Neuronske mreže i njihova primjena u detekciji objekata

Žunić, Ante

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI DIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Ante Žunić

DIPLOMSKI RAD

NEURONSKE MREŽE I NJIHOVA PRIMJENA U DETEKCIJI OBJEKATA

Zagreb, prosinca 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni diplomski studij informacijskih tehnologija smjer programsko inženjerstvo i informacijski sustavi
Student:	**Ante Žunić**
Matični broj:	2021109

Zadatak diplomskog rada

Predmet:	Statistika
Naslov:	**Neuronske mreže i njihova primjena u detekciji objekata**
Zadatak:	Definirati osnovne pojmove iz teorije i primjene umjetnih neuronskih mreža. Objasniti neke vrste neuronskih mreža i načine na koje uče. Na skupu zadanih podataka (slika) u raspoloživom softveru izraditi model koji uči detektirati objekte i interpretirati rezultate.
Mentor:	Marijan Čančarević, v. pred.
Zadatak uručen kandidatu:	4.11.2024.
Rok za predaju rada:	30.9.2025.
Rad predan:	_____

Povjerenstvo:

Dragana Čulina, pred.	član predsjednik	_____
Marijan Čančarević, v. pred.	mentor	_____
Jurica Đurić, v. pred.	član	_____

Sadržaj

1. UVOD 7

2. NEURONSKE MREŽE 9

2.1. Komponente neuronskih mreža 9

2.2. Treniranje neuronskih mreža 13

2.3 Vrste neuronskih mreža 16

3. UMJETNA INTELIGENCIJA I DETEKCIJA OBJEKATA 18

3.1. Konvolucijske neuronske mreže 18

3.2. YoloV8 20

4.5. PRAKTIČNI RAD - Neuronske mreže i njihova primjena u detekciji objekata 23

4.1. Plan rada 23

4.2. Odabir neuronske mreže 24

4.3. Implementacija YOLOV8 softvera 26

4.4. Izrada modela 29

4.5. Rezultati učenja 32

4.6. Logika segmentacije videozapisa 34

4.7. Poboljšanja 37

5. ZAKLJUČAK 39

LITERATURA 41

SAŽETAK 42

SUMMARY 43

Popis slika

Slika 1: Jednostavni prikaz neurona (Ultralytics, 2024)..... 9
Slika 2: Prikaz neuronske mreže (Medium, 2017) 10
Slika 3: Grafovi aktivacijskih funkcija..... 13
Slika 4: Prikaz konvolucijske neuronske mreže (LinkedIn, 2023)..... 20
Slika 5: Primjer detekcije objekta u YOLO softveru(Medium, 2020) 21
Slika 6: Primjer podataka korištenih u projektu 22
Slika 7: Prikaz alata za anotaciju podataka 26
Slika 8: Prikaz sposobnosti YOLO softvera (Ultralytics, 2024) 26
Slika 9: Arhitektura YOLOv8 modela (Medium, 2024)..... 27
Slika 10: Strukturna razlika između C3 i C2f modula (Medium, 2024) 29
Slika 11: Razlika razdovjenog i nerazdvojenog “Head” modula (Medium, 2024) 29
Slika 12: Prikaz rezultata učenja modela..... 31
Slika 13: Broj instanci klasa i njihove pozicije na slikama 33
Slika 14: Matrica zabune 33
Slika 15: Normalizirana matrica zabune 34
Slika 16: Prikaz segmentacije video podataka za obradu u audio format 35
Slika 17: Kôd koji izvršava lokalizaciju objekata po segmentima..... 36

Popis tablica

Tablica 1: Popis klasa za projekt 23

Popis kôdova

Kôd 1: linija koda za pokrenuti treniranje modela 30
Kôd 2: dohvaćanje modela za upotrebu 30
Kôd 3: Pozivanje kôda za detekciju objekata, pretvorbu u audio format i ispis informacija 37

1. UVOD

U posljednjem desetljeću, umjetna inteligencija (engl. *Artificial Intelligence,* AI) doživjela je neviđeni porast, revolucionarizirajući brojne industrije i transformirajući način na koji se komunicira s tehnologijom. Među brojnim izvanrednim primjenama, integracija umjetne inteligencije s računalnim vidom nedvojbeno se pokazala kao najvažnije postignuće. Sa sposobnošću repliciranja, pa čak i nadmašivanja ljudske vizualne percepcije, sustavi AI vida otključali su bezbroj mogućnosti u različitim domenama, od zdravstvene zaštite i autonomnih vozila do zabave i nadzora.

Ovaj rad zadire u zadivljujuće područje detekcije objekata s umjetnom inteligencijom, objašnjava inovativna otkrića, metodologije i implementacije u stvarnom svijetu koje nose čovječanstvo u eru u kojoj strojevi mogu "vidjeti" i interpretirati vizualne podatke s izuzetnom preciznošću. U potrazi za stvaranjem istinski inteligentnih sustava, konvergencija računalnog vida i umjetne inteligencije omogućila je strojevima ne samo prepoznavanje objekata, već i razumijevanje konteksta, emocija, pa čak i predviđanje događaja, značajno povećavajući njihovu korisnost i utjecaj u našim svakodnevnim životima.

Temeljno načelo načina na koji računala mogu vidjeti leži u spajanju algoritama dubokog učenja, neuronskih mreža i masivnih skupova podataka. Razvoj konvolucijskih neuronskih mreža (engl. *Convolutional Neural Network,* CNN) i njihovih varijanti otvorio je put neusporedivim mogućnostima prepoznavanja slika i videa, omogućujući strojevima da razaznaju obrasce i značajke sa zapanjujućom točnošću. Ovi moćni algoritmi, uvježbani na golemim repozitorijima označenih slika, uspješno su postigli izvedbu na ljudskoj razini u različitim zadacima vizualnog prepoznavanja, podižući vidno polje umjetne inteligencije na nove visine.

Kroz ovaj rad istražit će se neke od najutjecajnijih arhitektura i tehnika vida umjetne inteligencije, uključujući početak mehanizama pažnje, prijenos paradigmi učenja i generativne protivničke mreže (engl. *Generative Adversarial Networks,* GAN). Svaka inovacija dodatno je poboljšala sustave koji omogućuju vid računalima, dajući mogućnost prilagodbe, generalizacije i generiranja novog sadržaja s neviđenim realizmom.

Osim svog znanstvenog značaja, sustavi umjetne inteligencije već su započeli duboki društveni utjecaj. Napredak u zdravstvu vođen umjetnom inteligencijom doveo je do rane dijagnoze bolesti, personaliziranih planova liječenja i poboljšanog medicinskog snimanja. Osim toga, istražit će se uloga računalne percepcije u revoluciji transporta, napredovanju u autonomnim vozilima,

poboljšanju sigurnosti na cestama i optimiziranju upravljanja prometom. Međutim, uz ove izvanredne napretke, sve veća integracija tehnologije umjetne inteligencije postavlja pitanja o etičkim razmatranjima, privatnosti i mogućim pristranostima u algoritmima za donošenje odluka. Potencijal za modele detekcije objekata da preoblikuju industrije i osnaže društvo je neograničen, ali je imperativ da se tim putem kroči sa znanstvenom strogošću i etičkom pažnjom. Prihvaćanjem budućnosti odgovorne umjetne inteligencije mogu se omogućiti nove inovacije i osigurati da ova transformativna tehnologija koristi cijelom čovječanstvu.

2. NEURONSKE MREŽE

Neuronske mreže su se pojavile kao snažna paradigma za aplikacije strojnog učenja i umjetne inteligencije. Ovaj rad daje precizan pregled neuronskih mreža, razjašnjavajući njihove temeljne principe te uporabe za simuliranje ljudske vizualne percepcije. Zalazi se u osnovne koncepte neurona, slojeva, funkcija aktivacije i algoritama za obuku, razjašnjavajući njihove uloge u dizajnu i radu neuronske mreže. Dodatno, raspravlja se o vrstama neuronskih mreža, njihovim primjenama i nedavnim dostignućima u tom području.

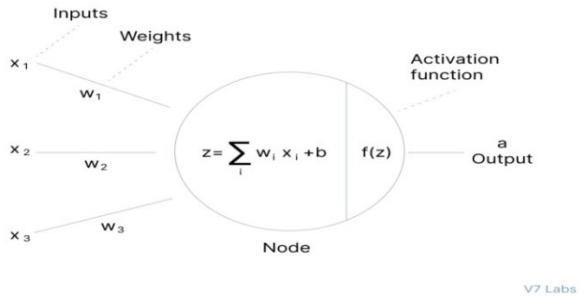
Neuronske mreže, inspirirane biološkim mozgom, stekle su ogromnu popularnost posljednjih godina zbog svojih iznimnih performansi u raznim zadacima kao što su prepoznavanje slika, obrada prirodnog jezika i igranje igrica. Ovo poglavlje ima za cilj razjasniti temeljne koncepte i komponente neuronskih mreža kako bi se omogućilo dublje razumijevanje njihovog unutarnjeg rada.

2.1. Komponente neuronskih mreža

Neuronska mreža sastoji se od nekoliko ključnih komponenti: neurona (ili čvorova), slojeva neurona (uključujući ulazne, skrivene i izlazne slojeve), aktivacijske funkcije i funkcije gubitaka.

2.1.1. Neuroni

Glavna komponenta neuronske mreže nalazi se u umjetnim neuronima, također poznatim kao čvorovi ili jedinice. Ti neuroni oponašaju funkcionalnost bioloških neurona primanjem ulaznih podataka, izvođenjem izračuna i stvaranjem izlaznih podataka. Svaki neuron skuplja svoje ulaze, primjenjuje matematičku transformaciju i prosljeđuje rezultat sljedećem sloju. Analitički, izlaz (aktivacija) neurona može se predstaviti kao na slici (Slika *1*):



Slika 1: Jednostavni prikaz neurona (Ultralytics, 2024)

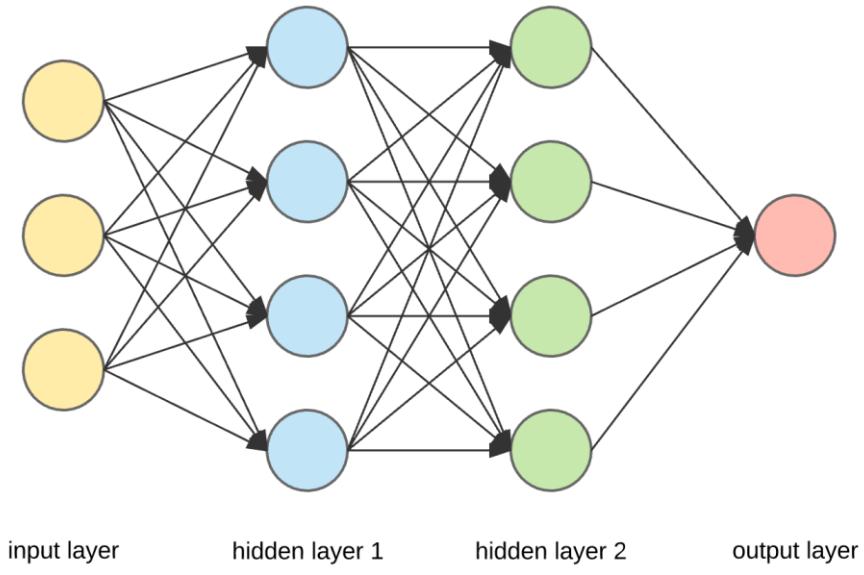
Objašnjenje:

- „$output$“ ili „a” je aktivacija neurona,
- $f(z)$ je aktivacijska funkcija,
- w su težine koje su povezane s ulaznim parametrima,
- x su ulazni parametri,
- b označava pristranost.

