

*Zahvaljujem se svojem mentoru  
izv. prof. dr. sc. Zoranu Kalafatiću za pomoć i ažurnost tijekom pisanja rada.*

# Sadržaj

<b>1</b>	<b>Uvod.....</b>	<b>1</b>
<b>2</b>	<b>Umjetna inteligencija.....</b>	<b>2</b>
2.1	Neuronske mreže .....	3
2.2	Strojno učenje.....	6
2.3	Duboko učenje.....	7
2.4	Treniranje .....	8
2.4.1	Nadzirano učenje .....	9
2.4.2	Nenadzirano učenje.....	9
<b>3</b>	<b>Detekcija i praćenje objekata.....</b>	<b>11</b>
3.1	Obrada slike i predprocesiranje.....	11
3.2	Ekstrakcija značajki i segmentacija i detekcija.....	11
3.3	Klasifikacija slika i videa .....	13
3.3.1	Vrste klasifikacije slika .....	14
<b>4</b>	<b>Korišteni alati.....</b>	<b>15</b>
4.1	Python .....	15
4.2	YOLOv8.....	16
4.2.1	Instalacija .....	17
4.2.2	Treniranje .....	17
4.2.3	Predviđanje.....	18
4.3	Flask .....	19
4.4	Jupyter .....	19
<b>5</b>	<b>Implementacija.....</b>	<b>20</b>
5.1	Hand Detector .....	20

<b>6</b>	<b>Eksperimenti i rezultati</b> .....	<b>23</b>
<b>6.1</b>	<b>Model tuned-hand-gestures</b> .....	<b>25</b>
<b>6.2</b>	<b>Model treniran na skupu podataka</b> .....	<b>27</b>
<b>6.3</b>	<b>Eksperimenti</b> .....	<b>32</b>
<b>7</b>	<b>Zaključak</b> .....	<b>36</b>
	<b>Literatura</b> .....	<b>37</b>

# 1 Uvod

Umjetna inteligencija je u zadnjih par godina postala jedna od najaktualnijih i zanimljivijih grana računalne znanosti, stoga je jedna od najbrže rastućih polja tehnologije u današnjem svijetu. Mnoge grane umjetne inteligencije koriste se za identifikaciju i praćenje objekata na slikama i video snimkama. Jedna od takvih je računalni vid koja omogućava jednostavno i efikasno rješavanje takvih problema. Iako daljnji razvoj i dalje traje, trenutno jedan od najrazvijenijih modela je YOLOv8 koji koristi konvolucijske neuronske mreže za identifikaciju i klasifikaciju objekata. YOLOv8 najnovija je verzija YOLO modela i predstavljen je početkom 2024. godine. Model pokazuje preciznije, brže i kvalitetnije rezultate od svojih prethodnika.

Mnogi unaprijed trenirani modeli omogućuju rješavanje problema detekcije mnogih različitih objekata koji pripadaju različitim klasama. Modeli se razlikuju u veličini i brzini, koja ovisi o vrsti treniranja i skupu podataka nad kojima se treniralo. Osim unaprijed treniranih modela, moguće je trenirati vlastite nad skupom podataka koji treba biti raznovrstan, reprezentativan i podijeljen na 3 dijela: skup podataka za učenje, skup podataka za provjeru i skup podataka za testiranje. Treniranje se odvija u iteracijama proizvoljan broj ponavljanja, čime se podešavaju težine neurona u mreži i time se mreža uči nad podacima.

Ovaj rad rješava konkretan problem računalnog vida, a to je razvijanje sustava za praćenje i detekciju dlana na slikama i video snimkama. To je zahtjevan problem koji zahtjeva korištenje mnogih tehnologija, algoritama i metoda. Definirani su pojmovi umjetne inteligencije, neuronskih mreža, strojnog i dubokog učenja i treniranja. Objašnjene su metode detekcije i klasifikacije objekata i detaljno su objašnjene korištene tehnologije.

Za potrebe rada je izrađena jednostavna web aplikacija koja omogućuje korištenje razvijenih modela za detekciju i praćenje dlana na slikama i video snimkama.

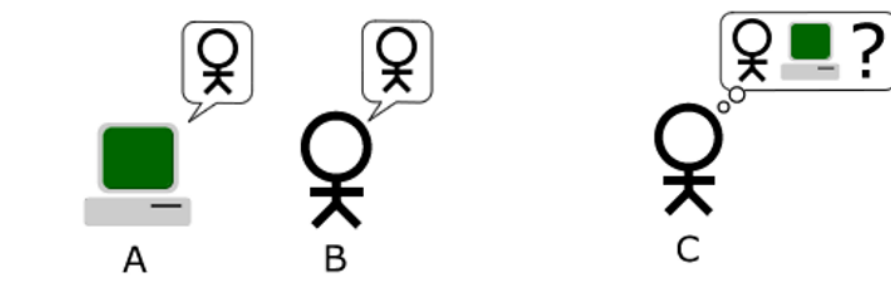
## 2 Umjetna inteligencija

Ciljevi računala su imitiranje čovjekove inteligencije i izvedba zadataka neizvedivih čovjeku. Eksponencijalnim razvojem umjetne inteligencije rješenja mnogih takvih problema postaje stvarnost i razvijaju se napredni algoritmi, softveri i modeli koji omogućavaju strojevima da percipiraju svoje okruženje i koriste učenje i inteligenciju kako bi to i čovjek i time povećavaju šanse za postizanje ciljeva. Takve strojeve nazivamo AI (Artificial Intelligence).

Tehnologije umjetne inteligencije imaju mnoge primjene u razvoju softvera i algoritama, u znanosti, politici. Neki primjeri su korištenje algoritama umjetne inteligencije za napredne web tražilice poput Google-a koji koristi umjetnu inteligenciju za računanje i prikazivanje najrelevantnijih rezultata i preporuka, interakcije čovjeka i računala, na primjer govorom, u autonomnim vozilima i daleko najaktualniji i najpopularniji primjer korištenja umjetne inteligencije danas su generativni modeli poput GPT-a i slično.

Alan Turing bio je prva osoba koja je provodila istraživanja pod nazivom strojne inteligencije i smatra se osnivačem umjetne inteligencije te se 1956. godine umjetna inteligencija osniva kao akademska disciplina. Poznati test inteligencije pod nazivom Turingov test, originalno zvan *igra imitacije*, test je koji provjerava inteligenciju računala i određuje smatra li se ono *intelligentnim*. Testira se sposobnost računala da pokaže inteligentno ponašanje slično, ili nerazlučivo čovjekovu, time se određuje je li računo sposobno razmišljati poput čovjeka. Neke informacije i opis Turing testa preuzeti s [1]

Test funkcionira tako da igrač C ispituje igrače A i B i pokušava odrediti koji je računo, a koji čovjek. Postupak se jasno vidi na slici **Slika 2.1** Danas, više od 70 godina od Turingova prijedloga, nijedna tehnologija umjetne inteligencije nije uspjela proći test ispunjavajući specifične uvjete.



**Slika 2.1** Opis postupka Turingova testa gdje igrač C pokušava odrediti koji od igrača A i B je računo, a koji čovjek. Slika je preuzeta s [1].

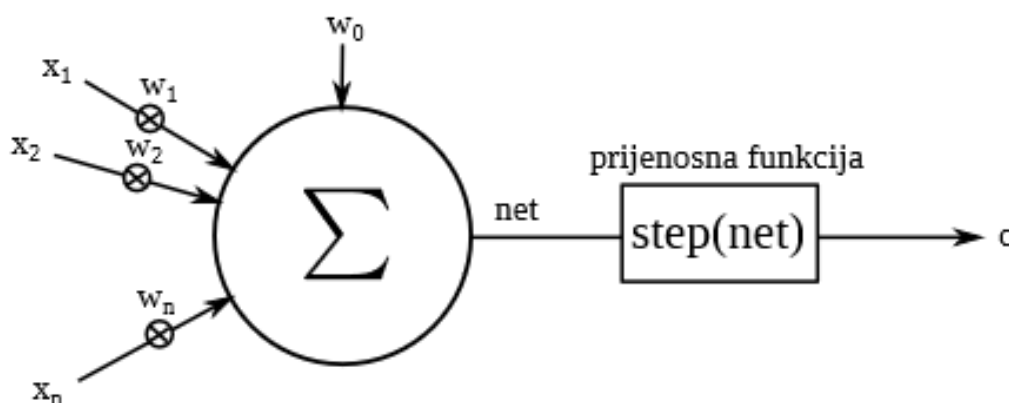
Umjetna inteligencija obuhvaća strojno učenje, duboko učenje kao i neuronske mreže. Ove discipline uključuju razvoj algoritama umjetne inteligencije, modeliranih prema procesima donošenja odluka u ljudskom mozgu, koji mogu učiti iz dostupnih podataka i svoje modele koristiti za rješavanje zahtjevnih problema i zadataka.

## 2.1 Neuronske mreže

Poznato je da se ljudski mozak sastoji od mnoštva međusobno povezani neurona koji rade istovremeno. Izgrađuju se sustavi *umjetnih neuronskih mreža* koji imitiraju rad ljudskog mozga. Umjetna neuronska mreža (ANN) replika je ljudskog mozga kojom se simuliraju postupci učenja i obrade podataka.

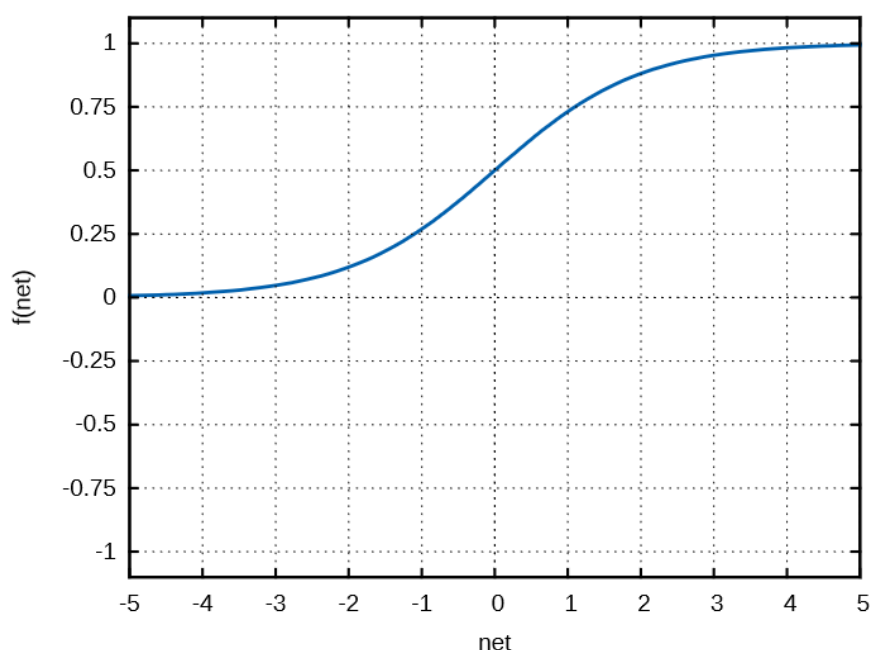
Neuroni su procesni elementi neuronske mreže inspirirani biološkim neuronima u mozgu. Biološki neuron se sastoji od tijela za obradu impulsa, dendrite za primanje impulsa i akson za prijenos impulsa do ostalih neurona. Arhitektura umjetnog neurona slična je biološkom, sadrži ulazni sloj neurona, skriveni sloj neurona koji predstavljaju tijelo biološkog neurona i izlazni sloj neurona koji računa izlazne vrijednosti.

Funkcionalnost umjetnog neurona temelji se na vrijednostima težina i pristranosti svakog od međusobno povezanih neurona u mreži. Vrijednost sa svakog ulaza  $x_i$  u neuron se množi težinom tog ulaza  $w_i$  i akumulira u tijelu. Ukupnoj sumi dodaje se pomak  $w_0$ , još poznat kao pristranost (eng. bias). Time je definirana akumulirana vrijednost koja se propušta kroz prijenosnu funkciju čime nastaje izlazna vrijednost. Prikaz modela umjetnog neurona vidi se na slici (Slika 2.2).



Slika 2.2 Model umjetnog neurona. Slika je preuzeta s [2].

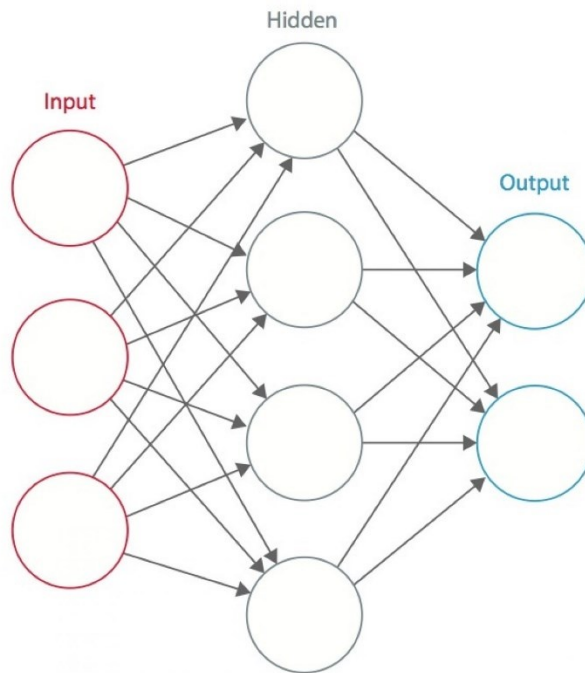
Prijenosna funkcija smanjuje linearnost i omogućuje rješavanje nelinearnih problema. Neke od često korištenih prijenosnih funkcija su funkcija identiteta, funkcija skoka, sigmoidalna funkcija (**Slika 2.3**), tangens hiperbolni.



