna stohastička neuronska mreža koja može naučiti distribuciju vjerojatnosti preko skupa ulaza. Imaju samo 2 sloja – vidljivi i skriveni. Čvorovi pojedinog sloja nisu povezani, ali su svi čvorovi jednog sloja povezani sa svim čvorovima sljedećeg sloja. [20] RBM ima dvosmjerne veze, funkcionirajući tako da od vidljivog sloja prema skrivenim slojevima se podaci analiziraju, a zadnjeg sloja prema prvome se generiraju.



Slika 26 Ograničeni Boltzmannov stroj

2.3.8. Mreža dubokog uvjerenja

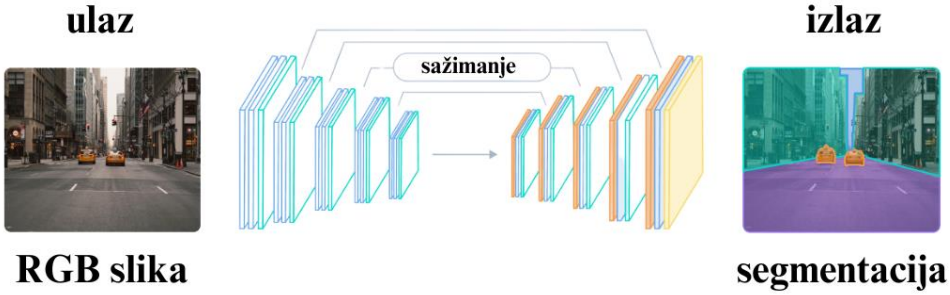
Mreža dubokog uvjerenja (eng. *DBN – deep belief network*) je grafički generativni model koji sadrži više slojeva stohastičkih, latentnih varijabli. [20] Ova vrsta je napravljena kao rješenje problema kod treninga tradicionalnih neuronskih mreža poput sporog učenja, potrebe za puno setova podataka itd. Arhitektura DBN se temelji na RBM.



Slika 27 DBN mreža

2.3.9. Autoenkoderi

Autoenkoderi su posebna vrsta neuronske mreže gdje su ulaz i izlaz isti. [21] Arhitektura je oblika grla boce (eng. *bottleneck*) te postoje 3 dijela enkoder, kod i dekoder. Ovi slojevi služe za raščlambu podataka, od kojih se čuvaju najbitnije značajke i na kraju stvara komprimirani prikaz ulaz.



Slika 28 Autoenkoder

3. RAČUNALNI VID

Računalni vid je područje umjetne inteligencije u kojem se računalu daje sposobnost da izdvoji podatke i kontekst iz slika i videa. Može se reći i da omogućava računalima da vide i razumiju vizualne prikaze dobivene kamerama ili senzorima.

Osnovna zadaća računalnog vida je prepoznavanje uzoraka, odnosno značajki, na temelju čega se vrše svi ostali zadaci koji omogućavaju razumijevanje slikovnog prikaza. Jedan od načina za istrenirati računalo, odnosno neuronsku mrežu, da prepoznaje uzorke je izvršiti algoritme nad označenim podacima. Primjenom algoritama, računalo će naučiti analizirati boje, oblike, udaljenosti između oblika i ostale značajke na slikama.

Duboko učenje oslanja se na neuronske mreže, koje mogu riješiti svaki problem koji se može prikazati primjerima. [22] Kada se označeni primjeri podvrgnu neuronskoj mreži, ona će moći izlučiti česte uzorke među tim primjerima i pretvoriti ih u matematičke funkcije koje će se kasnije koristiti za klasifikaciju budućih primjera.

U računalnom vidu postoje 3 osnovna koraka: dobivanje, procesiranje i razumijevanje slike. Slikovni prikazi se mogu dobivati putem slika, videa ili 3D tehnologija. Zatim se dobiveni prikaz procesira pomoću nekog modela računalnog vida, koji je najčešće treniran na raznim primjerima slika. Konačni korak je iterativan koji prepoznaje ili klasificira sliku.

U većini slučajeva, algoritmi dubokog učenja ovise o velikoj količini označenih podataka za treniranje i podešavanju parametara (npr. vrsta i broj slojeva). Usporedivši s prethodnim vrstama strojnog učenja, duboko učenje je lakše i jednostavnije za razvoj i implementaciju.

3.1. Razvoj

Prvi eksperimenti vezani za računalni vid sežu iz 1950-ih godina, gdje su se neuronske mreže koristile za prepoznavanje rubova objekata i za sortiranje jednostavnih objekata u kategorije. U 1970-ima, prva komercijalna aplikacija za računalni vid je mogla interpretirati napisani ili natipkani tekst koristeći optičko prepoznavanje znakova (eng. *OCR – optical character recognition*). Razvojem interneta 1990-ih, pojavljuje se i sve veći broj lako dostupnih skupova podataka, pa tako i slika. To je omogućilo razvoj raznih softvera računalnog vida, poput programa za prepoznavanje lica. [23]

Računalni vid se danas razvija brže nego ikada. Postoje mnogi razlozi za to. Mobilna tehnologija s ugrađenim kamerama je omogućila prikupljanje velike količine slika i videa,

potrebni hardver je danas vrlo dostupan i cjenovno prihvatljiv te su razvijeni mnogi algoritmi koji učinkovitije koriste i softver i hardver. U odnosu na prijašnja vremena, korištenje raznih softvera je također pojednostavljeno i vrlo ga je lako koristiti, čak i bez duge i opširne naobrazbe.

Razvoj je očitovan i rezultatima – stope točnosti prepoznavanja i klasifikacije su porasle s 50% na 99% u periodu kraćem od jednog desetljeća, a vrijeme potrebno za te operacije se sve više smanjuje, čineći računalne sustave moćnije u odnosu na ljude u mnogim zadacima računalnog vida. Iako je područje računalnog vida i umjetne inteligencije znatno napredovalo, posebice u zadnjem desetljeću, još uvijek postoji mnogo prostora za napredak.

3.2. Primjena i izazovi

Područje računalnog vida je postiglo značajni napredak u zadnjih nekoliko godina te se sve više primjenjuje u raznim industrijama. Korištenjem matematičkih i statističkih modela, računala su dostigla mogućnosti za obavljanje zadataka za koje se nekoć smatralo da je isključivo u domeni čovjeka. [24]

Jedna od najaktualnijih primjena je u autonomnim vozilima. Autonomna vozila se oslanjanju na sustave računalnog vida za identifikaciju i navigaciju u okolini s preprekama, za slijeđenje cestovnih oznaka i izbjegavanje sudara. Osim za autonomna vozila, u prometu se može koristiti za nadzor. Nadzorom prometa se može pratiti promet, smanjiti gužva i povećati sigurnost. U slučaju kršenja pravila, kod nekih sustava je moguće automatsko propisivanje kazni.

Također se koristi i u robotici za navigaciju kroz kompleksne okoline i za izvođenje zadataka za koje je potrebna analiza i interpretacija slike. Osim za navigaciju i izvođenje zadataka, može se koristiti i za interakciju s ljudima.

Veliki napredak je prisutan i u medicini, gdje se povećala točnost slika medicinskih pretraga, a razvijeni su i algoritmi za analizu slika pretraga pomoću kojih je moguće prepoznati razne anomalije koje mogu promaknuti ljudskom oku ili otkriti rane znakove poremećaja ili bolesti. Također se primjenjuje kao pomoć pri medicinskim zahvatima.

Računalni vid se može koristiti u marketingu za stvaranje interaktivnih oglasa i u turizmu za poboljšanje iskustva turista uz svakojaki digitalni sadržaj i informacije. Sve češća je i primjena u agrikulturi, gdje se prati stanje usjeva i određuje prisutnost štetočina i bolesti.

Česta primjena je i kod prepoznavanja lica. Analizom crta lica može se ustanoviti o kome se radi i dopustiti pristup, kao kod otključavanja mobitela, ili djelovati na temelju prepoznatih osoba, kao kod prepoznavanja osumnjičenika na sigurnosnim snimkama.

Još jedno od područja primjene je sigurnost i nadzor u stvarnom vremenu. Postoje kamere s ugrađenim softverima koji mogu prepoznati prijetnju i zatim obavijestiti sigurnosno osoblje.

Osim računalnog vida, danas se razvijaju i mnoga druga područja poput raznih primjena metoda umjetne inteligencije i interneta stvari (eng. *IoT – internet of things*). Njihovim spajanjem se mogu dobiti inteligentni sustavi koji će davati još točnije rezultate i proširiti primjenu.

Postoje tehnologije proširene stvarnosti (eng. *XR – extended reality*), a razlikuju se virtualna stvarnost (eng. *VR – virtual reality*), dorađena stvarnost (eng. *AR – augmented reality*) i mješovita stvarnost (eng. *MR – mixed reality*). Virtualna stvarnost je ne najčešći oblik proširene stvarnosti u kojem se uspostavlja interakcija u potpuno digitalnim, umjetno stvorenim svjetovima putem VR naočala. Dorađena stvarnost u stvarni svijet dodaje digitalno stvorene elemente, a za njihovo korištenje je dovoljan pametni telefon. Mješovita stvarnost se odnosi na kombinaciju prethodne dvije stvarnosti, u kojoj je moguće uspostaviti interakciju i s digitalnim i sa stvarnim objektima. Ove tehnologije su široko primjenjive, od raznih industrija do društvenih mreža.

Prema navedenim primjerima, vidljivo je kako je računalni vid sveprisutan i u koliko se raznih industrija i područja života koristi, u profesionalne i privatne svrhe.

Iako je napredak bio izuzetan, postoje još mnogi izazovi kojima se treba posvetiti. Jedan od najvećih izazova su podaci, velika količina s velikom kvalitetom prema kojima modeli mogu učiti. Stvaranje ovakvih skupova može biti i skupo i vremenski zahtjevno.

Osim potrebe za podacima, javlja se i potreba za raznolikošću podataka. Ako se među trening podacima nalazi nedovoljno različitih uzoraka, sustav može postati pristran pojedinoj rasi, spolu, kulturnoj pripadnosti i sl., i davati netočne rezultate.

Još jedan od izazova je interpretabilnost. Kako algoritmi postaju kompleksniji i sofisticiraniji, sve je teže razumjeti kako funkcioniraju i donose zaključke. No da bismo vjerovali algoritmu da donosi ispravne odluke, potrebno ga je razumjeti. Iz navedenog razloga se danas istražuje objašnjiva umjetna inteligencija (eng. *XAI – explainable artificial intelligence*), kod koje se promatra mogućnost sustava umjetne inteligencije da objasni kako je došao do danog zaključka.

Također se javlja problem privatnosti. Uz sve veću prisutnost kamera u raznim uređajima, raste zabrinutost oko prikupljanja i upotrebe podataka. Iako veći broj podataka pomaže unaprijediti

modele, važno je razviti metode koje štite pojedinca i ne dovode do zlouporabe njegovih podataka, a istovremeno omogućavaju učinkovitost. Potencijalno rješenje ovog problema je rubno računalstvo (eng. *edge computing*), tijekom kojeg se procesiranje podataka odvija blizu izvora umjesto slanja na središnji server. To smanjuje potrebu za prijenosom velikog broja podataka, čineći ga istovremeno učinkovitijim i sigurnijim.

Među izazovima računalnog vida nalazi se i potreba za robusnim i učinkovitim algoritmima koji mogu podnijeti toliku količinu slikovnih podataka u stvarnom vremenu uz nisku latenciju i visoku točnost.

U mnogim primjenama postoji potreba za procesiranje u stvarnom vremenu, poput autonomnih vozila. Broj podataka je velik jer sami slikovni prikaz treba biti dovoljne rezolucije kako bi se omogućila sigurna vožnja, ali to usporava algoritam. Postizanje brzih, ali točnih algoritama može biti teško za postići. Istražuju se razna rješenja za ovaj problem, koja uključuju upotrebu posebnih hardverskih komponenti i razvoj novih algoritama, specifičnih za primjenu u stvarnom vremenu.

Jedan od problema koji je izuzetno važan je problem sigurnosti. Premda su sustavi najčešće razvijani u kontroliranoj okolini, teško je pretpostaviti hoće li dobro reagirati na nekontroliranu okolinu. To je posebno važno za primjene u stvarnom vremenu. Takvi sustavi se moraju testirati u raznim uvjetima i za različite situacije kako bi bili sigurni za korisnike i cijelu okolinu.