2.1.2. Slojevi

Neuroni su organizirani u slojeve unutar neuronske mreže (Slika *2*). Obično postoje tri glavne vrste slojeva:

- **Ulazni sloj**: Ovaj sloj prima početne podatke ili značajke i prosljeđuje ih sljedećim slojevima. Broj neurona u ulaznom sloju određen je dimenzionalnošću ulaznih podataka,
- **Skriveni sloj**: Ovi slojevi, smješteni između ulaznog i izlaznog sloja, obavljaju većinu računanja u neuronskoj mreži. Skriveni slojevi omogućuju mreži da nauči složene obrasce i prikaze,
- **Izlazni sloj**: Završni sloj proizvodi mrežni izlaz, koji može biti rezultat klasifikacije, regresijsko predviđanje ili bilo koji drugi željeni ishod. Struktura izlaznog sloja ovisi o specifičnom zadatku.



Slika 2: Prikaz neuronske mreže (Medium, 2017)

2.1.3. Aktivacijske funkcije

Aktivacijske funkcije uvode nelinearnost u mrežu, omogućujući joj modeliranje složenih odnosa u podacima. Ovisno o problemu koji se rješava svi skriveni slojevi će koristiti istu aktivacijsku funkciju koja je najčešće drugačija od aktivacijskih funkcija izlaznog sloja. Najčešće korištene funkcije aktivacije su:

- **Step funkcija:** $f(x) = \begin{cases} 1; & \text{x} \geq 0 \\ 0; & x < 0 \end{cases}$,
- **Linearna funkcija:** $f(x) = kx$,
- **Nelinearne funkcije:** $Sigmoid(x) = \frac{1}{1+ e^{-x}}$,

  $\text{Tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$,

  $ReLU(x) = \max(0, x)$,

  $\text{S}oftmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$.

Različite aktivacijske funkcije se biraju ovisno o vrsti izlaza koji određeni problem zahtjeva te se mogu dobiti bolji ili gori rezultati za isti ulaz zato što svaka aktivacijska funkcija radi na drugi način.

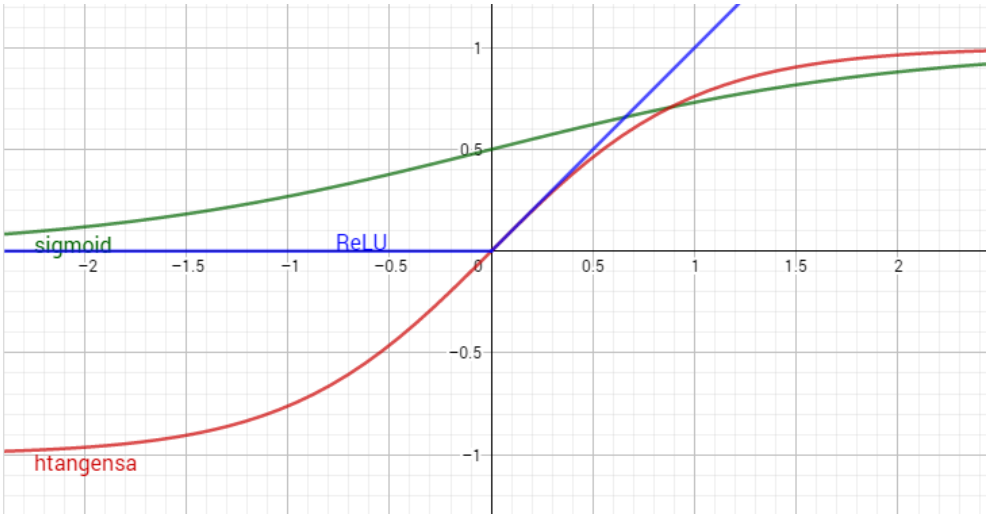
Analizirajući snage i slabosti ove 3 vrste aktivacijskih funkcija postaje evidentno da step funkcija i linearna funkcija imaju ograničenu upotrebljivost, razlog tome je to što nemaju mogućnost korištenja povratne propagacije i višestrukih skrivenih slojeva neurona. Najčešće korištene funkcije su nelinearne, specifično to bi bile Sigmoid, Tanh, ReLU i Softmax funkcije. Postoji također puno varijacija na te 4 osnovne funkcije kada problem koji se rješava zahtjeva neku od karakteristika koje se mogu samo dobiti od tih funkcija.

Sigmoid funkcija ulazne parametre transformira u broj između 0 i 1. Ta karakteristike omogućuje pretvaranja izlaza u neku vjerojatnost. Zbog toga se koristi u izlaznom sloju kada je problem binarne klasifikacije. Također, kod ove funkcije nailazi se na teškoće pri računanju gradijenta. Što x brže raste, funkcija sporije raste jer se približava asimptoti $y = 1$. Ovo je nepoželjno jer neuronska mreža treba taj gradijent kako bi mogla učiti kroz aktivacije neurona.

Tanh (engl. *Hyperbolic tangent,* Hiperbolni tangens) funkcija radi slično kao Sigmoid funkcija samo što ne komprimira vrijednosti na interval od 0 do 1 nego -1 do 1. Iako slično izgledaju na grafu, ove dvije funkcije imaju dosta različite primjene. Tanh funkcija služi više u skrivenom sloju zbog toga što daje veću nelinearnost. Još jedna prednost koja ova funkcija ima je to što je srednja vrijednost 0, ovo olakšava distinkciju između negativnih, neutralnih i pozitivnih izlaznih podataka. Nažalost i tanh pati od iste pojave kao i sigmoidna funkcija, ta pojava je nestajući gradijent. U slučaju Tanh funkcije gradijent čak brže otpada od Sigmoid, ali samo zbog korisnosti negativnih vrijednosti često je slučaj da Tanh bude poželjniji nad sigmoidnom funkcijom.

ReLU funkcija. ReLU je kratica za *Rectified Linear Unit,* ili "ispravljena linearna jedinica". To je funkcija koja pruža dobre rezultate dok je ujedno i efikasna što se tiče računanja. Njezina formula je zapravo identična običnoj linearnoj funkciji uz jednu znatnu promjenu, ako je ulazni podatak manji od 0 neuron se neće uopće aktivirati. Druga prednost ReLU funkcije osim njene efikasnosti je također smanjenje funkcije gubitka, to je funkcija koja dopušta da se mjeri koliko je precizan izlaz modela (to jest preciznost neuronske mreže) naspram očekivanog ili točnog izlaza. Ova funkcija ne nailazi na problem nestajućeg gradijenta, ali zato ima svoj vlastiti problem umirućeg ReLU-a. Ovaj problem se odnosi na situaciju gdje neki neuroni postaju neaktivni i prestaju učiti tijekom procesa treniranja. Razlog ovakvom ponašanju je to što je ulaz na ReLU neuron konzistentno negativan, tako da je izlaz neurona uvijek nula (Slika *3*). Nakon nekog vremena neuron za bilo koji ulaz počne vraćati nulu, jer je tako naučen.

Softmax je funkcija koja radi pretvorbu vektora od N realnih brojeva u distribuciju vjerojatnosti od N mogućih događaja. Ona je korištena kada projekt sadrži više klasa, te služi za dodjeljivanje vjerojatnosti klasama. Usporedba grafova aktivacijskih funkcija (Slika *3*):



Slika 3: Grafovi aktivacijskih funkcija

2.2. Treniranje neuronskih mreža

Postoje razne metodologije učenja neuronske mreže:

- Nadzirano,
- Nenadzirano i
- Polunadzirano.

Nadzirano učenje podrazumijeva da neuronska mreža ima na raspolaganju neki oblik označenih podataka koji se mogu sastojati od slika, numeričkih podataka ili drugih vrsta podataka ovisno o tome što se od modela traži. Jednom kada model ima podatke, proces obuke može započeti i s pomoću oznaka model može učiti i zauzvrat testirati svoju točnost u svojim predviđanjima.

Nenadzirano učenje koristi algoritme za prolazak kroz podatke i njihovo grupiranje u skupine prema neodređenim kriterijima. Ova se metoda često koristi kada postoji potreba za pronalaženjem obrazaca u podacima koje ljudi ne mogu pronaći. Postoje 3 glavna načina korištenja učenja bez nadzora, a to su: grupiranje, pridruživanje i smanjenje dimenzionalnosti. Grupiranje se odnosi na pronalaženje točaka podataka koje su slične i njihovo stavljanje u grupe diferencijacijom. Asocijacija pronalazi odnos između varijabli u skupovima podataka (jedna od upotreba je preporučivanje sličnih proizvoda kupcima koji su nešto kupili). Smanjenje dimenzionalnosti koristi se za smanjenje količine dimenzija, a time i složenosti skupa podataka. Time se stvara skup podataka kojim se može upravljati bez narušavanja kvalitete podataka.

Često postoje problemi s odlučivanjem koju metodologiju treba koristiti za određeni zadatak budući da se te dvije opcije uvelike razlikuju. Učenje pod nadzorom bolje je za jednostavnije zadatke gdje ima manje podataka, ali zahtijeva profesionalnu izradu skupa podataka. S druge strane, učenje bez nadzora može biti snažno, ali treba imati ogromnu količinu podataka, pa čak i tada može dati netočne rezultate osim ako ga ne provjeri osoba koja razumije cilj rješenja. Međutim, postoji srednji put, polunadzirano učenje. Ova metoda kombinira preciznost nadziranog učenja i minimalnu ljudsku interakciju nenadziranog učenja. Ovdje se koriste i označeni i neoznačeni skupovi podataka. Kada ima previše podataka za označavanje, djelomično označavanje omogućuje relativno dobro učenje modela. Nakon toga, korištenje modela za označavanje samih podataka i korištenje ljudskog znanja za ispravljanje nepreciznosti i pogrešaka koje se neizbježno pojavljuju.

Neovisno o tome koja se metodologija koristi za obuku modela, postoje koraci koji se gotovo uvijek poduzimaju.

Proces obuke je objašnjen na primjeru, u ovom primjeru neuronska mreža se trenira s označenim skupom podataka. To omogućuje pravljenje predviđanja i prilagođavanje parametara kako bi se smanjila pogreška predviđanja. Ključni koraci u obučavanju neuronske mreže su sljedeći:

1. **Inicijalizacija**: težine i pristranosti neuronske mreže moraju biti inicijalizirane s malim slučajnim vrijednostima. Optimizirana inicijalizacija može pomoći da trening brže konvergira. Funkcija gubitka ili troška se također bira u ovom koraku. Ona kvantificira razliku između predviđenih rezultata i stvarnih ciljnih vrijednosti (osnovna istina). Uobičajene funkcije gubitka uključuju srednju kvadratnu pogrešku (MSE, engl. *Mean Squared Error*) za zadatke regresije i unakrsnu entropiju za zadatke klasifikacije.
2. **Prolaz unaprijed (engl. *Feedforward*)**: Treba osigurati ulazne podatke neuronskoj mreži i izračunati predviđanja (izlaze) koristeći trenutne težine i pristranosti. Ovo je poznato kao prolaz unaprijed ili *feedforward* prolaz. Predviđanja se zatim uspoređuju s temeljnim oznakama istine.
3. **Prolaz unatrag (širenje unatrag, engl. *backpropagation*)**: Uključuje iterativno prilagođavanje težine veza između neurona širenjem pogreške unatrag od izlaznog sloja do ulaznog sloja. To omogućuje mreži da uči na svojim pogreškama i poboljša svoju izvedbu tijekom vremena.