**Slika 2.3** Sigmoidalna funkcija, vraća vrijednosti između 0 i 1, simetrična je. Slika preuzeta s [2].

Duboke neuronske mreže proširenje su umjetnih neuronskih mreža. Za razliku od konvencionalnih neuronskih mreža duboke neuronske mreže obično sadrže mnogo skrivenih slojeva i velik broj neurona čiji broj dostiže milijune u nekim kompleksnijim primjerima.

Duboke neuronske mreže su se pokazale da probleme računalnog vida podižu na još višu razinu točnosti i učinkovitosti, zahvaljujući konvolucijskim neuronskim mrežama (CNN), te zbog toga su najčešće rješenje problema klasifikacije slika. Konvolucijska neuronska mreža proširena je inačica umjetne neuronske mreže (ANN) koja se primarno koristi za obradu podataka matričnog oblika, poput slika. CNN-ovi oponašaju neuronske mreže ljudskog mozga u više slojeva: slojevi ulaznih podataka, konvolucijski slojevi, ReLU slojevi, slojevi sažimanja i sloj izlaznih podataka. Pojednostavljena arhitektura konvolucijske neuronske mreže prikazana je na slici **Slika 2.4**. Informacije o neuronskim mrežama preuzeti s [2] i [4].



**Slika 2.4** Arhitektura konvolucijske mreže s ulaznim slojem s 3 neurona, jednim međuslojem s 4 neurona i izlaznim slojem s 2 neurona. Slika preuzeta s [3]

Konvolucijski slojevi izvode linearno-transformacijske operacije među dvije matrice, gdje je prva matrica filter koji se može naučiti, a druga matrica je ograničeni dio ulazne matrice. Filter klizi po visini i širini slike stvarajući slikovnu reprezentaciju čime se dobiva aktivacijska mapa, dvodimenzionalni prikaz slike. Treniranjem na podacima mreža će naučiti sve filtre.

Sloj ReLU (Rectified Linear Unit) postao je popularan u zadnjih par godina. ReLU sloj se generalno nalazi iza konvolucijskog sloja i on svaku negativnu vrijednost pretvara u nulu.

Sloj sažimanja smanjuje veličinu slika što čini izračun bržim te smanjuje memorijsku i vremensku kompleksnost. Neke od vrsta sažimanja su maksimalno sažimanja i prosječno sažimanje.

Sposobnost neuronskih mreža da identificiraju obrasce, rješavaju kompleksne zadatke i prilagode se promjenjivom okruženju je ključna. Razvoj umjetne inteligencije uvelike ovisi o neuronskim mrežama, koje također pokreću inovacije i utječu na smjer razvoja tehnologije. Neuronske mreže osnovna su komponenta mnogih tipova učenja umjetne inteligencije poput strojnog i dubokog učenja.



## 2.2 Strojno učenje

Strojno učenje koristi podatke i algoritme kako bi omogućilo umjetnoj inteligenciji da imitira čovjekov način učenja, postepeno poboljšavajući točnost i uspješnost.

Algoritam algoritma strojnog učenja generalno se dijeli na tri glavna dijela:

1. **Proces odluke:** Na temelju ulaznih podataka algoritam proizvodi procjenu uzoraka u njima.
2. **Funkcija pogreške:** Procjenjuje se predviđanje modela, na temelju poznatih primjera može se odrediti točnost modela.
3. **Proces optimizacije modela:** Težine se prilagođavaju kako bi se smanjila pogreška. Algoritam će ponoviti ovaj iterativni proces "procjene i optimizacije", autonomno ažurirajući težine dok se ne postigne zadovoljavajući rezultat.

Klasično strojno učenje često ovisi o ljudskoj intervenciji u učenju. Ljudski stručnjaci određuju skup značajki kako bi razumjeli razlike između podataka. Postoje mnogi algoritmi i načini strojnog učenja koji se ovisno o problemu, dubini i kompleksnosti odabiru za rješenje:

- **Neuronske mreže**
- **Linearna regresija:** Postupak predviđanja vrijednosti na temelju linearne zavisnosti između varijabli.
- **Logistička regresija:** Predviđanja kategorijskih varijabli poput da/ne
- **Grupiranje:** Identificiranje uzoraka u podacima i njihovo grupiranje u razrede.
- **Stablo odluke:** Koriste se za predviđanje vrijednosti i za razvrstavanje podataka u razrede. Koristi niz grananja povezanih odluka koje se mogu prikazati dijagramom stabla.
- **Nasumične šume:** Kombinacijom izlaza više stabala odluke postiže se jedan rezultat.

Mnoge su prednosti strojnog učenja jer je moguće uz malu ljudsku intervenciju uočiti uzorke i trendove u ogromnim skupovima podataka. Algoritmi se neprestano poboljšavaju s više unosa podataka, što znači da su potrebne velike količine podataka za obuku, što smanjuje vjerojatnost za greškom.

## 2.3 Duboko učenje

Strojno učenje i duboko učenje koriste *neuronske mreže* za učenje iz ogromnih količina podataka, razlikuju se u vrsti neuronske mreže koju koriste i u količini ljudske intervencije.

Klasično strojno učenje se najčešće uči nadziranim učenjem, što znači da koristi podatke koji su strukturirani ili označeni, dok algoritmi dubokog učenja koriste duboke neuronske mreže koje se sadrže od više skrivenih slojeva te se koristi za nenadzirano učenje koje automatizira ekstrakciju značajki iz nestrukturiranih podataka. Duboke neuronske mreže se obučavaju na velikim količinama podataka kako bi identificirale i klasificirale fenomene, prepoznale obrasce i odnose, procijenile vjerojatnosti te donosile predviđanja i odluke.

Svaki od međusobno povezanih čvorova se nadograđuje na prethodni sloj kako bi se poboljšalo i optimiziralo predviđanje ili kategorizacija. Napredovanje kroz mrežu, u kojem se mreži predaju ulazni podaci da bi se dobilo ciljno predviđanje poznato je pod nazivom *forward pass*. Algoritam širenja pogreške unazad povratno se širi kroz mrežu izračunavajući grešku zatim prilagođavajući težine i pristranosti mreže pomicanjem unatrag kroz slojeve u nastojanju da se uvježba model. Širenje unaprijed i natrag omogućuje neuronskoj mreži da napravi predviđanja i ispravi što više pogrešaka i na taj način algoritam postupno postaje precizniji. Detalji izvedbe učenja duboke mreže preuzeti su s [5].

Duboko učenje zahtijeva ogromnu količinu računalne snage. Grafičke procesorske jedinice (GPU) visokih performansi idealne su za učenje modela dubokim učenjem jer mogu podnijeti veliku količinu izračuna u više jezgri s velikom dostupnom memorijom. Grafičke kartice su optimizirane za matrične operacije, a same neuronske mreže i njihove težine i pristranosti mogu se prikazati kao matrice.

Generativni modeli aktualan su primjer primjene dubokog učenja. To su modeli umjetne inteligencije koji se odnose na modele dubokog učenja koji mogu uzeti neobrađene podatke i naučiti statistički vjerojatne rezultate. Porast dubokog učenja omogućio je njihovo proširenje i jednostavno korištenje dostupno svima.

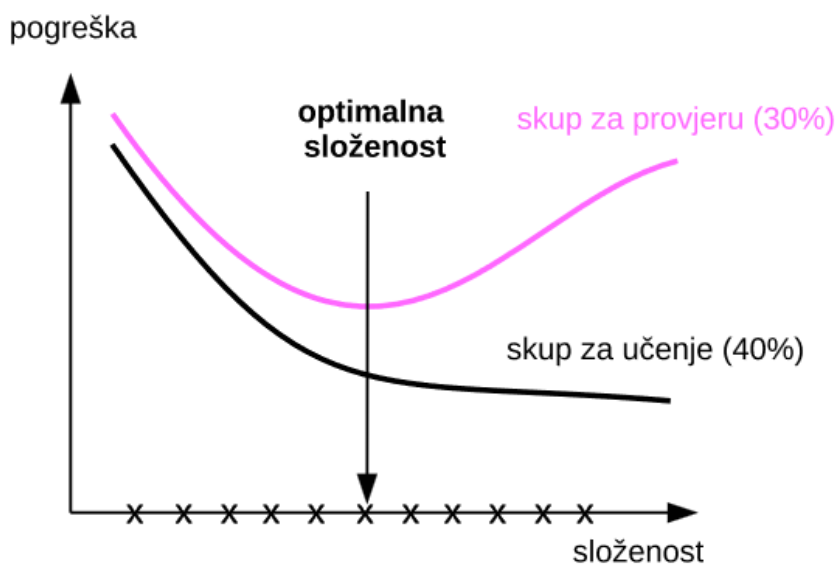
Detalji strojnog i dubokog učenja preuzeti su s [6].

## 2.4 Treniranje

Postoje različite metode treniranja koje ovise o dostupnim podacima i zadatku kojeg pokušavamo odraditi. Osim metoda treniranja bitno je kvalitetno odrediti skup podataka jer će kvaliteta mreže ovisiti o njegovoj kvaliteti. Loše reprezentativni ili mali skupovi podataka rezultirat će mrežama koje neprecizno generalizira naučeno na neviđene ulazne podatke.

Čest je problem prenaučivosti (engl. *overfitting*), što predstavlja model koji se previše naučio nad podacima za treniranje i njih savršeno klasificira, ali loše i s niskom preciznošću prepoznaje neviđene ulaze, model postaje *štreber*. Generalno se savjetuje podjela skupa podataka na podskup za učenje (engl. *train*), podskup za provjeru (engl. *valid*) kojim se provjerava uspješnost i podskup za testiranje (engl. *test*). Omjer podjele obično je 40% na podskup za učenje, 30% na podskup za testiranje i 30% na podskup za provjeru. Ovaj postupak naziva se unakrsna provjera i njime se određuje optimalna složenost modela, nju je moguće odrediti iz grafa (Slika 2.5), naime optimalna složenost je u točki gdje pogreška skupa za provjeru počinje rasti, a pogreška skupa za učenje i dalje se smanjuje.

Omjeri skupova i detalji treniranja i vrsta treniranja preuzeti su s [8].

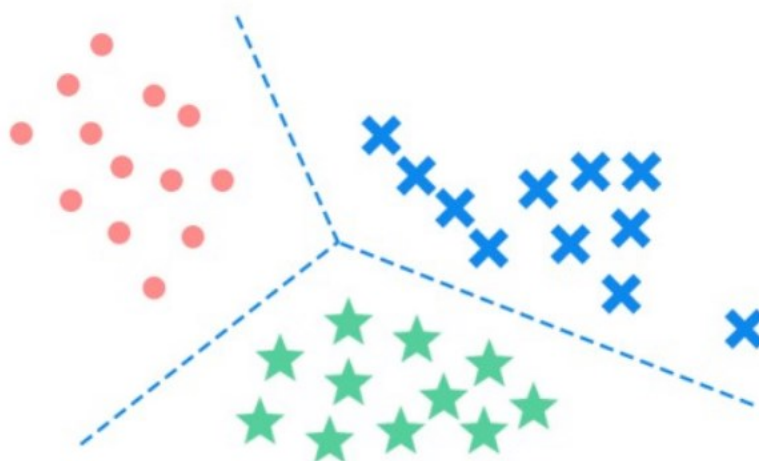


**Slika 2.5** Graf postotka pogreške na skupovima za provjeru i učenje, određuje se optimalna složenost unakrsnom provjerom. Slika preuzeta s [9].

## 2.4.1 Nadzirano učenje

Nadzirano strojno učenje algoritam je učenja na označenim skupovima podataka, s već određenim odnosom ulaza i ispravnih izlaza. Skup podataka slika je već označen i klasificiran. Na primjer ulazni skup podataka može se sastojati od slika životinja koje su označene ovisno o vrsti, slike pasa pripadaju klasi pas, slike mačaka pripadaju klasi mačka i slično. Dopršetkom učenja, analizom novih ulaza algoritam na temelju podataka za učenje zaključuje i predviđa razrede novih slika.

Algoritmi nadziranog učenja dijele se na klasifikaciju s jednim razredom koji slici dodjeljuje samo jedan razred poput dlana ruke i klasifikaciju s više razreda (**Slika 2.6**). Mnogi algoritmi implementiraju neku vrstu nadziranog strojnog učenja. Jedan od takvih je stablo odluke koja se lako implementira i široko je primjenjiva. Stablo odluke funkcionira poput dijagrama toka gdje se na svakom čvoru određuje u kojem će se smjeru dalje kretati sve dok se ne dođe do lista koji predstavlja ciljnu vrijednost.



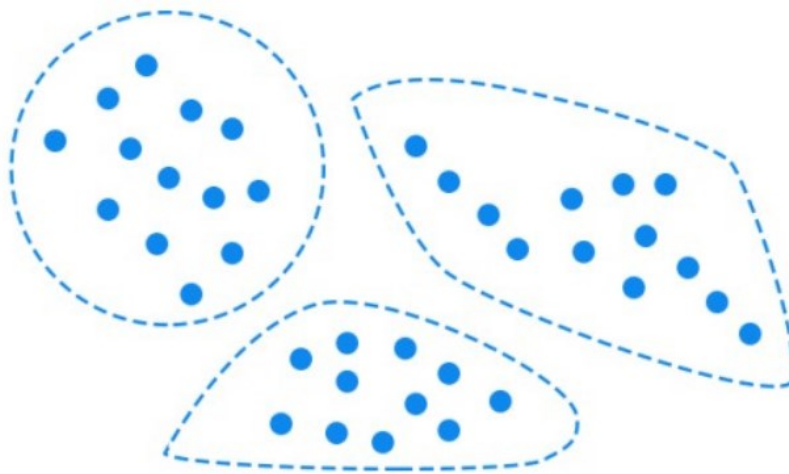
**Slika 2.6** Primjer klasifikacije u 3 razreda ulaznog skupa podataka uz nadzirano učenje. Slika preuzeta s [10].