3.3. Softver

Svi softverski programi su izgrađeni u nekom od programskih jezika u nekom operacijskom sustavu. Programi se mogu razlikovati između operacijskih sustava, no mogu biti i primjenjivi za sve. Najpoznatiji su Windows, Linux i macOS. U njima se koriste uređivači teksta (eng. *editors*) za pisanje koda algoritama koji će izvršavati operacije. Uređivači teksta mogu biti upotrebljivi za više programskih jezika (npr. Visual Studio Code) ili specifični za jedan (npr. PyCharm). Kod pisanja koda, koriste se ekstenzije programskih jezika, odnosno tzv. knjižnjice ili paketi, čijom su upotrebom omogućene dodatne funkcije.

Osim instalacije svih potrebnih navedenih softvera, moguće je i upotreba oblaka (eng. *cloud*). Računalstvo u oblaku (eng. *cloud computing*) je kada se umjesto dostupnog hardvera koristi mreža udaljenih servera te je dostupno putem neke od platformi oblaka (npr. Google Cloud Platform).

Za potrebe računalnog vida često se koriste programski jezici C/C++ i Python, a najpoznatije knjižnice su OpenCV, TensorFlow i Keras. Također se mogu koristiti i već programirani modeli, čija je uporaba još jednostavnija.

3.4. Hardver

Operacije koje koriste programi računalnog vida zahtijevaju hardverske komponente s dovoljno resursa za njihovo izvođenje. Zahtjevi ovise o samoj funkciji programa.

Jedna od komponenti koja je potrebna za svaki oblik programa računalnog vida je jedinica za procesiranje (eng. *processing unit*). Najčešće su korišteni procesori (eng. *CPU – central processing unit*), grafičke kartice (eng. *GPU – graphical processing unit*) ili kombinacija navedenih uz mogućnost dodavanja još nekih komponenti. Još se javljaju i programabilni sklopovi s vratima (eng. *FPGAs – field programmable gate arrays*), aplikativno specifični integrirani sklopovi (eng. *ASICs – application-specific integrated circuit*), mikrokontrolerske pločice i ugradbeni računalni sustavi (eng. *embedded systems*). Svaka opcija ima prednosti i mane u uporabi, a usporedba najčešće korištenih dana je u tablici 2. [25]

Tablica 2 Usporedba

	CPU	**GPU**	**CPU+GPU**	**ostalo**
performanse	dobre	bolje	najbolje	najsporije
jednostavnost programiranja	lako	teško	vrlo teško	lako
skalabilnost	dobra	ograničena	najbolja	nije skalabilno
cijena	umjerena	umjerena	skupa	jeftina
paralelna obrada	dobra	najbolja	najbolja	najlošija
faktor oblika	veliki	mali	veliki	najmanji

Performanse jedinice za procesiranje podrazumijevaju brzinu izvođenja zadataka. Skalabilnost se odnosi na nastavak funkcioniranja pri promjeni količine korištenih podataka u programu. Kod paralelne obrade se veliki problemi dijele na manje koji se obrađuju istovremeno, a ovisno o procesorskoj jedinici razlikuje se veličina manjih problema.

Tablica 3 Zahtjevi hardvera

SPECIFIKACIJA HARDVERA	ZAHTJEVI
memorija	- minimalno: 8 GB - preporučeno: 16 GB
CPU	- minimalno: 2 jezgre - preporučeno: 4 jezgre
GPU	- minimalno: 8 GB VRAM - preporučeno: 16 GB VRAM
pohrana	- minimalno: 30 GB slobodno

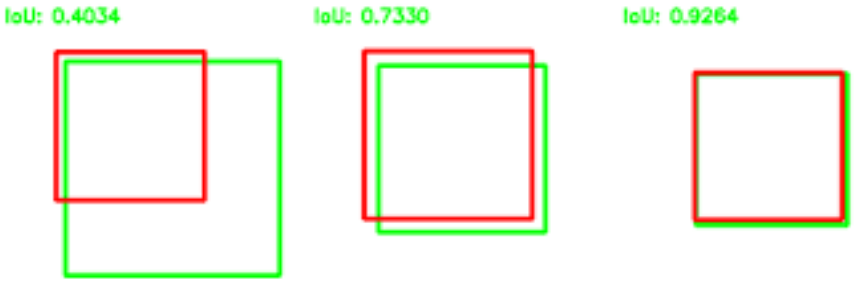
Premda je u programima računalnog vida potreba visoka kvaliteta grafike, često se koriste grafičke kartice različitih specifikacija. Glavna razlika između potrošačkih grafičkih kartica i onih za virtualizaciju je ta da grafičke kartice s virtualizacijom mogu učitati slike veće rezolucije na ulazu u model, no ne postižu značajnu razliku u brzini.

3.5. Vrednovanje modela

Svaki model se vrednuje prema nekoliko mjera koji pokazuje koje su bolje, a koje lošije strane pojedinog modela. Osnovne mjere prema kojima se model vrednuje su točnost, preciznost i odziv. [26] Te mjere se dobivaju usporedbom označenih slika koje prikazuju točne podatke i predviđenih rezultata koje daje model.

Kod klasifikacije, moguća su 4 različita ishoda: istinski pozitivno (eng. *TP – true positive*), istinski negativno (eng. *TN – true negative*), lažno pozitivno (eng. *FP – false positive*) i lažno negativno (eng. *FN – false negative*). Oni su svrstani u matricu konfuzije (eng. *confusion matrix*) prikazanu tablicom 4. Navedeni ishodi se dobivaju uspoređujući prijedloge područja objekta unutar graničnih okvira (eng. *bounding box*) s temeljnom istinom (eng. *ground truth*) i određeni su kroz mjeru zvanu presjek iznad unije (eng. *IoU – intersection over union*). Izračunava se kao omjer područja preklapanja i područja unije prema jednadžbi (7), slika 29.

$$IoU = \frac{područje\ preklapanja}{područje\ unije} \tag{7}$$



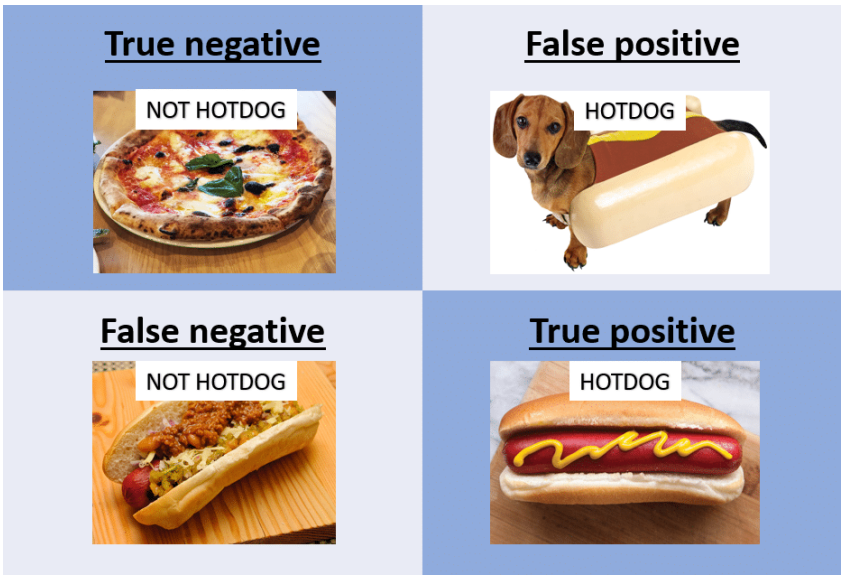
Slika 29 Primjeri različitih iznosa IoU

Kako će model dalje klasificirati objekte ovisi o pragu (eng. *threshold*) koji je proizvoljno definiran. Na temelju tih vrijednosti, granični okvir se može svrstati u jednu od opcija u matrici konfuzije (eng. *confusion matrix*). Najčešće se koristi prag iznosa 0,5. Što je veći IoU iznos, to je bolje predviđanje.

Tablica 4 Matrica konfuzije

		predviđeno	
		P	N
temeljna istina	P	TP	FN
	N	FP	TN

Matrica konfuzije na primjeru je vidljiva na slici 30.



Slika 30 Primjer matrice konfuzije

3.5.1. Točnost

Ispravno klasificirani primjeri se nalaze na dijagonali matrice konfuzije. Točnost (eng. *accuracy*) se računa kao omjer ispravno klasificiranih primjera i ukupne količine primjera, jednadžba (8), te prikazuje postotak točno klasificiranih primjera u odnosu na sve primjere.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

3.5.2. Preciznost

Preciznost (eng. *precision*) se odnosi na kvalitetu pozitivnih predviđanja modela, odnosno prikazuje postotak točno klasificiranih primjera među svim pozitivno klasificiranim primjerima. Računa se kao omjer istinski pozitivnih primjera u odnosu na ukupne pozitivne primjere, jednadžba (9).

$$P = \frac{TP}{TP + FP} \tag{9}$$

Kod vrednovanja modela postoje dvije prosječne preciznosti: AP i mAP. AP (eng. *average precision*) prikazuje prosječnu preciznost jedne klase, a mAP (eng. *mean average precision*) prikazuje prosječnu preciznost za sve klase.

3.5.3. Odziv

Odziv (eng. *recall*) prikazuje sposobnost modela da pronađe sve pripadnike neke klase, odnosno postotak točno klasificiranih primjera u odnosu na sve pozitivne primjere. Prema tome, računa se kao omjer pozitivno označenih primjera i količine ukupnih pozitivnih primjera, jednadžba (10).

$$R = \frac{TP}{TP + FN} \tag{10}$$

3.6. Tradicionalne metode

Kod tradicionalnih metoda, prepoznavanje objekata se vrši u 3 koraka. Prvi korak je prijedlog regija za lokalizaciju objekata na pojedinoj slici u bazi slika. Drugi korak je izdvajanje značajka

prema predloženim regijama, prema metodama SIFT, HOG i sl. Ovisno o korištenoj metodi, rezultati će biti drugačiji jer se svaka metoda koristi za izdvajanje drugačijih značajki. Posljednji korak je klasifikacija. Najpoznatiji klasifikatori su SVM, AdaBoost itd. [27]

Algoritmi za prepoznavanje objekata su navedeni u tablici 5 uz njihovu usporedbu.

Tablica 5 Usporedba tradicionalnih metoda

metoda	učinkovitost prepoznavanja	vrijeme prepoznavanja
HOG+SVM	dobra	dugo
HOG+Cascade	loša	kratko

Još se koristi i DPM algoritam (eng. *deformable parts model*), koji je robusniji kod promjene pozicije, stoga ima veću točnost kod prepoznavanja u kompleksnim slikama, ali je zbog velike količine računanja spor i loš za primjenu u stvarnom vremenu.

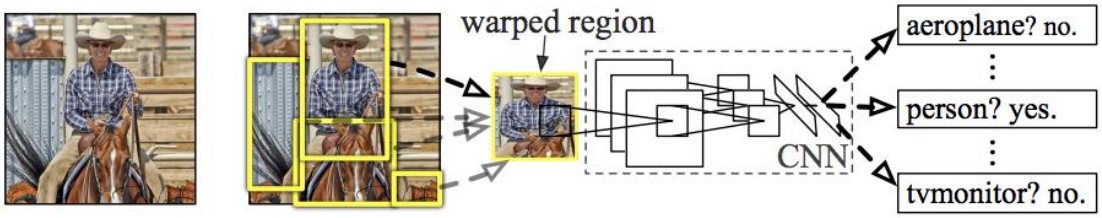
3.7. Dvofazni detektori

Nakon tradicionalnih metoda koje uključuju 3 koraka, pojavili su se dvofazni detektori (eng. *two-stage*) koji su to sveli na 2 koraka ili faze. Prva faza je prijedlog skupa područja za lokalizaciju objekata. Nakon slijedi klasifikacija lokaliziranih objekata. Ovakvi modeli pokazuju visoku točnost, no mogu biti spori. Popularni dvofazni detektori su R-CNN, SPP-Net, Brza R-CNN, Brža R-CNN, Maskirana R-CNN i FPN.

3.7.1. R-CNN

Konvolucijske neuronske mreže temeljene na regiji (eng. *region based convolutional neural networks*) se najčešće koriste pod engleskim akronimom R-CNN. Jedan su od modela koji je omogućio znatan napredak u području računalnog vida. [28]

R-CNN je nadogradnja na običnu CNN, gdje se ispred nje dodaje još jedan korak predlaganja regija (eng. *region proposal*), slika 31. U tom koraku prepoznaju se regije u kojima se nalaze objekti, neovisno o pripadnosti klasi. Nakon što je taj proces izvršen, dobiv se skup kandidata za detekciju (eng. *set of candidate detections*). Slijedi klasična konvolucijska neuronska mreža koja detektira značajke i na temelju toga klasificira objekte, no kao ulaz ne koristi cijelu sliku, već samo predložene regije, što je čini bržom u odnosu na klasične CNN, a rezultati pokazuju i veću točnost.