4. **Gradijentno spuštanje**: Ovo je proces koji za cilj ima minimizirati funkciju gubitka (to jest naći globalni ili barem lokalni minimum funkcije). Ovaj korak uključuje množenje gradijenata sa stopom učenja i oduzimanje rezultata od trenutnih vrijednosti parametra. Varijante gradijentnog spuštanja, kao što su Adam, RMSprop i stohastički gradijentni spust (engl. *Stochastic Gradient Descent,* SGD), mogu se koristiti za učinkovitije treniranje.
5. **Ponavljanje**: Koraci od 2 do 5 ponavljaju se iterativno za više epoha. Svaka se epoha sastoji od potpunog prolaska kroz skup podataka za obuku. Ponavljanje ovog procesa omogućuje neuronskoj mreži da postupno prilagodi svoje parametre i poboljša svoja predviđanja.
6. **Validacija**: Periodički se izvedba modela ocjenjuje na zasebnom validacijskom skupu podataka kako bi se pratila njegova sposobnost generalizacije i izbjeglo prekomjerno prilagođavanje. Prekomjerno prilagođavanje se događa kada model dobro radi na podacima za obuku, ali loše na neviđenim podacima.
7. **Rano zaustavljanje**: Uvježbavanje se može zaustaviti kada izvedba modela na validacijskom skupu podataka počne slabiti, što ukazuje na prekomjerno prilagođavanje. To sprječava model da zapamti podatke o obuci i potiče ga na bolju generalizaciju.
8. **Testiranje**: Nakon završetka obuke, performanse modela se procjenjuju na zasebnom skupu podataka testiranja kako bi se pružila nepristrana procjena njegove točnosti i generalizacija na neviđene podatke.
9. **Implementacija**: Nakon što je model obučen i testiran na zadovoljavajući način, može se implementirati za predviđanje novih, neviđenih podataka u stvarnim aplikacijama.

Važna razmatranja tijekom učenja neuronske mreže uključuju izbor hiperparametara (npr. stopa učenja, broj skrivenih slojeva, broj neurona po sloju) i arhitekturu same neuronske mreže (npr. vrstu aktivacijskih funkcija, broj slojeva, tehnike regulacije).

Proces obuke se nastavlja sve dok model ne konvergira u stanje u kojem je gubitak sveden na najmanju moguću mjeru, što ukazuje da je naučio napraviti točna predviđanja na podacima o obuci. Pravilna obuka zahtijeva ravnotežu između dobrog uklapanja podataka o obuci uz izbjegavanje pretjeranog prilagođavanja (engl. *overfitting*) i osiguravanja generalizacije na nove podatke. To često uključuje fino podešavanje hiperparametara i pažljivo praćenje procesa treninga.

2.3 Vrste neuronskih mreža

Neuronske mreže dolaze u različitim arhitekturama, a svaka je dizajnirana za specifične zadatke i vrste podataka. Ovdje su neke od uobičajenih vrsta neuronskih mreža:

1. **Feedforward neuronska mreža** (engl. *Feedforward Neural Network*, FNN): Također poznat kao višeslojni perceptron (engl. *Multilayer perceptron,* MLP). Sastoji se od ulaznog sloja, jednog ili više skrivenih slojeva i izlaznog sloja. Koristi se za širok raspon zadataka, uključujući regresiju i klasifikaciju.

2. **Konvolucijska neuronska mreža** (engl. *Convolutional Neural Network,* CNN): Posebno dizajnirana za obradu podataka nalik mreži, kao što su slike i videozapisi. Koristi konvolucijske slojeve za automatsko učenje hijerarhijskih značajki. Pogodan za zadatke poput klasifikacije slika, otkrivanja objekata i segmentacije slika.
3. **Rekurentna neuronska mreža** (engl. *Recurrent Neural Network,* RNN): Dizajnirana za sekvencijalne podatke, kao što su vremenske serije, jezik i govor. Sadrži rekurentne veze koje dopuštaju postojanost informacija tijekom vremena. Učinkovito za zadatke poput jezičnog modeliranja, prepoznavanja govora i analize osjećaja ili značenja.
4. **Mreža dugog kratkoročnog pamćenja** (engl. *Long Short-Term Memory,* LSTM): Specijalizirana vrsta RNN-a dizajnirana za ublažavanje problema nestajanja gradijenta. Koristi zatvorene jedinice za hvatanje dugotrajnih ovisnosti u sekvencama. Široko se koristi u zadacima kao što su strojno prevođenje, prepoznavanje govora i predviđanje vremenskih serija.
5. **Ograđena Rekurzivna Jedinica** (engl. *Gated Recurrent Unit,* GRU) mreža: Slično LSTM-ovima, ali s pojednostavljenim mehanizmom zatvaranja. Lakši za treniranje i zahtjeva manje parametara od LSTM-ova. Koristi se u aplikacijama kao što su obrada prirodnog jezika i prepoznavanje govora.
6. **Autoenkoder** (engl. *Autoencoder,* AE): Sastoji se od kodera i dekodera. Koristi se za smanjenje dimenzionalnosti, učenje značajki i generativne zadatke. Varijante uključuju autoenkodere za uklanjanje šuma i varijacijske autoenkodere (engl. *Variational Autoencoder,* VAE).

7. **Mreža radijalne osnovne funkcije** (engl. *Radial Basis Function,* RBF): Koristi radijalne osnovne funkcije kao aktivacijske funkcije. Prikladno za aproksimaciju funkcije, interpolaciju i grupiranje.
8. **Samoorganizirajuća karta** (engl. *Self-Organizing Map,* SOM): Nenadzirana neuronska mreža koja se koristi za grupiranje i vizualizaciju visokodimenzionalnih podataka. Organizira podatkovne točke u strukturu nalik mreži na temelju sličnosti.
9. **Hopfieldova mreža**: Asocijativna memorijska mreža koja se koristi za prepoznavanje uzoraka i adresabilnu memoriju sadržaja. Pohranjuje i dohvaća uzorke na temelju naučenih veza.
10. **Boltzmannov stroj** (engl. *Boltzmann Machine,* BM): Vrsta stohastičke neuronske mreže s vidljivim i skrivenim jedinicama. Koristi se za različite zadatke učenja bez nadzora, uključujući ograničene Boltzmannove strojeve i mreže dubokog uvjerenja.
11. **Generativna kontradiktorna mreža** (engl. *Generative Adversary Network,* GAN): Sadrži generator i diskriminatorsku mrežu. Koristi se za generiranje realističnih podataka, kao što su slike i tekst.
12. **Transformatori**: Dizajniran za zadatke obrade jezika. Koristi mehanizme pažnje za učinkovito rukovanje sekvencama. Doveo do značajnog napretka u NLP-u, uključujući modele poput BERT-a i GPT-a.

Ovo su samo neke od mnogih vrsta neuronskih mreža koje su se razvijale tijekom godina. Odabir arhitekture neuronske mreže ovisi o specifičnom problemu i karakteristikama podataka, budući da su različite arhitekture prikladne za različite zadatke i vrste podataka. Istraživači nastavljaju istraživati nove mrežne arhitekture i prilagodbe za rješavanje raznih izazova u strojnom učenju i umjetnoj inteligenciji.

3. UMJETNA INTELIGENCIJA I DETEKCIJA OBJEKATA

Ovo poglavlje dublje zadire u funkcioniranje umjetne inteligencije pri obradi vizualnih podataka.

3.1. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (CNN ili ConvNets) klasa su modela dubokog učenja posebno dizajniranih za obradu podataka nalik mreži, s primarnim fokusom na zadatke poput analize slike i videa. Vrlo su uspješni u računalnom vidu, a imaju i primjenu u raznim drugim domenama. CNN-ovi su inspirirani ljudskim vizualnim sustavom, a njihova arhitektura uključuje specijalizirane slojeve za automatsko učenje hijerarhijskih značajki iz ulaznih podataka. Konvolucijske neuronske mreže čine:

- **Konvolucijski sloj**:

U središtu CNN-a nalaze se konvolucijski slojevi koji izvode operaciju konvolucije. Konvolucija uključuje primjenu filtra (koji se također naziva kernel) na ulaz, množenje po elementima i zbrajanje rezultata. Filtar je postavljen preko ulaza i na svakoj poziciji izračunava ponderirani zbroj lokalnih ulaznih vrijednosti. Ova operacija pomaže u otkrivanju lokalnih uzoraka ili značajki unutar ulaznih podataka. Filter: mala matrica (npr. 3x3 ili 5x5) koja se postavlja preko ulaznih podataka. Mapa značajki (engl. *feature maps*): Izlaz primjene filtra na ulaz naziva se mapa značajki. Svaki filtar uči otkrivati određenu značajku. Konvolucijski slojevi obično se sastoje od više filtara, od kojih svaki hvata različite značajke. Tijekom obuke, CNN-ovi uče optimalne vrijednosti za te filtre putem širenja unatrag.

- **Sloj sažimanja/objedinjujući sloj** (engl. Pooling *Layer)*:

Slojevi sažimanja koriste se za smanjenje prostornih dimenzija (širine i visine) mape značajki uz zadržavanje bitnih informacija. Najčešća vrsta sažimanja je maksimalno sažimanje (engl. *max pooling*), gdje se maksimalna vrijednost unutar malog prozora (npr. 2x2 ili 3x3) zadržava, a ostatak se odbacuje. Sažimanje pomaže u smanjenju računalnog opterećenja, smanjenju prekomjernog prilagođavanja i održavanju invarijantnosti prijenosa (prepoznavanje značajki bez obzira na njihov točan položaj).

- **Aktivacijska funkcija:**

Aktivacijske funkcije uvode nelinearnost u mrežu, omogućujući CNN-u da modelira složene odnose u podacima. Uobičajene aktivacijske funkcije koje se koriste u CNN-ovima uključuju ispravljenu linearnu jedinicu (ReLU), sigmoidu i hiperbolički tangens (tanh). ReLU je najčešće korišten zbog svoje jednostavnosti i učinkovitosti u obuci dubokih mreža.

- **Potpuno povezani sloj (gusti sloj, engl. *Dense layer*):**

Potpuno povezani slojevi tradicionalni su slojevi neuronske mreže gdje je svaki neuron povezan sa svakim neuronom u prethodnom sloju. Obično se koriste u kasnijim fazama CNN-a za kombiniranje značajki visoke razine naučenih konvolucijskim i skupnim slojevima. Nakon potpuno povezanih slojeva slijedi aktivacijska funkcija, često softmax u slučaju zadataka višestruke klasifikacije, kako bi se izračunala vjerojatnost klase.

- **Arhitektura:**

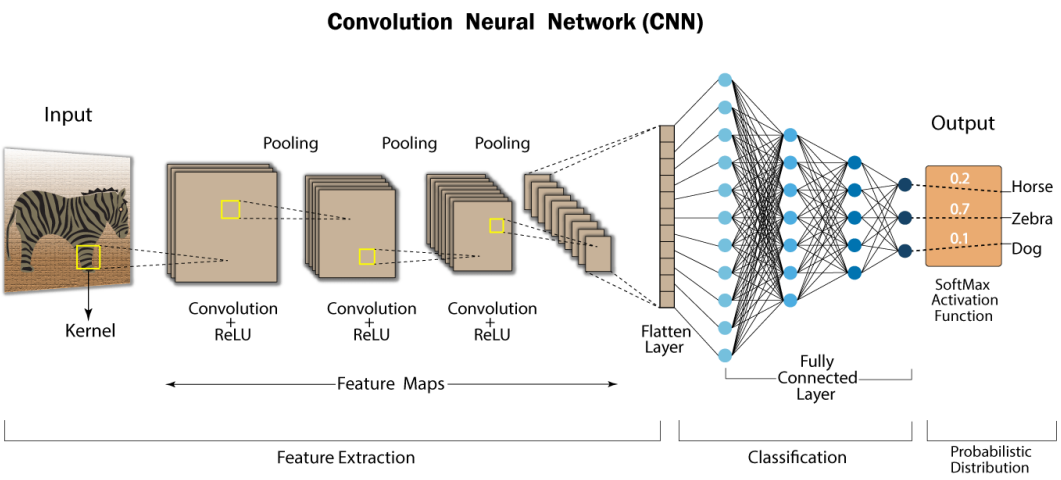
CNN-ovi često slijede specifičan obrazac arhitekture:

  - Ulazni sloj: prima neobrađene ulazne podatke, poput slike.
  - Konvolucijski slojevi: Sadrži više konvolucijskih filtara za otkrivanje značajki.
  - Sažimanje slojeva: Smanjuje prostorne dimenzije i zadržava važne informacije.
  - Potpuno povezani slojevi: Kombiniraju značajke visoke razine za zadatke klasifikacije ili regresije.
  - Izlazni sloj: Pruža konačno predviđanje ili izlaz.