## 2.4.2 Nenadzirano učenje

Za razliku od nadziranog učenja, skupovi podataka u algoritmima nenadziranog učenja nisu označeni niti unaprijed klasificirani, ovi algoritmi pronalaze pravilnosti i sličnosti u podacima grupirajući, otkrivajući stršećih vrijednosti i smanjenjem dimenzionalnosti. Skup podataka mogu biti slike životinja, ali bez klasifikacija, algoritam grupira podatke na temelju nekih

sličnosti pa će na primjer psi biti grupirani u jednu grupu na temelju sličnih karakteristika, dok će mačke biti grupirane u drugu grupu. Primjer grupiranja ulaznih podataka vidimo na slici **Slika 2.7**.

Cilj nenadziranog učenja je istražiti i razumjeti prirodu danih podataka. Algoritam tumači podatke prema vlastitim uvjetima, prepoznaje uzorke i izvlači zaključke. Najčešći koncept koji koristi nenadzirano učenje je grupiranje podataka na temelju nekih sličnosti i uzoraka kojim dobijemo grupe podataka čiji će se razredi kasnije odrediti.



**Slika 2.7** Primjer grupiranja podataka uz nenadzirano učenje. Slika preuzeta s [10].

## 3 Detekcija i praćenje objekata

Detekcija i praćenje objekata je proces pronalaženja, klasificiranja i praćenja objekata na slikama i videima. Koraci postupka detekcije i praćenja objekta preuzet je s [11]. Postupak se izvodi u nekoliko slijednih koraka:

1. **Obrada slike i predprocesiranje**
2. **Ekstrakcija značajki i segmentacija i detekcija**
3. **Klasifikacija**
4. **Postprocesiranje**
5. **Sinteza**

### 3.1 Obrada slike i predprocesiranje

Svaka slika sačinjena je od tisuća piksela koji se u tehnikama klasifikacije tretiraju kao matrice, čije veličine ovise o rezoluciji same slike. Svaki algoritam implementira neke metode koje će poboljšati kvalitete slike i pripremiti ih za daljnju analizu. Slikama se često mijenjaju dimenzije (visina i širina) ili izrezuju samo dijelovi slike radi uklanjanja nebitnih dijelova slike ili radi smanjenja veličine i kompleksnosti.

Koristi se metoda normalizacije slike gdje se vrijednost svakog piksela podešava na normalnu distribuciju. Svakom pikselu se oduzima srednja vrijednost i dijeli se sa standardnom devijacijom vrijednosti piksela. Mnoge slike sadrže šum koji može negativno utjecati na efikasnost modela pa je potrebno koristiti neke od tehnika filtriranja slike poput Gaussovog filtriranja ili srednjeg filtriranja. Ove metode pospješuju uspješnost i kvalitetu modela. Nakon prve obrade slike ona je spremna za daljnju analizu. Postupak obrade slike preuzet s [11].

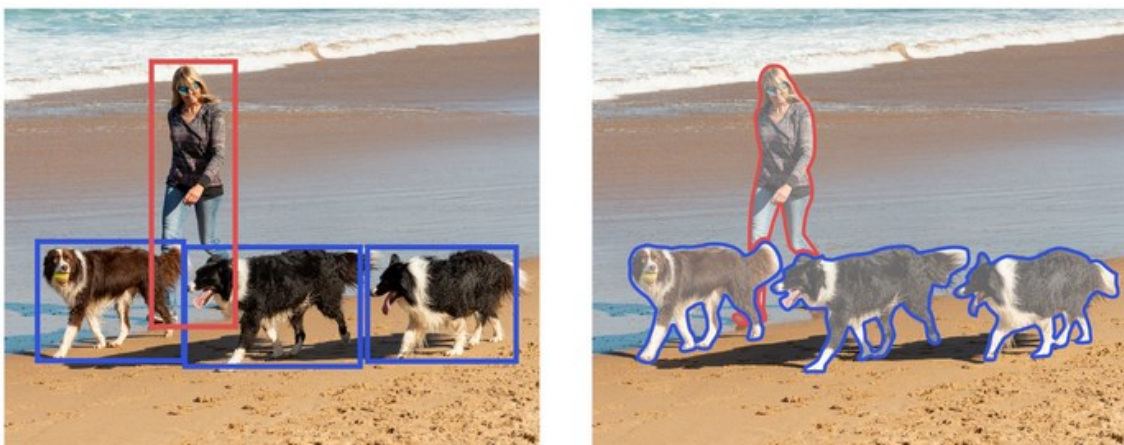
### 3.2 Ekstrakcija značajki i segmentacija i detekcija

Ekstrakcija značajki je bitan proces prepoznavanja uzoraka unutar slike koji će se koristiti za razlikovanje jednog objekta od drugog. Uzorci su obično specifični za svaki razred što rezultira jasnom razlikom razreda. U slučaju klasifikacije jabuka i banana postoje značajke poput oblika i boje koje će se koristiti za razlikovanje ta dva razreda. Postupak i detalji ekstrakcije značajki i segmentacije i detekcije slika preuzeti su sa [11].

Ekstrakcija značajki uvelike poboljšava performanse modela fokusiranjem na najrelevantnije dijelove svake slike, u suprotnom bi se svaki put trebala analizirati cijela slika što utječe na kompleksnost i točnost modela. Otkrivanje rubova na slici korisna je i bitna metoda jer s pomoću nje uočavamo granice između područja na slici. Najčešće metoda otkrivanja rubova na slici je korištenje operatora gradijenta koji određuje di dolazi do prijelaza između elemenata na slici. Analiza teksture analizira sliku i pronalazi ponavljajuće uzorke što se koristi za prepoznavanje različitih materijala ili površina objekata, ovime se prepoznaju tumori na medicinskim slikama jer se tekstura zdravog tkiva razlikuje od teksture kancerogenog tkiva.

Slike se sastoje od informacija bitnih za analizu i onih nebitnih. Koriste se mnogi načini podjele slike na bitne dijelove za analizu čime se rezultati postupka klasificiranja objekata na slici značajno poboljšava. Segmentacija je postupak podjele slike ili videa na područja interesa za identificiranje i razlikovanje objekata ili zanimljivih dijelova. Područja od interesa mogu biti bilo kakvi objekti koji se ističu na slici – ljudi, automobili, zrakoplovi, životinje. U kontekstu detekcije i praćenja dlana dlan i ruka su područja interesa. Najčešći oblik je semantička segmentacija koja svakom pikselu na slici daje oznaku.

Detekcija lokalizira određeni objekt unutar slike često pružajući granične okvire njegove lokacije, što olakšava i ubrzava proces klasifikacije objekta unutar slike. Segmentacija slike daje detaljnu analizu gdje je izlazna slika podijeljena piksel po piksel dok nam detekcija daje granične okvire objekata. Primjer razlike segmentacije i detekcije vidi se na slici **Slika 3.1**.



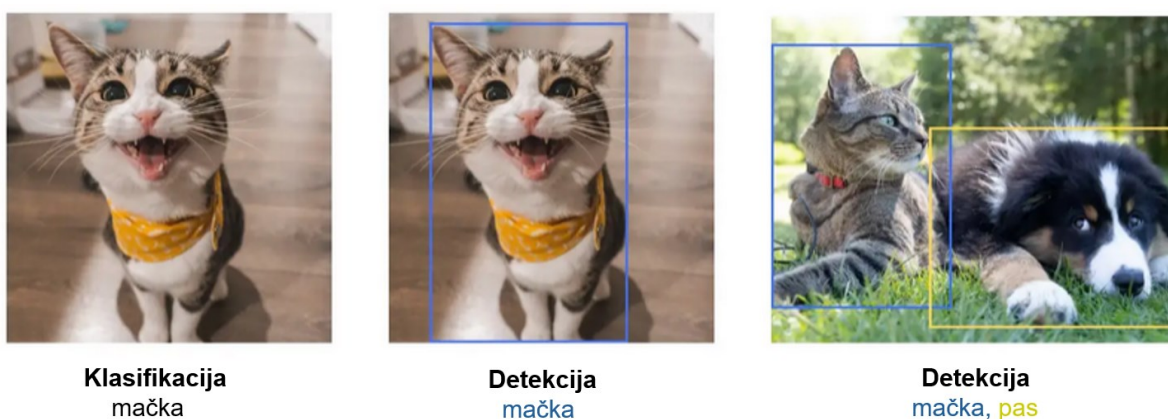
**Slika 3.1** Razlika između detekcije(lijevo) i segmentacije(desno). Slika je preuzeta s [12].

### 3.3 Klasifikacija slika i videa

Kako bi model mogao odrediti nalazi li se na slici dlan, potrebno je odrediti razrede segmentiranih i detektiranih objekata, to se izvodi postupkom *klasifikacije*. Klasifikacija slika i videa zadatak je strojnog učenja da identificira i odredi što slika ili video predstavljaju. Model se trenira na skupu podataka za učenje koji sadrži različite razrede i oznake. Budući da je video skup slika, mnoga se jednaka pravila koriste za klasifikaciju slika kao i za klasifikaciju videa. Slici se dodjeljuje jedna ili više oznaka ili razreda na temelju postojećih podataka za treniranje već označenih slika. Dobivamo rezultate koji nam govore nalaze li se na slici određeni objekti, atributi ili uzorci. Neki od mnogih slučaja upotrebe klasifikacije slika i videa su: automatska inspekcija i kontrola kvalitete artikala, prepoznavanje objekata u autonomnim vozilima, prepoznavanje lica u sigurnosnim snimkama. Detalji klasifikacije slike i videa preuzeti s [11].

Napretkom strojnog učenja u zadnjih nekoliko godina brzo napreduju i modeli klasifikacije i detekcije objekata. Danas, za razliku od prije samo deset godina klasifikacija objekata je brzo izvediva jer je hardver puno jači i jednostavnija je za implementaciju.

Potrebno je razlikovati klasifikaciju, i detekciju objekata. Klasifikacija dodjeljuje razred slici ili objektu dok detekcija objekta specificira područje fotografije ili videa na kojem se objekt određenog razreda nalazi i najčešće ga stavlja unutar graničnog okvira umjesto da klasificira cijelu sliku. Razlika je jasno vidljiva na slici **Slika 3.2**.



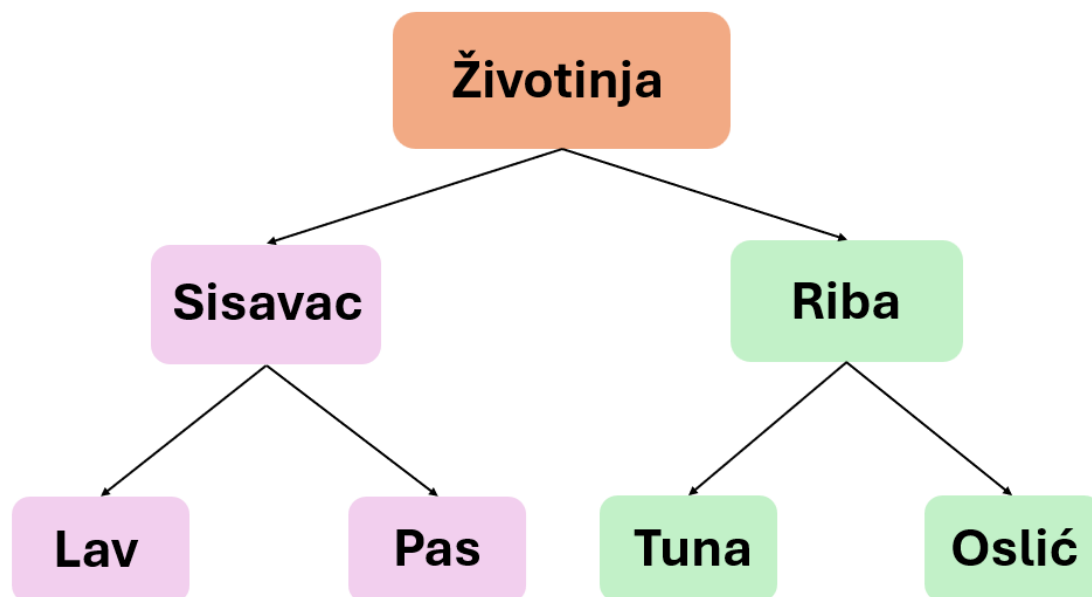
**Slika 3.2** Primjer klasifikacije i detekcije mačke i psa na 3 slike gdje se jasno vidi razlika između klasifikacije i detekcije. Slika preuzeta s [11].



### 3.3.1 Vrste klasifikacije slika

Postoje različite metodologije klasifikacije slike koje se mogu koristiti u rješavanju širokog opsega zadataka i problema. Detalji o vrstama klasifikacija slika preuzeti s [11].

- **Binarna klasifikacija** klasificira sliku u jedan od dva razreda. Slika će pripadati ili jednom ili drugom razredu, što ima mnoge i široke primjene, poput prepoznavanja tumora, uočavanje anomalija ili nešto poput klasifikacije životinje u razrede pas ili mačka.
- **Višerazredna klasifikacija** slična je binarnoj, kategorizira sliku u tri ili više razreda i ima širu primjenu jer se njome slike mogu detaljnije i kompleksnije analizirati i kategorizirati, moguće je napraviti model koji će na temelju slike prepoznati koju to životinju predstavlja, a ne poput binarne samo jednu od dvije moguće klasifikacije.
- **Hijerarhijska klasifikacija** za zadatak ima organizaciju klasa u hijerarhijsku strukturu na temelju njihove sličnosti, moguće je klasificirati životinje na sisavce i ribe te zatim sisavce i ribe dodatno podijeliti na specifičnije članove tog skupa kao što su lav, tuna, i slično. Primjer hijerarhijske klasifikacije slike vidimo na **Slika 3.3**.