Slika 31 R-CNN mreža

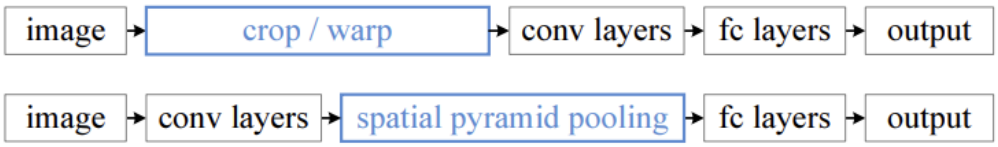
3.7.2. SPP-Net

Do razvoja SPP-Net mreže, u svim prethodnima je postojao problem što ulazna slika za potpuno povezane slojeve mora biti fiksnih dimenzija. [29] To je uzrokovalo probleme u točnosti kod prepoznavanja na slikama ili dijelovima slika različitih dimenzija jer bi slike dobivene nakon konvolucije i sažimanja sadržavale izrezane ili iskrivljene objekte, prilagođene zadanim dimenzijama, slika 32.

Slika 32 Primjer izrezanog i iskrivljenog objekta

Kako bi se eliminirao ovaj problem, predstavljena je nova arhitektura modela s novom vrstom sažimanja, prostornim piramidalnim sažimanjem (eng. *SPP – spatial pyramid pooling*). Ova metoda primjenjuje više operacija sažimanja koji daju izlaze različite veličine, no oni se kombiniraju i potom stvaraju izlaz fiksne veličine koji se prosljeđuje potpuno povezanim slojevima.

Arhitektura izgleda tako da se između konvolucijskih slojeva i potpuno povezanih slojeva dodaje sloj prostornog piramidalnog sažimanja, odnosno SPP sloj, slika 33.



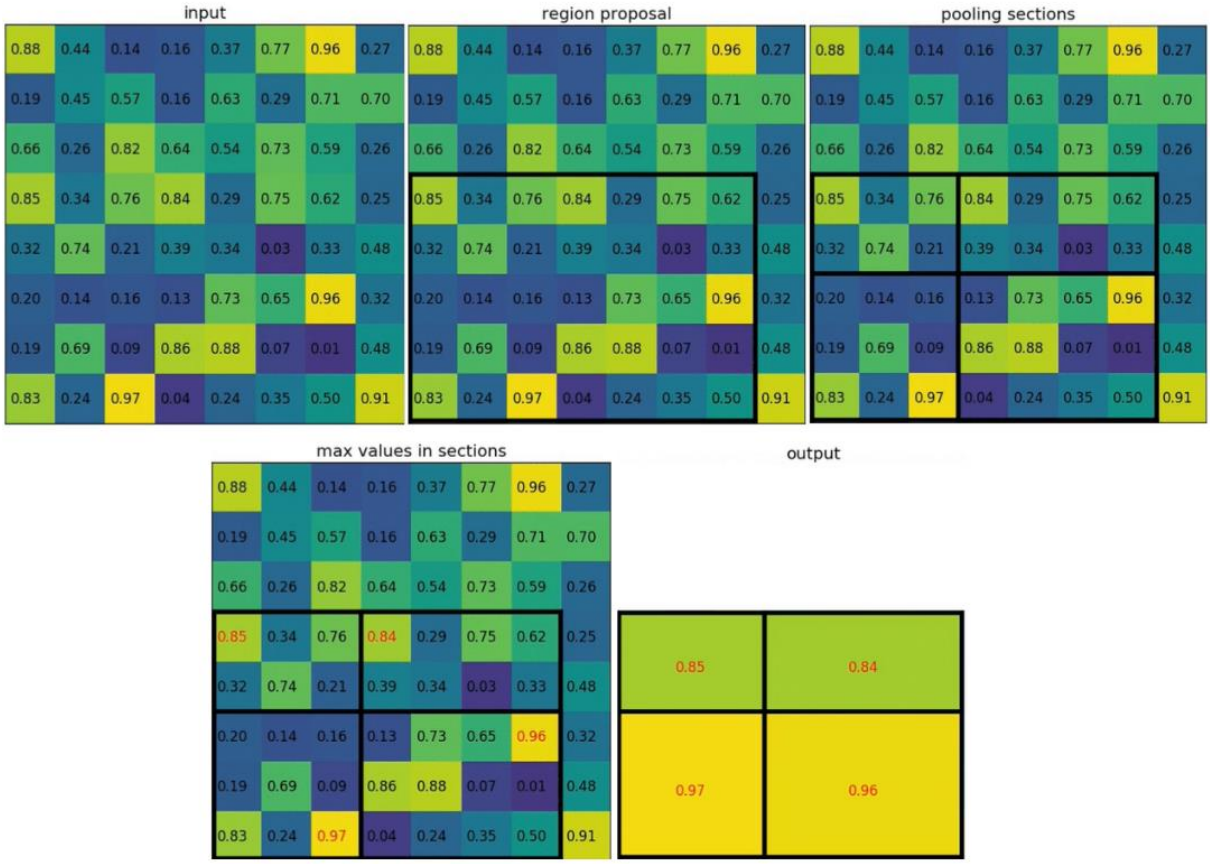
Slika 33 Usporedba R-CNN i SPP-net mreže

Ova mreža je brža jer se konvolucija provodi samo jednom na cijeloj slici, dok se kod R-CNN provodi za svaki prijedlog regije. Osim što je brža, daje i bolje rezultate.

3.7.3. Brza R-CNN

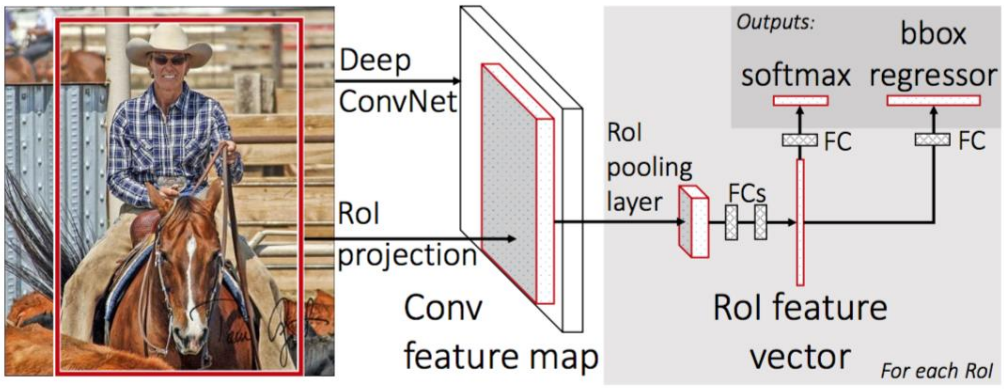
R-CNN mreže su spore jer se cijeli proces vrši za svaki prijedlog regije. Iako su SPP-Net mreže ubrzale vrijeme testiranja, vrijeme treninga je i dalje ostalo sporo. Kao rješenje ovih problema, predložena je brza R-CNN (eng. *fast R-CNN*).

Brza R-CNN predlaže novu vrstu sažimanja zvanu RoI sažimanje (eng. *region of interest pooling*). Sažimanje se vrši na regijama interesa (eng. *RoI – region of interest*), koje su podijeljene na dijelove proporcionalne regiji, no ne moraju biti istih veličina, slika 34. Na svakom dijelu se zatim vrši sažimanje maksimalnih vrijednosti. [30]



Slika 34 RoI sažimanje

Mreža na ulazu uzima cijelu sliku gdje prvo nekoliko konvolucijskih slojeva i slojeva sažimanja maksimalnih vrijednosti stvara mapu značajki s prijedlogom regija interesa. Zatim slijedi RoI sažimanje na regijama interesa te se stvara RoI vektor značajki, koji je ulaz u potpuno povezane slojeve. U tim posljednjim slojevima se stvaraju granični okviri i vrši se klasifikacija. Prikaz mreže dan je na slici 35.



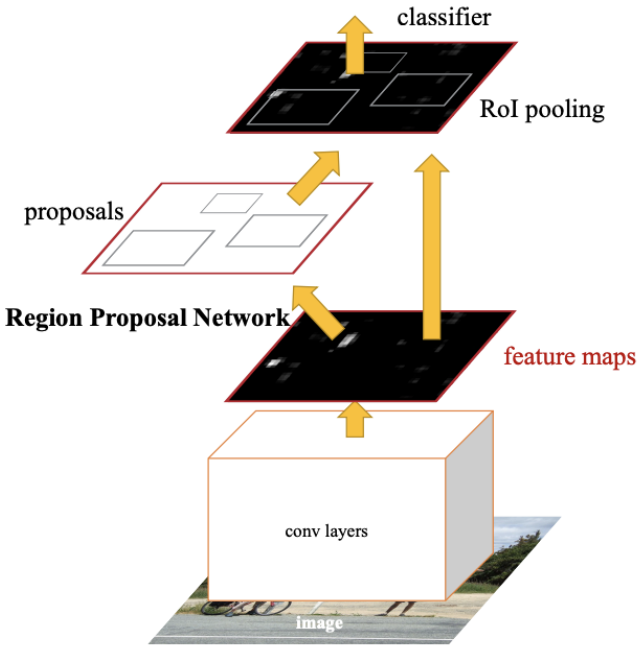
Slika 35 Arhitektura brze R-CNN mreže

3.7.4. Brža R-CNN

Brža R-CNN (eng. *faster R-CNN*) je mreža koja koristi drugačiji algoritam za pronalaženje prijedloga regija u odnosu na prethodne. Taj algoritam je zapravo posebna mreža koja predviđa prijedloge regije na mapi značajki dobivene od klasične CNN, a zove se mreža za predlaganje područja (eng. *RPN – region proposal network*).

Arhitektura brže R-CNN izgleda tako da se na početku nalaze konvolucijski slojevi koji uzimaju cijelu sliku na ulazu, a daju mapu značajki, slika 36. Na mapu značajki se primjenjuje RPN mreža koja daje prijedloge regija te se na tim regijama provodi RoI sažimanje. Slijede potpuno povezani slojevi koji stvaraju granične okvire i klasificiraju objekte. [31]

Može se reći da je ovo kombinacija brze R-CNN i RPN mreže.

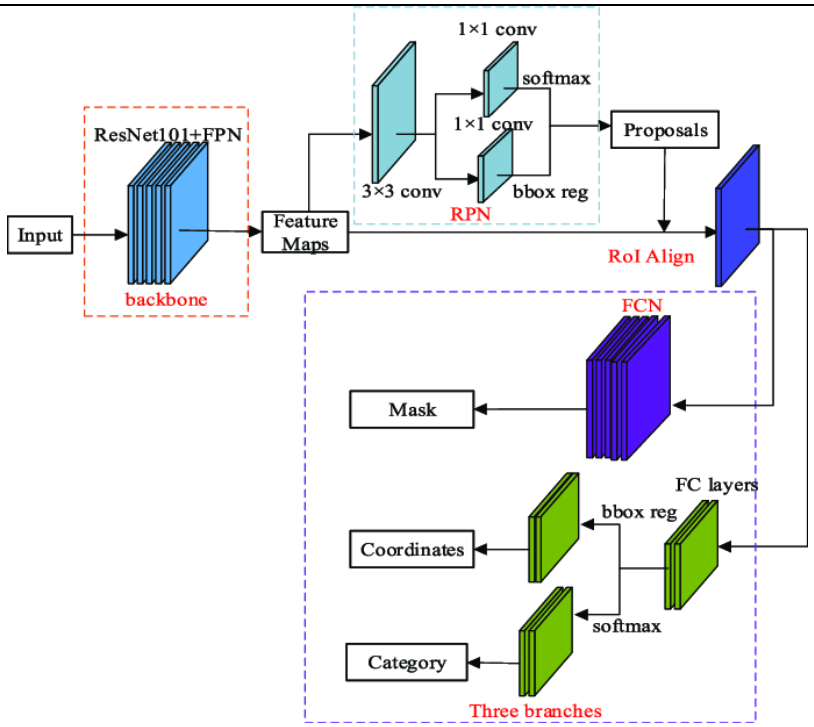


Slika 36 Brža R-CNN mreža

3.7.5. Maskirana R-CNN

Maskirana R-CNN (eng. *mask R-CNN*) je mreža koja u arhitekturu brže R-CNN dodaje stvaranje maske na svakoj regiji interesa pomoću segmentacije. Grafičko rješenje je maska oblika kao i objekt koja se nalazi unutar granice graničnog okvira. [32]

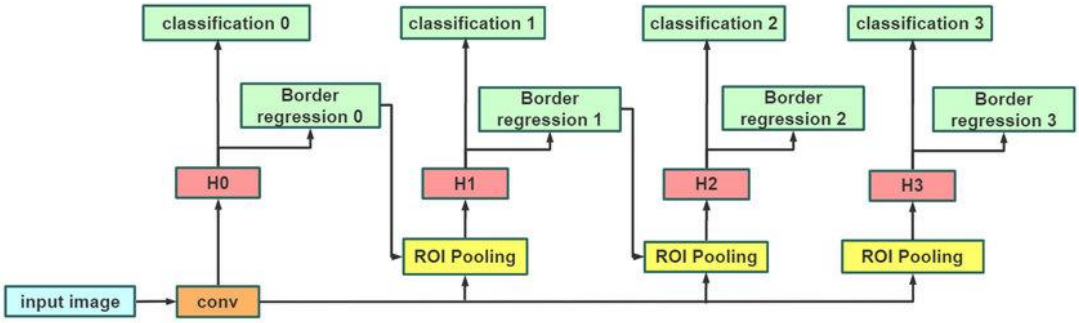
Kao što je već spomenuto, arhitektura liči na onu brže R-CNN – prva na redu je CNN, zatim RPN mreža, no onda slijedi promjena, slika 37. Uz potpuno povezane slojeve za stvaranje graničnih okvira i klasifikaciju, paralelno se nalazi još jedna mreža za stvaranje binarne maske za svaki RoI. Ta mreža je po vrsti potpuno konvolucijska mreža (eng. *FCN – fully convolutional network*) što znači da je to klasična s CNN s potpuno povezanim prvim slojem.