Konvolucijski slojevi i slojevi sažimanja obično se slažu više puta kako bi formirali duboke mreže. Dublje mreže mogu uhvatiti sve apstraktnije značajke. Slika *4* sadrži prikaz strukture prosječnog CNN-a.

- **Preneseno učenje:**

CNN-ovi mogu iskoristiti unaprijed obučene modele na velikim skupovima podataka kako bi poboljšali izvedbu na određenim zadacima. Preneseno učenje uključuje fino podešavanje unaprijed obučenog CNN-a na manjem skupu podataka ili zadatku, što je posebno korisno kada su dostupni ograničeni označeni podaci.



Slika 4: Prikaz konvolucijske neuronske mreže (LinkedIn, 2023)

3.2. YoloV8

YOLO (engl. ***Y**ou **O**nly **L**ook **O**nce*) je popularan algoritam za otkrivanje objekata u stvarnom vremenu koji se ističe u otkrivanju i lokaliziranju više objekata u slikama ili videima. Za razliku od nekih tradicionalnih metoda detekcije objekata koje uključuju procese u više faza, YOLO usvaja jednofazni pristup, što ga čini vrlo učinkovitim i prikladnim za aplikacije u stvarnom vremenu.

YOLO dijeli ulaznu sliku u mrežu ćelija. Svaka ćelija je odgovorna za predviđanje graničnih okvira i vjerojatnosti klase objekata koji se nalaze unutar te ćelije. Mreža pojednostavljuje proces otkrivanja objekata rastavljanjem složenih scena na manja, lokalizirana predviđanja.

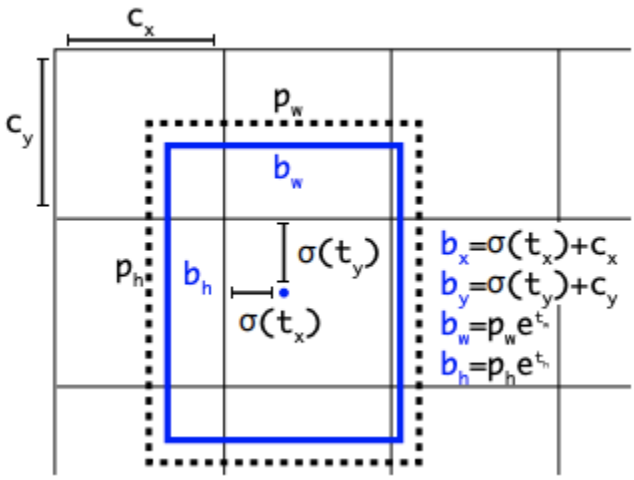
YOLO također koristi kvadratne okvire (takozvani sidreni okviri, engl. *anchor boxes*) za predviđanje veličine i oblika objekata unutar svake ćelije. Okviri su unaprijed definirani četverokuti graničnih okvira različitih omjera i veličina. YOLO predviđa dimenzije ovih okvira i prilagođava ih kako bi odgovarale otkrivenim objektima. To omogućuje YOLO-u učinkovito rukovanje predmetima različitih oblika i veličina.

Svaka ćelija u mreži predviđa granične okvire za objekte. Za svaki granični okvir, YOLO predviđa četiri vrijednosti:

- Središnje koordinate (x, y): koordinate središta okvira u odnosu na granice ćelije.
- Širina (w): širina okvira u odnosu na širinu ćelije.

- Visina (h): visina okvira u odnosu na visinu ćelije.

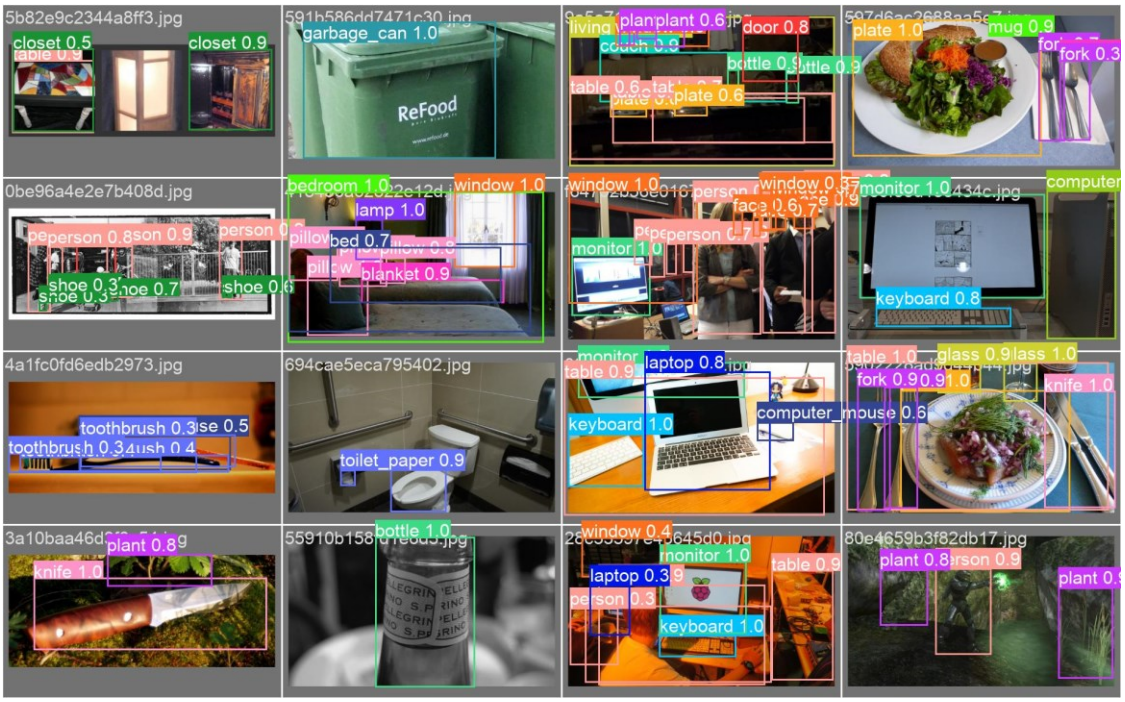
Koordinate (x, y) obično se odnose na ćeliju u kojoj se nalazi središte objekta, a širina i visina (w, h) predviđene su kao dio dimenzija ćelije mreže. Slika *5* je prikaz strukture sidrenih okvira unutar ćelija.



Slika 5: Primjer detekcije objekta u YOLO softveru(Medium, 2020)

Za svaku ćeliju YOLO također predviđa vjerojatnosti za prisutnost različitih klasa objekata. To se radi s pomoću funkcije aktivacije softmax. Broj klasa koje treba predvidjeti ovisi o specifičnoj aplikaciji, ali YOLO može predvidjeti više klasa istovremeno.

YOLO dodjeljuje ocjenu pouzdanosti svakom predviđanju sidrenog okvira, pokazujući koliko je vjerojatno da je objekt prisutan unutar tog okvira. Ovaj rezultat odražava (engl.) *intersection over union* (IoU) između predviđenog okvira i okvira temeljne istine. Viši rezultati pouzdanosti ukazuju na bolja predviđanja. Slika *6* prikazuje kako izgledaju slike nakon obavljenog predviđanja.



Slika 6: Primjer podataka korištenih u projektu

Nakon izrade predviđanja za svaku ćeliju i sidrenu kutiju, YOLO primjenjuje ne-maksimalno potiskivanje (engl. *Non-maximum suppression, NMS*) kako bi filtrirao duple detekcije ili detekcije niske pouzdanosti. NMS osigurava da se zadrže samo najpouzdaniji sidreni okviri koji se ne preklapaju, čime se uklanja suvišnost u konačnom izlazu.

YOLO se široko koristi u aplikacijama kao što su:

- Detekcija objekata na slikama i videozapisima,
- Nadzor i sigurnost u stvarnom vremenu,
- Autonomna vozila za detekciju pješaka i objekata,
- Robotika i navigacija dronovima,
- Proširena stvarnost i virtualna stvarnost za prepoznavanje objekata,
- Brojanje i praćenje objekata u prepunim scenama.

YOLO-ova sposobnost da izvrši točnu detekciju objekata u stvarnom vremenu u različitim domenama učinila ga je vrijednim alatom u aplikacijama računalnog vida i umjetne inteligencije.

4.5. PRAKTIČNI RAD - Neuronske mreže i njihova primjena u detekciji objekata

Cilj ovog softvera je omogućiti računalu da ispravno detektira objekte te kreirati "auditorno sučelje" za slabovidne osobe.

4.1. Plan rada

Projekt počinje određivanjem onoga što je potrebno za pomoć osobama s oštećenjem vida. Odabrani objekti uglavnom su oni koji bi se mogli naći unutar nečijeg doma, ali sadrži i neke koji bi mogli biti korisni za vanjsku navigaciju.

Na sljedećoj tablici se nalaze sve klase koje su odabrane za korištenje u ovom projektu:

Tablica 1: Popis klasa za projekt

ID	naziv klase	hrvatski prijevod
0	chair	stolica
1	table	stol
2	window	prozor
3	kitchen	kuhinja
4	living room	dnevna soba
5	bedroom	spavaća soba
6	computer	računalo
7	monitor	ekran
8	shoe	obuća
9	phone	mobitel
10	wallet	novčanik
11	keys	ključevi
12	bed	krevet
13	toothbrush	četkica za zube
14	bathroom	kupaona
15	lamp	svijetiljka
16	bag	vreća
17	plant	biljka
18	pillow	jastuk
19	blanket	pokrivač
20	car	automobil
21	person	osoba
22	face	faca
23	plate	tanjur

24	glass	čaša
25	mug	šalica
26	vase	vaza
27	bottle	boca
28	glasses	naočale
29	painting	slika
30	garbage can	koš za smeće
31	can	limenka
32	remote	daljinski upravljač
33	toilet	WC školjka
34	shower	tuš
35	bathtub	kada
36	TV	televizija
37	fork	vilica
38	knife	nož
39	spoon	žlica
40	door	vrata
41	doorknob	kvaka
42	fridge	hladnjak
43	joystick	joystick
44	paper roll	papirnati ubrus
45	pan	tava
46	sink	umivaonik
47	stove	štednjak
48	closet	ormar
49	couch	kauč
50	carpet	tepih
51	keyboard	tipkovnica
52	computer mouse	miš za računalo
53	toilet paper	toaletni papir
54	laptop	prijenosno računalo

Nakon što su objekti odabrani, sljedeći korak je prikupljanje slika koje će se koristiti kao skup podataka. Postoje online resursi koji imaju velike zbirke već označenih podataka. Resurs koji je korišten za ovaj projekt je *"OpenImages"* koji posjeduju 600 klasa objekata i otprilike 16 milijuna sidrenih kutija na pola milijuna slika.