**Slika 3.3** Primjer hijerarhijske klasifikacije slike

## 4 Korišteni alati

Problem detekcije i praćenja objekata na slici i videu nije jednostavan proces. Primjenjivost i efikasnost metoda ovisi o mnogim čimbenicima, na primjer kvaliteta slike, broj sličica po sekundi na videu, šumovi na podacima i slično. Složenost najvećim dijelom proizlazi iz samih uvjeta u kojima se vrši detekcija. Kretnja dlana i ruke nepredvidiva je, brza i često promjenjiva, osim tog i pozadina sama po sebi može biti nepredvidiva. Količina vanjskog svjetla, sjene i slično drastično utječu na efikasnost modela. Spoj ili kombinacija spomenutih svojstava definira uvjete u kojima se vrši detekcija, koje je nužno uzeti u obzir pri odabiru metoda.

Računalni vid danas je glavni alat za složenu analizu slike, prvenstveno u onim slučajevima kada je potrebno postići seciranje na onoj razini na kojoj to radi ljudski mozak. Računalni vid podvrsta je umjetne inteligencije gdje se promatraju vizualni podaci. Raznovrsnim se metodama pokušava postići viša razina razumijevanja vizualnih podataka kako bi se povećala primjenjivost u što većem broju slučajeva. Jedan od najčešćih zadataka računalnog vida je raspoznavanje objekata na slici ili videu.

Danas dolazi do ubrzanog razvoja ovog područja zbog novih specifičnih potreba za raspoznavanjem predmeta, stalnim napretkom, poboljšanjem starih metoda i stalni razvoji novih. Postoje mnogi sustavi temeljeni na odabranim algoritmima koji posjeduju modele koji imaju svoje parametre koji se uče, na primjer težine neuronskih mreža. Ti sustavi su prilagodljivi različitim zadacima i problemima. Neki od poznatijih konkretnih primjera su **YOLO** (You Only Look Once), **TensorFlow** i **EfficientDet**.

U sklopu ovog rada u svrhu rješavanja problema detekcije i praćenja dlana korišten je **YOLOv8** model koji je implementiran unutar **Flask** aplikacije koristeći **python**.

### 4.1 Python

Python [13] je objektno orijentirani programski jezik visoke razine s dinamičkom semantikom. Sadrži mnoge ugrađene podatkovne strukture, jednostavnu sintaksu koju je lako naučiti, čitljiv je i stoga smanjuje troškove održavanja programa što ga čini vrlo atraktivnim za brzi razvoj aplikacija, kao i za korištenje kao jezik za povezivanje postojećih komponenti. Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda.

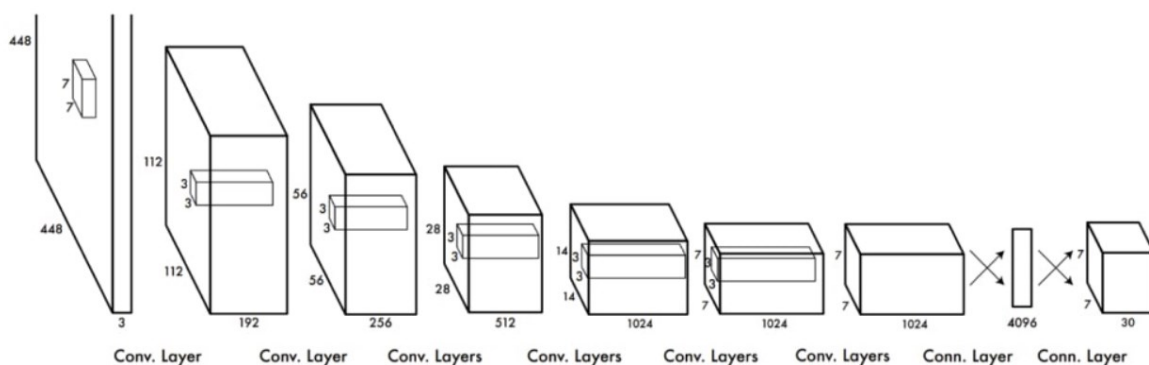
Pythonov *interpreter* i opsežna standardna biblioteka javno su dostupni i mogu se besplatno distribuirati.

Python-ova jednostavnost i brzina razlog su njegove rasprostranjenosti, jednostavan je i intuitivan za početnike i jednostavne programe, a implementira kompleksne i složene metode i algoritme za razvoj naprednijih programa.

## 4.2 YOLOv8

YOLO (You Only Look Once) vrhunski je model računalnog vida kojeg razvija *Ultralytics*, platforma umjetne inteligencije za stvaranje, treniranje i implementiranje modela strojnog učenja sa sučeljima bez izvornog koda za jednostavnost i praktičnost uporabe. Ultralytics pojednostavljuje proces i pruža rješenja koja se lako implementiraju.

Arhitektura dostiže veliki ugled i raširenost zahvaljujući brzini raspoznavanja s relativno visokom točnosti. Ova svojstva omogućavaju primjenu u stvarnom vremenu i kvalitetnu analizu videozapisa. YOLO arhitektura se razlikuje od ostalih jer na prepoznavanje predmeta gleda kao na problem regresije do prostorno odvojenih poznatih okvira i pridruženih vjerojatnosti mogućih klasa. Tijekom analize se promatra čitava slika što pospješuje preciznost jer se predviđanja temelje na globalnom kontekstu. Ovo su posljedice upotrebe samo jedne konvolucijske neuronske mreže koja dijeli sliku u niz podmreža kojima je raspodijeljen zadatak određivanja mogućih okvira, pridružene sigurnosti i klasificiranja. Prikaz jedne neuronske mreže vidimo na slici **Slika 4.1**.



**Slika 4.1** Arhitektura konvolucijske neuronske mreže YOLOv8 modela. Slika preuzeta s [15]

YOLOv8 ažurirana je verzija popularnog YOLO algoritma za raspoznavanje objekata, koji je predstavljen 2016. Model sadrži brojne arhitektonske i razvojne promjene i poboljšanja u odnosu na prijašnje verzije. Utvrđeno je kako uspijeva pronaći i klasificirati objekte različitih razreda na slikama s visokom točnošću.

Novo razvijena arhitektura YOLOv8 koristi princip raspoznavanja bez sidra, što znači da sve objekte predviđa direktno u njihovu centru umjesto koristeći granični okvir poznat kao *anchor box*. Prepoznavanje bez sidra smanjuje kompleksnost i ubrzava proces prepoznavanja. Informacije o YOLOv8, njegovoj arhitekturi, povijesti i specifikacijama preuzeti s [14].

### 4.2.1 Instalacija

YOLOv8 primarno koristi programski jezik *python* za implementaciju i pokretanje, stoga je prije same instalacije potrebno na vlastito računalo instalirati programski jezik python sa službene stranice. Potreban je i *pip*, alat koji nam omogućava i olakšava instalaciju python paketa. Kada su te dva zahtjeva ispunjena moguće je instalirati paket *ultralytics* i sve njegove zahtjeve pozivanjem sljedeće komande unutar naredbenog retka:

```
pip install ultralytics
```

Yolov8 može se koristiti direktno u sučelju naredbenog retka (CLI) komandom `yolo`. Može se koristiti na razne načine i za mnoge zadatke, prima mnoge argumente i sadrži različite načine rada. Postupak instalacije preuzet s [14].

### 4.2.2 Treniranje

Glavna funkcionalnost YOLOv8 modela je treniranje i učenje na skupu podataka. Poziva se funkcija treniranja koja kao argumente prima skup podataka (*data*), veličinu slike (*imgsz*) i broj epoha (*epoch*). Epoha predstavlja jedan prolaz algoritma kroz sve podatke na kojima se model trenira. Postoje mnogi argumenti koji se mogu postaviti, ali ovo su najbitniji i oni koji su se koristili u sklopu ovog rada. Metoda treniranja može se pozvati iz naredbenog retka:

```
yolo detect train data=coco8.yaml model=yolov8n.pt epochs=100  
imgsz=640
```

Način treniranja u YOLOv8 osmišljen je za djelotvorno treniranje modela raspoznavanja objekata, uz potpuno korištenje modernih hardverskih mogućnosti. Prije samog treniranja potrebno je pripremiti skup podataka za učenje. Potrebno je prikupiti slike, označiti ih i pohraniti u odgovarajućem formatu. Koristan alat je **Roboflow** koji se koristi za označavanje i predprocesiranje slika i podataka te sadrži javno dostupne gotove skupove podataka za sve korisnike, nudi formate za treniranje YOLOv8 modela. Važno je napomenuti da svaki skup podataka sadrži skup slika za učenje, skup slika za provjeru i skup slika za testiranje koji se koriste za optimizaciju i pronalaženje najpreciznijeg rješenja i pomaže u smanjenju prenaučivosti.

Potrebno je uzeti u obzir brzinu treniranja, mnoga današnja računala nisu optimizirana za zadatke matičnog množenja što može uvelike usporiti treniranje podataka. Jedna epoha treniranja podataka s više od 1000 slika na računalu bez ugrađene GPU može trajati do 15 minuta, dok se s ugrađenom GPU treniranje od 100 epoha na jednakom skupu podataka može izvesti u sat vremena.

Nakon završetka treniranja modela, dobiva se skup istreniranih težina *best.pt* koji se dalje može koristiti za detekciju objekata. Dobiju se i mnogi rezultati, analize i statistike treniranja modela. Svi rezultati spremaju se u datoteku *runs*.

Modele je moguće spremati funkcijom *export* te postoji mogućnost validacije modela kojim se procjenjuje kvaliteta obučanih modela. *Val* je funkcija koja pruža širok skup alata i metrika za procjenu izvedbe vlastitih modela prepoznavanja objekata. Detalji metrika objašnjeni su u kasnijim poglavljima u sklopu evaluacije rezultata vlastitih modela.

Postupak treniranja preuzet s [14].

### 4.2.3 Predviđanje

YOLOv8 nudi funkcionalnost predviđanja i raspoznavanja objekata koja koristi gotove modele ili vlastito trenirane. Moguće je provoditi detekcije objekata na slikama, videima i prijenosu uživo u stvarnom vremenu.

Predviđanje se izvodi pozivom modela i prijenosom argumenata izvora na kojem će se izvoditi predviđanje s mogućnošću spremanja novonastale slike ili videa na kojem je izvedeno prepoznavanje objekata. Uz svaku sliku ili kadar videa dobiju se i rezultati koji opisuju detalje koji su objekti detektirani i na kojim koordinatama.

Predviđanje je moguće sljedećim programskim kodom:

```
model = YOLO("yolov8n.pt")  
model.predict("slika.jpg", save=True, imgsz=320)
```

Rezultat predviđanja bit će slika ili video na kojem su unutar graničnih okvira označeni objekti za koje je model treniran da prepoznaje. Moguće je manipulirati elementima *Boxes* koji predstavljaju granične okvire, može im se mijenjati format, oblik, indeks i dodatne informacije koje se ispisuju poput vjerojatnosti raspoznavanja točnog razreda i sami nazivi razreda.

Predviđanje je moguće samostalno pokrenuti na slikama i videima ili je moguće ih implementirati u sklopu aplikacija da se dobiveni rezultati koriste za daljnje funkcionalnosti.

Postupak predviđanja preuzet s [14].

## 4.3 Flask

Flask [16] je okvir za izradu web aplikacija, to je python paket koji omogućava jednostavan razvoj aplikacija i koristi se kao alat za razvoj programske potpore za web. Brz je i jednostavan, osmišljen je kako bi se brzo izradile jednostavne aplikacije, uz mogućnost skaliranja do složenih aplikacija. Ima malu jezgru koju je lako proširiti. Jednostavnost i brzina razvoja Flask aplikacije čini ga savršenim za izbor za razvoj jednostavne aplikacije koja koristi trenirane modele za detekciju i praćenje dlana. Instalacija Flask paketa izvršava se pozivanjem sljedeće komande unutar naredbenog retka:

```
pip install -U Flask
```

Instalacijom može se početi s izradom aplikacije ili se može koristiti za pokretanje gotovih aplikacija lokalno na računalu.

## 4.4 Jupyter

**Jupyter** [17] je interaktivni alat koji se koristi za stvaranje *bilježnica* koje sadrže elemente poput programskog koda, teksta, slika, grafova i slično. U sklopu ovog rada koristi se Jupyter bilježnica za treniranje modela. Glavna i najzanimljivija funkcionalnost Jupyter-a je implementacija mnogih programskih jezika poput Python-a, R-a i Julie u jednu bilježnicu ekstenzije .ipynb.

## 5 Implementacija

### 5.1 Hand Detector

Razvijena je web aplikacija Hand Detector u sklopu ovog rada koja omogućuje detekciju i praćenje dlana na slici i videu. Pokretanje aplikacije moguće je unutar naredbenog retka unutar direktorija unutar koje se aplikacija nalazi komandom:

```
flask run
```

Implementacija je podijeljena u više datoteka s nekoliko *python* modula i skripti te nekoliko *html* i *css* datoteka.