Slika 37 Maskirana R-CNN

3.7.6. Kaskadna R-CNN

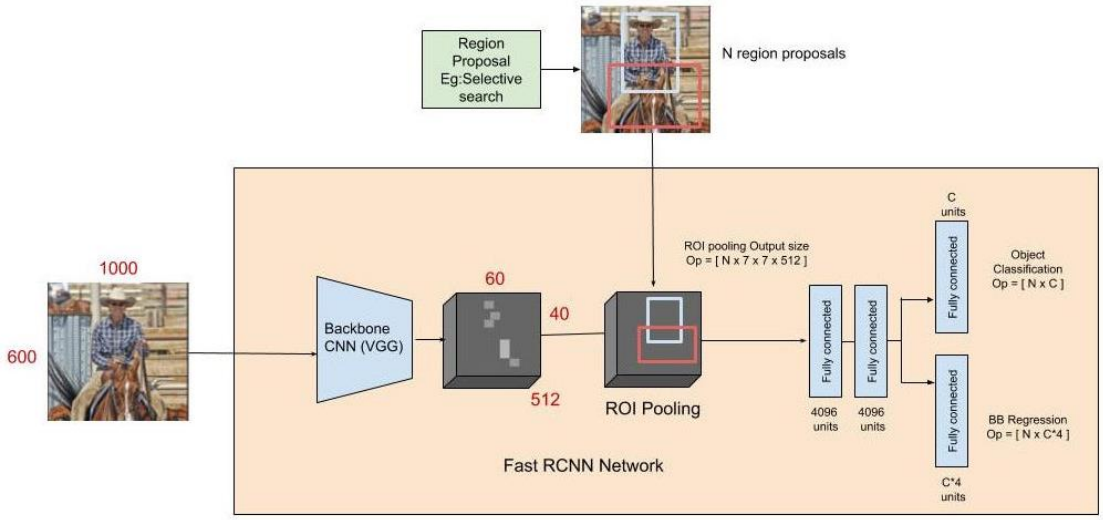
Kaskadna R-CNN (eng. *cascade R-CNN*) je vrsta R-CNN mreže specifična za zadatke prepoznavanja objekata. Može se smatrati višefaznim proširenjem R-CNN mreže. Na početku se nalaze CNN i RPN mreže. Zatim dolazi do sažimanja i nakon toga do klasifikacije i predviđanja graničnih okvira. No tu nije kraj jer postoji više faza, odnosno kaskada, slika 38. [33] U svakoj sljedećoj kaskadi dolazi do prepravljanja graničnih okvira prema presjeku iznad unije.



Slika 38 Kaskadna R-CNN

3.7.7. FPN

Piramidalna mreža značajki (eng. *FPN – feature pyramid network*) je mreža koja se sastoji od RPN i brze R-CNN mreže, slika 39. Prva faza ja konvolucijska mreža koja stvara mape značajki različitih veličina i raspoređuje ih u hijerarhiju. [34] Na taj način se postiže da se u zadnjem sloju nalaze mape značajki s najizraženijim značajkama. U drugoj fazi, za svaki sloj hijerarhije, se provodi klasifikacija. Zbog različite veličine slojeva i izraženosti značajki, ova mreža je vrlo pogodna za klasifikaciju objekata slike različitih veličina, no može se koristiti samo kod testiranja.



Slika 39 FPN mreža

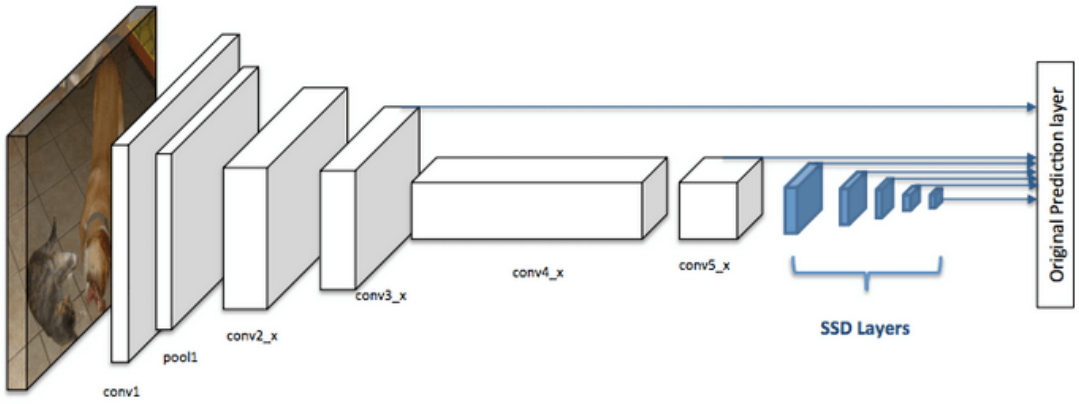
3.8. Jednofazni detektori

Jednofazni detektori su još više ubrzali proces tako da su 2 koraka sveli na 1. Konstruirani su tako da istovremeno daju prijedlog graničnog okvira i vjerojatnost pripadnosti klasi za više klasa u jednom prolazu kroz mrežu. Zbog toga se često primjenjuju u zadatke koji se odvijaju u stvarnom vremenu. Iako zbog jednog prolaza kroz mrežu postoji mogućnost manje točnosti, neki modeli su uspjeli postići zadovoljavajuću točnost te postigli ukupnu nadmoćnost nad dvofaznim detektorima.

3.8.1. SSD

Jednonamjenski detektor (eng. *SSD – single shot detector*) je vrsta mreže koja na početku ima baznu mrežu koja je predtrenirana na velikom skupu slika za klasifikaciju. Nakon toga slijede

dodatni slojevi koji su odgovorni za prepoznavanje objekta različitih veličina i obično su to konvolucijski slojevi i slojevi sažimanja, slika 40. [35]

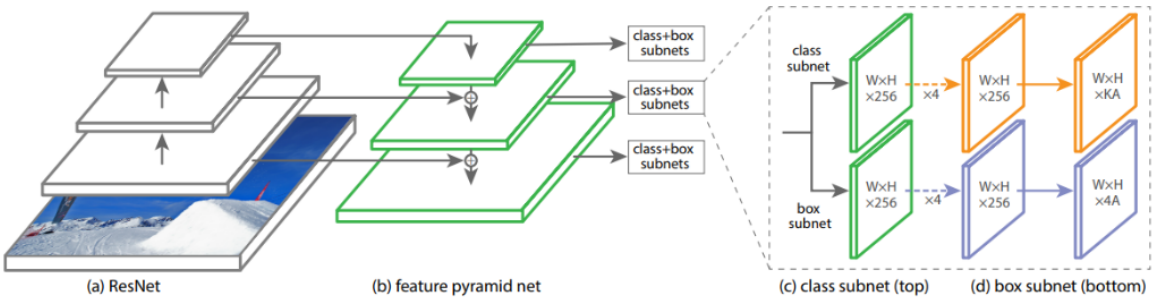


Slika 40 SSD detektor

Ovi modeli su vrlo brzi i učinkoviti. Prema tome, pogodni su za primjenu u stvarnom vremenu. Točni su, no ne koliko i R-CNN arhitekture te mogu biti osjetljive na velike promjene u veličini objekata.

3.8.2. Retinanet

Retinanet na početku ima FPN mrežu za stvaranje mape značajki na prethodno spomenut način te se izlazni sloj toga dijela podvrgava klasifikacijskoj i regresijskoj mreži za predviđanje graničnih okvira i klasifikaciju, slika 41. [36]



Slika 41 Retinanet

3.8.3. YOLO

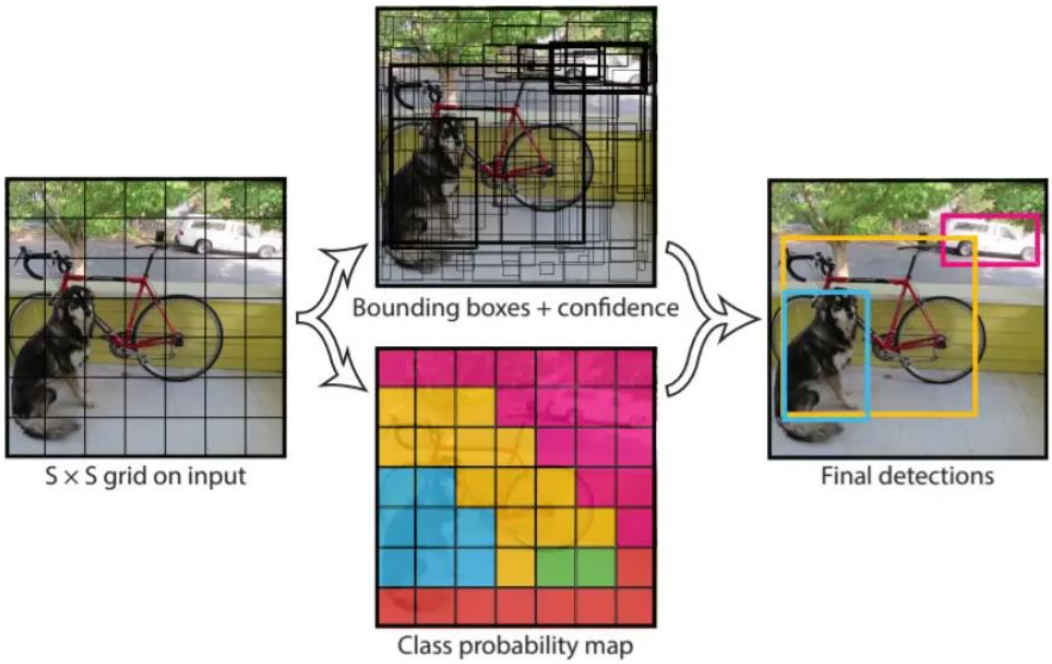
YOLO (eng. *you only look once*) je algoritam za detekciju objekata predstavljen 2016. godine. Taj algoritam nudi značajan napredak u odnosu na R-CNN po pitanju prepoznavanja objekata u stvarnom vremenu. Prvi je model za primjenu u stvarnom vremenu s *end-to-end* principom. [37]

Ovaj algoritam se razvija velikom brzinom, tako da je do danas izašlo mnogo verzija, a posljednja je YOLOv8, koja se koristi i u ovom radu.

Arhitektura mreže se sastoji od 3 dijela, a to su *backbone*, *neck* i *head*. *Backbone* je predtrenirana konvolucijska neuronska mreža koja za ulaz uzima sliku te na izlazu daje mapu niskih, srednjih i visokih značajki. *Neck* za ulaz uzima izlaznu mapu značajki mreže *backbone* i spaja ih koristeći *PAN* mrežu (*path aggregation network*), kojoj je cilj zabilježiti dugoročne ovisnosti u slikama. Dugoročne ovisnosti u slikama podrazumijevaju korelacije među pikselima ili značajkama koje nisu međusobno prostorno bliske, a njihovo uočavanje je krucijalno za prepoznavanje kompleksnih uzoraka.

