PRIMJER PRIMJENE UMJETNE INTELIGENCIJE

Putnik, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
STROJARSKI FAKULTET U SLAVONSKOM BRODU

DIPLOMSKI RAD

sveučilišnog diplomskog studija

Marko Putnik
0152209436

Mentor diplomskog rada:
prof. dr. sc. Roberto Lujić

Slavonski Brod, 2022.

I. AUTOR

Ime i prezime: Marko Putnik
Mjesto i datum rođenja: Slavonski Brod, 02.09.1997. g.
Adresa: Donja Vrba Sv. Filipa i Jakova 95, 35208 Ruščica

STROJARSKI FAKULTET U SLAVONSKOM BRODU

II. DIPLOMSKI RAD

Naslov: PRIMJER PRIMJENE UMJETNE INTELIGENCIJE

Naslov na engleskom jeziku: EXAMPLE OF THE APLICATION OF ARTIFICIAL INTELLIGENCE

Ključne riječi: Umjetna inteligencija, informatička podrška

Ključne riječi na engleskom jeziku: Artificial intelligence, IT support

Broj stranica: 54 slika: 41 tablica: 4 bibliografskih izvora: 30

Ustanova i mjesto gdje je rad izrađen: STROJARSKI FAKULTET U SLAVONSKOM BRODU

Stečen akademski naziv:
Sveučilišni magistar inženjer strojarstva

Mentor rada: prof. dr. sc. Roberto Lujić

Obranjeno na **Strojarskom fakultetu** u Slavonskom Brodu

Dana: _____

Oznaka i redni broj rada: _____

SVEUČILIŠTE U SLAVONSKOM BRODU
POVJERENSTVO ZA DIPLOMSKI ISPIT

Slavonski Brod, 11. siječnja 2022.

DIPLOMSKI ZADATAK br. 2021-2022

Pristupnik: **Marko Putnik (0152209436)**
Studij: Diplomski studij: Strojarstvo
Smjer: Logistika proizvodnje

Zadatak: **PRIMJER PRIMJENE UMJETNE INTELIGENCIJE**

Opis zadatka:

1. UMJETNA INTELIGENCIJA, PODJELA, KARAKTERISTIKE...
2. INFORMATIČKA PODRŠKA IZ PODRUČJA PREPOZNAVANJA SLIKA
3. IZRADA PROGRAMSKOG KODA ZA ODABRANI PRIMJER
4. PRIMJENA PROGRAMSKOG KODA NA ODABRANOM PRIMJERU
5. ZAKLJUČAK

Zadatak uručen pristupniku: 18. siječnja 2022.
Rok za predaju rada: 18. srpnja 2022.

Mentor:

prof. dr. sc. Roberto Lujić

Predsjednik povjerenstva za diplomski ispit:

prof. dr. sc. Ivica Kladarić

I Z J A V A

Izjavljujem da sam završni rad izradio samostalno, koristeći se vlastitim znanjem i literaturom. Izjavljujem da rad nije napisan protiv pravila i da ne krši bilo čija autorska prava. Također izjavljujem da u radu nije korišten dio bilo kojih drugih radova.

Marko Putnik

SAŽETAK

U radu je obrađena tema korištenja tehnologije umjetne inteligencije. Napredovanje tehnologije mijenja način na koji svijet funkcionira u nadi za poboljšanjem i lakšim životom. Vizualni sustav (računalni vid i prepoznavanje uzoraka) je naziv za tehnološku i znanstvenu disciplinu koja olakšava način na koji računala mog razumjeti video ili sliku.

U radu je pojašnjen pojam umjetne inteligencija, izvršena se podjela i navedene ključne karakteristike. Opisan je i pojam mekog računalstva.

Potom je dan pregled informatičke podrške iz područja umjetne inteligencije.

U nastavku rada opisan je program Visual Studio Code i programski jezik Python koji su korišteni u praktičnom dijelu. Opisane su dvije biblioteke programskog jezika Python, Numpy koja je standardna biblioteka programskog jezika i OpenCV biblioteka.

U praktičnom dijelu rada fokusiralo se na mogućnosti integracije biblioteke OpenCV za prepoznavanje i označavanje objekata koji su registrirani. Na kraju je izrađen programa koji koristi kameru kao izvor slike i videa koja omogućava računalu vid kao i označavanje prepoznatih objekata.

ABSTRACT

The paper deals with the topic of using artificial intelligence technology. Advances in technology are changing the way the world works in hopes of making life better and easier. Visual systems (computer vision and pattern recognition) is the name for the technological and scientific discipline that facilitates the way computers can understand a video or image.

The paper explains the concept of artificial intelligence, divides it and lists key characteristics. The concept of soft computing is also described.

Then an overview of IT support in the field of artificial intelligence was given.

In the continuation of the paper, the program Visual Studio Code and the programming language Python, which were used in the practical part, are described. Two Python programming language libraries are described, Numpy, which is the standard programming language library, and the OpenCV library.

In the practical part of the work, the focus was on the possibilities of integrating the OpenCV library for recognizing and marking registered objects. At the end, a program was created that uses a camera as a source of image and video, which enables the computer to see and mark recognized objects.

S A D R Ž A J

1 UVOD

Svakim danom svjedoci smo sve bržih i značajnijih promjena ne samo u prirodi i društvu već i u gospodarstvu. Uzrok promjena je čovjek koji stjecanjem novih znanja na dnevnoj bazi izaziva iste u svim područjima ljudskog djelovanja, pa tako i same proizvodnje.

Razvojem ne samo novih znanja već i informacijsko-komunikacijskih tehnologija omogućilo je poduzećima prijelaz sa klasičnog načina planiranja i kontrole proizvodnje na oblike podržane novim znanjima i tehnologijama. Neka od takvih rješenja dana su primjenom alata i metoda baziranih na načelima umjetne inteligencije.

Kako bi poduzeće opstalo na tržištu mora ispuniti zahtjeva kupca u smislu kvalitete, cijene i roka isporuke. Kao jedan od problema postavlja se i problem kvalitete proizvoda/usluge. Dosadašnji način u kojem se u pravilu u završnoj kontroli zaključivalo da li proizvod/usluga odgovara zahtjevima, traži se u proaktivnom zamjenskom načinu kontrole cijelog procesa. Time bi se u konačnici smanjio škart ili dorada a samim time i nepotrebni troškovi. Jedan od alata koji u tome može pomoći je svakako vizualni sustav (računalni vid i prepoznavanje uzoraka) na koji bi način proizvodi/usluge koji ne zadovoljavaju uvjete kvalitete već u ranoj fazi bili identificirali i uklonjeni iz daljnjeg procesa,

Cilj rada je pokazati mogućnost primjene vizualnog sustava na problemu kontrole proizvoda/usluge.

Međutim, uvođenjem i primjenom istog trebalo bi osim direktnog poboljšanja same funkcije kontrole i minimalizacije troškova iste u sustavima imalo i indirektna poboljšanja u smislu promjena u samoj organizacijskoj strukturi poduzeća; kvalitetnijoj pripremi proizvodnje i same proizvodnje; oslobađanja radnika od neposredne odgovornosti za kvalitetu proizvoda/usluge.

2 UMJETNA INTELIGENCIJA

Umjetna inteligencija (UI) je dio računalne znanosti koja uključuje razvoj računalnih programa za izvršavanje zadataka za koje bi se inače zahtijevala ljudska inteligencija. Algoritmi koje umjetna inteligencija može izvoditi je učenje, percepcija, razumijevanje jezika, logičko zaključivanje i/ili rješavanje problema. UI sa koristi na mnogo načina u današnjem svijetu, od osobnih asistenata do autonomnih automobila. Ona se jako brzo razvija. [1]

Karakteristike umjetne inteligencije [2]:

- Sposobnost reagiranja na informacije dostupne okolišu
- Sjećanje i učenje iz određenih iskustava
- Sposobnost rješavanja specifičnih problema
- Prilagodljivost
- Sposobnost osjetilne percepcije
- Sposobnost upravljanja
- Sposobnost optimizacije
- Sposobnost učinkovitog rukovanja velikim količinama informacija
- Mjerljivo za kvantificiranje izvedbe i buduća ulaganja

Postoji širok skup tehnika u umjetnoj inteligenciji kao što su planiranje, automatizacija, lingvistika, obrada prirodnog jezika, znanost o odlučivanju, automatizacija robota, vizija, pristranost, i druge.

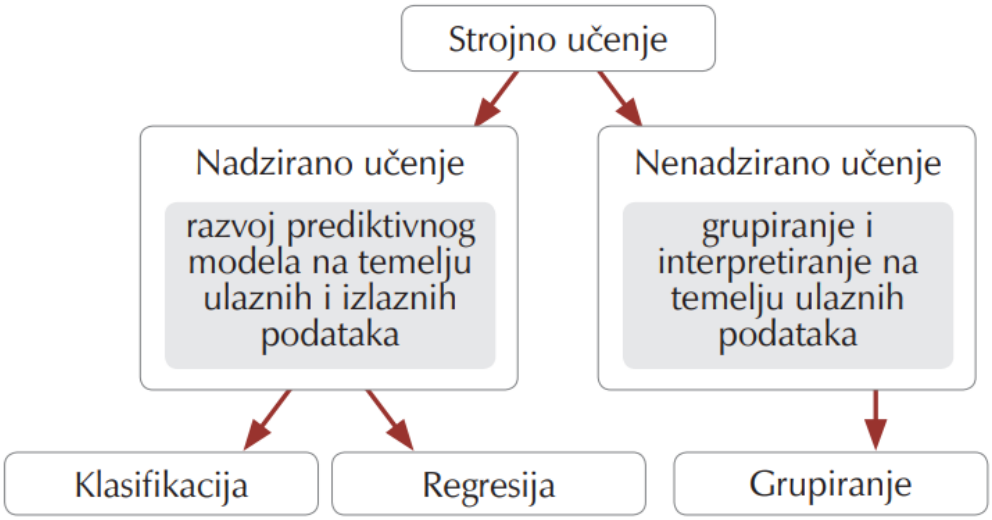
Glavne podjele umjetne inteligencije [3]:

1. Strojno učenje
2. Neuronske mreže
3. Robotika
4. Ekspertni sustavi
5. Neizrazita logika
6. Obrada prirodnih jezika
7. Genetički algoritmi
8. Meko računalstvo
9. Prepoznavanje uzoraka…

U nastavku će biti pojašnjeni neki od pojmova AI.