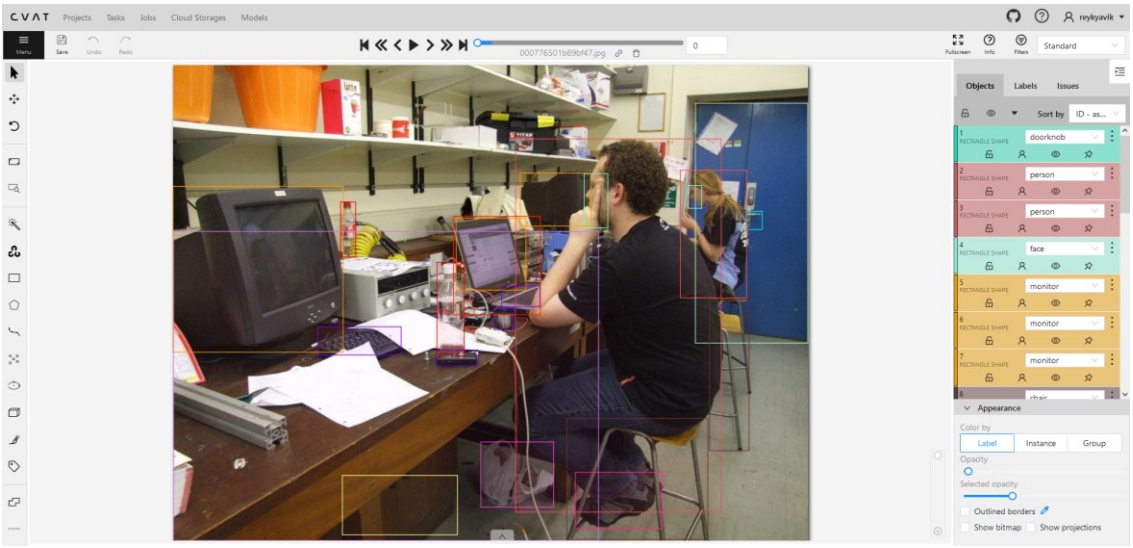
4.2. Odabir neuronske mreže

Razlog za korištenje YOLO obitelji neuronskih mreža je činjenica da su njeni rezultati i izvedba bili precizni, s čestim ažuriranjem se pokazalo jako dobrim za rješavanje postavljenog projekta.

Međutim, problem koji postoji je taj što unaprijed obučeni modeli nisu bili obučeni na dovoljno velikom skupu podataka da bi bili pogodni. Nadalje, veći je problem što zadani model ne sadrži potrebne objekte koje je trebalo detektirati u svrhu projekta, a ima one koji nisu potrebni.

Za svaku klasu je odabrano 20 slika koje sadrže tu klasu. Druge klase su također prisutne na tim slikama kako bi podaci bili brojniji. Bolje je imati više objekata na jednoj slici jer se uz pomoć toga stvara kontekstualno znanje o objektima. Uzimajući u obzir da ima 55 klasa trebalo bi biti 1100 slika međutim zbog duplikata taj broj se smanjio na 980.

Nakon preuzimanja slika potrebno je anotirati sve sidrene kutije koje se mogu prevući preko svakog objekta na pojedinim slikama. Za ovaj korak pomaže alat koji se zove CVAT (engl. *Computer Vision Annotation tool*, to jest Anotacijski Alat za Računalni Vid). Korisnost ovog alata je u tome što može ubrzati i olakšati proces postavljanja sidrenih kutija na slike. To se radi tako da se slike učitaju preko stranice, što znači da nema potrebe za preuzimanje ikakvog softvera. Kada se slike učitaju, na raspolaganju su brojni alati s kojima se mogu objekti označiti. Moguće je koristiti sidrene kutije, ili ako je to potrebno mogu se označiti na precizniji način, tako da se ciljano može obuhvatiti objekt bez ikakve okoline. Iako taj način zvuči preciznije, nije nužno bolji od sidrenih kutija. Taj način bi znatno povećao količinu rada za svaku sliku, ali čak i da to nije problem ne podudara se s ciljevima projekta jer nema intuitivnog načina kako bi se predstavio oblik i orijentacija svakog objekta kroz tekst. Na slici (Slika *7*) je prikazano sučelje koje se koristi za anotaciju slika gdje svaka sidrena kutija predstavlja označeni objekt. Različite boje predstavljaju različite klase te na desnoj strani se vidi kojoj klasi pripadaju.

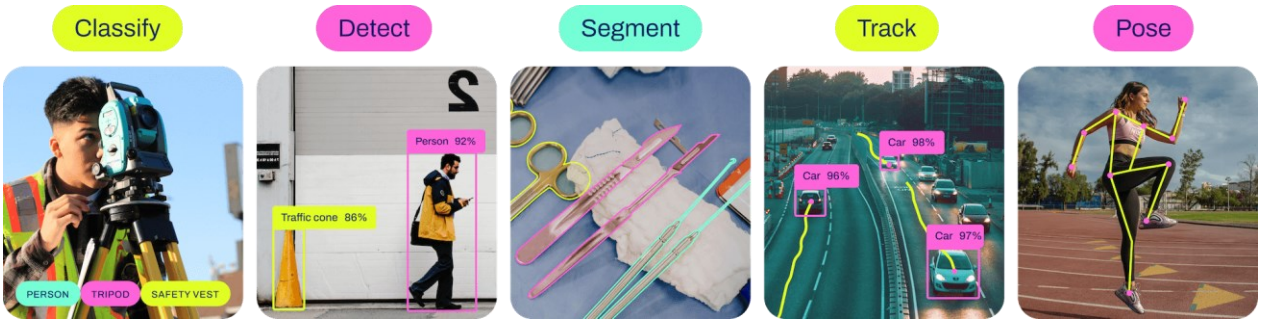


Slika 7: Prikaz alata za anotaciju podataka

Nakon što se završi postupak označavanja svih slika, sljedeći korak je dovršiti obuku nad podacima koji su stvoreni.

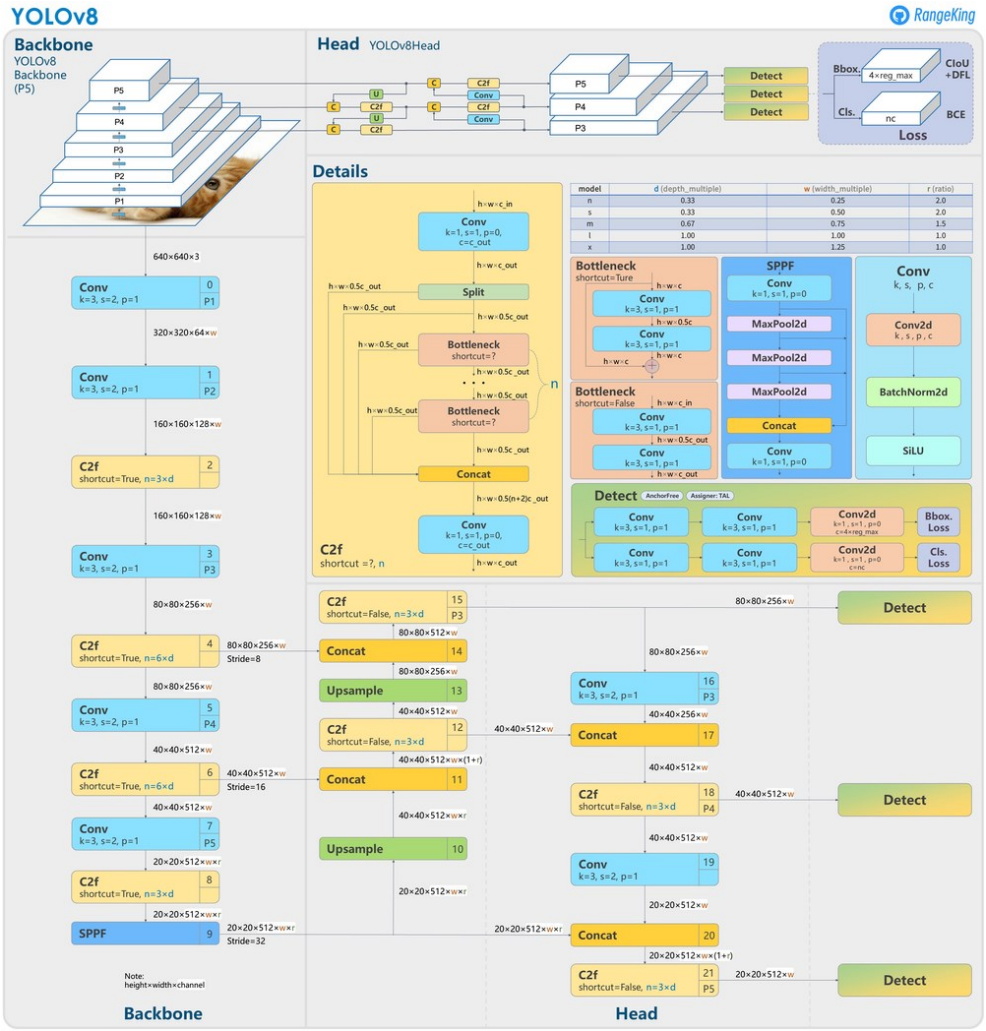
4.3. Implementacija YOLOV8 softvera

Razvoj i napredak softvera za umjetnu inteligenciju i otkrivanje objekata postao je sve popularniji način rješavanja problema koji bi inače zahtijevali složene algoritme i zahtijevali puno više vremena za kodiranje. YOLO omogućuje programerima upotrebu optimizirane neuronske mreže koja je specijalizirana za otkrivanje objekata, otkrivanje poza, praćenje objekata itd. Na slici (Slika *8*) su prikazane sve funkcionalnosti koje YOLO omogućuje.



Slika 8: Prikaz sposobnosti YOLO softvera (Ultralytics, 2024)

danas YOLOV8 predstavlja najprecizniji softver otvorenog koda za otkrivanje objekata s poboljšanjima od otprilike 20-30% u odnosu na prethodnika. To je nova verzija koja još uvijek nema vlastiti rad napisan o njoj kao prethodne verzije. Međutim, postoji izvorni kod i dijagrami koji se mogu pogledati kako bi se analizirao način poboljšanja i kako bi se unaprijedilo razumijevanje načina na koji model u ovom projektu funkcionira. Slika *9* pokazuje kako je strukturiran.



Slika 9: Arhitektura YOLOv8 modela (Medium, 2024)

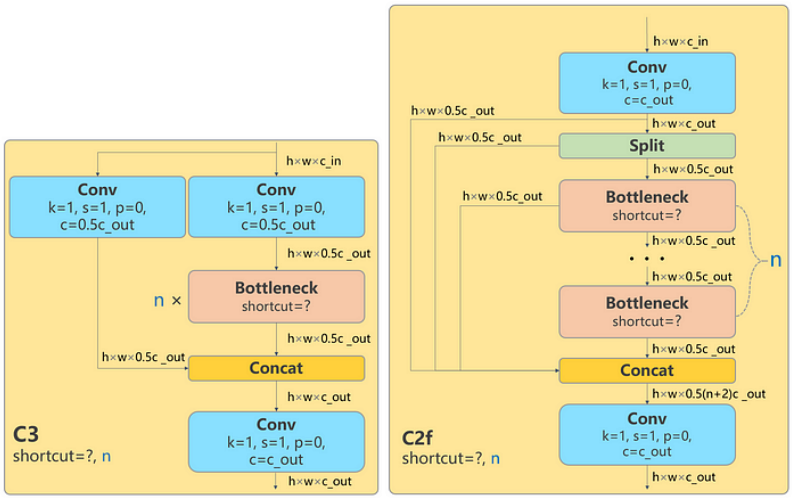
Struktura YOLO-a podijeljena je na 3 dijela, kralježnicu, vrat i glavu:

- Kralježnica ili okosnica (*backbone*) ima ulogu prepoznavanja uzoraka na više verzija prilagođene slike (dimenzije slike se mijenjaju kao što se vidi na slici 9). Uzorci koje prepoznaje su rubovi, teksture i drugi uzorci na različitim apstrakcijama slike.