3.8.3.1. YOLOv1

YOLOv1 je jednofazni detekcijski model koji je nastao unapređivanjem standardnog R-CNN detekcijskog mehanizma s bržim i boljim performansama. U ovoj verziji problem prepoznavanja je postavljen kao regresija sa zadatkom predviđanja graničnih okvira i vjerojatnostima klase sa samo jednim prolaskom slike, slika 42. [38]



Slika 42 Rad YOLOv1 modela

Slika je podijeljena prema mreži te se izračunavaju bodovi pouzdanosti (eng. *confidence scores*) i granični okviri za svaki pojedini dio mreže u kojem postoji mogućnost da se nalazi objekt koji pripada nekoj klasi. Ako postoji mogućnost prisutnosti objekta, algoritam uspoređuje objekt s

postojećim klasama kako bi odredio kojoj pripada, i kao rezultat daje ukupni postotak pripadnosti i granični okvir.

U odnosu na R-CNN, YOLOv1 ima bolju prosječnu preciznost (mAP, eng. *mean average precision*) u iznosu od 63,4 i brzinu zaključivanja (eng. *inference speed*) u iznosu od 45 fps na setu podataka *Pascal Visual Object Classes 2007*.

3.8.3.2. YOLOv2

Ova verzija algoritma može prepoznati preko 9000 kategorija. Novitet su sidreni okviri (eng. *anchor boxes*), to su predtrenirani granični okviri priori (eng. *priors*) koji se koriste kako bi se precizno odredio idealan položaj objekta.

Također je uveden presjek nad unijom (IoU, eng. *intersection over union*) za predviđeni granični okvir u odnosu na sidreni okvir; ako IoU prijeđe graničnu vrijednost, model stvara predviđanje.

Na setu podataka *VOC 2007*, YOLOv2 postiže mAP 76,8 pri 67 fps.

3.8.3.3. YOLOv3

U verziji YOLOv3 postignute su još bolje performanse, s preciznošću mAP 28,2 pri 22 milisekunde. Za predviđanje klasa je za *backbone* dio modela korišten Darknet-53 s logističkim klasifikatorima umjesto softmax aktivacijske funkcije i BCE greške (eng. *binary cross-entropy*).

Darknet-53 je konvolucijska neuronska mreža koja koristi uzastopne 3x3 i 1x1 slojeve, a ima ih 53. [39] Uspoređujući s mrežama ResNet-101 i ResNet-152, Darknet-53 ima iste ili bolje performanse i brža je od obje. Također ima i najviši iznos FLOPS-a (eng. *floating point operations per second*), što je mjera računskih performansi, odnosno koju količinu *floating-point* operacija računalni sustav može odraditi u jednoj sekundi. Premda ima viši iznos FLOPS-a, znači da bolje koristi resurse GPU-a.

3.8.3.4. YOLOv4

YOLOv4 predstavlja 2 nova koncepta: vreće besplatnih stvari (eng. *BoF - bag of freebies*) i vreće posebnosti (eng. *BoS - bag of specials*).

BoF je grupa tehnika koje povećavaju preciznost bez utjecaja na brzinu zaključivanja, dok BoS metode znatno povećavaju preciznost s malim povećanjem na brzinu zaključivanja. BoF uključuje metoda poput CutMix, CutOut i Mixup povećanja podataka te novu metodu mozaika.

Mozaik augmentacija (eng. *mosaic augmentation*) koristi 4 različite trening slike koji omogućuju model s boljim kontekstom. BoS metode uključuju značajke poput nelinearne aktivacije i prečaca.

Model postiže 43,5 mAP pri otrpilike 65 fps na MS COCO setu podataka.

3.8.3.5. YOLOv5

Peta verzija je izašla neposredno nakon prethodne te je vrlo laka za korištenje i treniranje jer je implementacija PyTorch-a.

Za *backbone* arkitekturu je korišten dijelom kroz faze (eng. *CSP - cross-stage partial*) konekcijski blok radi boljeg gradijenta toka kako bi se smanjio računski utrošak. Za konfiguraciju modela umjesto CFG datoteke koristi YAML (eng. *yet another markup language*).

3.8.3.6. YOLOv6

Ovu verziju je predstavila kineska prodavačka platforma Meituan u želji za modelom za industrijsku primjenu s boljim performansama. Razlika u odnosu na prijašnje modele je prepoznavanje bez sidrenih točaka (eng. *anchor-free detection*) i razdvojeni *head* dio mreže, što znači da je jedna glava zadužena za klasifikaciju, dok druga provodi regresiju za predviđanje koordinata graničnih okvira.

Ove promjene su rezultirale preciznošću mAP od 37,5 pri 1187 fps za nano verziju, odnosno mAP 45 pri 484 fps za malu verziju.

3.8.3.7. YOLOv7

Na ljeto 2022. godine je izašao model YOLOv7, do tada najbrži i najprecizniji model za prepoznavanje objekata s preciznošću mAP 56,8% pri fps između 5 i 160.

Kao *backbone* dijela ovog modela koristi se E-ELAN (eng. e*xtended efficient layer aggregation network*), mreža koja omogućuje učinkovitije učenje i konvergenciju kontrolom gradijenta. Za modele koji se temelje na ulančavanju, model koristi složeno skaliranje kako bi se pri klasifikaciji mogle koristiti različite brzine zaključivanja.

3.8.3.8. YOLOv8

Posljednja verzija modela je izašla u siječnju 2023. godine te je najnaprednija do sada. Iako nedostaje službena dokumentacija, dokazana je nadmoć ovog modela nad drugima. Na COCO setu podataka je izmjerena preciznost mAP 50,2 pri 1,83 milisekundi na A100 TensorRT GPU.

Moguća je implementacija kao Python paketa ili CLI-based, čineći ga lakim za korištenje, a postoji u 5 inačica: nano (n), small (s), medium (m), large (l) i extra large (x), te se sve mogu koristiti za klasifikaciju, prepoznavanje objekata i segmentaciju. Parametri skaliranja za pojedinu inačicu su prikazani tablicom 6.

Tablica 6 Inačice modela YOLOv8

inačica	d (depth_multiple)	w (width_multiple)	mc (max_channels)
n	0,33	0,25	1024
s	0,33	0,5	1024
m	0,67	0,75	768
l	1,00	1,00	51
x	1,00	1,25	512

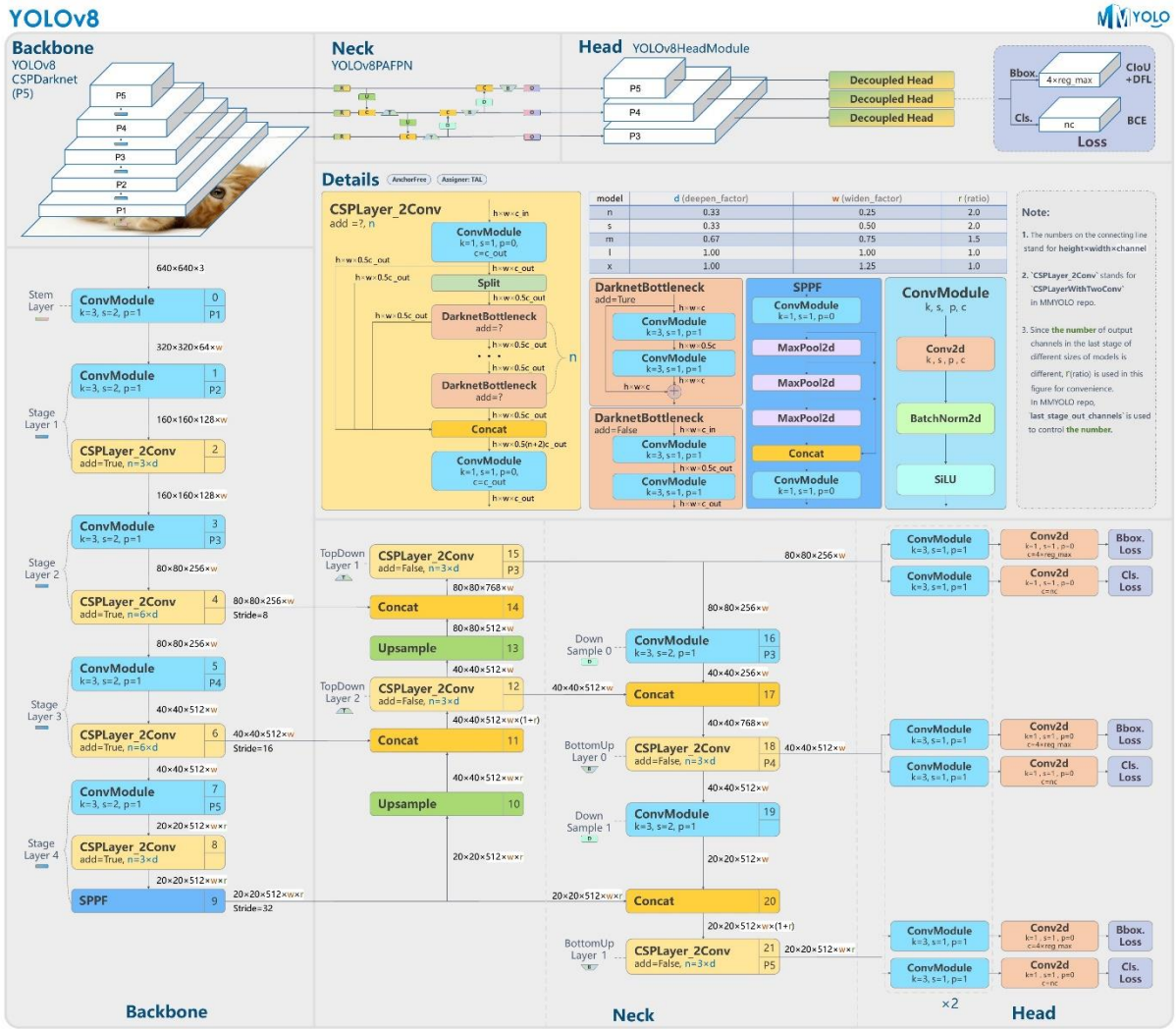
Glavne značajke ovog modela su mozaik augmentacija podataka, prepoznavanje bez sidrenih točaka, C2f modul, razdvojena glava i modificirana funkcija gubitaka.

- Mozaik augmentacija podataka – miješanje 4 slike kako bi se omogućio model s boljim kontekstualnim informacijama; augmentacija staje u zadnjih 10 trening epoha kako bi se poboljšale performanse
- Prepoznavanje bez sidrenih točaka – poboljšava generalizaciju, model izravno predviđa središte objekta i smanjuje broj predviđenih graničnih okvira te pomaže ubrzati NMS (eng. *non-max suppression*), korak pretprocesiranja koji odbacuje netočna predviđanja
- C2f modul – koristi se umjesto C3 modula, spaja izlaze svih *bottleneck* modula umjesto posljednjeg čime ubrzava trening i poboljšava gradijent toka
- Razdvojena glava (eng. *decoupled head*) – klasifikacija i regresija se izvode odvojeno, što poboljšava performanse modela
- Gubitak – moguća neusklađenost (lokalizacija jednog i klasifikacija drugog objekta), odabire najbolje k pozitivne uzorke i izračunava klasifikacijski gubitak koristeći BCE i

regresijski gubitak koristeći CIoU (*complete intersection over union*) i DFL (*distribution focal loss*)

- BCE mjeri razliku između stvarnih i predviđenih oznaka,
- CIoU uzima u obzir točnost predviđenih graničnih okvira prema temeljnoj istini po pitanju središta i mjerila,
- DFL optimizira distribuciju granica graničnih okvira, fokusirajući se na uzorke klasificirane kao lažno pozitivne.

Arhitektura ovog modela je prikazana slikom 43.



Slika 43 YOLOv8 arhitektura

Kao *backbone* dio mreže za izdvajanje značajki je korištena modificirana CSPDarknet53 mreža. Umjesto CSP sloja je korišten C2f modul, nakon kojeg slijede razdvojene glave za učenje semantičke segmentacije. SPPF (eng. *spatial pyramid pooling fast*) sloj ubrzava računanje sažimanjem značajki u mapu fiksne veličine. U *neck* dijelu postoji blok *upsample*

koji povećava rezoluciju. U *concat* bloku se spajaju značajke dva ulaza i zatim se slika prosljeđuju u *head* dio mreže koji služi za prepoznavanje. Između *concat* blokova postoje konvolucijski slojevi koji smanjuju dimenzije ulaza. Na taj način je omogućeno prepoznavanje malih objekata, odnosno nakon prve konvolucije prepoznavanje objekata srednje veličine i nakon druge konvolucije prepoznavanje velikih objekata. [40]

4. PRAKTIČNI DIO

Praktični dio zadatka se odnosi na stvaranje novog modela iz predtreniranog YOLOv8 modela u programskom jeziku python. Pri tome postoje 3 koraka: treniranje, provjeravanje (validacija) i testiranje. Treniranje se vrši nad označenim skupom podataka i za rezultat daje podešene težine modela te matricu konfuzije i grafove u kojem su prikazani parametri modela kroz epohe. Epoha predstavlja jedan prolazak cijelog skupa podataka za treniranje i određuju se proizvoljno. Cilj je postaviti broj epoha što manji kako se proces ne bi nepotrebno odužio, no dovoljan da se postigne željena točnost. Kada je model istreniran, slijedi provjera u kojoj se koristi model modificiran prema težinama, u kojem je moguće odabrati posljednje korištene ili one koje su dale najbolje rezultate. Provjera se također vrši nad označenim podacima u svrhu promatranja performansi modela na skupu na kojem nije treniran. Ako u ovom koraku model ne postiže zadovoljavajuće rezultate, može se ponoviti proces treniranja. Ako su pak rezultati zadovoljavajući, slijedi testiranje. Kod testiranja se model primjenjuje na dosad nepoznatim podacima, gdje se ostvaruje cilj samog učenja mreže.