STROJNO UČENJE

Algoritam strojnog učenja ima zadatak pronaći poveznice i prirodne uzorke u podatcima ta prema tome steći uvid i tek onda predviđati i odlučivati. Svakodnevno se primjenjuju za donošenje bitnih odluka u medicinskoj dijagnostici, majstorstvom trgovanja dionicama, predviđanje potrošnje energije i drugo. Na slici 2.1 može se vidjeti osnovna podjela strojnog učenja. [4]



Slika 2.1. *Osnovna podjela strojnog učenja* [4]

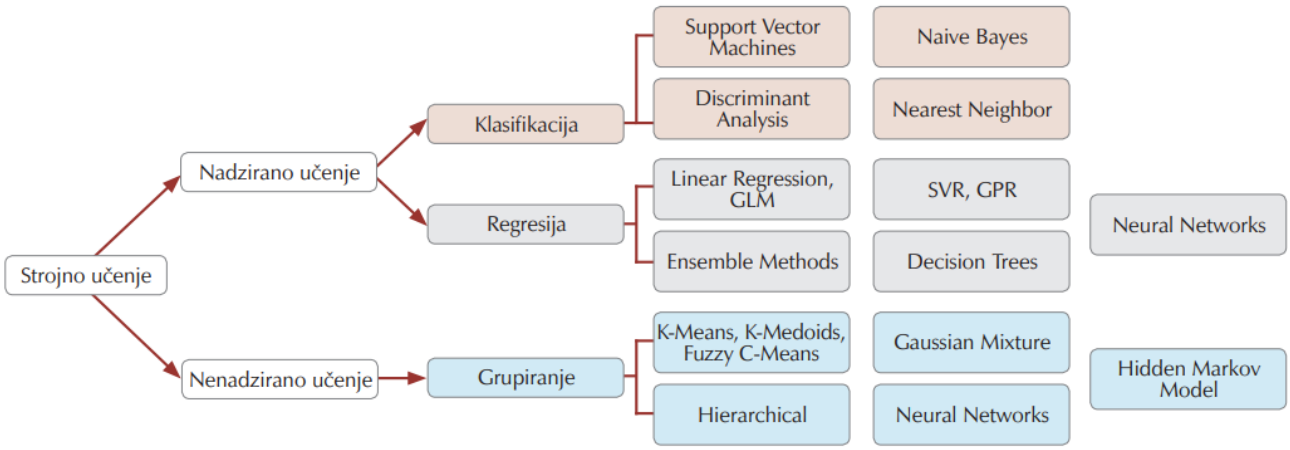
Algoritmi u slučaju nadziranog učenja rade s poznatim skupom ulaznih podataka i poznatim izlaznim skupom podataka, te zatim uvježbavaju model za predviđanje.

Nadzirano učenje se primjenjuje u postupcima:

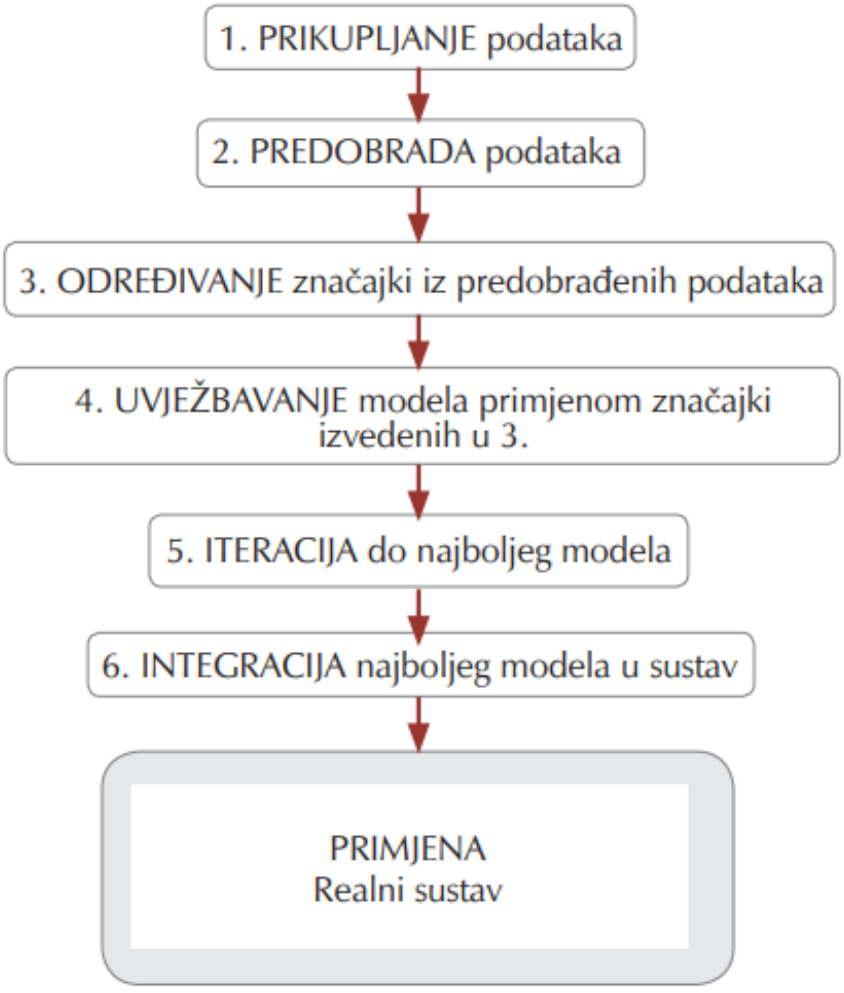
- Razvrstavanje ili klasifikacija – predviđanje direktnih odziva
- Regresija – predviđanje kontinuiranih odziva

Nenadzirano učenje može pronaći unutarnje strukture ili skrivene uzorke u podatcima. Tako se otkrivaju strukture i grupiraju uzorci. Primjena im je za izvođenje zaključaka iz niza podataka koje čine samo ulazni podatci bez da poznaju izlazni skup podataka.

Strojno učenje se primjenjuje kod rješavanja kompleksnih problema ili zadataka sa velikom količinom podataka i velikim brojem varijabli bez postojanih razrađenih formula ili jednadžbi (modela). Na slikama 2.2 i 2.3 se može vidjeti podjela metoda strojnog učenja i koraci razvoja modela primjenom strojnog učenja. [4]



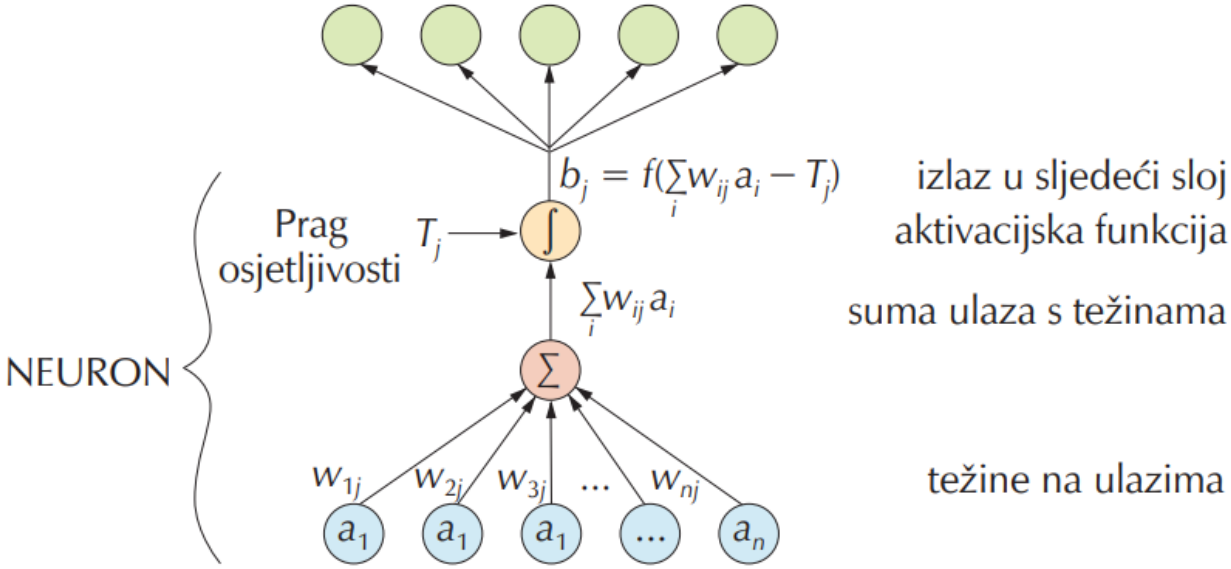
Slika 2.2 *Podjela metoda strojnog učenja* [4]



Slika 2.3 *Koraci razvoja modela primjenom strojnog učenja* [4]

NEURONSKE MREŽE

Razumijevanje strukture i funkcije ljudskog mozga i neurona olakšalo je istraživanje i razvoj umjetnih neuronskih mreža. Svaka neuronska mreža se sastoji od ulaznog sloja, skrivenog sloja i izlaznog sloja. Ulazni sloj sadrži ulazne varijable, te se u skrivenom sloju sve ulazne varijable množe sa težinskim koefcijentom w_i, te se ti signali zbrajaju i njihov zbroj se uspoređuje s pragom osjetljivosti neurona T_j. U skrivenom sloju ulazi se zbrajaju pomoću funkcije sumiranja i tako svoju internu aktivaciju. Većim zbrojem otežanih signala od praga osjetljivosti neurona se aktivira funkcija f koja generira izlazni signal neurona iznosa b_j. Na slici 2.4 se može vidjeti model neurona.[5]



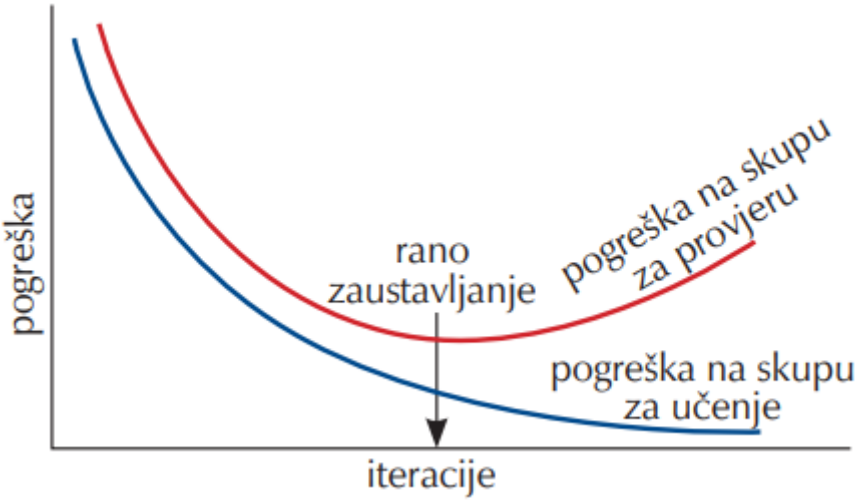
Slika 2.4 *Prikaz neurona* [5]

Postupkom učenjem podrazumijeva se iterativni postupak podešavanja tj. optimiranja težinskih faktora prema osnovi pogreške između proračunatog modela i prave vrijednosti veličine koja je mjerena. Nakon jednog prolaza informacije neuronskom mrežom tada se generira vrijednost koja se zatim uspoređuje sa stvarnom vrijednosti. Tim razlikama izračunate i stvarne vrijednosti korigira se težinski faktor.