**routes.py** - središnji modul koji postavlja rute web aplikacije, to je tipični princip pisanja Flask aplikacije gdje su sve moguće rute upisane unutar jednog modula

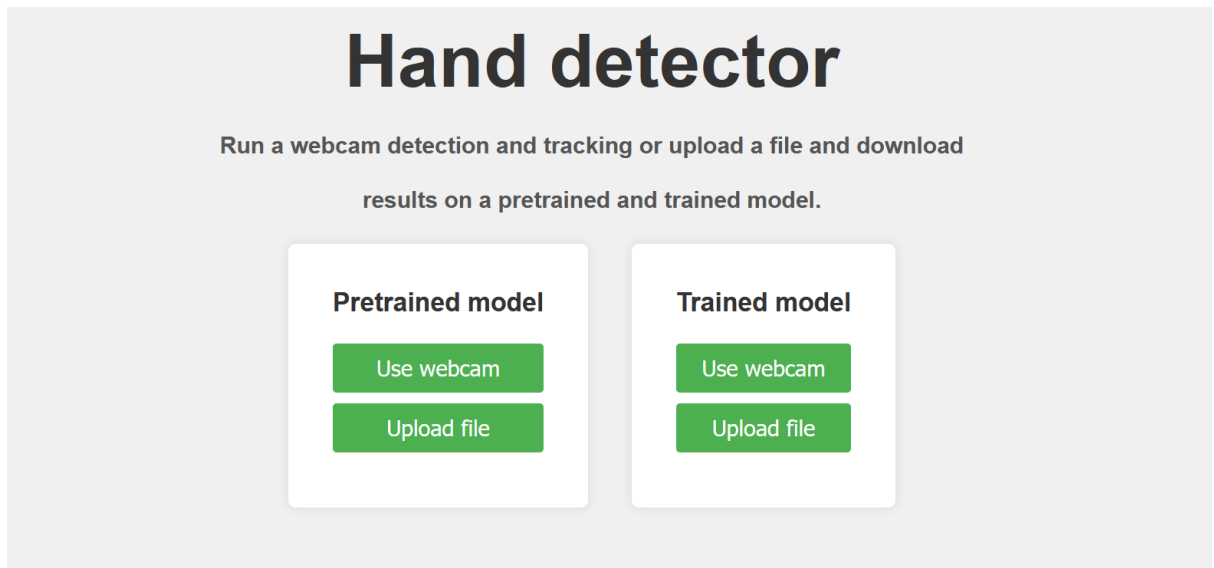
**templates** – datoteka koja sadrži *html* datoteke poput početne stranice, stranice za učitavanje vlastitih datoteka i stranice za pokretanje web kamere

**upload.py** – *python* modul koji se pokreće učitavanjem datoteke i njenim predviđanjem, unutar modula stvara se YOLOv8 model pomoću kojeg se zatim izvodi detekcija dlana, predviđanje razreda i u slučaju videa praćenje dlana, nakon uspješno provedene detekcije korisnik se preusmjerava na novu stranicu na kojoj može preuzeti novonastalu datoteku

**webcam.py** - *python* modul koji se pokreće prilikom pokretanja web kamere, modul je zadužen za njeno pokretanje i za izvođenje predviđanja, detekcije i praćenja dlana u stvarnom vremenu na web kameri korištenjem YOLOv8 modela koji se stvara unutar modula, web kamera se otvara kao skočni prozor te je moguće zaustaviti proces pritiskom tipke **ESC**

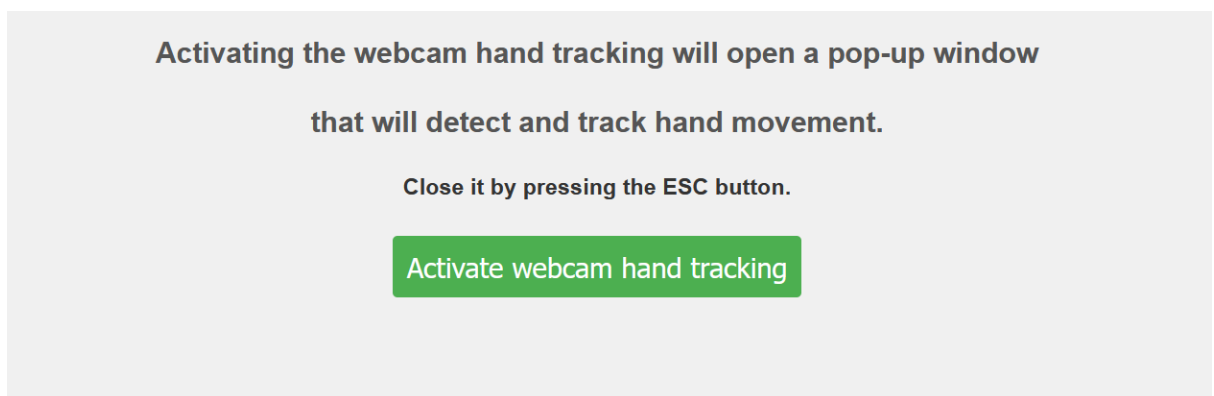
**best.pt** – datoteka koja predstavlja vlastito istrenirani YOLOv8 *PyTorch* model

Pokretanjem poslužitelja aplikacije možemo ju otvoriti lokalno na web-pregledniku. Aplikacija omogućuje dvije mogućnosti: upaliti web kameru te detektirati dlan u stvarnom vremenu i učitavanje videa ili slike s računala. Moguće je pokrenuti detekciju korištenjem već istreniranog modela ili korištenjem modela koji je istreniran u svrhu ovog rada. Početna stranica aplikacije prikazana je na slici **Slika 5.1**.



**Slika 5.1** Početna stranica web aplikacije *Hand Detector* na kojoj je moguć odabir detekcije dlana web kamerom ili učitavanjem datoteke

Odabirom detekcije u stvarnom vremenu dobivamo mogućnost upaliti web kameru i pokrenuti proces detekcije. U pozadini se pokreće *python* skripta *webcam.py* koja će pokrenuti web kameru i u stvarnom vremenu detektirati i pratiti dlan. Stavlja se granični okvir oko dlana i ispisuje razred kojem prepoznati objekt pripada s kojom vjerojatnošću. Za kompleksniji model postoji više razreda poput otvorenog dlana, zatvorene šake, geste za brojeve jedan i dva i slično, dok je za jednostavniji model moguće detektirati samo dlan. Sučelje stranice za praćenje i detekciju dlana u stvarnom vremenu korištenjem web kamere prikazano je na slici **Slika 5.2**.



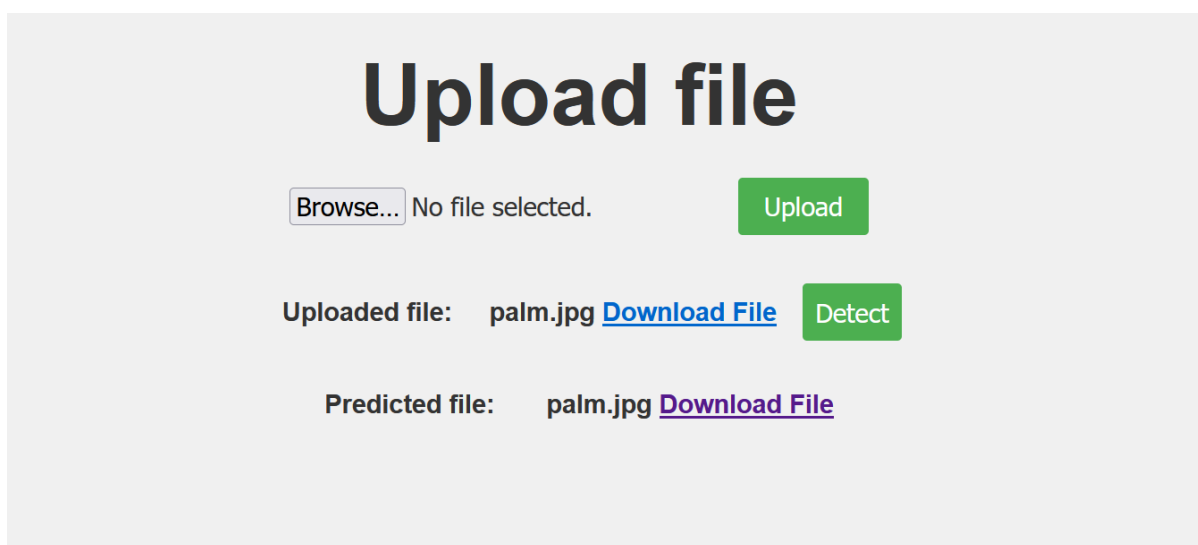
**Slika 5.2** Sučelje za pokretanje web kamere kao izvor za detekciju dlana



Učitavanje vlastitih slika i videa funkcionira tako da se korisniku da mogućnost odabira datoteke na kojoj želi obaviti procjenu, točnije nad kojom želi provest proces detekcije i praćenja dlana. Formati datoteke na kojom se može provesti procjena su .jpg, .png i .mp4.

Nakon što korisnik učitava datoteku, on ima mogućnost tu istu preuzeti na vlastito računalo kako bi se moglo provjeriti je li točna datoteka učitana. Zatim korisnik pritiskom na gumb za predviđanje pokreće *python* skriptu *upload.py* koja se pokreće u pozadini i za zadanu datoteku izvodi proces detekcije i praćenja dlana.

Pri završetku korisnik ima mogućnost preuzeti novu datoteku nad kojom je izvedeno predviđanje. Slika ili video bit će jednaka onoj učitanoj gdje će se detektirani dlan staviti unutar graničnog okvira i ispisati razred kojem prepoznati objekt pripada s kojom vjerojatnošću. Izgled stranice nakon što je korisnik učitao datoteku i proveo predviđanje prikazan je na slici **Slika 5.3**.



The screenshot displays a web interface titled "Upload file". It features a file selection area with a "Browse..." button and the text "No file selected.", alongside a green "Upload" button. Below this, the "Uploaded file:" section shows "palm.jpg" with a blue "Download File" link and a green "Detect" button. The "Predicted file:" section at the bottom also shows "palm.jpg" with a purple "Download File" link.

**Slika 5.3** Stranica za učitavanje lokalne datoteke nakon što je provedena detekcija

## 6 Eksperimenti i rezultati

Praćenje i detekcija dlana na slici i videu zahtjevan je zadatak za koji je potrebno odabrati kvalitetan model koji će precizno prepoznavati i klasificirati dlan. U sklopu rada, kao što je ranije navedeno, koriste se dva modela, istrenirani javno dostupni model te model istreniran na temelju skupova podataka.

U sklopu YOLOv8 postoje više dostupnih modela, koji se mogu koristiti za predikciju ili za treniranje vlastitih modela na temelju skupa podataka. Glavna razlika između varijanti YOLOv8 modela je u veličini i složenosti, oni veći su složeniji i točniji, ali zatim i sporiji dok su oni manji manje složeni i precizni, ali brži. Razlika u modelima preuzeta je s portala Medium [18]. Svojstva različitih modela vidimo u tablici **Tablica 6.1** Tablica prikazuje svojstva različitih YOLOv8 Modela. Ovisno o zadatku koji se rješava odabire se onaj model koji najbolje odgovara potrebama, na primjer za potrebe prepoznavanja u stvarnom vremenu uzimaju se manji, brži modeli, a za kompleksnije zadatke koji zahtijevaju visoku razinu preciznosti uzimaju se složeniji, veći modeli.

**Tablica 6.1** Tablica prikazuje svojstva različitih YOLOv8 Modela. Tablica je preuzeta s [18].

Varijanta	Veličina	Brzina	Srednja prosječna preciznost
<b>YOLOv8-XS</b>	2.4 MB	40 FPS	41.1% mAP
<b>YOLOv8-S</b>	3.4 MB	30 FPS	46.0% mAP
<b>YOLOv8-M</b>	8.1 MB	20 FPS	51.1% mAP
<b>YOLOv8-L</b>	53.6 MB	15 FPS	55.3% mAP
<b>YOLOv8-X</b>	111.1 MB	10 FPS	61.2% mAP

Treniranjem modela iz skupova podataka pomoću YOLO-a stvaraju se grafovi i tablice iz kojih se može doći do mnogih zanimljivih zaključaka vezanih uz upotrebljene podatke i modele. Jedan od takvih je graf koji prikazuje **F1** svojstvo određenih modela. Detalji računanja F1 svojstva preuzeti su s portala v7labs [20]. Detalji o izračunu:

- **TP** (engl. *true positives*): Broj slučajeva koji su stvarno bile pozitivne i koje je model ispravno predvidio kao pozitivne.
- **FP** (engl. *false positives*): Broj slučajeva koji su zapravo bili negativni, ali ih je model netočno predvidio kao pozitivne.
- **TN** (engl. *true negatives*): Broj slučajeva koji su stvarno bile negativne i koje je model ispravno predvidio kao negativne.
- **FN** (engl. *false negatives*): Broj slučajeva koji su zapravo bili pozitivni, ali ih je model netočno predvidio kao negativne.
- **Preciznost (P)**: Računa omjer točnih pozitivnih predikcija na cijelom skupu pozitivnih predikcija, računa se formulom (1).

$$\frac{TP}{TP+FP} \quad (1)$$

- **Odziv (R)**: Odziv ili osjetljivost računa omjer točnih pozitivnih predikcija u odnosu na sveukupan broj pozitivnih instanci, računa se formulom (2),

$$\frac{TP}{TP+FN} \quad (2)$$

- **F1**: Spaja Preciznost i Odziv u jednu formulu, računa se formulom (3).

$$\frac{2*P*R}{P+R} \quad (3)$$

Svojstvo F1 poprilično je korisno pri vrednovanju rada modela te nam može pružiti bolji uvid nego prethodne komponente zasebno. Grafovi F1 svojstva ukazuju na ocjenu modela (od 0 do 1, y os) za predmete za koje je siguran do određene razine (x os).

Od značaja je i graf koji prikazuje **PR** svojstvo, na njemu je vidljiv kompromis između preciznosti i odziva za različite pragove. Područje ispod krivulje predstavlja njihov odnos, iznos te površine ovisi o njihovom odnosu.