4.1. Skup podataka

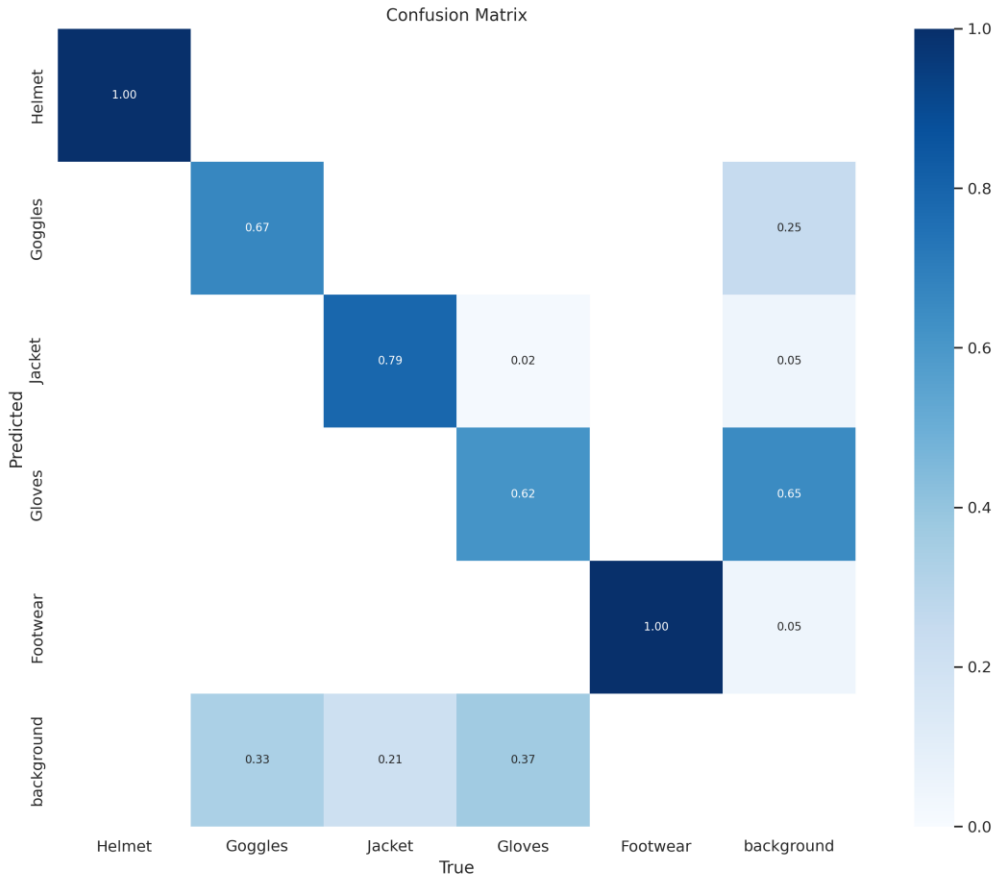
Skup podataka je zapravo baza slika koja je za ovaj primjer preuzeta s repozitorija. To je mapa kojoj se nalaze još 3 mape i jedna *.yaml* datoteka. 3 mape su *train*, *val* i *test* te se svaka koristi za pojedinu fazu učenja modela. U svakoj mapi se nalazi mapa *images*, u kojoj se nalaze slike, i mapa *labels*, u kojoj su prikazane oznake. Za ovaj format, oznake trebaju biti prikazane u obliku <*class*><*cx*><*cw*><*w*><*h*>, gdje <*class*> označava pripadnost klasi, <*cx*><*cw*> prikazuje središte graničnog okvira, a <*w*><*h*> prikazuje njegovu širinu i visinu. Za svaku sliku postoji jedna *.txt* datoteka s navedenim oznakama. U *.yaml* datoteci prikazan je put mape za treniranje i provjeru te broj i nazivi klasa. Klase u ovom skupu podataka su: *Helmet*, *Goggles*, *Jacket*, *Gloves* i *Footwear*.

4.2. Treniranje

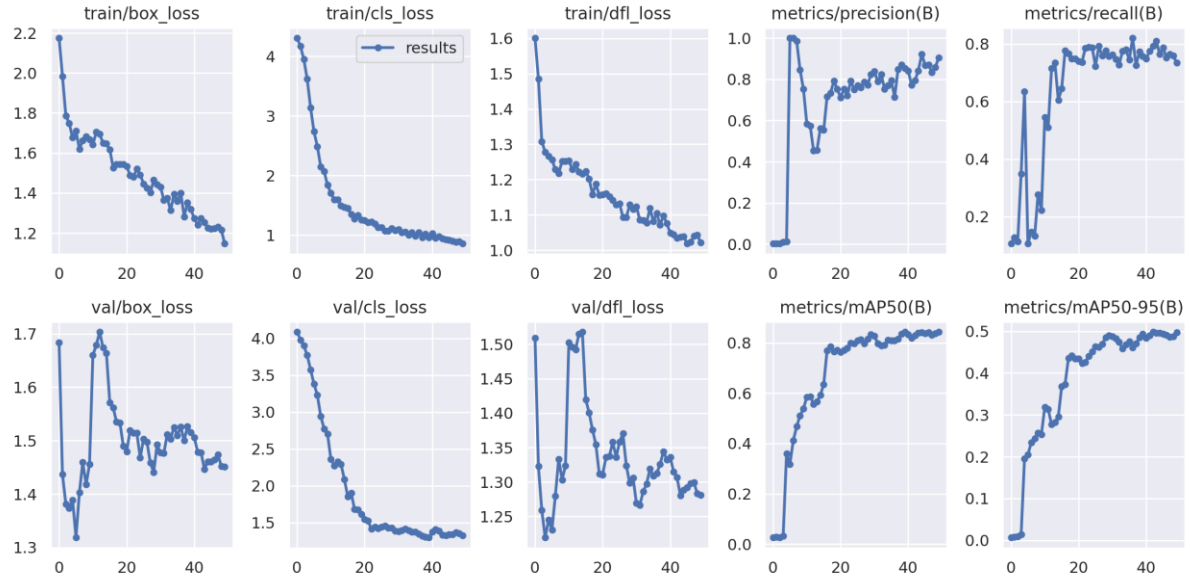
Za treniranje je korištena inačica *yolov8n.pt*. Za njeno korištenje potrebna je instalacija knjižnice *ultralytics*. Modelu je zadano treniranje na odgovarajućim podacima kroz 50 epoha kroz sljedeću naredbu:

!yolo task=detect mode=train model=yolov8n.pt data=data.yaml epochs=50 imgsz=275 plots=True

Kao rezultat treniranja mreže, dobivena je mapa s posljednjim i najboljim težinama te rezultati samog modela. Na slici 44 se nalazi dobivena matrica konfuzije, a na slici 45 se vidi vrijednost gubitaka i točnosti kroz epohe.



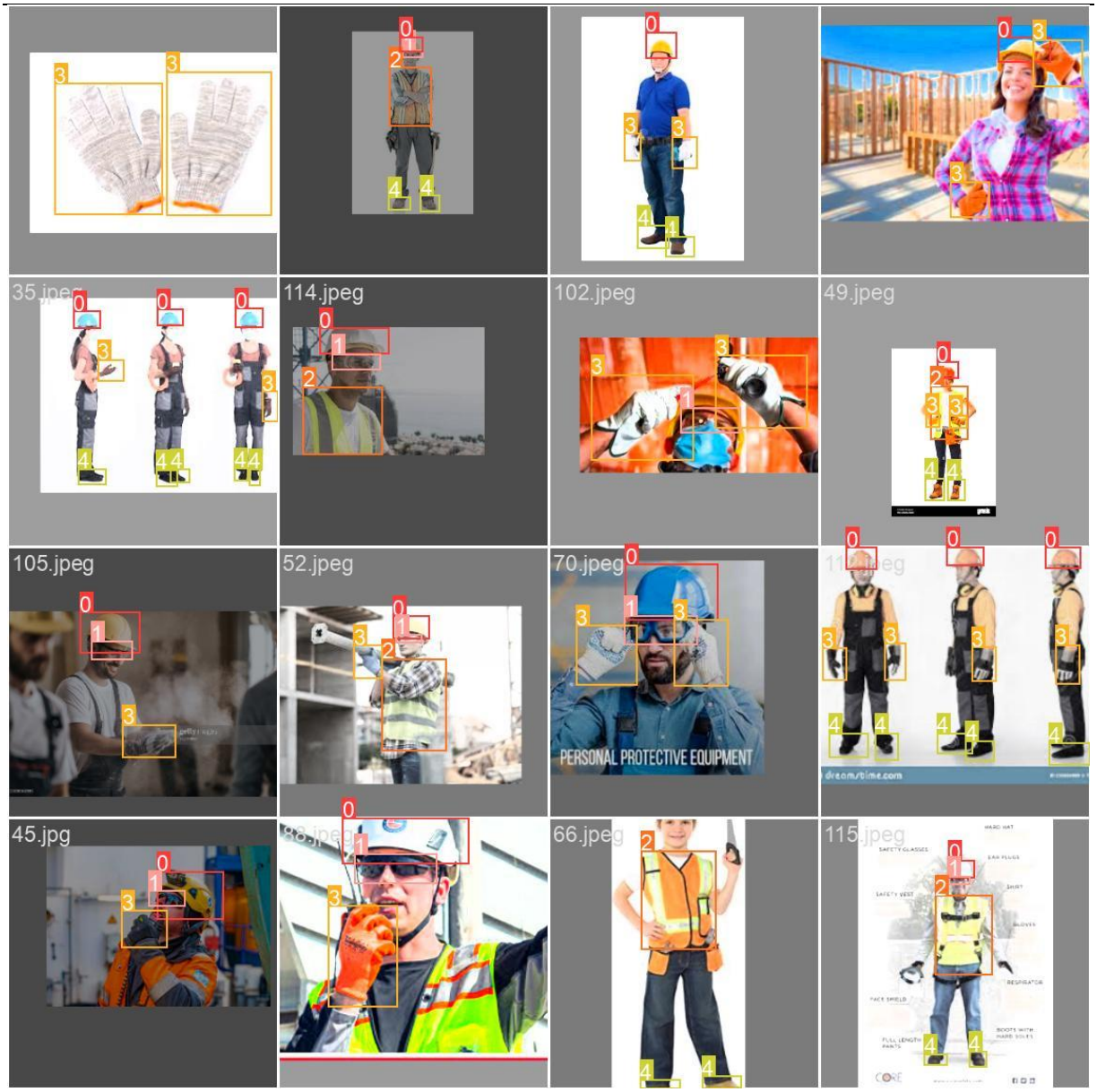
Slika 44 Matrica konfuzije treniranog modela



Slika 45 Rezultati parametara modela nakon treniranja

U matrici konfuzije je vidljivo kako su postignuti vrlo dobri rezultati treninga, osim za pozadinu jer pozadinu nije potrebno prepoznavati pa ne postoje slike za treniranje pozadine. Prema rezultatima, može se vidjeti da, iako neki parametri ne rastu ili ne padaju monotono, u konačnici gubici postaju sve manji, dok točnost raste, što je i cilj samoga treninga.

Još jedan parametar treniranja je serija (eng. *batch*). To je količina uzoraka nakon koje se provodi podešavanje parametara. Količina tih promjena ovisi o količini podataka za trening i iznosu serije. Primjer jedne serije iz treniranja ovog modela je na slici .