Pravilo širenja unatrag predstavlja podešavanje težinskih faktora. Danas postoji puno struktura i algoritama učenja zahvaljujući intenzivnom razvoju i primjene neuronskih mreža u praksi.[4]

Neuronska mreža, pomoću korekcije težinskih faktora, uči predviđati stvarne vrijednosti se smanjuje razliku između predviđene i stvarne vrijednosti izlazne veličine. O generalizaciji mreže i kvaliteti govori kriterij pogreške. Pretreniranje mreže se sprječava provjerom mreže na novom skupu podataka prilikom učenja.[5]

Kada mreža s visokom točnošću opisuje vladanje podataka na skupu podataka na kojem je razvijana, a izvan toga skupa daje loše rezultate, javlja se pretreniranje. Pogreška učenja i pogreška provjere se smanjuje prije točke minimuma pogreške provjere, te nakon te točke pogreška učenja i dalje pada, a greška provjere raste, te je to i pokazatelj pretreniranja neuronske mreže. Pretreniranje neuronskih mreža može se izbjeći tako da se učenje zaustavi u trenutku kada pogreška provjere počne rasti. Na slici 2.5 je prikazano učenje neuronske mreže tehnikom ranog zaustavljanja.[5]



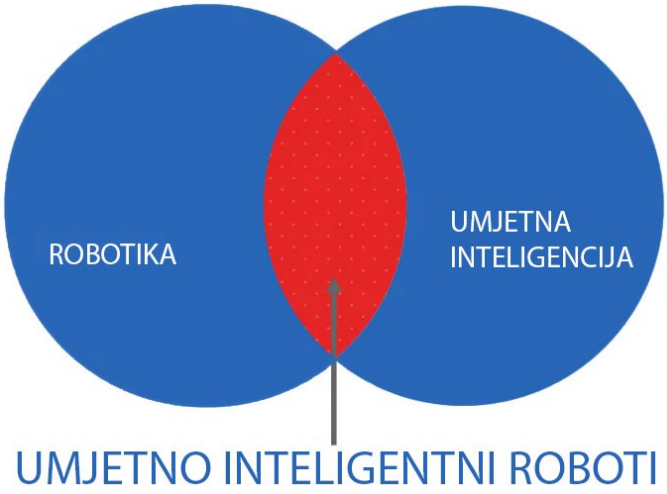
Slika 2.5 *Prikaz ovisnosti pogreške učenja i pogreške provjere na pretreniranje* [5]

Koraci razvoja neuronske mreže:

1. Planiranje i provedba eksperimenta
2. Prikupljanje i obrada podataka
3. Odabir strukture modela neuronske mreže
4. Učenje neuronske mreže i vrednovanje modela [5]

ROBOTIKA

Kod robota računalni vid osigurava umjetna inteligencija da pronađu put, otkriju put i reagiraju. Robotika danas ne bi bila moguća bez umjetne inteligencije. S godinama razvoja polja robotike ono se razvilo i uključilo zadatke kao što su navigacija i prepoznavanje objekata. Autonomna vozila su jedna od ponajboljih primjena umjetne inteligencije. Takva vozila mogu voziti sama bez potrebe za ljudskim vozačem, jer koriste senzore i kamere za navigaciju u svom okruženju. Na slici 2.6 je prikazana shema povezanosti robotike i umjetne inteligencije. Robotika i umjetna inteligencija se koriste i u medicini za precizne operacije u što kraćem roku s minimalnim rizicima. [6]

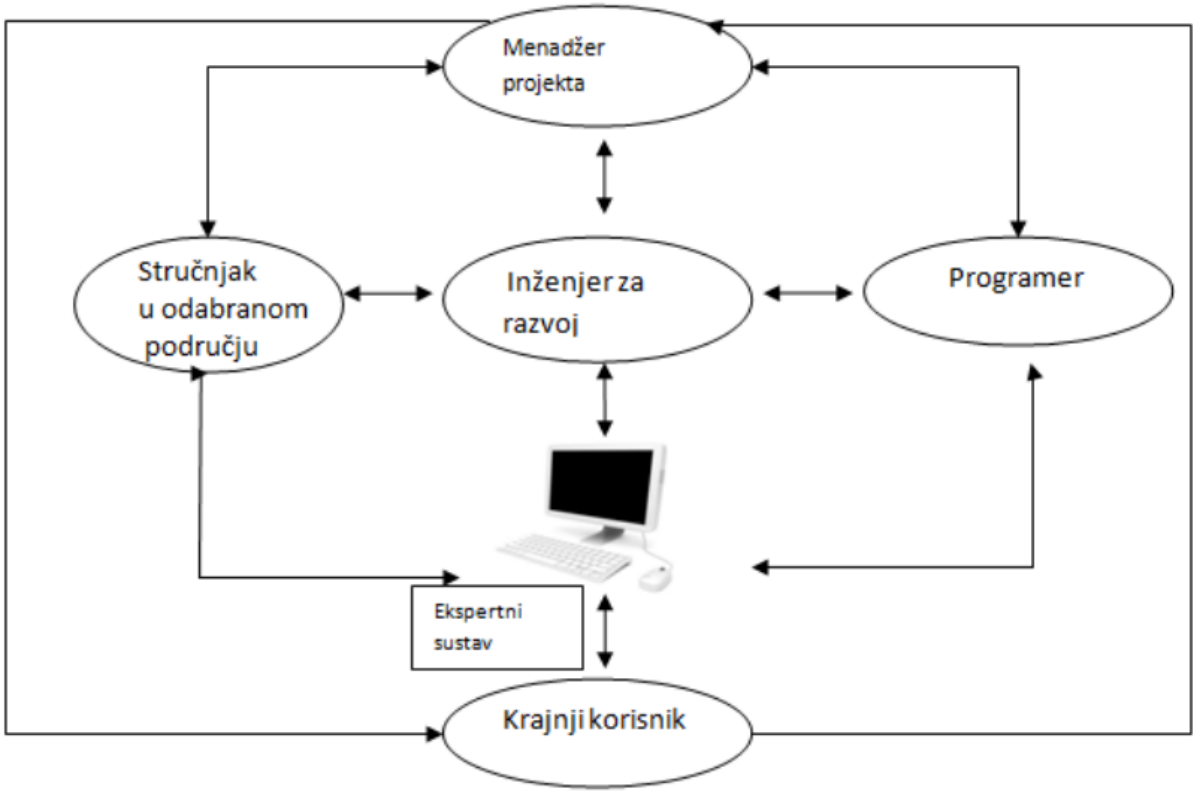


Slika 2.6 *Prikaz povezanosti robotike i umjetne inteligencije* [6]

EKSPERTNI SUSTAVI

Da bi se razvio ekspertni sustav potrebno je pet članova (slika 2.7.):

- inženjer za razvoj,
- stručnjak u promatranom području,
- programer,
- menadžer projekta i
- korisnik [7].



Slika 2.7 *Prikaz povezanosti članova ekspertnog sustava* [8]

Ljudi koji posjeduju znanje i koji su u stanju to znanje predstaviti kao niz pravila koja vode do rješenja određenih problema nazivaju se ekspertima. [7]

Uspjeh ekspertnog sustava ovisi znatno o tome kako članovi funkcioniraju kao tim. Izrada ekspertnog sustava zasnovana je da bi ljudi rješavali svoje probleme koristeći svoje znanje. Ekspertni sustav se sastoji od, može se vidjeti na slici 2.8: [9]

- baze znanja,
- baze podataka,
- algoritma za zaključivanje,
- modul za objašnjenja i
- korisničkog sučelja.

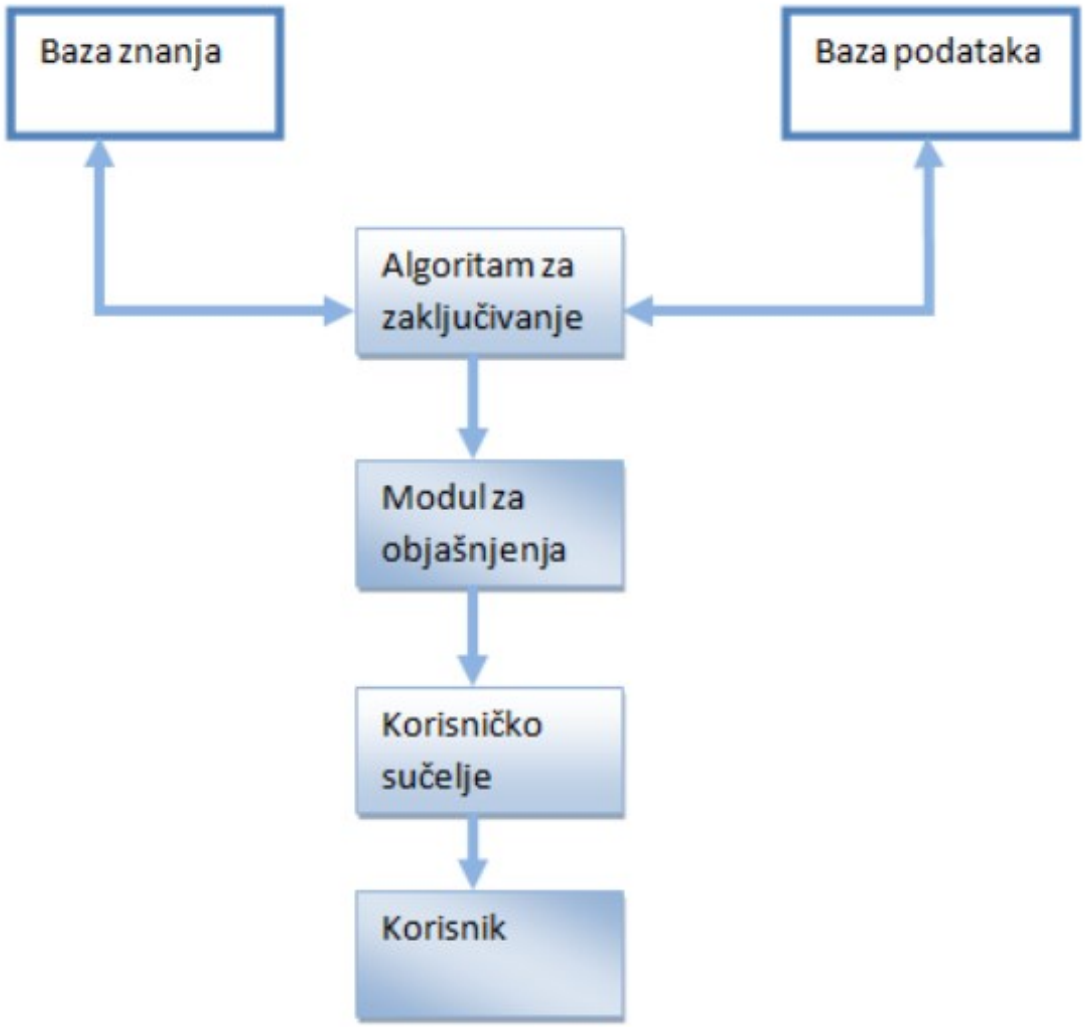
Baza znanja sadrži podatke o području koje je korisno za rješavanje problema, ono je prikazano kao niz pravila sa prikazanim vezama, strategijama, preporukama i *if* ili *then* strukturu svakog pravila.