**Matrica zabune** [19] (engl. *confusion matrix*) kvadratna je matrica koja na sažet način prikazuje točne i netočne predikcije zajedno sa stvarnim razredom. Za binarnu klasifikaciju, matrica zabune je dimenzija 2x2 i prikazana je tablicom **Tablica 6.2**.

Matrica zabune nudi temeljitiju i vjerodostojniju analizu predviđanja, preciznosti, točnosti i učinkovitosti modela jer pruža sveobuhvatnu procjenu izvedbe modela za svaki od razreda. Na primjer ako skup podataka sadrži velik broj primjera razreda **A**, a malen broj primjera razreda **B** i rezultati nam pokazuju da je većina slika koje pripadaju razredu **A** točno klasificirano, a nijedna slika koja pripada razredu **B** nije točno klasificirana, to je loš model

iako je preciznost visoka jer jedan razred neuspješno klasificira. Razlog visokoj preciznosti je velik broj primjera razreda **A**. Informacije

**Tablica 6.2** Matrica zabune binarne klasifikacije za razred Pas

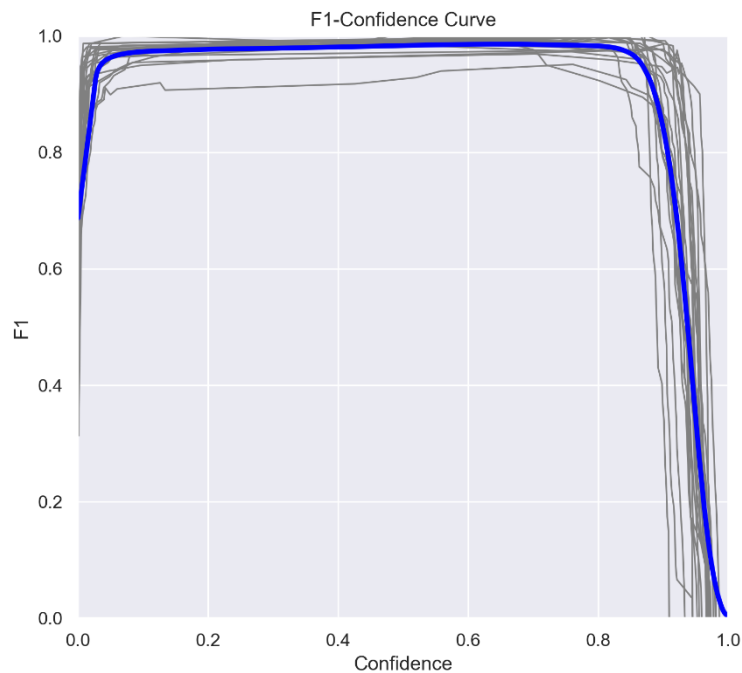
		Točna vrijednost	
		Pas	Nije Pas
Predvideno	Pas	TP	FP
	Nije Pas	FN	TN

## 6.1 Model tuned-hand-gestures

Prvi korišteni model je YOLOv8 javno dostupni model za detekciju dlana i gesti ruke *tuned-hand-gestures*. Model i izlazne slike treniranja modela preuzete su s [22]. Model je treniran koristeći *ultralytics* na skupu podataka označenih slika dlana i gesti ruke. Treniranje modela se izvršilo u 10 epoha. Model sadrži dvadeset klasa od kojih su neke: otvoreni dlan, šaka, broj jedan, broj dva i slično.

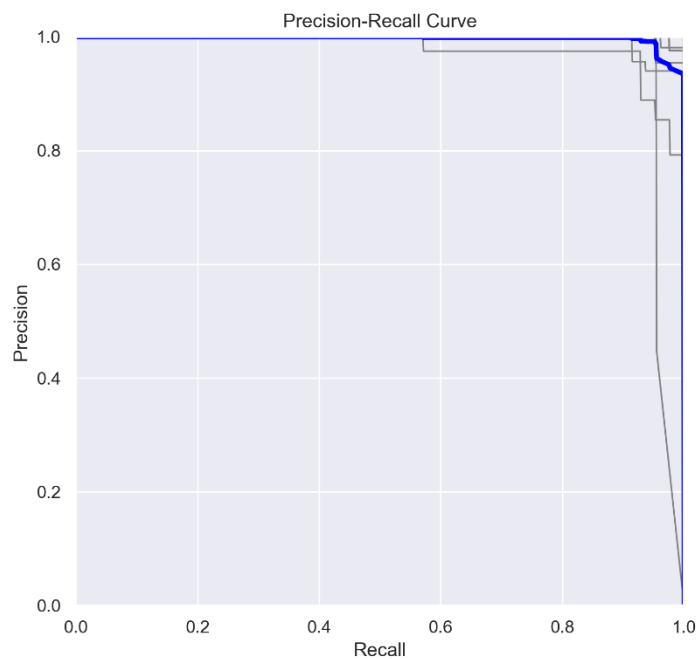
Model detektira i prati dlan s visokom preciznošću, ali je zato spor i kompleksan. Radi se o YOLOv8-X varijanti modela stoga pokretanje videa je *sporo*, prikazuje se 15 sličica u sekundi.

Slika **Slika 6.1** F1 graf za validaciju modela tuned-hand-gestures prikazuje F1 svojstvo za validaciju modela. Za očekivani raspon sigurnosti model postiže vrlo visoke rezultate, skoro 100%. Ovime se zaključuje da je ovo kvalitetan model koji će s visokom preciznošću izvršavati zadatak detekcije i praćenja dlana.



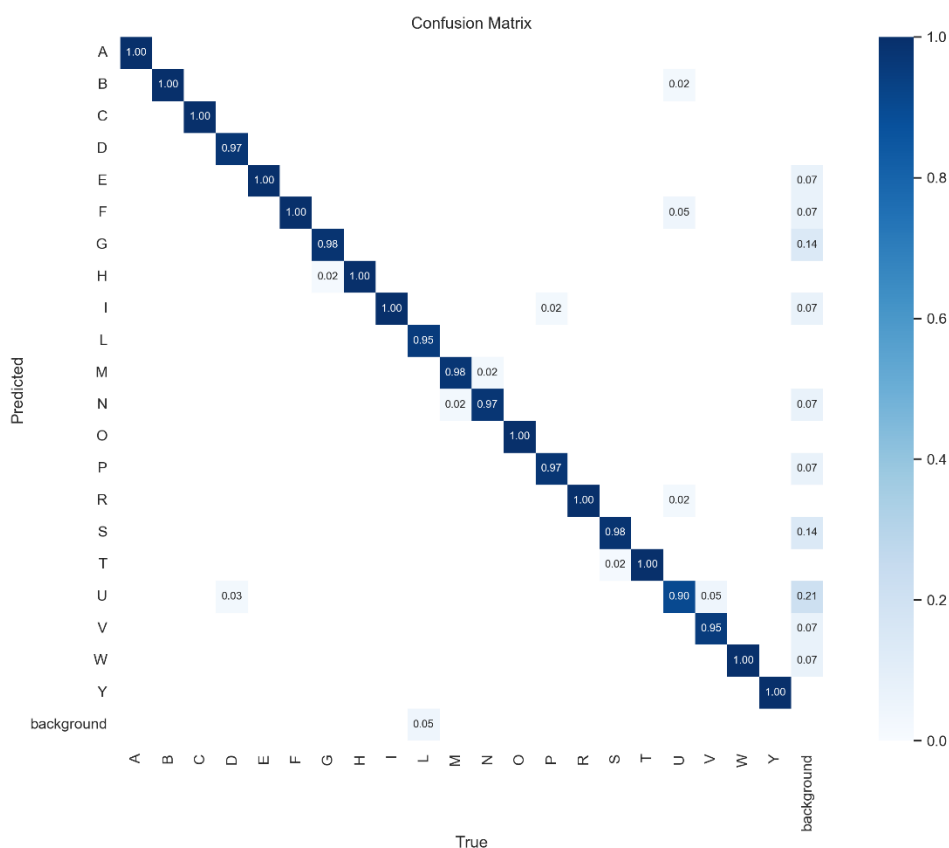
**Slika 6.1** F1 graf za validaciju modela tuned-hand-gestures

Na slici **Slika 6.2** PR graf modela tuned-hand-gestures je vidljiv PR graf ovog modela, budući da je površina ispod krivulje velika zaključujemo da su odziv i preciznost za ovaj model veliki što pokazuje visoku preciznost ovog modela.



**Slika 6.2** PR graf modela tuned-hand-gestures

Na temelju ova dva grafa može se zaključiti da će i matrica zabune prikazivati kvalitetne rezultate uz minimalna odstupanja što vidimo na slici **Slika 6.3**. Matrica zabune prikazuje mala odstupanja, ali generalno svi razredi su točno klasificirani sa skoro 100% preciznošću. Dijagonalne matrice ukazuju na uspješnost modela.



**Slika 6.3** Matrica zabune za model tuned-hand-gestures

## 6.2 Model treniran na skupu podataka

Za treniranje modela potrebni su pravilno označeni ulazni podaci. Čitav proces zadovoljavajućeg modela specijaliziranog za određene svrhe oslanja se upravo na te ulazne podatke. Preciznost i generalizacija modela ovisit će o odabiru početnih slika stoga je taj korak veoma bitan.

U svrhu ovog rada treniran je skup podataka *Hand Detecting* [23]. Skup podataka sastoji se od 847 slika dlana u različitim položajima i s različitim pozadinama. Neke slike sadrže različite količine zamućenosti što pomaže u treniranju modela da provodi točne predikcije i na

slikama lošije kvalitete. Slike su podijeljene za skup slika za učenje (744 slika), skup slika za provjeru (72 slika) i skup slika za testiranje (31 slika). Skup je podijeljen na dvije klase:

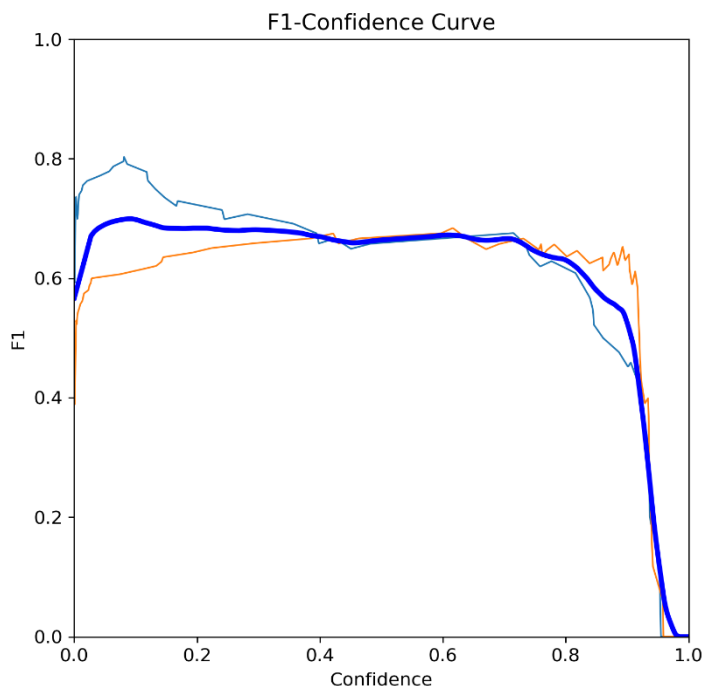
- -
- *dt - v2*

Budući da obje klase predstavljaju dlan, obje su preimenovane u *palm* u svrhu ovog rada.

Treniranje modela odrađeno je unutar *Jupyter* bilježnice koristeći YOLOv8 model u 100 epoha u trajanju od sat vremena i 10 minuta. Rezultati treniranja se spremaju u obliku modela *last.pt* i *best.pt* što predstavlja YOLOv8 modele spremne za korištenje za detekciju objekata. Koristi se *best.pt* jer on sadrži težine iteracije koja postiže najbolje rezultate.

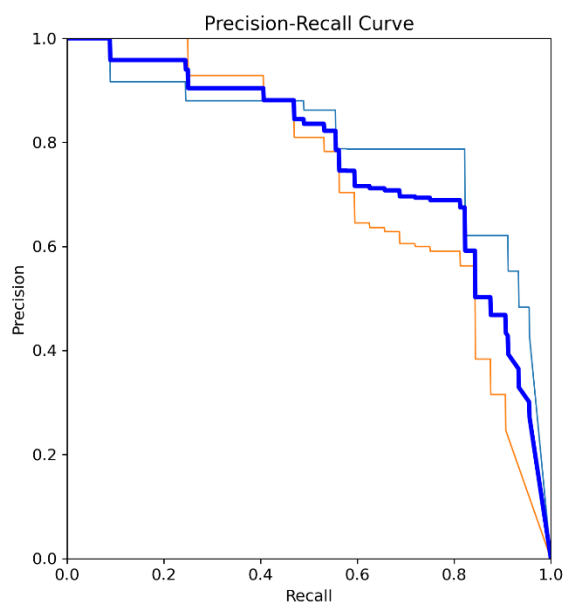
Model detektira i prati dlan s relativno visokom preciznošću i visokom brzinom. Radi se o YOLOv8-S varijanti modela stoga pokretanje videa je *brzo*, prikazuje se 30 sličica u sekundi.

Slika **Slika 6.4** prikazuje F1 svojstvo za validaciju modela. Za očekivani raspon sigurnosti model postiže lošije rezultate, oko 60%. Model postiže nedovoljno kvalitetne preciznosti, točnije lošije od modela *tuned-hand-gestures* koji je treniran na većem skupu podataka i u više epoha. Također možemo vidjeti kako sigurnosti prepoznavanja predmeta naglo padaju nakon 80%, što se može objasniti malim brojem predmeta u toj kategoriji.