Slika 46 Serija treniranog modela

U poruci dobivenoj izvršavanjem treninga je prikazana struktura korištene mreže, slika 47.

```
                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.Conv                  [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.Conv                  [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.C2f                   [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.Conv                  [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.C2f                   [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.Conv                  [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.C2f                   [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.Conv                  [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.C2f                   [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.SPPF                  [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.Concat                [1]
 12                  -1  1    148224  ultralytics.nn.modules.C2f                   [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.Concat                [1]
 15                  -1  1     37248  ultralytics.nn.modules.C2f                   [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.Conv                  [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.Concat                [1]
 18                  -1  1    123648  ultralytics.nn.modules.C2f                   [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.Conv                  [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.Concat                [1]
 21                  -1  1    493056  ultralytics.nn.modules.C2f                   [384, 256, 1]
 22        [15, 18, 21]  1    752287  ultralytics.nn.modules.Detect                [5, [64, 128, 256]]
Model summary: 225 layers, 3011823 parameters, 3011807 gradients, 8.2 GFLOPs
```

Slika 47 Struktura korištene mreže

U poruci su prikazani rezultati svake epohe, gdje se može usporediti kako su iznosi gubitaka veći i iznosi točnosti manji na početku, slika 48, odnosno iznosi gubitaka manji, a točnost veća na kraju treninga, slika 49.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/50     0.803G      2.175      4.305      1.601         84        288: 100% 8/8 [00:25<00:00,  3.14s/it]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:26<00:00, 26.37s/it]
             all         31        101    0.00278      0.106      0.026    0.00655
          Helmet         31         18          0          0          0          0
        Goggles          31          9          0          0          0          0
         Jacket          31         14    0.00497      0.357     0.0921     0.0182
         Gloves          31         52    0.00892      0.173      0.038     0.0146
       Footwear          31          8          0          0          0          0

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/50     0.803G      1.984      4.166      1.486         67        288: 100% 8/8 [00:02<00:00,  2.75it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.14it/s]
             all         31        101    0.00383      0.128     0.0289    0.00858
          Helmet         31         18          0          0          0          0
        Goggles          31          9          0          0          0          0
         Jacket          31         14    0.00672      0.429      0.104      0.025
         Gloves          31         52     0.0124      0.212     0.0402     0.0179
       Footwear          31          8          0          0          0          0

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/50     0.803G      1.786      3.948      1.308         79        288: 100% 8/8 [00:03<00:00,  2.35it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.63it/s]
             all         31        101    0.00418      0.114     0.0264    0.00896
          Helmet         31         18          0          0          0          0
        Goggles          31          9          0          0          0          0
         Jacket          31         14    0.00657      0.357     0.0667     0.0179
         Gloves          31         52     0.0143      0.212     0.0653     0.0269
       Footwear          31          8          0          0          0          0
```

Slika 48 Rezultati epoha na početku

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      47/50      0.816G      1.224     0.9002      1.023         31        288: 100% 8/8 [00:02<00:00,  3.82it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.79it/s]
                   all         31        101      0.871      0.752      0.842      0.491
                Helmet         31         18      0.973          1      0.995      0.751
               Goggles         31          9      0.707      0.667      0.726       0.33
                Jacket         31         14      0.834      0.786      0.856      0.594
                Gloves         31         52      0.841      0.385      0.636      0.322
              Footwear         31          8          1      0.921      0.995      0.461

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50      0.816G      1.231     0.8859      1.041         44        288: 100% 8/8 [00:01<00:00,  5.69it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.91it/s]
                   all         31        101      0.835      0.764      0.832      0.486
                Helmet         31         18      0.939          1      0.995      0.741
               Goggles         31          9      0.696      0.667      0.699      0.333
                Jacket         31         14      0.793      0.786      0.857      0.581
                Gloves         31         52      0.745      0.385      0.612      0.302
              Footwear         31          8          1      0.983      0.995      0.474

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50      0.816G      1.216     0.8888      1.044         29        288: 100% 8/8 [00:01<00:00,  5.09it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.96it/s]
                   all         31        101      0.857      0.759      0.838      0.488
                Helmet         31         18       0.95          1      0.995      0.734
               Goggles         31          9      0.719      0.667      0.696      0.331
                Jacket         31         14      0.791      0.786      0.854      0.586
                Gloves         31         52      0.824      0.361      0.649       0.32
              Footwear         31          8          1      0.981      0.995      0.466

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      0.816G      1.146      0.857      1.022         35        288: 100% 8/8 [00:01<00:00,  4.78it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.43it/s]
                   all         31        101      0.904      0.735      0.843      0.498
                Helmet         31         18      0.968          1      0.995      0.742
               Goggles         31          9      0.932      0.667      0.711      0.339
                Jacket         31         14      0.808      0.786      0.859      0.611
                Gloves         31         52      0.814      0.346      0.657       0.33
              Footwear         31          8          1      0.875      0.995      0.466

50 epochs completed in 0.074 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.2MB
```

Slika 49 Rezultati epoha na kraju

4.3. Provjera

Tijekom provjere ili validacije, korišten je model s najboljim težinama. Provjera modela je izvršena prema naredbi:

!yolo task=detect mode=val model=runs/detect/train/weights/best.pt data=data.yaml

Dobivena je poruka, slika 50, u kojoj se vidi sažetak modela i iznose koje postiže.

```
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3006623 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.47it/s]
                   all         31        101      0.906      0.735      0.843      0.498
                Helmet         31         18      0.969          1      0.995      0.742
               Goggles         31          9      0.938      0.667      0.711      0.339
                Jacket         31         14       0.81      0.786      0.859      0.611
                Gloves         31         52      0.814      0.346      0.656       0.33
              Footwear         31          8          1      0.875      0.995      0.466
Speed: 0.1ms pre-process, 0.8ms inference, 0.0ms loss, 1.5ms post-process per image
Results saved to runs/detect/train
```

Slika 50 Rezultat provjere modela

Postignuti rezultati su zadovoljavajući te se isti model koristi za testiranje.

4.4. Testiranje

Testiranje se vrši nad neoznačenim skupom podataka koji su modelu podvrgnuti po prvi put, prema naredbi:

!yolo task=detect mode=predict model=runs/detect/train/weights/best.pt conf=0.7 source=data/test/images

Izvršavanjem testiranje, dobivena je poruka u kojoj se vide rezultati testiranja pojedine slike, slika 51.

```
Model summary (fused): 168 layers, 3006623 parameters, 0 gradients, 8.1 GFLOPs
image 1/31 /content/drive/MyDrive/YOLOv8/data/test/images/1.jpeg: 224x288 93.1ms
image 2/31 /content/drive/MyDrive/YOLOv8/data/test/images/10.jpeg: 192x288 61.0ms
image 3/31 /content/drive/MyDrive/YOLOv8/data/test/images/11.jpeg: 288x288 8.8ms
image 4/31 /content/drive/MyDrive/YOLOv8/data/test/images/12.jpeg: 224x288 7.9ms
image 5/31 /content/drive/MyDrive/YOLOv8/data/test/images/13.jpeg: 160x288 66.6ms
image 6/31 /content/drive/MyDrive/YOLOv8/data/test/images/14(1).jpeg: 288x288 7.7ms
WARNING ⚠ NMS time limit 0.550s exceeded
image 7/31 /content/drive/MyDrive/YOLOv8/data/test/images/14.jpg: 224x288 2 Helmets, 7.5ms
image 8/31 /content/drive/MyDrive/YOLOv8/data/test/images/15.jpeg: 288x288 7.8ms
image 9/31 /content/drive/MyDrive/YOLOv8/data/test/images/16.jpeg: 288x192 1 Helmet, 1 Goggles, 1 Jacket, 59.2ms
image 10/31 /content/drive/MyDrive/YOLOv8/data/test/images/17.jpeg: 224x288 1 Helmet, 1 Jacket, 1 Gloves, 7.8ms
image 11/31 /content/drive/MyDrive/YOLOv8/data/test/images/18.jpeg: 288x192 1 Helmet, 1 Jacket, 7.9ms
image 12/31 /content/drive/MyDrive/YOLOv8/data/test/images/19.jpeg: 288x288 1 Helmet, 1 Jacket, 7.7ms
image 13/31 /content/drive/MyDrive/YOLOv8/data/test/images/2.jpeg: 192x288 8.7ms
image 14/31 /content/drive/MyDrive/YOLOv8/data/test/images/20.jpeg: 288x192 1 Jacket, 1 Footwear, 8.2ms
image 15/31 /content/drive/MyDrive/YOLOv8/data/test/images/21.jpeg: 288x192 1 Jacket, 1 Gloves, 7.4ms
image 16/31 /content/drive/MyDrive/YOLOv8/data/test/images/22.jpeg: 288x160 1 Helmet, 1 Jacket, 1 Footwear, 71.5ms
image 17/31 /content/drive/MyDrive/YOLOv8/data/test/images/23.jpeg: 288x192 1 Helmet, 1 Jacket, 1 Gloves, 2 Footwears, 7.5ms
image 18/31 /content/drive/MyDrive/YOLOv8/data/test/images/24.jpeg: 288x224 1 Helmet, 1 Goggles, 1 Jacket, 99.4ms
image 19/31 /content/drive/MyDrive/YOLOv8/data/test/images/25.jpeg: 288x256 1 Helmet, 1 Jacket, 1 Gloves, 105.4ms
image 20/31 /content/drive/MyDrive/YOLOv8/data/test/images/26.jpeg: 288x288 4 Glovess, 11.7ms
image 21/31 /content/drive/MyDrive/YOLOv8/data/test/images/27.jpeg: 288x256 15.6ms
image 22/31 /content/drive/MyDrive/YOLOv8/data/test/images/28.jpeg: 256x288 64.1ms
image 23/31 /content/drive/MyDrive/YOLOv8/data/test/images/29.jpeg: 160x288 1 Gloves, 9.0ms
image 24/31 /content/drive/MyDrive/YOLOv8/data/test/images/3.jpeg: 224x288 1 Helmet, 1 Gloves, 8.7ms
image 25/31 /content/drive/MyDrive/YOLOv8/data/test/images/30.jpeg: 192x288 1 Gloves, 8.5ms
image 26/31 /content/drive/MyDrive/YOLOv8/data/test/images/4.jpeg: 288x192 1 Helmet, 8.1ms
image 27/31 /content/drive/MyDrive/YOLOv8/data/test/images/5.jpeg: 192x288 1 Helmet, 9.4ms
image 28/31 /content/drive/MyDrive/YOLOv8/data/test/images/6.jpeg: 192x288 1 Helmet, 7.4ms
image 29/31 /content/drive/MyDrive/YOLOv8/data/test/images/7.jpeg: 192x288 1 Gloves, 7.9ms
image 30/31 /content/drive/MyDrive/YOLOv8/data/test/images/8.jpeg: 224x288 1 Helmet, 1 Goggles, 1 Jacket, 1 Gloves, 8.6ms
image 31/31 /content/drive/MyDrive/YOLOv8/data/test/images/9.jpeg: 192x288 1 Helmet, 1 Gloves, 8.0ms
Speed: 0.4ms pre-process, 26.4ms inference, 21.8ms postprocess per image at shape (1, 3, 288, 288)
```

Slika 51 Rezultat provjere modela

Iz rezultata se može uvidjeti kako se testiranjem primjera, vrijeme zaključivanja smanjuje, čak za više od 10 puta. Sada je model spreman za daljnje korištenje.

4.5. Praćenje objekta

Za praćenje objekata će se koristiti isti model na 2 primjera. To su videi skinuti s interneta za slobodnu uporabu.

Za učitavanje videa je potrebno koristiti i knjižnicu *cv2*, specifičnu za potrebe računalnog vida, a koristi se i *numpy*, no samo za potrebe spremanja videa.

Praćenje se izvršava prema skripti *tracking.py*. Na kadrovima iz prvog videa, slike 52 i 53, su prepoznate kacige u graničnim okvirima s prikazanim vjerojatnosti pripadnosti klasi. Taj se

iznos stalno mijenja jer je video zapravo niz uzastopnih kadrova te model za svaki kadar prepoznaje objekt. Jakna nije prepoznata jer je prekrivena papirom te gubi dio svojih značajki zbog kojih model nije u mogućnosti prepoznati taj objekt.

Slika 52 Kadar u prvom primjeru

Slika 53 Kadar u prvom primjeru

Drugi primjer je video u kojem se pojavljuju različite klase. Iako model dobro funkcionira, u nekim kadrovima ne prepoznaje sve prisutne objekte, slike 54, 55 i 56. Potencijalno rješenje

tog problema je povećanje broja epoha. Mogući problem je i u premaloj bazi slika, odnosno bazi slika koja ne nudi dovoljno raznolikost unutar klase.

Slika 54 Kadar u drugom primjeru

Slika 55 Kadar u drugom primjeru

Slika 56 Kadar u drugom primjeru

Kada ne bismo koristili trenirani model, ne bismo mogli dobiti navedene klase. Klasični *yolov8n* model je treniran na drugom skupu podataka, stoga bi za oba primjera davao drugačije rezultate, slika 57 i 58. Izvršava se pokretanjem skripte *tracking.py* uz izmjenu korištenog modela.