Baza podataka se sastoji od niza činjenica za usporedbu sa *if* dijelovima pravila u bazi znanja. Korisnik unosi informacije poznate ekspertnom sustavu za traženje rješenja problema.

Algoritam za zaključivanje povezuje bazu podataka i bazu znanja te daje krajnje rješenje problema. Centralni dio svakog ekspertnog sustava je algoritam za zaključivanje i predstavlja pokušaj kopiranja ljudskog znanja eksperata u algoritam.

Modul za objašnjenja korisniku omogućuje da dobije korake kako je sustav došao do zaključka i zašto je potrebna određena činjenica. Može pomoći korisniku da potvrdi dobiveno rješenje ili da ga može doraditi prateći logiku ekspertnog sustava prilikom rješavanja problema.

Korisničko sučelje je komunikacijska veza izmeću ekspertnog sustava koji daje rješenje i korisnika. Trebalo bi biti što lakše i jednostavno za razumjeti.[7]



Slika 2.8 *Elementi ekspertnog sustava* [9]

NEIZRAZITA LOGIKA

Neizrazita logika je namijenjena za modeliranje logičkog razmišljanja s nepreciznim ili nejasnim izjavama. Istinite vrijednosti se tumače kao stupnjevi istine. Istinita vrijednost logički složene tvrdnje određena je istinitom vrijednošću njenih komponenti, tj. kao kod klasične logike nameće se istinska funkcionalnost.

Neizvjesnost u neizrazitoj logici se odnosi na neizrazitosti, nejasnoće, kolebanja i promjenjivosti u mišljenu i izražavanju (ne)izvjesnost pojave događanja kod računa vjerovatnosti se nepristrano prosuđuje ponavljanjem pokusa. Neizrazita logika je formalizacija i matematizacija pristranosti, neizrazitosti i neodređenosti.

Neizrazita logika se primjenjuje u prirodnim, tehničkim i društvenim disciplinama kao što su:

- Sustavi vođenja i upravljanja
- Sustavi podrške u odlučivanju
- Tehnički sustavi umjetne inteligencije.

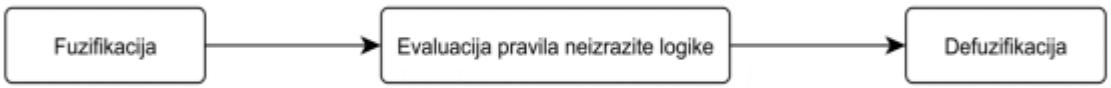
Prednosti neizrazite logike:

- Veća vjerojatnost da će odražavati probleme stvarnog svijeta od klasične logike
- Algoritmi imaju manje hardverske zahtjeve od klasične logike
- Mogu proizvesti točne rezultate s nepreciznim ili čak netočnim podacima.

Nedostaci neizrazite logike:

- Zahtjeva široku provjeru valjanosti
- Sustavi upravljanja ovise o ljudskom znanju.

Često se koristi u umjetnoj inteligenciji i kontrolerima strojeva, te se može primijeniti i na softverima za trgovanje. Zbog toga što neizrazita logika oponaša kako ljudi donose odluke, najkorisnija je u modeliranju složenih problema s nejasnim ili dvosmislenim ulazima Na slici 2.9 se vidi proces odlučivanja neizrazite logike.



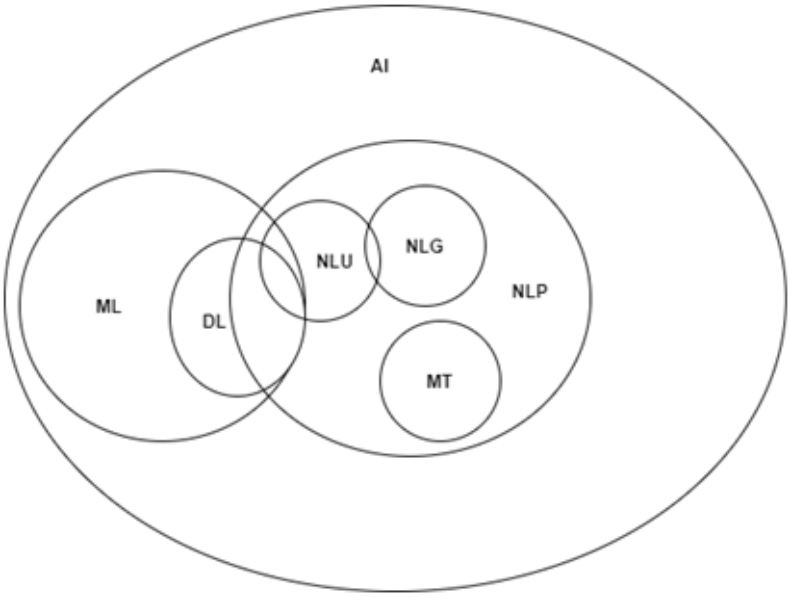
Slika 2.9 *Proces odlučivanja neizrazite logike*

Glavni dio sustava upravljanja kod neizrazite logike su pravila te su oni najbolji pokazatelji načina kako je sustav dizajniran. Pravilima je moguće jasno utvrditi međusobni odnos između ulaznih veličina te na koji način će se njihovo djelovanje u danom trenutku odraziti na izlazne veličine. Zbog velike varijabilnosti svaki sustav je specifičan obzirom na namjenu za koju je izrađen. Obzirom na broj ulaznih varijabli raste i broj pravila, te zbog toga je iznimno važno razmišljanje idealnog načina konstrukcije pravila.

OBRADA PRIRODNIH JEZIKA

Obrada prirodnih jezika je dio područja umjetne inteligencije povezane s znanošću o jeziku. Ona se koristi kako bi računalo „razumjelo“ prirodne jezike. Obrada prirodnih jezika se bavi pronalaženjem načina upotrebe računala da razumije ili obrađuje ljudske jezike – prirodni jezici. Koristi se za prevođenje ili pretvaranje podataka s prirodnog jezika na jezik razumljiv računalu. Na slici 2.10 se može vidjeti gdje se u cijelom sustavu umjetne inteligencije nalazi obrada prirodnih jezika. [6]

AI – Umjetna inteligencija
NLP – Obrada prirodnih jezika
NLU – Razumijevanje prirodnih jezika
NLG – Generiranje prirodnog jezika
MT – Strojno prevođenje
ML – Strojno učenje
DL – Duboko učenje



Slika 2.10 *Prikaz pojmova povezanih s obradom prirodnih jezika*

Sa znanstvene perspektive, cilj obrade prirodnih jezika je izrada modela pokusnih principa na kojima se zasniva razumijevanje i izrada ljudskih jezika. Sa inženjerske perspektive, obrada prirodnih jezika pokriva razvoj raznih aplikacija za lakšu interakciju između čovjeka i računala.[6] Obično aplikacije obrade prirodnih jezika uključuju:

- raspoznavanje govora
- leksičku i semantičku analizu jezika
- strojno prevođenje
- automatsko sumiranje
- analizu mišljenja
- dohvaćanje informacija
- odgovaranje na pitanja i dr. [10]

Kod obrade prirodnih jezika najviše se obraća pažnja na analizu, primjenu računalnih algoritama i dizajn te na način predstavljanja za obradu prirodnih jezika. Cilj obrade prirodnih jezika je implementacija efikasnih algoritama, analiza rečenične strukture i značenje zadane rečenice. [10]

Obrada prirodnih jezika istražuje primjenu računalnih algoritama na zadatke kao što su izvlačenje informacija iz teksta, prevođenje teksta na druge jezike, mogućnost automatiziranog odgovora na pitanja, vođenje razgovora s računalom na prirodnom jeziku i razumijevanje računala ljudskih tekstualnih ili govornih naredbi.

Danas se obradi prirodnih jezika pristupa korištenjem strojnog učenja, kojim se omogućava stvaranje složenih programa na temelju ulazno – izlaznih primjera. [11]

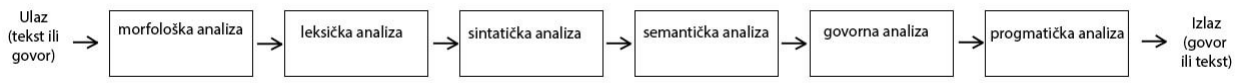
Najveći izazov i problem kod obrade prirodnih jezika je dvosmislenost na više razina [11]:

- značenje riječi
- morfologija
- sintatička svojstva
- uloge i veze između dijelova teksta.

Problem dvosmislenosti ljudi rješavaju na način da uzimaju u obzir širi smisao, prethodno znanje i iskustvo, ali ni tada se ne mogu izbjeći problemi u komunikaciji. [12] Računala rješavaju problem dvosmislenosti slično kao i ljudi tako što uzimaju široki kontekst oko riječi i zaključuje na temelju prošlih slučajeva.

Razine obrade prirodnih jezika dijele i međusobno isprepliću, prikazane na slici 2.11: [10]

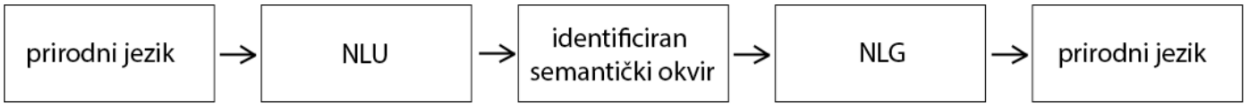
- morfološka razina – prva faza analize, ako se radi o opisanom jeziku – tekstu
- leksička razina – svrstavanje riječi u leksičke kategorije
- sintaktička razina – podjela rečenica na sastavne dijelove
- semantička razina – otkrivanje značenja u rečenicama
- govorna razina – određivanje značenja rečenice
- pragmatička razina – jasno određivanje konteksta rečenice.