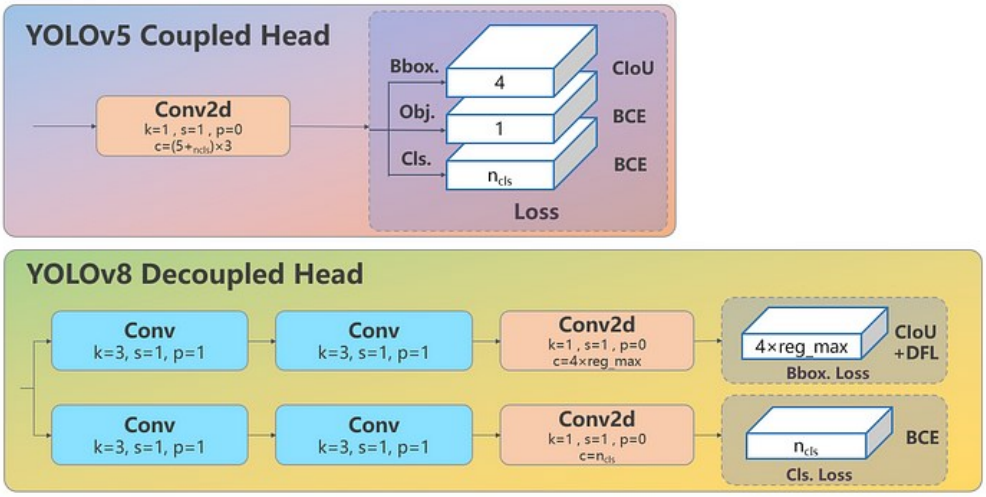
- Vrat (engl. *Neck*) povezuje kralježnicu s glavom i ima funkciju stvaranja "piramide značajki" kombiniranjem mapa značajki koje stvara kralježnica. Također pruža kontekstualne informacije o svakom objektu u odnosu na njegovu okolinu (formiranje odnosa između objekata npr. WC školjke su obično u kupaonicama).
- Glava (engl. *Head*) je odgovorna za generiranje izlaza, a to su sidreni okviri te njihova ocjena predviđanja, izražena u postotku.

Postoji više promjena i optimizacija koje su napravljene od prethodne iteracije YOLO-a, kao što su:

- Zamjena C3 modula za C2f modul. C"N" moduli označavaju CSP (engl. *Cross Stage Partial*) module uskog grla (usko grlo je prijevod za 3x3 konvoluciju s preostalom vezom) gdje N predstavlja broj korištenih konvolucija. To znači da C2f ima 2 konvolucije, plus f koje označava broj značajki. Slika *10* pruža bolji pogled na strukture.
- Sidrene kutije su ažurirane, model sada predviđa gdje je središte objekta, umjesto okvira ili rubova sidrene kutije. Ovo čini posao jednostavnijim, a time i učinkovitijim u smislu računanja smanjenjem broja predviđanja okvira, što je postignuto stvaranjem nove verzije modula „glave“ (engl. *Head*). Modul *Head* glavni je razlog zašto je to moguće jer je promijenjen u razdvojenu (engl. *decoupled*) strukturu, što omogućuje razdvajanje klasifikacije i detekcije. Razlike su opisane na slici (Slika *11*), na kojoj se vidi da se podjela posla radi prije konvolucije u YOLOV8 a tek nakon konvolucije u YOLOV5. Također je izbačena (engl.) *Objecteness* grana (Obj. Grana na desnoj strani slike 11)
- Promjene u izračunu gubitka (engl. *Loss*, sada s pomoću (engl.) *TaskAlignedAssignera*) i regresijskog gubitka (s pomoću (engl.) *Distribution Focal Loss, DFL*). Vlasnici YOLOv8 (*Ultralytics*) ne dijele izvorni kod za funkcije gubitka koje koriste dakle nije poznato što su promijenili u funkcionalnosti, no podatci pokazuju da su rezultati pozitivni.
- Ostale promjene kao što je prosječni iznos epoha (povećan je s 300 na 500 kako bi se osigurali bolji rezultati), povećanje podataka (stvaranje novih podataka iz postojećih, tj. zrcaljenje slika, njihovo kombiniranje itd.), povećanje mozaika (ovo je zaustavljeno za zadnjih 10 epoha jer može biti štetno za model).



Slika 10: Strukturna razlika između C3 i C2f modula (Medium, 2024)



Slika 11: Razlika razdovjenog i nerazdvojenog "Head" modula (Medium, 2024)

4.4. Izrada modela

Izrada modela je najlakši dio cijele operacije. Potrebno je samo nekoliko stvari za početak:

- Yaml datoteka koja sadrži lokaciju (na disku) skupova podataka za obuku, testiranje i sve tekstualne datoteke povezane s anotacijama za slike. U istima se nalaze i popisi klasa i njihovih ID-ova.

- Sljedeći korak je određivanje količine epoha koje bi bile optimalne (preporučeno 500), nakon čega je potreban samo red koda (Kod 1) za početak procesa obuke.

```
model.train(data=”coco128.yaml”, epochs = 100)
```

Kôd 1: linija koda za pokrenuti treniranje modela

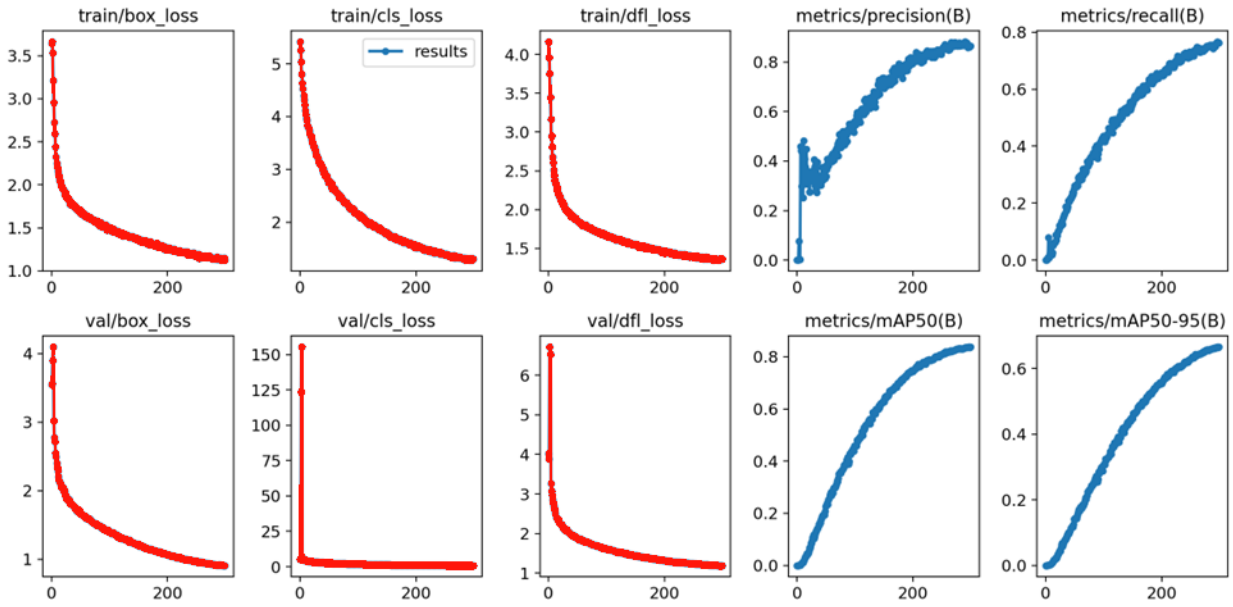
Za ovaj projekt korišteno je 300 epoha za treniranje modela, za što je bilo potrebno otprilike 70 sati. Ovaj proces obuke može se pauzirati i nastaviti u bilo kojem trenutku. Nakon što je model obučen, lako je učitati model koji će se koristiti kad god je to potrebno s pomoću iduće linije koda.

```
model = YOLO(“E:/Personal Data/Lokacija modela/model.pt”)
```

Kôd 2: dohvaćanje modela za upotrebu

Nakon što se model uvježba, može se koristiti za potrebe ovog projekta ili se može koristiti za daljnje poboljšanje samoga sebe kroz stvaranje šireg skupa podataka. To se može učiniti uzimanjem novih podataka i puštanjem modela da ih predvidi, nakon čega bi korisnik morao provjeriti kada je model stvorio dobre a kada loše podatke (tj. je li predviđanje bilo točno ili ne). Kako korisnik ne bi morao ručno označavati nove podatke, potrebno je samo izraditi relativno stabilan model i tako ga kontinuirano nadograđivati. Sve što bi bilo potrebno jest stvoriti izlaz koji se može potvrditi i vratiti kao ulaz, a zatim samo uvježbati model s novim podacima.

Svaki put kada je model gotov s obukom, YOLO generira neke korisne grafikone za prikaz napredovanja u kvaliteti modela kroz više parametara. Na slici (Slika *12*) predstavljeni su rezultati treniranja modela korištenog za ovaj projekt.



Slika 12: Prikaz rezultata učenja modela

Na ovoj slici nalazi se 10 različitih grafikona koji predstavljaju različite aspekte metrike gubitka i preciznosti koje model ima tijekom procesa obuke. Grafikoni imaju broj epoha na x osi i vrijednosti za svaki atribut na y osi. Opće pravilo za ove grafikone je da se gubitak smanjuje s brojem epoha, dok preciznost raste jer je to naznaka da model ispravno uči.

Grafikoni se mogu podijeliti u 2 odjeljka, 6 crvenih i 4 plavih. Crveni se odnose na gubitak i podijeljeni su na gornja 3 koja su iz podataka o obuci (engl. *train*) i na donja 3 (engl. *val*) koja su iz validacijskih podataka. Plavi se odnose na preciznost predviđanja modela.

Imena predstavljaju sljedeće:

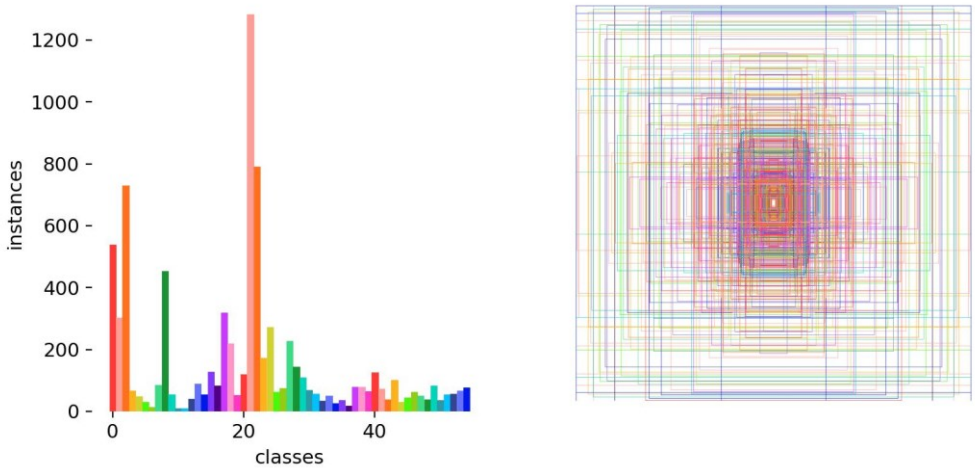
- **box_loss** - Regresijski gubitak sidrenog okvira. Koristi se srednja kvadratna pogreška (engl. *Mean Squared Error*) da bi se pronašlo koliki je pomak predviđenog okvira u usporedbi sa stvarnim okvirom podataka,

- **cls_loss** - Predstavlja koliko često model zamijeni jednu klasu za drugu,
- **dfl_loss** - Distribucijski fokalni gubitak (engl. *Distribution Focal Loss*), ovo je još jedna funkcija koja se koristi za regresiju sidrenog okvira.

Grafikoni preciznosti (plavi) predstavljaju preciznost i točnost (engl. *Recall*). Ove dvije koriste različite formule, ali obje koriste iste parametre koji su definirani kao x, y i z. „x“ predstavlja istinske pozitivne rezultate (objekt je bio tamo i otkriven je od strane modela), „y“ predstavlja lažno pozitivne (objekt nije bio tamo, ali je otkriven), a „z“ predstavlja lažno negativne rezultate (objekt je bio tamo, ali nije otkriven). Za graf preciznosti formula bi bila $\frac{x}{x+y}$, a graf točnosti $\frac{x}{x+z}$. To znači da preciznost izračunava koliko je predviđanja graničnih okvira točnih, a točnost mjeri koliko je istinitih sidrenih okvira bilo točno predviđeno. Značenje iza mAP50 je srednja prosječna preciznost (mAP) na IoU (engl. *Intersection over Union*), prag od 0.5 (ovo u osnovi predstavlja koliko je predviđeni okvir ispravan u usporedbi s okvirom podataka, to jest predstavlja njihovo preklapanje). MAP_0.5:0.95 je prosječni mAP preko različitih IoU pragova, u rasponu od 0.5 do 0.95.