**Slika 6.4** F1 graf za validaciju modela HandDetection

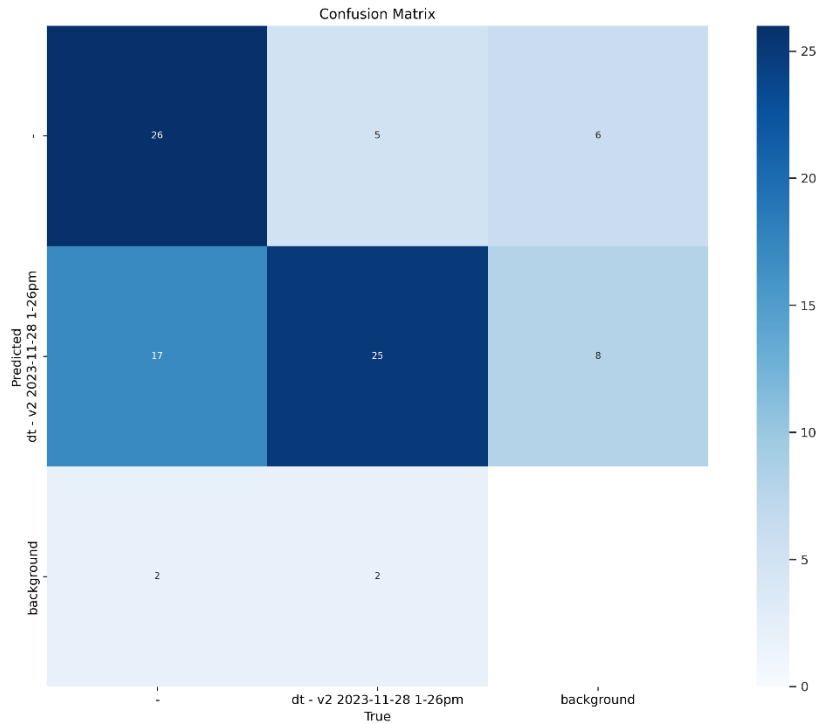
Na slici **Slika 6.5** je vidljiv PR graf ovog modela, površina ispod krivulje ne zauzima velik dio grafa stoga se zaključuje da odziv i preciznost za ovaj model nisu jednako dobri kao oni modela *tuned-hand-gestures*. Vidljivo je iz grafa da krivulja okružuje gornji desni kut grafa što znači da je ovo dobar klasifikator koji održava i dobru preciznost i visok odziv.



**Slika 6.5** PR graf modela HandDetection

Dodatno svojstvo koje pokazuje da je ovo, iako ne savršen, dobar model i klasifikator je relativno dijagonalna matrica zabune (**Slika 6.6**) uz par stršećih vrijednosti.





**Slika 6.6** Matrica zabune za model HandDetection

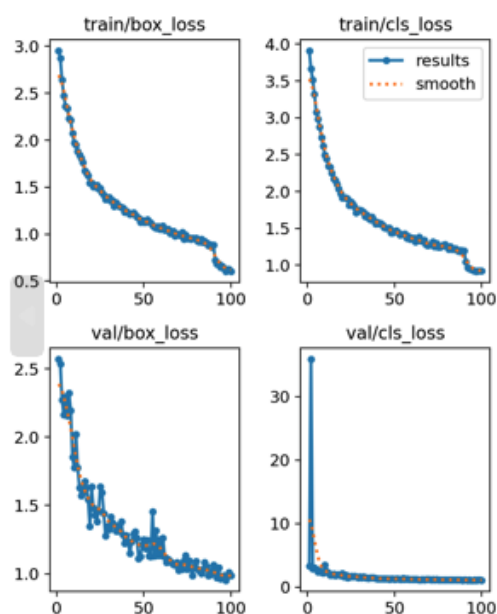
Analizom treniranja dostupni su grafovi koji prikazuju dodatne krivulje koje pokazuju uspješnost treniranja. Detalji o metrikama preuzeti su sa [20] i [24]:

- ***box\_loss***: Mjeri pogrešku predviđenog graničnog okvira. Potiče model da se prilagodi preciznijem određivanju okvira. Računa se kao srednja kvadratna pogreška između predviđenih parametara graničnog okvira i točnih vrijednosti.
- ***cls\_loss***: Mjeri pogrešku klasifikacije objekta. Potiče model da točno klasificira objekt. Izračunava se kao prosjek entropijskog gubitka za predviđanja za sve klase.
- ***mAP50*** (*engl. mean average precision*): Prosječna preciznost izračunata nad primjerima koji se lako detektiraju, točnije na onim modelima na kojima se predviđen i točan razred preklapaju do 50%.
- ***mAP50-95*** (*engl. mean average precision*): Prosjek srednje prosječne preciznosti izračunate na pragovima gdje se točan i predviđen razred preklapaju u rasponu od 50% do 95%. Daje sveobuhvatan pregled performansi modela na različitim razinama težine detekcije.

Slika **Slika 6.7** prikazuje *box\_loss* i *cls\_loss* grafove na skupovima podataka za treniranje i skupu za provjeru. Kod skupa za treniranje vidimo konzistentan pad grafa što znači da se

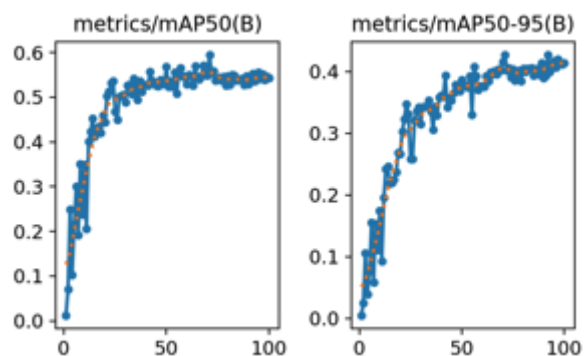
svakom epohom poboljšava predviđanje graničnog okvira i razreda objekta. Iz tog podataka se zaključuje da model nije istreniran do najkvalitetnijeg oblika s danim podacima te u slučaju treniranja na više epoha bi se model dodatno poboljšavao jer su grafovi i dalje padajući. Pogreške iznose manje od 1%, što se vidi na y-osima na grafovima.

Skup podataka za provjeru daje slične rezultate, osim što je pogreška predviđanja graničnog okvira manje konzistentna i ima par *šiljaka*. Pogreška predviđanja razreda se već u prvih 3 epohe drastično smanji što znači da model brzo uči prepoznavanje razreda, što ima smisla s obzirom na to da postoje samo dva različita razreda.



**Slika 6.7** Slika prikazuje rezultate treniranja; 4 grafa koji predstavljaju metrike pogreške graničnog okvira i pogreške razreda na skupu za provjeru i skupu za treniranje

Na slici **Slika 6.8** se vide iznosi metrika  $mAP50$  i  $mAP50-95$ , krivulje su rastuće što znači da treniranjem raste preciznost modela. Unatoč rastu funkcija, nijedna ne dostiže vrijednosti veće od 55% što nije savršen iznos preciznosti i znači da model ne daje najkvalitetnije rezultate. Kako bi se to promijenilo potrebno je ili koristiti drugačiji skup podataka s više slika, koristiti drugi YOLOv8 model (neki drugačije veličine) ili trenirati u više epoha.



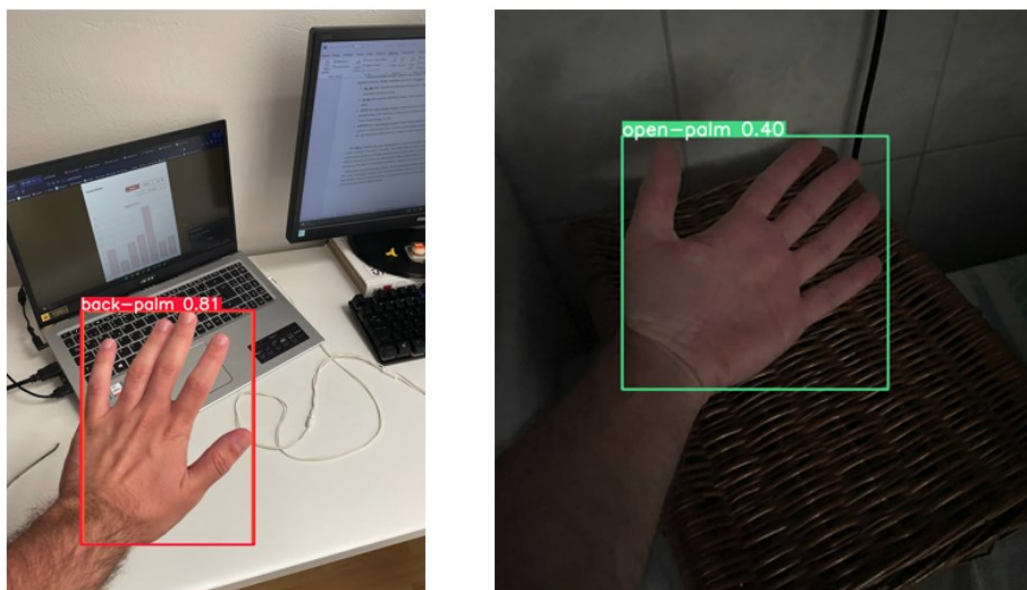
**Slika 6.8** Slika prikazuje mAP50 i mAP50-95 grafove za trenirani model

## 6.3 Eksperimenti

Eksperimentalni dio odnosi se na pokretanje raznih dobivenih modela nad skupovima podataka, pri čemu su dolazi do mnogih zaključaka. Modeli su korišteni za testiranje funkcionalnosti i kvalitete rada algoritma za praćenje i detekciju dlana nad vlastitim podacima, preuzetim podacima i korištenjem web kamere.

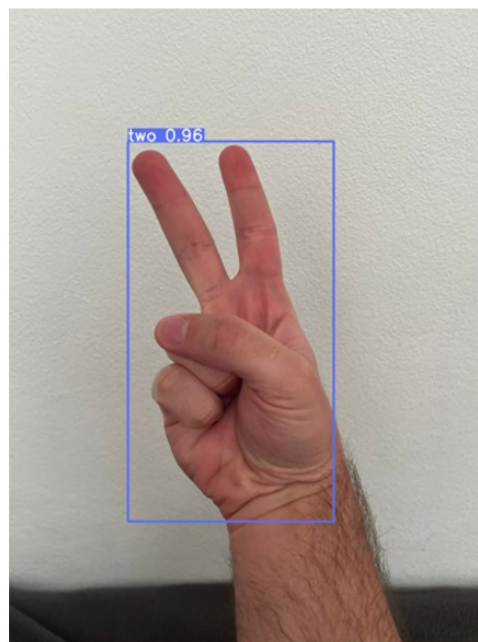
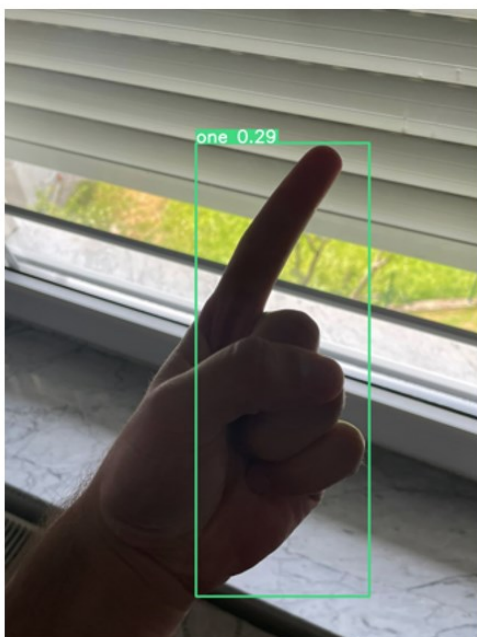
Testiranje nad prvim, javno dostupnim modelom gesta ruke daje kvalitetne rezultate, detekcija dlana precizno je napravljena s visokom točnosti na većini primjera, a praćenje dlana na videima i korištenjem web kamere konzistentno je i precizno.

Slika **Slika 6.9** prikazuje rezultate predviđanja i detekcije dlana za dva različita primjera, uočljivo je da mnogi elementi slike poput osvjetljenja i pozadine utječu na preciznost i sigurnost predviđenog razreda.

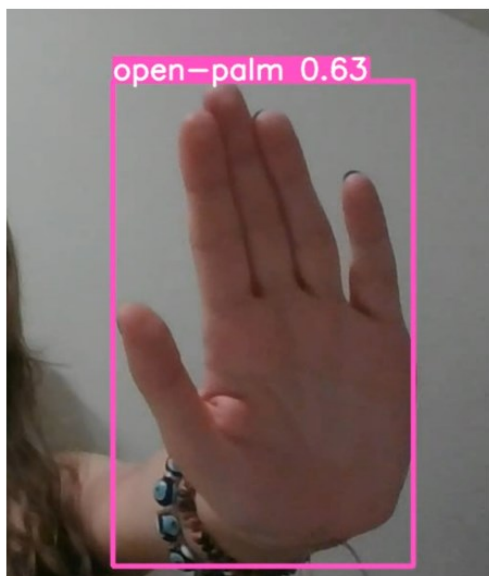


**Slika 6.9** Slika prikazuje dva primjera detekcije dlana model hand-tuned-gestures. Lijeva slika sadrži pozadinske elemente koji mogu loše utjecati na preciznost dok je desna slika slabijeg osvjetljenja.

Na slikama **Slika 6.10** se također uočava da ovaj model uspješno klasificira dlan u različitim položajima, prepoznaje se da prva slika predstavlja gornju stranu dlana, a druga slika otvoreni dlan. Nadalje, slika pokazuje da model prepoznaje i kompleksnije razrede poput geste brojeva. Na lijevoj slici se prepoznaje da se drži jedan prst u zraku, dok desna slika predstavlja dva prsta. Lijeva slika manje je sigurnosti zbog slabijeg osvjetljenja i jasnoće ruke od desne slike. Na slikama **Slika 6.11** vidi se da model uspješno prepoznaje i klasificira i objekte slabije kvalitete, ali s manjom sigurnošću.