Slika 2.11 *Opći model razumijevanja prirodnih jezika* [13]

Identifikatori prirodnih jezika i generator prirodnih jezika dijelovi su istraživačke tematike i razine obrade prirodnih jezika. Oni predstavljaju različite dijelove procesa za reproduciranje ljudske komunikacije i ponašanja putem računala.

Strojno prevođenje je istraživačko područje koje pokriva procesiranje i lingvistiku prirodnih jezika. Vrlo je slično procesu govora stroja i čovjeka na nekom od prirodnih jezika jer generiranje odgovora računala na prirodnom jeziku uključuje i jednu vrstu prevođenja. Strojno prevođenje kao i strojno generiran razgovor mapira znakove u niz unutar koda prevođenja kako bi sam razumio. Na primjer kao niz riječi iz hrvatskog jezika u niz riječi na njemačkom jeziku, a kod strojnog razgovora pretvara niz riječi i znakova prirodnog jezika u niz znakova računalom izrađenog odgovora. Na slici 2.12 je prikazan model razumijevanja prirodnih jezika. [13]



Slika 2.12 *Model razumijevanja prirodnih jezika* [13]

Razvojem snage procesora računala ostvaruju se nove sposobnosti u obradi velikog broja različitih podataka nestrukturiranog tipa, kao što je to video, slika, govor i tekst. Kako su se sa godinama razvijale neuronske mreže računala pojavila se nova vrsta učenja koja se naziva „duboko učenje“. Umjesto da Duboko učenje koristi skupinu već definiranih svojstava, ono funkcionira iz mnogo velikog niza primjera. Učenje je hijerarhijsko jer počinje osnovnim elementima (riječima ili znakovima) i nastavlja identifikacijom složenijih struktura (niz riječi ili izraza) dok ne dobije potpunu analizu objekta koji se analizira. Metode dubokog učenja koje su zasnovane na neuronskim mrežama pokazuju porast u obradi prirodnih jezika, a i u području strojnog prevođenja postaju popularne. [8]

GENETSKI ALGORITAM

Genetski algoritam koristi se za rješavanje problema optimizacije u strojnom učenju i temeljen je na pretraživanju. Važan je jer rješava probleme za čije bi rješavanje trebalo puno vremena jer su teški. Koristi se u različitim aplikacijama u stvarnom životu kao što su podatkovni centri, razbijanje kodova , dizajn elektroničkih sklopova, umjetna kreativnost i obrada slika. [14]

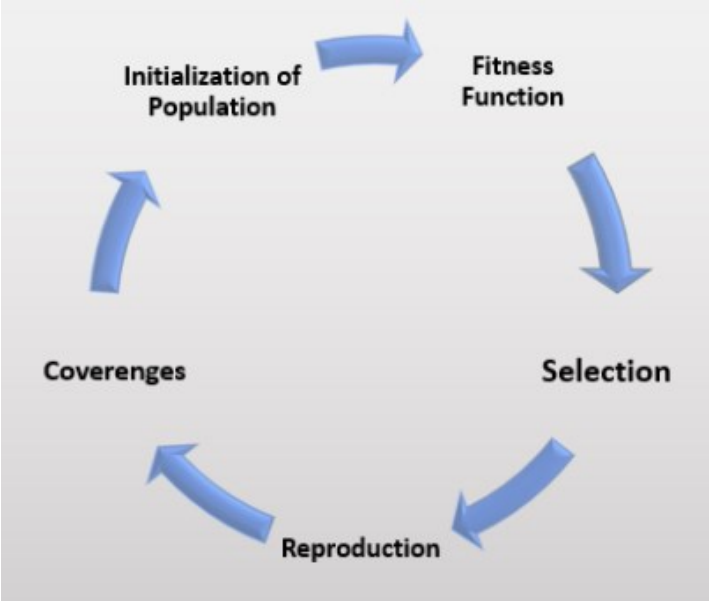
Ovaj algoritam je heuristički algoritam pretraživanja korišten za rješavanje problema optimizacije i pretraživanja. Ovaj algoritam je podskup genetskih algoritama koji se koriste za računanje. Oni koriste model genetike i prirodnog odabira kako bi pružili rješenja za probleme.

Genetski algoritmi imaju bolju inteligenciju od algoritama pretraživanja nasumično jer koriste podatke iz povijesti za prevođenje pretraživanja u područje s najboljom verzijom unutar prostora rješenja. Genetski algoritmi se također baziraju na ponašanju kromosoma i njihovih genetskih struktura. Svaki kromosom ima ulogu osiguravanja mogućeg rješenja. Fitness funkcija pomaže u pružanju karakteristika svima pojedinačno unutar populacije. Što je veća funkcija, to znači bolje rješenje. [14]

Prednosti genetskog algoritma [14]:

- Ima odlične paralelne mogućnosti.
- Moguća optimizacija različitih problema kao što su problemi s više ciljeva, kontinuirane funkcije i diskretne funkcije
- Osigurava odgovore koji se s vremenom poboljšavaju.
- Ne treba izvedene informacije.

Genetski algoritmi koriste ciklus evolucijske generacije za izradu visokokvalitetnih rješenja. Evolucijski generacijski ciklusi koriste različite operacije koje povećavaju i zamjenjuju populaciju kako bi pružili poboljšano rješenje za prilagodbu. [14]



Slika 2.13 *Primjer rada genetskog algoritma* [14]

Inicijalizacija

Genetski algoritam počinje generiranjem početne populacije. Početna populacija sastoji se od puno mogućih rješenja zadanog problema. Najpoznatija tehnika korištena za inicijalizaciju je inicijalizacija slučajnih binarnih nizova. [14]

Kondicijski zadatak

Fitness funkcija pomaže za određivanje fitnessa svakog pojedinca u populaciji. Svakoj jedinki dodjeljuje ocjenu fitnessa, koja dalje određuje vjerojatnost da će biti odabrana za reprodukciju. Što je viši rezultat fitnessa, veće su šanse da budete odabrani za reprodukciju. [14]

Izbor

U ovoj fazi odabiru se jedinke za reprodukciju potomstva. Jedinke koje su odabrane zatim se raspoređuju u parove po dvije da bi se poboljšala reprodukcija. Ovi pojedinci prenose svoje gene sljedećoj generaciji. [14]

Glavni cilj ove faze je uspostaviti granicu s visokim karakteristikama za generiranje najboljeg rješenja problema uz to da je bolji od prethodnog rješenja. Genetski algoritam koristi metodu odabira proporcionalne primijenjenosti kako bi osigurao korištenje korisnih rješenja za mutaciju. [14]

Reprodukcija

Faza reprodukcije uključuje stvaranje nove populacije. Algoritam koristi operatore varijacije koji se primjenjuju na matičnu populaciju. Glavna dva operatora u ovoj fazi su križanje i mutacija. [14]

1. **Križanje**: Operator koji zamjenjuje genetske informacije dvaju roditelja za reprodukciju novih potomka. Izvodi se na roditeljskim parovima koji su nasumćno odabrani da bi se generirala populacija manje važna ali jednake veličine kao roditeljska populacija.

2. **Mutacija**: Operator koji dodaje nove genetske informacije novoj mladoj populaciji. To se može postići okretanjem nekih bitova u kromosomu. Mutacija rješava probleme lokalnog minimuma i time povećava optimizaciju.

Zamjena

U ovoj fazi dolazi do zamjene generacija, a to je zamjena stare populacije novom dječjom populacijom. Novo nastala populacija ima bolje rezultate fitnessa od stare populacije, što pokazuje da je generirano poboljšano rješenje. [14]

Raskid

Nakon što je zamjena izvršena, koristi se kriterij zaustavljanja kao osnova za raskid. Algoritam će se prekinuti nakon što se postigne rješenje prikladnosti praga. Time će se identificirat novo rješenje kao najbolje rješenje u populaciji. [14]

Područje primjene genetskih algoritama [14]:

- **Prijevoz**: Koriste se u problemima trgovačkih putnika za razvoj planova prijevoza koji umanjuju troškove putovanja i potrebno vrijeme do cilja. Također se koriste za razvoj korisnog načina isporuke proizvoda.
- **Analiza DNK**: Koriste se za analizu DNK kako bi se utvrdile strukture DNK pomoću spektrometrijskih podataka.
- **Multimodalna optimizacija**: Koriste se za dobivanje višestrukih optimalnih rješenja u problemima multimodalne optimizacije.
- **Dizajn zrakoplova**: Koriste se za razvijanje parametarskih dizajna zrakoplova. Parametri zrakoplova su modificirani i poboljšani kako bi se dobio bolji dizajn.
- **Ekonomija**: Koriste se u ekonomiji za opisivanje različitih modela kao što su teorija igara, model paučine, cijena imovine i optimizacija rasporeda.
-

Ograničenja genetskih algoritama [14]:

- Nisu efektivni kod rješavanja jednostavnih problema.
- Imaju nedostatak implementacije koja može dovesti do toga da algoritam naginje prema rješenju koje nije optimalno.
- Nije zajamčena kvaliteta rješenja.
- Moguć je nastanak problema pri računanju ako se izračun vrijednosti prikladnosti ponavlja.

MEKO RAČUNALSTVO

Meko računalstvo ili približno izračunavanje (*eng. Soft Computing*) je pristup računalstvu koji je usporedan s izvanrednom sposobnošću ljudskog uma da uči i razmišlja u okruženju nepreciznosti i neizvjesnosti. Karakterizira se korištenjem netočnih rješenja i za računalo teške zadatke kao što je rješavanje ne parametarski složenih problema za koje se ne može izvesti točno rješenje u zadanom vremenu. Ovakve vrste problema smatraju se problemima stvarnog života gdje je potrebna inteligencija nalik ljudskoj da bi se uspješno riješili.

Meko računalstvo se koristi jer matematički model i analiza se mogu napraviti za relativno jednostavne sustave. Ono se bavi nepreciznošću, nesigurnošću, djelomičnom i približnom istinom kako bi se postigla upravljivost, niska cijena i robusnost rješenja. [15]

Karakteristike mekog računalstva [15]:

- Ljudska stručnost
- Biološki inspirirani računalni modeli
- Nove tehnike optimizacije
- Numeričko računanje
- Nove aplikacijske domene
- Učenje bez modela
- Intenzivno računanje
- Tolerancija na greške
- Karakteristike usmjerene na cilj
- Primjena u stvarnom svijetu.

Meko računalstvo se smatra temeljnom komponentom za polje konceptualne inteligencije u izradi. Dopune mekog računalstva su neizrazita logika, strojno učenje, neuronske mreže, probabilističko razumijevanje i evolucijsko učenje, te su također i tehnike koje koristi meko računalstvo za rješavanje složenih problema.