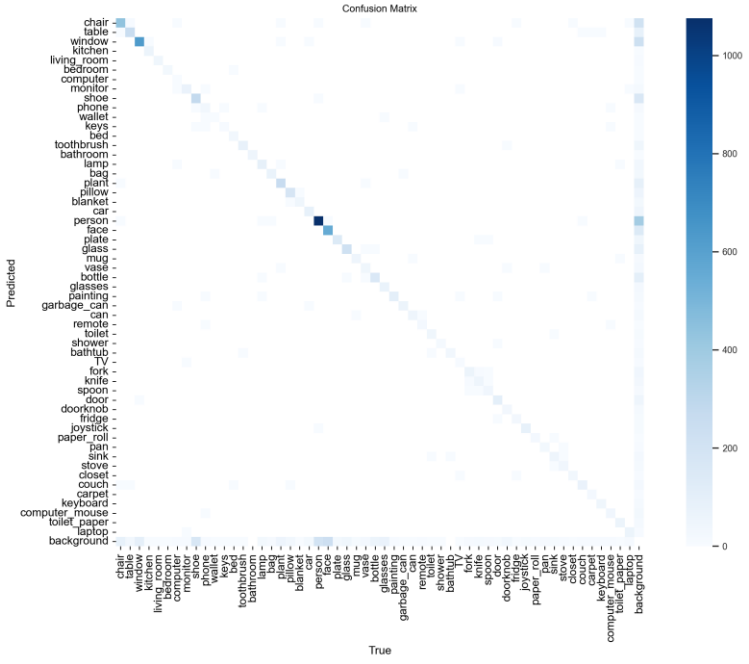
4.5. Rezultati učenja

Nakon učenja također se stvore grafovi s podacima o preciznosti modela. Slika *13* ima graf s brojem instanci svih klasa u modelu. Imena klasa nisu na grafu, ali su u istom poretku kao i objekti na slikama (Slika *14* i Slika *15*). Najbrojnija klasa je klasa osoba, zatim faca, prozor, stolica, cipela. Na desnoj strani slike su sve sidrene kutije iz podataka. Sveukupan broj svih instanci je 7937 (1282 od tih instanci spada pod osobe). Za YOLOv8 je preporučeno 2000 instanci za objekt kako bi se osigurala preciznost od otprilike 95%. Prema tom pravilu broj instanci za objekte u ovom projektu bi trebao biti 110 000 kako bi se osigurala odlična detekcija svih objekata.



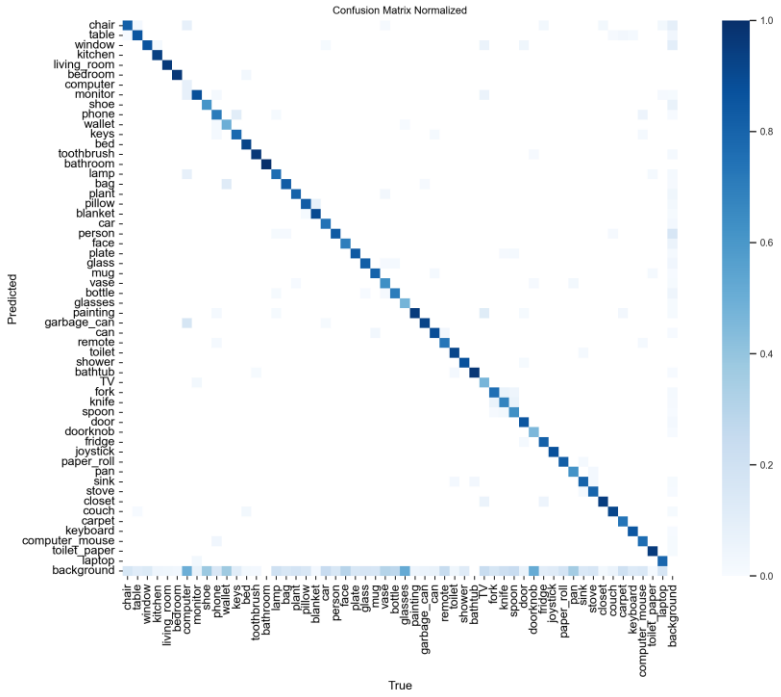
Slika 13: Broj instanci klasa i njihove pozicije na slikama

Matrica zabune (Slika *14*) je alat koji se koristi za procjenu izvedbe modela detekcije objekata. To je tablica koja uspoređuje stvarne oznake objekata u skupu podataka s oznakama predviđenim modelom. Ovaj grafikon predstavlja sve prave pozitivne, lažno pozitivne, prave negativne i lažno negativne rezultate koje je stvorio model. Prisutne su sve klase objekata uz dodatak još jedne klase, pozadine. Ovo pomaže u razumijevanju koliko dobro model radi u smislu ispravnog identificiranja objekata i gdje bi mogao činiti pogreške (npr, koji se objekti često pogrešno zamjenjuju jedni s drugima).



Slika 14: Matrica zabune

Normalizirana verzija grafa pomaže u prikazu koji objekti imaju najgore performanse u usporedbi jedni s drugima.

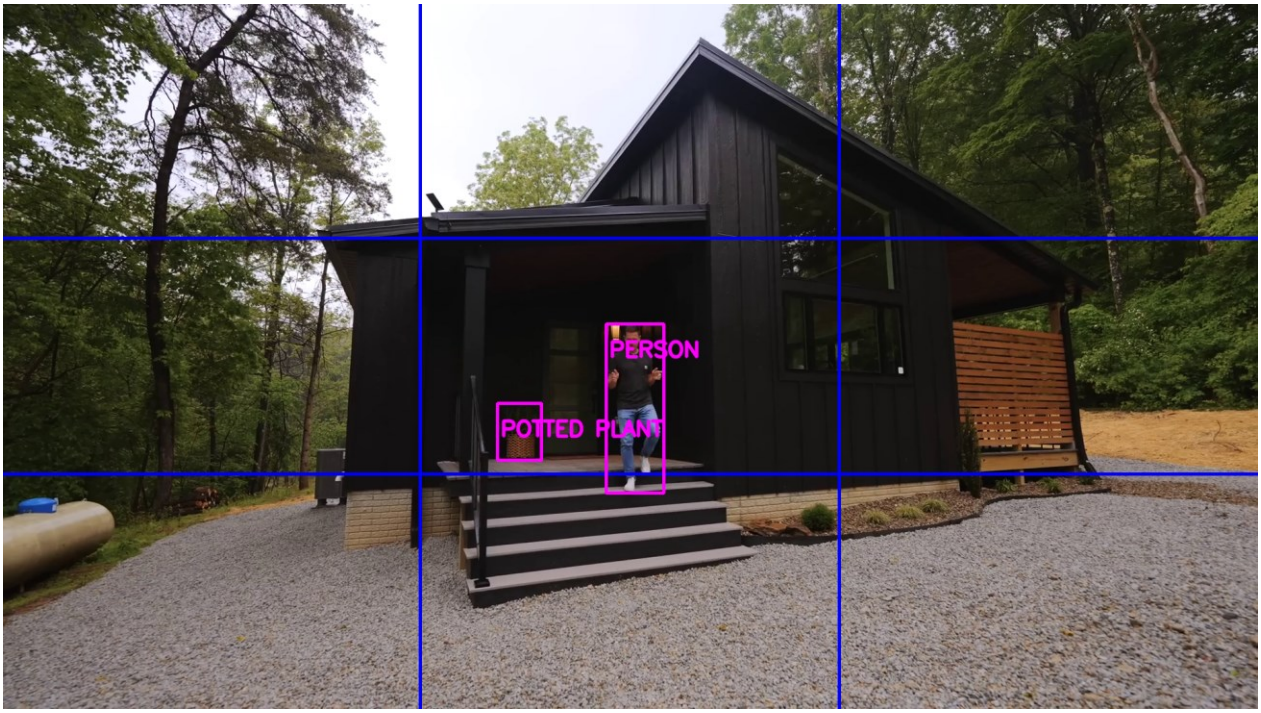


Slika 15: Normalizirana matrica zabune

4.6. Logika segmentacije videozapisa

Nakon što je potvrđeno da model funkcionira i detektira objekte na videozapisu sa zadovoljavajućom preciznošću preostaje problem komuniciranja generiranih podataka korisniku. S obzirom na to da je fokus u ovom projektu na pomaganju slabovidnim ili slijepim ljudima cilj bi bio predstaviti zvukovne informacije na dinamičan i precizan način.

Poanta ovog projekta je omogućiti stroju prevođenje vizualnih podataka u audio podatke na učinkovit način. Budući da je dio otkrivanja objekata gotov, može se prijeći na pretvaranje tih informacija u probavljiv način. Glavna primijenjena strategija je distribucija zaslona na 9 jednakih dijelova raspoređenih u mrežu (Slika *16* je primjer te distribucije).



Slika 16: Prikaz segmentacije video podataka za obradu u audio format

Budući da su sidrene kutije za svaki objekt dostupne, sljedeći korak je utvrditi gdje svaka sidrena kutija najbolje pristaje u jedan ili više odjeljaka od 9 odjeljaka mreže. Najjednostavniji i najučinkovitiji način za to je pronalaženje lokacije najgornjeg lijevog i najdonjeg desnog ruba sidrene kutije. Kada se zna koji odjeljak mreže zauzimaju te 2 točke, jednostavno je odrediti sve presječene dijelove (npr. ako je gornja lijeva točka u dijelu 1, a donja desna točka u dijelu 5, zaključak je da ova sidrena kutija dodiruje dijelove 1, 2, 4 i 5).

Idući problem koji se pojavljuje je problem lokacije sidrenog okvira, npr. na slici 9, gdje sidreni okvir osobe pokriva odjeljke 5 i 8. Ovaj sidreni okvir ima središnju točku (jednake udaljenosti od svih bridova) koja se nalazi u jednom od ova dva odjeljka. U kojem god da se većina sidrenog okvira nalazi u tom području. Ovo je rješenje naizgled najefikasnija poziciona metoda koja se može koristiti s podacima koji su na raspolaganju. Ovo se može poboljšati daljnjom segmentacijom zaslona, ali samo se proširuje problem pretvorbe tih informacija u razumljive tekstualne informacije. Slika *17* prikazuje kôd koji izvršava te operacije.

```
def find_location_of_rect(width, height, list_of_rect_coords, name_of_item):
    width_third = int(width/3)
    height_third = int(height/3)
    square_coords_in_camera = [[0, 0], [width_third, height_third], [width_third * 2, 0], [width, height_third],
                               [0, height_third * 2], [width_third, height], [width_third * 2, height_third * 2],
                               [width, height]]
    # prva kocka je izmedu nulte i prve tocke itd...
    # 0,1 : 1,2 : 2,3 : 1,4 : 1,6 : 3,6 : 4,5 : 5,6 : 6,7
    nonadrant_point_couples = [[0,1],[1,2],[2,3],[1,4],[1,6],[3,6],[4,5],[5,6],[6,7]]
    middle_of_rect = [int(list_of_rect_coords[0] + (list_of_rect_coords[2] - list_of_rect_coords[0]) / 2),
                      int(list_of_rect_coords[1] + (list_of_rect_coords[3] - list_of_rect_coords[1]) / 2)]

    for nonadrant in nonadrant_point_couples:
        if between(list_of_rect_coords[0], square_coords_in_camera[nonadrant[0]][0],
                   square_coords_in_camera[nonadrant[1]][0]) and between(list_of_rect_coords[1],
                                                                         square_coords_in_camera[nonadrant[0]][1],
                                                                         square_coords_in_camera[nonadrant[1]][1]):
            print("x1: {}, y1: {}".format(list_of_rect_coords[0], list_of_rect_coords[1]))
            nonadrant_of_first_point = nonadrant_point_couples.index(nonadrant) + 1
            print("first point of the rectangle is in the {} nonadrant".format(nonadrant_of_first_point))
        if between(list_of_rect_coords[2], square_coords_in_camera[nonadrant[0]][0],
                   square_coords_in_camera[nonadrant[1]][0]) and between(list_of_rect_coords[3],
                                                                         square_coords_in_camera[nonadrant[0]][1],
                                                                         square_coords_in_camera[nonadrant[1]][1]):
            print("x2: {}, y2: {}".format(list_of_rect_coords[2], list_of_rect_coords[3]))
            nonadrant_of_second_point = nonadrant_point_couples.index(nonadrant) + 1
            print("second point of the rectangle is in the {} nonadrant".format(nonadrant_of_second_point))
        if between(middle_of_rect[0], square_coords_in_camera[nonadrant[0]][0],
                   square_coords_in_camera[nonadrant[1]][0]) and between(middle_of_rect[1],
                                                                         square_coords_in_camera[nonadrant[0]][1],
                                                                         square_coords_in_camera[nonadrant[1]][1]):
            print("middle_x: {}, middle_y: {}".format(middle_of_rect[0], middle_of_rect[1]))
            nonadrant_of_middle_point = nonadrant_point_couples.index(nonadrant) + 1
            print("middle point of the rectangle is in the {} nonadrant".format(nonadrant_of_middle_point))

    result = find_which_nonadrants_box_is_touching(nonadrant_of_first_point, nonadrant_of_second_point)
    #percent_of_nonadrant_that_the_box_covers(result, square_coords_in_camera)
    return start_text_gen(nonadrant_of_middle_point, name_of_item, result)
```