**Slika 6.10** Slika prikazuje dva primjera detekcije dlana i geste ruke modela hand-tuned-gestures. Lijeva slika sadrži pokazuje gestu broja jedan i slabijeg je osvjetljenja, a desna pokazuje gestu broja dva.

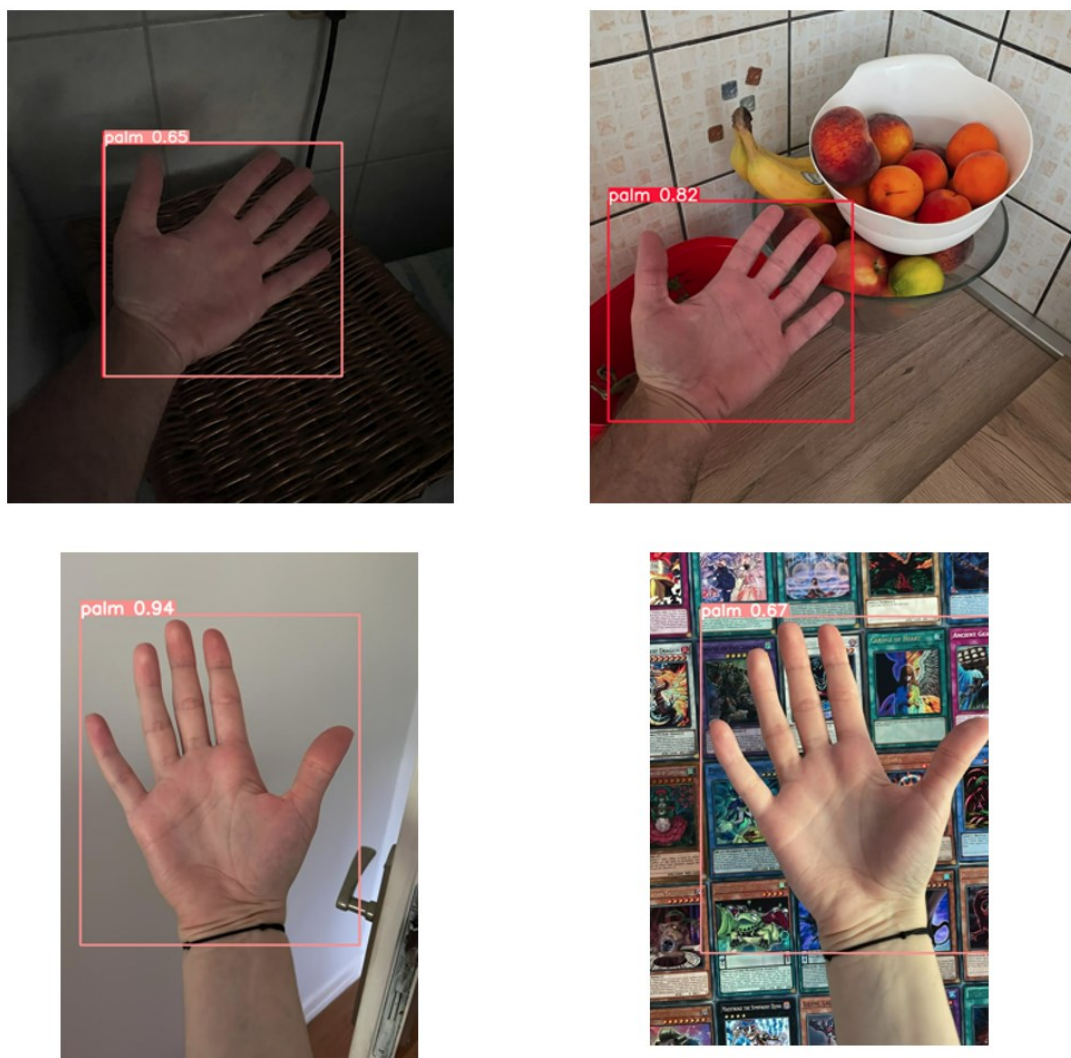


**Slika 6.11** Slika prikazuje dva primjera detekcije dlana i geste ruke modela hand-tuned-gestures. Lijeva slika predstavlja dlan, a desna pokazuje položaj šake.

Možemo primijetiti da bi nam ovakvi modeli pomogli u kreiranju algoritama koji mogu prepoznavati geste ruku te nekih kompleksnijih zadataka poput prepoznavanja **znakovnog jezika** i automatskog generiranja teksta.

Drugi model također daje precizne i točne rezultate, za neke primjere daje rezultate veće sigurnosti od prvog modela. Taj model je brži, ali zato manje kompleksan, samo prepoznaje dlan, ne može prepoznati ni klasificirati različite položaje ruke poput prvog modela.

Slika pokazuje da je ovaj model uspješan za detekciju dlana u raznim uvjetima. Također primijeti se da će sigurnost predviđanja ovisiti o pozadini i osvjetljenju na slici. Zaključujemo da na sigurnost najviše utječe osvjetljenje dlana, točnije slike na kojima se dlan više *uklapa* u sliku lošije i teže će klasificirati i detektirati dlan.



**Slika 6.12** Slika pokazuje rešetku od 4 različita primjera rezultata predviđanja drugog modela. Slike pokazuju dlan u različitim uvjetima, s različitim pozadinama i osvjetljenjem.

## 7 Zaključak

Praćenje i detekcija dlana aktualan je problem koji ima širok raspon primjena i koristi se u raznim područjima. Odabrao sam rješavati taj problem jer ima zanimljive primjene poput prepoznavanja gesta i praćenja dlana u svrhe manipulacije virtualnim objektima u sklopu tehnologija virtualnih stvarnosti, u video igricama koje koriste te tehnologije za stvaranje realističnih iskustava i interakcije računala i čovjeka i nekih korisnijih primjera poput prepoznavanja gesta ruke i znakovnog jezika.

Unatoč eksponencijalnom razvoju algoritama i modela koje rješavaju ovaj problem i daleko većoj i dostupnijoj računalnoj moći u odnosa na prošlo stoljeće, ovaj problem i dalje zahtijeva daljnje usavršavanje i razvoj novih tehnologija. Modeli korišteni u sklopu rada dovoljno su dobri te svaki imaju svoju svrhu, prvi je kompleksniji i uspijeva prepoznati različite razrede, dok drugi je brz, ali prepoznaje samo dlan. Za neke kompleksnije probleme koji implementiraju sustave praćenja dlana ovi bi modeli trebali doradu, treniranjem na većem skupu podataka, treniranjem u više epoha ili korištenjem kompleksnijih modela. Sustavu se mogu dodati neke nove funkcionalnosti poput prepoznavanja gesta ruke i njihov ispis ili korištenje već razvijene funkcionalnosti praćenja dlana za jednostavno crtanje po virtualnom ekranu.

Razvojem sustava za detekciju i praćenje dlana sam stekao kvalitetno iskustvo i upoznao se s mnogim alatima koji se mogu koristiti za razvoj modela, analizu podatkovnih skupova i stvaranje jednostavnih web aplikacija.



# Literatura

- [1] Wikipedia, *Artificial Intelligence*, Poveznica: [https://en.wikipedia.org/wiki/Artificial\\_intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), pristupljeno: 12. svibnja 2024.
- [2] Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder, *Umjetne neuronske mreže*, Poveznica: [https://www.fer.unizg.hr/download/repository/UI\\_12\\_UmjetneNeuronskeMreze\[1\].pdf](https://www.fer.unizg.hr/download/repository/UI_12_UmjetneNeuronskeMreze[1].pdf), pristupljeno: 11. lipnja 2024.
- [3] Alliance, *Using Convolutional Neural Networks for Image Recognition*, Poveznica: <https://www.edge-ai-vision.com/2015/11/using-convolutional-neural-networks-for-image-recognition/>, pristupljeno: 7. lipnja 2024.
- [4] Geeks4Geeks, *Introduction to Convolutional Neural Networks*, Poveznica: <https://www.geeksforgeeks.org/introduction-convolution-neural-network/>, pristupljeno: 7. lipnja 2024.
- [5] Medium, *Deep Learning Course — Lesson 5: Forward and Backward Propagation*, Poveznica: <https://medium.com/@nerdjock/deep-learning-course-lesson-5-forward-and-backward-propagation-ec8e4e6a8b92>, pristupljeno: 14. lipnja 2024.
- [6] IBM, *What is machine learning (ML)?*. Poveznica: <https://www.ibm.com/topics/machine-learning>, pristupljeno: 29. svibnja 2024.
- [7] Marko Čupić, *Umjetna inteligencija, Umjetne neuronske mreže*, Poveznica: <http://java.zemris.fer.hr/nastava/ui/ann/ann-20180604.pdf>, pristupljeno: 22. svibnja 2024.
- [8] Marko Čupić, *Umjetna inteligencija, Uvod u strojno učenje*, Poveznica: <http://java.zemris.fer.hr/nastava/ui/ml/ml-20220430.pdf>, pristupljeno: 22. svibnja 2024.
- [9] Bojana Dalbelo Bašić, Jan Šnajder, *Strojno učenje*, Poveznica: <https://www.fer.unizg.hr/download/repository/UI-2020-10-StrojnoUcenje.pdf>, pristupljeno: 11. lipnja 2024
- [10] LinkedIn, *Supervised vs Unsupervised Learning*, Poveznica: <https://www.linkedin.com/pulse/supervised-vs-unsupervised-learning-whats-difference-smriti-saini>, posjećeno: 4. lipnja 2024.
- [11] SuperAnnotate, *What is image classification? Basics you need to know*. Poveznica: <https://www.superannotate.com/blog/image-classification-basics>, posjećeno: 17. lipnja 2024.



- [12] Qualcomm, *Classification, Object Detection and Image Segmentation*. Poveznica: <https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation>, pristupljeno: 10. lipnja 2024.
- [13] Python. Poveznica: <https://www.python.org/>, pristupljeno: 13. lipnja 2024.
- [14] Ultralytics, *YOLOv8*. Poveznica: <https://docs.ultralytics.com/>, pristupljeno: 7. lipnja 2024.
- [15] Encord, *Yolo Object Detection Explained*. Poveznica: <https://encord.com/blog/yolo-object-detection-guide/>, pristupljeno: 13. lipnja 2024.
- [16] Flask. Poveznica: <https://flask.palletsprojects.com/en/3.0.x/>, pristupljeno: 3. lipnja 2024.
- [17] Jupyter. Poveznica: <https://jupyter.org/>; pristupljeno 22. svibnja 2024.
- [18] Muhammad Abdullah, *YOLO Working principle, difference between its different Variants and Versions*. Poveznica: <https://medium.com/@muhabd51/yolo-working-principle-difference-between-its-different-variants-and-versions-95b8ad7b95ab>; pristupljeno 2. lipnja 2024.
- [19] Geeks4Geeks, *Confusion Matrix in Machine Learning*. Poveznica: <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>, pristupljeno: 13. lipnja 2024.
- [20] Rohit Kundu, *F1 Score in Machine Learning: Intro & Calculatio.*, Poveznica: <https://www.v7labs.com/blog/f1-score-guide>, pristupljeno 3. lipnja 2024.
- [21] Jan Šnajder, *Vrednovanje modela Strojno učenje 1, UNIZG FER, ak. god. 2022./2023*. Poveznica: <https://www.fer.unizg.hr/download/repository/SU1-2022-P21-VrednovanjeModela.pdf>; pristupljeno 9. svibnja 2024.
- [22] Lewis Watson, *YOLOv8x-tuned-hand-gestures model*. Poveznica: <https://huggingface.co/lewiswatson/yolov8x-tuned-hand-gestures/tree/main>; pristupljeno 24. svibnja 2024.
- [23] Roboflow, *Hand Detecting Computer Vision Project*. Poveznica: <https://universe.roboflow.com/learn-rpis1/hand-detecting-h7z1n>; pristupljeno 12. svibnja 2024.
- [24] Erfan Akbarnezhad, *YOLOv8 Projects #1 "Metrics, Loss Functions, Data Formats, and Beyond"*. Poveznica: <https://www.linkedin.com/pulse/yolov8-projects-1-metrics-loss-functions-data-formats-akbarnezhad>, posjećeno: 10. lipnja 2024.

# Sustav za detekciju i praćenje dlana

## Sažetak

U svrhu ovog rada napravljen je sustav za detekciju i praćenje dlana korištenjem koncepata računalnog vida, poput klasifikacije slika i videa, segmentacije i detekcije objekata i praćenja objekata. Sustav je ostvaren koristeći dva modela YOLOv8 temeljena na konvolucijskim neuronskim mrežama. Prvi model je javno dostupni unaprijed trenirani model velike kompleksnosti, a drugi je trenirani model nad javno dostupnim skupom podataka. Razvijena je jednostavna web aplikacija koja omogućava korištenje modela za detekciju i praćenje dlana koja sadrži jednostavno sučelje čime se omogućava korištenje modela i njegovih funkcionalnosti na vlastitim primjerima. Rad daje uvide u rezultate treniranja i uspoređuje ova dva modela s primjerima.

**Ključne riječi:** konvolucijske neuronske mreže, duboko učenje, detekcija objekata, YOLOv8 model

## Abstract

In this paper, a system for hand detection and tracking was developed using computer vision concepts such as image and video classification, segmentation, object detection, and object tracking. The system was implemented using two YOLOv8 models based on convolutional neural networks. The first model is a publicly available pre-trained model of high complexity, and the second is a trained model on a publicly available dataset. A simple web application was developed that allows users to use models for hand detection and tracking, featuring a simple interface that allows image and video input. The paper provides insights into the training results and compares these two models with examples.

**Keywords:** convolutional neural networks, deep learning, object detection, YOLOv8 model