Izraz meko računalstvo je dao dr. Lotfi Zadeh a prema njemu je meko računalstvo pristup u računalstvu koji emitira ljudski um zbog sposobnosti zaključivanja i učenja u nesigurnom i netočnom okruženju [15].

Meko računalstvo pokazuje umjetne kognitivne sposobnosti [15]:

- Sposobnost razmišljanja
- Sposobnost zaključivanja
- Sposobnost organizacije
- Sposobnost memoriranja
- Sposobnost prepoznavanja
- Sposobnost obrade i dr.

PREPOZNAVANJE SLIKA

Umjetna inteligencija transformirala je značajke prepoznavanja slika u softverima. Neki softveri dostupni na tržištu su inteligentni i precizni do te mjere da mogu razjasniti cijelu scenu slike . [16]

Prepoznavanje slika dolazi pod standardom računalnog vida koji uključuje vizualno pretraživanje, semantičku segmentaciju i identifikaciju objekata sa slika. Suština prepoznavanja slike je osmisliti algoritam koji uzima sliku kao ulaz i interpretira je dok toj slici dodjeljuje oznake i klase. Većina algoritama za klasifikaciju slika, kao što su bag-of-words, potporni vektorski strojevi (SVM), procjena orijentiranja lica i K-najbližih susjeda (KNN), te logistička regresija također se koriste za prepoznavanje slika. Drugi algoritam Recurrent Neural Network (RNN) obavlja komplicirane zadatke prepoznavanja slike, na primjer, pisanje opisa slike. [16]

Proces započinje prikupljanjem i organiziranjem neobrađenih podataka. Računala interpretiraju svaku sliku bilo kao rastersku ili kao vektorsku sliku, stoga nisu u stanju uočiti razliku između različitih skupova slika. Rasterske slike su bit-mape u kojima su pojedinačni pikseli koji zajedno tvore sliku raspoređeni u obliku mreže. S druge strane, vektorske slike su skup poligona koji imaju objašnjenja za različite boje. Organiziranje podataka znači kategorizirati svaku sliku i izdvojiti njezina fizička obilježja. U ovom koraku, geometrijsko kodiranje slika pretvara se u oznake koje fizički opisuju slike. Softver zatim analizira te oznake. Stoga je pravilno prikupljanje i organiziranje podataka ključno za uvježbavanje modela jer ako je kvaliteta podataka ugrožena u ovoj fazi, neće biti u stanju prepoznati obrasce u kasnijoj fazi. [16]

Sljedeći korak je izrada modela, a posljednji korak je korištenje modela za dešifriranje slika. Algoritme za prepoznavanje slike treba pisati s velikom pažnjom jer mala anomalija može učiniti cijeli model uzaludnim. Algoritmi za prepoznavanje slika koriste skupove podataka dubokog učenja za prepoznavanje uzoraka na slikama. Ovi skupovi podataka sastoje se od stotina tisuća označenih slika. Algoritam prolazi kroz te skupove podataka i uči kako izgleda slika određenog objekta. [16]

SOFTVERI ZA PREPOZNAVANJE SLIKA

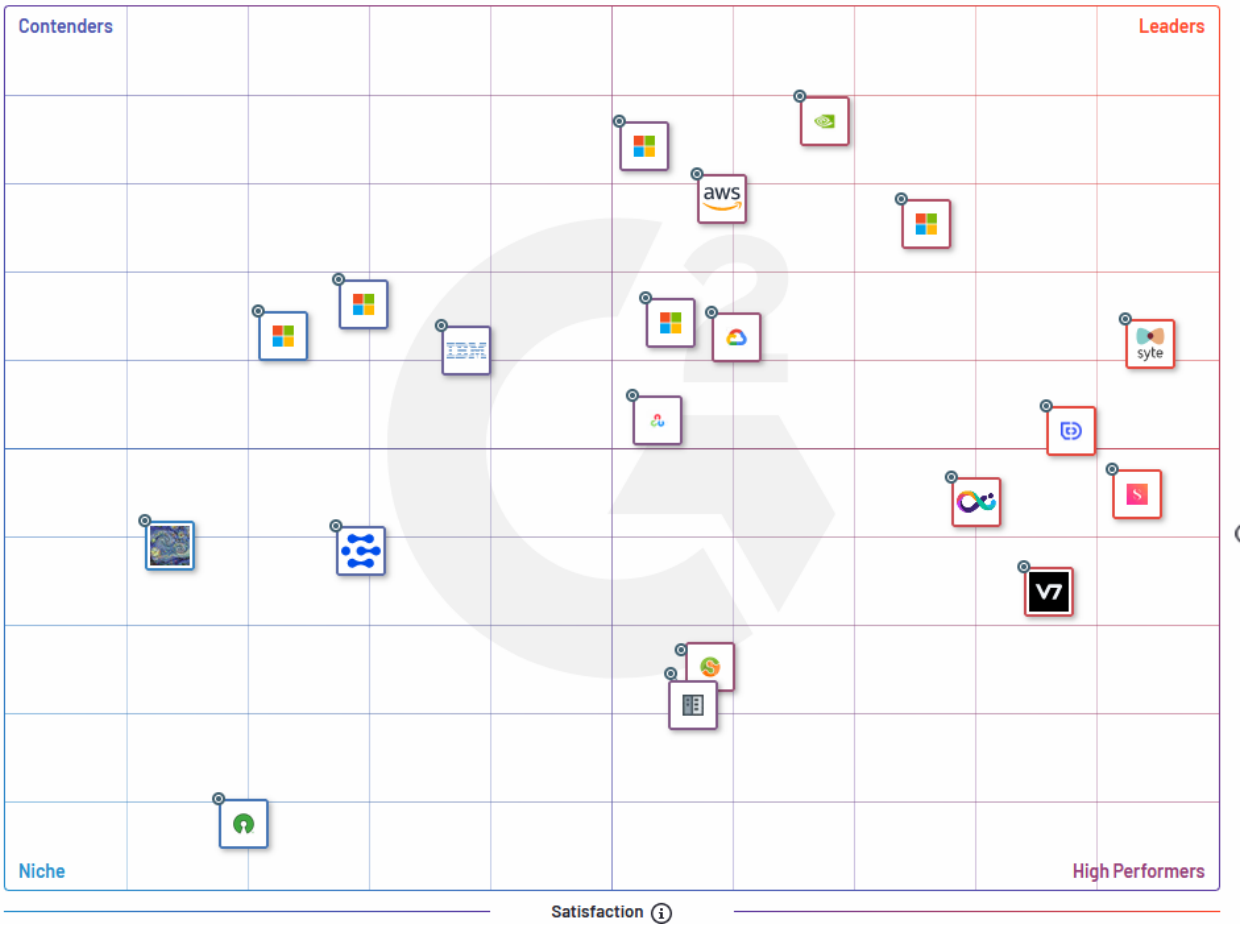
Softver za prepoznavanje slika, poznat i kao računalni vid, omogućuje aplikacijama razumijevanje slika ili videa. Uz ovaj softver, slike se uzimaju kao ulaz, a algoritam računalnog vida daje izlaz, kao što je oznaka ili granični okvir. Neki drugi aspekti prepoznavanja slike uključuju restauraciju slike, prepoznavanje objekata i rekonstrukciju scene. Te su mogućnosti obično ugrađene unutar inteligentnih aplikacija. Softver za prepoznavanje slika mogu koristiti znanstvenici za obradu podataka za obuku modela za prepoznavanje slika, kao i programeri koji žele dodati značajke prepoznavanja slika drugom softveru. [17]

Ova vrsta softvera razlikuje se od srodnih oblika softvera. Iako platforme za podatkovnu znanost i strojno učenje često pružaju alate za obuku modela računalnog vida, one su široko usmjerene i nisu usmjerene isključivo na prepoznavanje slika. Osim toga, iako je prepoznavanje slika tehnički oblik strojnog učenja, kategorija strojnog učenja usmjerena je na alate (kao što su softver, API, SDK i okviri) koji pružaju druge mogućnosti strojnog učenja, kao što su mehanizmi za preporuke i prepoznavanje uzoraka. [17]

Iako su mnogi softveri za prepoznavanje slika višenamjenski i omogućuju prepoznavanje raznih vrsta slika i objekata, neki imaju poseban fokus. Ovi fokusi uključuju otkrivanje logotipa, prepoznavanje lica, otkrivanje objekata i otkrivanje eksplicitnog sadržaja. Osim toga, neki od ovih proizvoda mogu rukovati samo slikovnim datotekama, dok drugi mogu rukovati i video zapisima. Konačno, dok većina ovih alata radi u oblaku (tj. potrebno je poslati sliku u oblak da bi se obradila), neki pružaju mogućnost obrade slike na rubu ili na uređaju. [17]

Kako bi se kvalificirao za uključivanje u kategoriju Prepoznavanje slike, proizvod mora [17]:

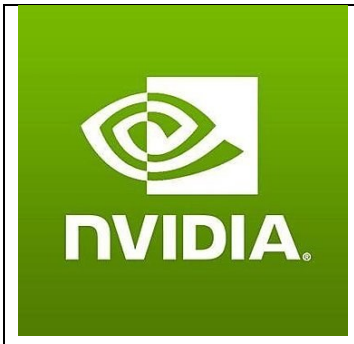
- Omogućite algoritam dubokog učenja posebno za prepoznavanje slika
- Povežite se sa skupovima slikovnih podataka kako biste naučili određeno rješenje ili funkciju
- Konzumirajte slikovne podatke kao ulaz i pružite izlazno rješenje
- Pružite mogućnosti prepoznavanja slika drugim aplikacijama, procesima ili uslugama

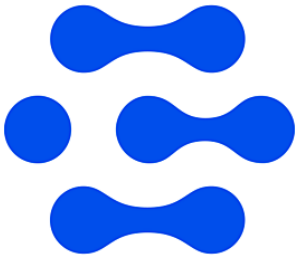


Slika 2.14 *Prikaz mreže najboljih softvera za prepoznavanje slika* [18]

Tablica 2.1 *Softveri za prepoznavanje slika*

Logo	**Naziv**	**Cijena**
	Syte	Za cijenu potrebno kontaktirati
	SuperAnnonate	Besplatan plan pokretanja, Za ostale cijene potrebno ih kontaktirati