Slika 17: Kôd koji izvršava lokalizaciju objekata po segmentima

Jednom kada je lokacija objekta definirana, postupak stvaranja teksta koji će se čitati naglas korisniku je jednostavan. Kada korisnik pritisne gumb, on pravi snimku trenutnog stanja pregledanih objekata i očitava segmente u kojima se objekti nalaze. Nakon što to učini, generira jednostavan tekst koji opisuje gdje se na slici sve nalazi. Korištenjem ekstenzije pyttsx3 za TTS (engl. *text-to-speech)* ovaj će se tekst čitati dok se gumb drži. Kôd 3 prikazuje kako se dolazi do teksta pozivanjem kôda sa slike 17.

```
if score > threshold:
    cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (255,
0, 255), 3)

    text_to_say = find_location_of_rect(int_width, int_height, [x1,
y1, x2, y2],    class_name_dict[int(class_id)])
    if keyboard.is_pressed('s'):
        engine.say(text_to_say)
        engine.runAndWait()
```

Kôd 3: Pozivanje kôda za detekciju objekata, pretvorbu u audio format i ispis informacija

Ipak, postoji prostor za poboljšanje, postoji više mogućnosti za poboljšanje trenutnog stanja projekta dodavanjem jednog ili više od sljedećih rješenja.

4.7. Poboljšanja

Glavni problem kod ovog modela je to što ne postoji način za prepoznavanje dubine, to jest udaljenosti određenih objekata u prostoru.

- Rješenje s više kamera: To bi se moglo zaobići korištenjem više kamera i triangulacijom položaja svakog od objekata. Ovo rješenje izgleda dovoljno jednostavno na površini, ali stvara više problema nego što ih rješava. Nakon što se uvede više kamera, glavni problem koji se javlja je točnost modela da prepozna neki objekt iz više perspektiva (ovo također ovisi o tome koliko su dvije kamere udaljene, ne samo o modelu).
- Model predviđanja udaljenosti: Još jedno rješenje za koje postoji argument je podučavanje zasebnog modela za predviđanje udaljenosti svakog objekta na temelju još više podataka koji sadrže ne samo objekte koje je potrebno otkriti, već i udaljenost na kojoj se nalaze. Nedostaci ovog rješenja pojavljuju se kada određeni objekti izgledaju drugačije, ali pripadaju istoj klasi (npr. model treniran na udaljenosti od plavih automobila možda neće moći točno predvidjeti udaljenost crvenih automobila).

Ovo nije nerješiv problem budući da je sve što je potrebno je dovoljno velik uzorak podataka za svaki objekt, ali stvaranje skupa podataka predstavlja veliki zadatak jer bi zahtijevalo mjerenje udaljenosti do svakog objekta, što podrazumijeva sve podatke bi trebalo prikupljati ručno i mjeriti ručno.

- Senzori udaljenosti: Ovo je rješenje naizgled najjednostavnije i zahtijevalo bi samo da korisnik usmjeri senzor u ispravnom smjeru kako bi bio okrenut prema objektu. To bi se postiglo usmjeravanjem korisnika da okrene kameru/senzor putem zvuka, međutim stvaranje više posla za korisnika nije optimalno i bilo bi bolje zaobići to ako je moguće. Iz tog razloga drugo rješenje koje uključuje senzore je privlačnije. Moguće je koristiti ove senzore za izradu 3D modela prostorija koje se često koriste, uz mapiranje i pohranu koordinata. Tu su i senzori koji imaju već ugrađeno 3D mapiranje uz negativnu stranu da je cijena viša.

Jedno od poboljšanja koje ne uključuje problem dubine bi bilo kreacija jezičnog modela koji bi dodao mogućnost preciznijeg opisa informacija (npr. relacijske informacije kao što je “Mobitel je na stolu“).

5. ZAKLJUČAK

Zaključno, ovaj je rad istražio transformativni potencijal umjetne inteligencije u području otkrivanja objekata, s posebnim naglaskom na njezine duboke implikacije za pomoć osobama s oštećenjima vida i poboljšanje robotskih sposobnosti. Kroz dubinsko ispitivanje temeljnih tehnologija i njihovih primjena, pokazuju se da sustavi za otkrivanje objekata pokretani umjetnom inteligencijom obećavaju poboljšanje kvalitete života za osobe s oštećenjem vida i unaprjeđenje sposobnosti robota u raznim domenama.

Za osobe s oštećenjem vida detekcija objekata vođena umjetnom inteligencijom predstavlja ogroman korak prema postizanju veće autonomije i neovisnosti. Korištenjem algoritama računalnog vida, nosivi uređaji opremljeni kamerama mogu pružiti prepoznavanje objekata u stvarnom vremenu, pomoć u navigaciji i kontekstualne informacije, omogućujući osobama oštećena vida da se kreću u nepoznatim okruženjima s povećanim povjerenjem i sigurnošću. Nadalje, integracija obrade prirodnog jezika omogućuje pretvaranje detektiranih objekata u slušnu povratnu informaciju, stvarajući intuitivnije i informativnije iskustvo.

U području robotike, otkrivanje objekata s pomoću umjetne inteligencije otvara mnoštvo mogućnosti. Roboti opremljeni naprednim sustavima vida mogu raditi u dinamičnim i nestrukturiranim okruženjima, precizno obavljati složene zadatke i učinkovitije komunicirati s ljudima. Primjene sežu od autonomnih vozila i pametne proizvodnje do pomoći u zdravstvu i misija traganja i spašavanja. Detekcija objekata vođena umjetnom inteligencijom ne samo da poboljšava senzorne sposobnosti robota, već također olakšava sigurniju i učinkovitiju suradnju čovjeka i robota.

Iako su potencijalne prednosti umjetne inteligencije u otkrivanju objekata obećavajuće, izazovi ostaju. To uključuje brige vezane uz privatnost i sigurnost podataka, algoritamske pristranosti i potrebu za kontinuiranim istraživanjem robusnosti i pouzdanosti. Rješavanje ovih izazova ključno je kako bi se osiguralo da se tehnologije za otkrivanje objekata koje pokreće AI koriste etički i odgovorno.

Ukratko, umjetna inteligencija za otkrivanje objekata ima moć uvesti novu eru pristupačnosti za osobe s oštećenjem vida i značajno unaprijediti sposobnosti robota u različitim domenama. Dok se nastavlja usavršavanje i širenje ove tehnologije, također postoji odgovornost iskoristiti njihov potencijal za poboljšanje društva, nastojeći stvoriti svijet u kojem tehnologija osnažuje i obogaćuje živote svih, bez obzira na fizičke sposobnosti ili ograničenja. Put prema ovoj budućnosti je u tijeku, gdje inovacija, suradnja i etička razmatranja pomažu svima.

LITERATURA

1. Baheti, Pragati, **Activation Functions in Neural Networks,** https://www.v7labs.com/blog/neural-networks-activation-functions (pristupljeno 7.10.2024)
2. Delua, Julianna, **Supervised versus unsupervised learning,** https://www.ibm.com/blog/supervised-vs-unsupervised-learning/ (pristupljeno 7.10.2024)
3. Wikipedia, Activation function, https://en.wikipedia.org/wiki/Activation_function (pristupljeno 7.10.2024)
4. Solawetz, Jacob, **What is YOLOv8?** https://blog.roboflow.com/whats-new-in-yolov8/ (pristupljeno 7.10.2024)
5. Ultralytics, https://github.com/ultralytics/ultralytics (pristupljeno 7.10.2024)
6. Solawatez, Jacob, **What's New in YOLOv8 | Model Deep Dive,** https://www.youtube.com/watch?v=x0HlrCjJDjs (pristupljeno 7.10.2024)
7. OpenMMLab, **Dive into YOLOv8: How does this state-of-the-art model work?** https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1 (pristupljeno 7.10.2024)

SAŽETAK

U ovom radu opisana je funkcionalnost i korisnost AI vida kao procesa koji simulira ljudski vid s fokusom na neuronskim mrežama. Sama realizacija je postupna, počevši od osnovnih pojmova, perceptrona do zamršenih slojeva konvolucijskih neuronskih mreža.

Detaljno su definirani pojmovi vezani uz neuronske mreže i njihova funkcija unutar same mreže. Također, navedena je struktura različitih neuronskih mreža, vrste aktivacijskih funkcija, prednosti i nedostaci njihove primjene u raznim situacijama. Fokus rada je postupak treniranja i analiza dobivenih rezultata.

Osnovni koncept temelji se na primjeni strojnog učenja koje s pomoću vizualnih ulaza poput slika i videa sudjeluje u otkrivanju objekata. Istaknuta je važnost konvolucijskih neuronskih mreža za umjetni vid. Prikazana i korištena je YOLO (engl. *You Only Look Once*) neuronska mreža razvijena s ciljem primjene umjetne inteligencije za detektiranje objekata.

S pomoću softvera YOLO realiziran je projektni zadatak na temelju skupa ulaznih podataka, procesa obuke i implementacija kôda modela detekcije objekta. Interpretacijom rezultata istaknuta je potreba za primjenom umjetne inteligencije u otkrivanju objekata i posebno, njihova uloga kao pomoć slabovidnim osobama.

Ključne riječi: neuron, neuronska mreža, konvolucijska neuronska mreža, YOLO, aktivacijska funkcija

SUMMARY

This paper describes the functionality and usefulness of AI vision as a process that simulates human vision and has a focus on neural networks. The realization itself is gradual, starting from the basic concepts of perceptrons to intricate layers of convolutional neural networks.

Terms related to neural networks and their function within the network are defined in detail. Also, the structure of different neural networks, types of activation functions, advantages and disadvantages of their application in various situations are listed. The focus of the work is the training process and the analysis of the obtained results.

The basic concept is based on the application of machine learning, which uses visual inputs such as images and videos to participate in the detection of objects. The importance of convolutional neural networks for artificial vision is highlighted. The YOLO (You Only Look Once) neural network developed with the aim of applying artificial intelligence for object detection was presented and used.

Using the YOLO software, the project task was realized based on a set of input data, the training process and the code implementation of the object detection model. The interpretation of the results highlighted the need for the application of artificial intelligence in the detection of objects and, in particular, their role as an aid to the visually impaired.

Keywords: Neuron, Neural Network, Convolutional Neural Network, Yolo, Activation Function