Slika 57 Klasični *yolov8* model na prvom primjeru

Slika 58 Klasični *yolov8* model na drugom primjeru

4.6. Evaluacija rješenja

Softversko rješenje se pokazalo kao prilično dobro. U velikom broju slučajeva su objekti prepoznati točno, slika 59.


id:1 Helmet 0.94
id:2 Helmet 0.68


id:1 Helmet 0.76
id:2 Gloves 0.57


id:8 Helmet 0.56
id:2 Helmet 0.81
id:6 Jacket 0.47
id:1 Jacket 0.94
id:7 Footwear 0.32


id:2 Helmet 0.98
id:1 Jacket 0.64 .59
id:8 Jacket 0.69

Slika 59 Točno prepoznati i klasificirani objekti

U nekim primjerima, ili pojedinim kadrovima, je došlo do djelomičnog prepoznavanja objekata, slika 60.


id:3 Jacket 0.43
id:2 Footwear 0.37
id:1 Footwear 0.22


id:1 Helmet 0.83
id:2 Gloves 0.68


id:13 Jack
id:12 Foo

Slika 60 Djelomično prepoznati objekti

U samo 2 slučaja je postojala kriva lokalizacija objekta, slika 61. U oba slučaja je objekt bio kaciga.

Slika 61 Kriva lokalizacija

U nekoliko primjera je prepoznati objekt bio krivo klasificiran.

U dva primjera nije objekta na slici, no lokalizacija tih krivo prepoznatih objekata je bila točna, slika 62. Na prvom primjeru su rukavice prepoznate na rukama, a na drugom su rukavice prepoznate na rukama i naočale na očima.

Slika 62 Kriva klasifikacija s točnom lokalizacijom

U jednom primjeru je prepoznata kaciga gdje je uopće nije bilo jer je predmet na slici sličnog oblika i boje, slika 63.

Slika 63 Krivo prepoznati objekt

U nekim kadrovima je prepoznati objekt krivo klasificiran, slika 64.

Slika 64 Kriva klasifikacija

Najbolji rezultati vjerojatnosti pripadnosti klasi postignuti su za klase *Helmet* i *Jacket.* Ti objekti su površinom veći u odnosu na drugu, jednostavnijih su oblika i prisutne su manje varijacije unutar klase. Klase *Footwear* i *Gloves* su rjeđe prepoznate jer su prisutna veća varijacija te su kompliciranijih oblika te su iznosi vjerojatnosti pripadnosti klasi manji. *Footwear* je uvijek dobro prepoznata klasa, dok je u pojedinim slučajevima klasa *Gloves* prepoznata tamo gdje nije prisutna. Za klasu *Goggles* je lokalizacija u svim primjerima bila točna, čak i u neprisutnosti samog objekta, a iznosi vjerojatnosti pripadnosti klasi su slični onima kao kod klasa *Footwear* i *Goggles*.

Uspoređujući s modelom *yolov8n*, model treniran tijekom ovog rada daje dobre rezultate. rezultati vjerojatnosti pripadnosti klasi koje postiže trenirani model su isti ili nešto manji nego kod predtreniranog *yolov8n* modela, iako je baza slika za treniranje bila znatno manja.

Lokalizacija i klasifikacija je u većini slučajeva bila dobra i model je pokazao sposobnost praćenja objekta i pri promjeni perspektive gledanja i za varijacije unutar klase. Moguća poboljšanja modela uključuju povećanje baze slika uz veće varijacije unutar pojedine klase, daljnje treniranje postojećeg modela ili promjenu broja epoha ili broja serija, a moguće je na samom početku uzeti drugu inačicu *yolov8* modela prema kojoj će se trenirati model.

5. ZAKLJUČAK

U ovome radu je prikazana teorijska osnova za razumijevanje metoda umjetne inteligencije, točnije dubokog učenja. Za razumijevanje računalnog vida, krucijalno je razumjeti kako funkcioniraju konvolucijske neuronske mreže. Na njima se temelje svi moderni algoritmi za prepoznavanje, klasifikaciju i praćenje objekata.

Jedan od danas najpopularnijih i najboljih modela je YOLO, u najnovijoj verziji YOLOv8. to je predtrenirani model na velikoj bazi podataka koji se može koristiti kao takav, ili istrenirati na vlastitom skupu podataka. Pri treniranju, model koristi označene podatke za trening u kojima uči prepoznavati značajke kojima je definiran objekt. U ovom dijelu, model podešava težine kako bi bio što točniji i što brži. Nakon treninga se provodi validacija, u kojoj se promatraju performanse modela na označenim podacima na kojima nije treniran. Kao rezultat razvoja modela računalnog vida, koji koristi dobivene težine za izvršavanje zadataka na dotad neviđenim podacima. Premda je stvaranje novog modela od samog početka vrlo obuhvatan i dugotrajan proces, danas se gotovo uvijek koriste neki od postojećih modela te se oni nadograđuju i/ili modificiraju.

Iz praktičnog dijela se može zaključiti koliko korištenje predtreniranih modela olakšava stvaranje novog. Težine su tijekom treninga podešene za prepoznavanje klasa koje se nalaze među podacima za trening. Što je veći broj epoha proveden, to su gubici manji, a točnost veća. To dovodi do zaključka da je poželjan veći broj epoha, no ne pretjerano velik već do zadovoljavajuće točnosti kako se trening ne bi nepotrebno odužio. Također, iz dobivenih podataka je vidljivo kako se s većim brojem testiranih podataka smanjuje vrijeme zaključivanja.

LITERATURA

[1] Difference Between Artificial Intelligence vs Machine Learning vs Deep Learning, *https://www.geeksforgeeks.org/difference-between-artificial-intelligence-vs-machine-learning-vs-deep-learning/*, pristupljeno: 11.12.2023.

[2] Neuron, *https://hr.wikipedia.org/wiki/Neuron*, pristupljeno: 11.12.2023.

[3] Wang X., Liu Y. Xin H.: Bond strength prediction of concrete-encased steel structures using hybridmachine learning method, 2021.

[4] What is Perceptron: A Beginners Guide for Perceptron, *https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron*, pristupljeno: 12.12.2023.

[5] How Do Convolutional Layers Work in Deep Learning Neural Networks?, *https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/*, pristupljeno: 14.12.2023.

[6] Image Types, *https://www.nv5geospatialsoftware.com/docs/Image_Types.html*, pristupljeno: 14.12.2023.

[7] Convolutional Neural Networks (CNN) – Architecture Explained, *https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243*, pristupljeno: 14.12.2023.

[8] Convolution layer, Padding, Stride, and Pooling in CNN, *https://www.codingninjas.com/studio/library/convolution-layer-padding-stride-and-pooling-in-cnn*, pristupljeno: 15.12.2023.

[9] Activation Functions in Neural Networks [12 Types & Use Cases], *https://www.v7labs.com/blog/neural-networks-activation-functions#h3*, pristupljeno: 16.12.2023.

[10] Introduction To Pooling Layers In CNN, *https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn*, pristupljeno: 16.12.2023.

[11] Fully Connected Layer vs. Convolutional Layer: Explained, *https://builtin.com/machine-learning/fully-connected-layer*, pristupljeno: 16.12.2023.

[12] Top 10 Deep Learning Algorithms You Should Know in 2023, *https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm*, pristupljeno: 18.12.2023.

[13] A Brief Overview of Recurrent Neural Networks (RNN), *https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/*, pristupljeno: 18.12.2023.

[14] What is LSTM? Introduction to Long Shoert-Term Memory, *https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/*, pristupljeno: 18.12.2023.

[15] A Gentle Introduction to Generative Adversarial Networks (GANs), *https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/*, pristupljeno: 18.12.2023.

[16] What are Radial Basis Functions Neural Networks? Everything You Need to Know, *https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-are-radial-basis-functions-neural-networks*, pristupljeno: 18.12.2023.

[17] Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis, *https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141*, pristupljeno: 18.12.2023.

[18] Self Organizing Maps – Kohonen Maps, *https://www.geeksforgeeks.org/self-organising-maps-kohonen-maps/*, pristupljeno: 18.12.2023.

[19] A Beginner's Guide to Restricted Boltzmann Machines (RBMs), *https://wiki.pathmind.com/restricted-boltzmann-machine*, pristupljeno: 18.12.2023.

[20] Wang H., Raj B.: On the Origin of Deep Learning, 2017.

[21] What are Autoencoders? Introduction to Autoencoders in Deep Learning, *https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-are-autoencoders-in-deep-learning*, pristupljeno: 18.12.2023.

[22] Everything You Ever Wanted To Know About Computer Vision, *https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e*, pristupljeno: 22.12.2023.

[23] Computer Vision: What it is and why it matters, *https://www.sas.com/en_id/insights/analytics/computer-vision.html*, pristupljeno: 22.12.2023.

[24] Exploring the Applications and Challenges of Computer Vision Technology, *https://medium.com/@mostafa.ragheb.ghareeb.cse/exploring-the-applications-and-challenges-of-computer-vision-technology-5e445dd9b36e*, pristupljeno: 22.12.2023.

[25] Tonde C.: Hardware and Software Platforms for Computer Vision, Rutgers University, 2010.

[26] Đuričić T., Merćep A.: Strojno učenje, Bilješke s predavanja, Šnajder J., Dalbelo Bašić B., Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2015./2016.

[27] Du L., Zhang R., Wang X.: Overview of two-stage object detection algorithms, Beijing University of Post and Telecommunication, Peking, 2020.

[28] Girshick R., Donahue J., Darrell T., Malik J.: Rich feature hierarchies for accurate object detection and semantic segmentation, Tech report (v5), UC Berkley, 2014.

[29] He K., Zhang X., Ren S., Sun J.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 2015.

[30] Girshick R., Microsoft Research: Fast R-CNN, 2015.

[31] Ren S., He K., Girshick R., Sun J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2016.

[32] He K., Gkioxari G., Dollàr P., Girshick R., Facebook AI Research (FAIR): Mask R-CNN, 2018.

[33] Cai Z., Vasconcelos N.: Cascade R-CNN: Delving into High Quality Object Detection, UC San Diego, 2017.

[34] Lin T.-Y., Dollàr P., Girshick R., He K., Hariharan B., Belongie S., Facebook AI Research (FAIR), Cornell University and Cornell Tech: Feature Pyramid Networks for Object Detection, 2017.

[35] Liu W., Anguelov D., Erhan D., Szegedy C., Reed S. Fu C.-Y, Berg A. C., UNC Chapel Hill, Zoox Inc., Google Inc. University of Michigan, Ann-Arbor: SSD: Single Shot MultiBox Detector, 2016.

[36] Lin T.-Y., Goyal P., Girshick R., He K., Dollàr P., Facebook AI Research (FAIR): Focal Loss for Dense Object Detection, 2018.

[37] Terven J., Córdova-Esparza D.-M., Romero-González J.-A.: A Comprehensive Revoew of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Mach. Learn. Knowl. Extr.* 2023, 5, 1680–1716. https://doi.org/10.3390/make5040083

[38] A Guide to YOLOv8 in 2024, *https://viso.ai/deep-learning/yolov8-guide/*, pristupljeno: 28.12.2023.

[39] Redmon J., Farhadi A.: YOLOv3: An Incremental Improvement. University of Washington, 2018.

[40] YOLOv8 Architecture Detailed Explanation – A Complete Breakdown, *https://www.youtube.com/watch?v=HQXhDO7COj8*, pristupljeno: 29.12.2023.

PRILOZI

I. *tracking.py*

```
import cv2
from ultralytics import YOLO

# ucitavanje modela
model = YOLO('runs/detect/train/weights/best.pt')

# ucitavanje videa
video_path = "dataset/test_vid_1.mp4"
cap = cv2.VideoCapture(video_path)

# dimenzije videa za spremanje
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# stvaranje videa za spremanje rezultata
out =
cv2.VideoWriter('tracking.mp4',cv2.VideoWriter_fourcc('M','J','P','G'), 10,
(frame_width,frame_height))

# petlja za prikaz videa
while cap.isOpened():
    # ucitavanje kadra
    success, frame = cap.read()

    if success:
        frame = cv2.resize(frame,(1280,720),fx=0,fy=0, interpolation =
cv2.INTER_CUBIC)
        # pracenje kroistenjem modela
        results = model.track(frame, persist=True)

        # dodavanje rezultata na video i spremanje videa
        annotated_frame = results[0].plot()
        out.write(annotated_frame)

        # prikaz videa
        cv2.imshow("YOLOv8 Tracking", annotated_frame)

        # za prekid videa
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # prekid na kraju videa
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```