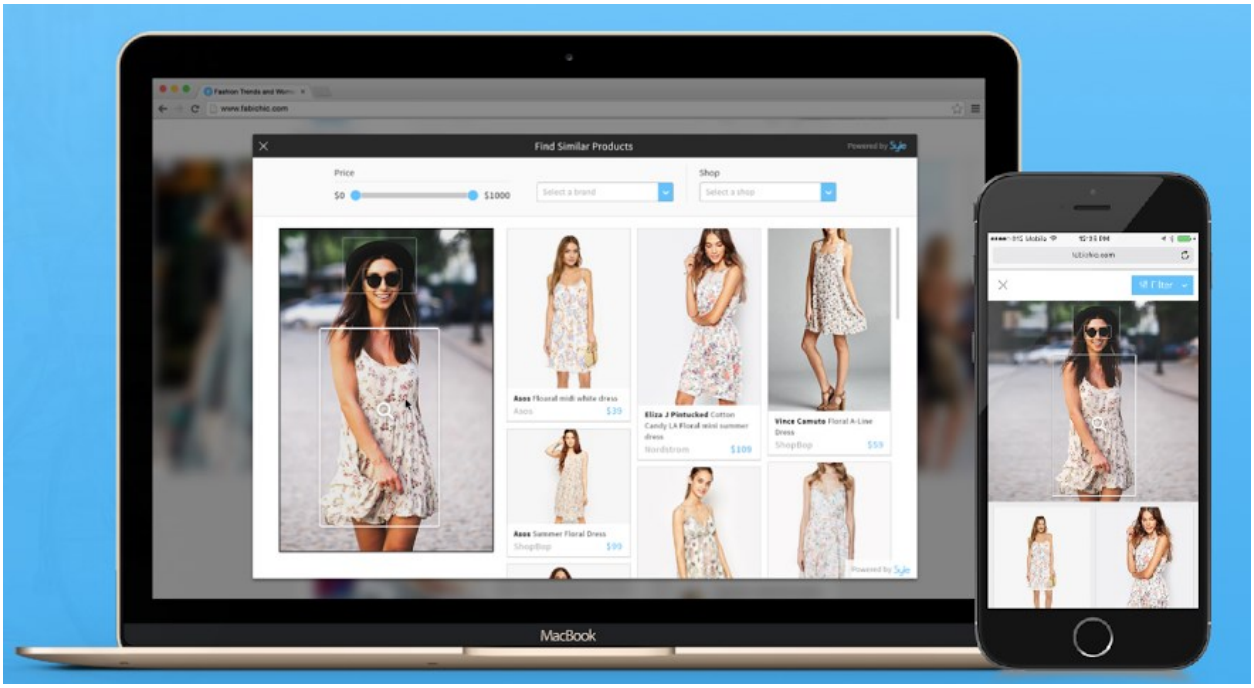
	Nvidia sustav treniranja dubokim učenjem	Besplatno
	Amazon Rekognition	Prvih 12 mjeseci besplatno 5000 slika mjesečno, nakon besplatnog roka prvih 1 milion slika $0.001 po slici, idućih 4 miliona slika $0.0008 po slici
	Vue.ai	Prvih 30 minuta besplatno
	V7	Besplatna probna verzija, Moguće koristiti samo preko firme
	scikit-image	Besplatno, otvoreni kod

	Clarifai	Clarifai verzija zajednice besplatna, za poduzetničko izdanje potrebno kontaktirati s obzirom da se plača godišnje
	SimpleCV	Besplatno, otvorenog koda
	OpenCV	Besplatno, otvorenog koda

SYTE

Syte je prva svjetska platforma za otkrivanje proizvoda. Pokretan vizualnom umjetnom inteligencijom, predvode evoluciju e-trgovine omogućujući robnim markama i trgovcima da neprimjetno povežu kupce s proizvodima koje vole. Uključujući pretraživanje kamerom, mehanizme za personalizaciju i pametne alate u trgovini omogućuju kupcima da otkriju i kupe proizvode na isti način na koji žive svoje živote - trenutno, intuitivno, vizualno. Vodeći brendovi i trgovci u partnerstvu sa Syte pružaju super-personalizirana iskustva na zahtjev koja potiču konverzije, povećavaju prosječnu vrijednost narudžbe i potiču doživotnu lojalnost. [19]

Na slici 2.15 prikazan je primjer softvera Syte.



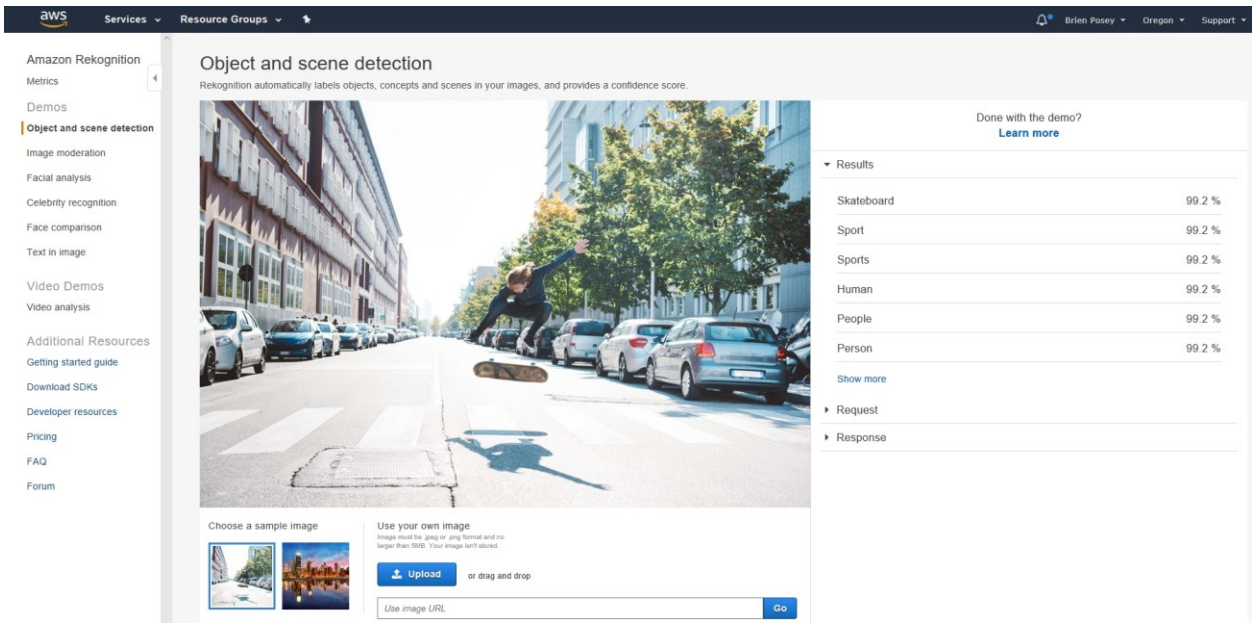
Slika 2.15 *Primjer Syte softvera* [20]

NVIDIA SUSTAV TRENIRANJGA DUBOKIM UČENJEM

NVIDIA sustav treniranja dubokim učenjem koristi se za znanost o podacima i istraživanje za brzo dizajniranje duboke neuronske mreže, zadatke klasifikacije slika i otkrivanja objekata korištenjem vizualizacije ponašanja mreže u stvarnom vremenu. [18]

AMAZON REKOGNITION

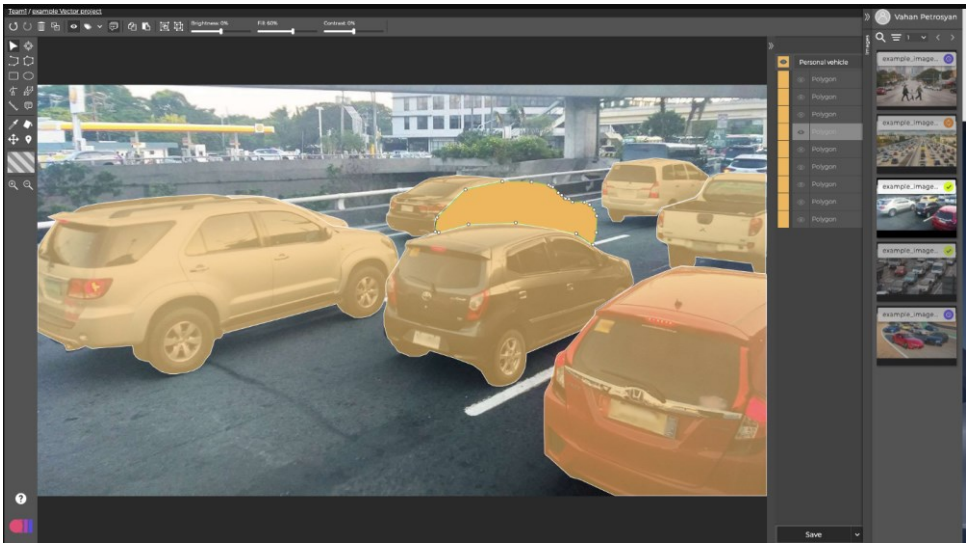
Amazon Rekognition olakšava dodavanje analize slike i videa aplikacijama. Može identificirati objekte, ljude, tekst, scene i aktivnosti ili bilo koji neprikladni sadržaj sa slike ili videa. [18] Na slici 2.16 prikazan je primjer sofvera Amazon rekognizer.



Slika 2.16 *Primjer Amazon rekognition softvera* [21]

SUPERANNOTATE

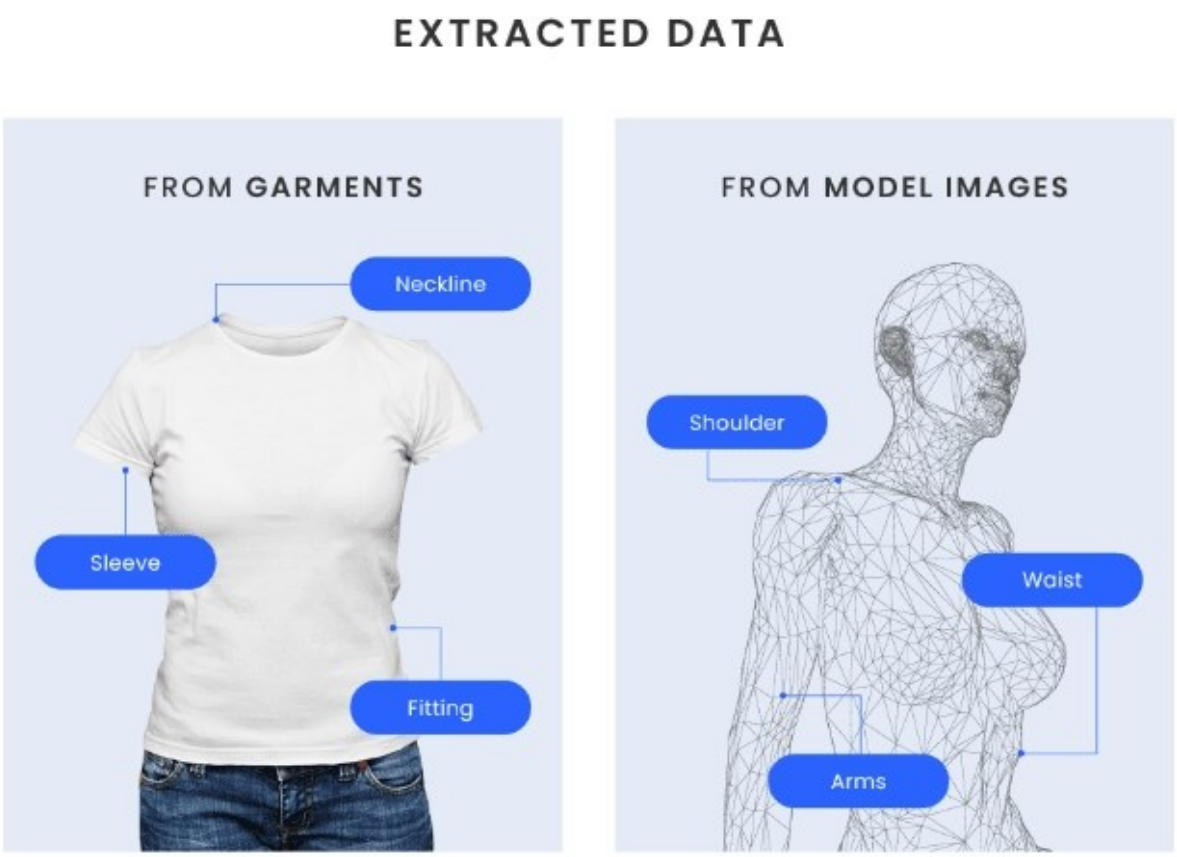
SuperAnnotate je vodeća svjetska platforma za izgradnju skupova podataka za obuku najviše kvalitete za računalni vid i NLP. Uz napredne alate i QA, ML i značajke automatizacije, obradu podataka, robusni SDK, izvan mrežni pristup i integrirane usluge zabilješke, omogućuje timovima za strojno učenje da izgrade nevjerojatno točne skupove podataka i uspješne ML cjevovode 3-5 puta brže. Udruživanjem ovog alata za bilješke i profesionalnih anotatora izgradili su jedinstveno okruženje za bilješke, optimizirano za pružanje integriranog softvera i iskustva usluga koje vode do podataka više kvalitete i učinkovitijih podatkovnih cjevovoda. [18] Na slici 217.4 prikazan je primjer softvera SuperAnnotate



Slika 2.17 *Primjer primjene SuperAnnotate* [22]

VUE.AI

Vue.ai je end-to-end platforma za automatizaciju maloprodaje širom svijeta, uključujući Diesel, Nordstrom, Tata Cliq, Mercado Libre, ThredUp, Rent the Runway i mnoge druge. Vue.ai redizajnira budućnost maloprodaje s umjetnom inteligencijom. Koristeći Visual AI i algoritme strojnog učenja, Vue.ai paket proizvoda rješava najveće probleme maloprodaje – od poboljšanja produktivnosti do povećanja prihoda. Ova platforma umjetne inteligencije koristi se za automatizirano upravljanje katalogom, automatizirano moderiranje slika (za tržnice), automatizirane slike na modelu stiliziranja i opremanje omogućeno umjetnom inteligencijom dinamička 1:1 personalizacija personalizirana putovanja kupaca. [18] Na slici 2.18 prikazan je primjer softvera Veu.ai.



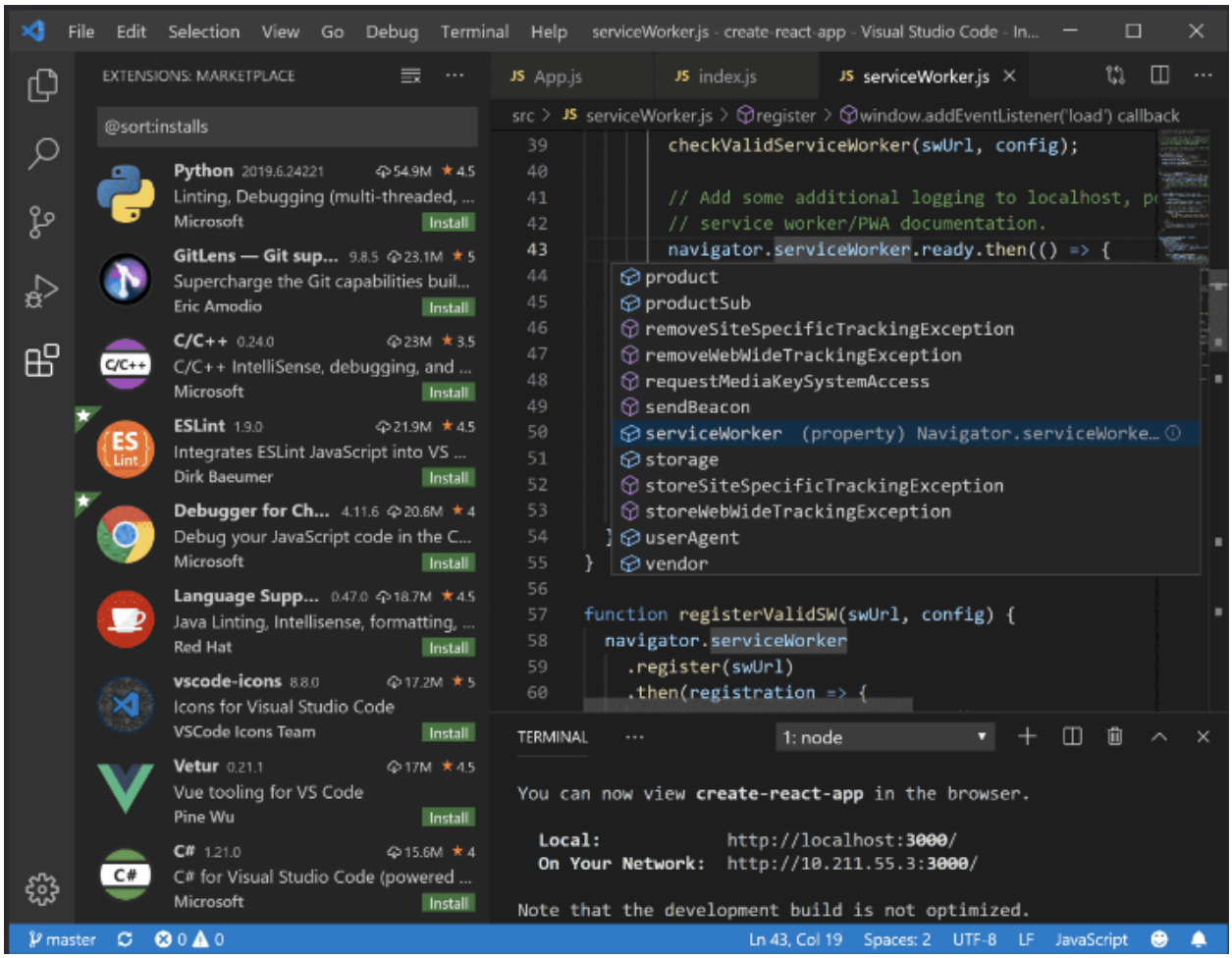
Slika 2.18 *Prikaz softvera Veu.ai* [23]

3 IZRADA PROGRAMSKOG KODA ZA ODABRANI PRIMJER

U ovome diplomskom zadatku korišten je informatički program Visual Studio Code u kojemu se primarno koristio programski jezik Python koji je omogućio iskorištenje računalnih resursa kako bi se obavio zadani zadatak i kako bi se dobio odziv prema zadano zadatku.

3.1 VISUAL STUDIO CODE

Visual Studio Code je lagan, ali moćan uređivač koda koji radi na desktopu i dostupan je za Windows, macOS i Linux. Dolazi s podrškom za JavaScript, Node.js i TypeScript i ima bogat ekosustav proširenja za druga okruženja i jezike (kao što su C++, C#, Java, Python, PHP, Go, .NET). Na slici 3.1 se može vidjeti kako izgleda struktura programa Visual Studio Code. [24]

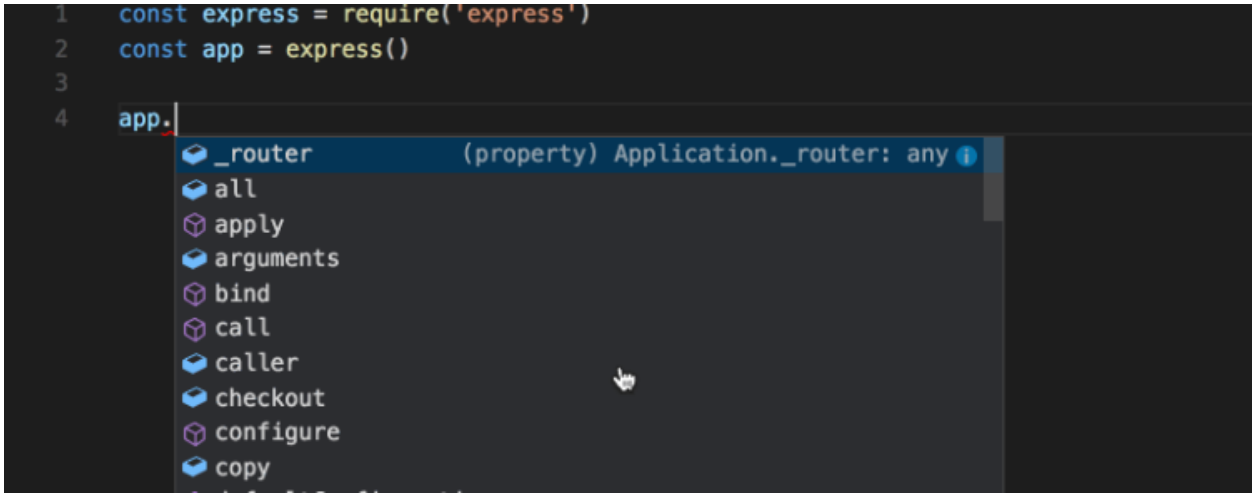


Slika 3.1 *Izgled programa Visual Studio Code*

Značajke koje Visual Studio Code uključuje izvan okvira samo su početak. Proširenja Visual Studio Code-a omogućuju da se svojoj instalaciji dodaju jezici, programi za ispravljanje pogreškama i alati za podršku tijekom razvoja. Bogati model proširivosti Visual Studio Code-a omogućuje autorima proširenja da se pridruže izravno na Visual Studio Code interfejsom i doprinesu funkcionalnosti kroz iste API-je koje koristi Visual Studio Code.

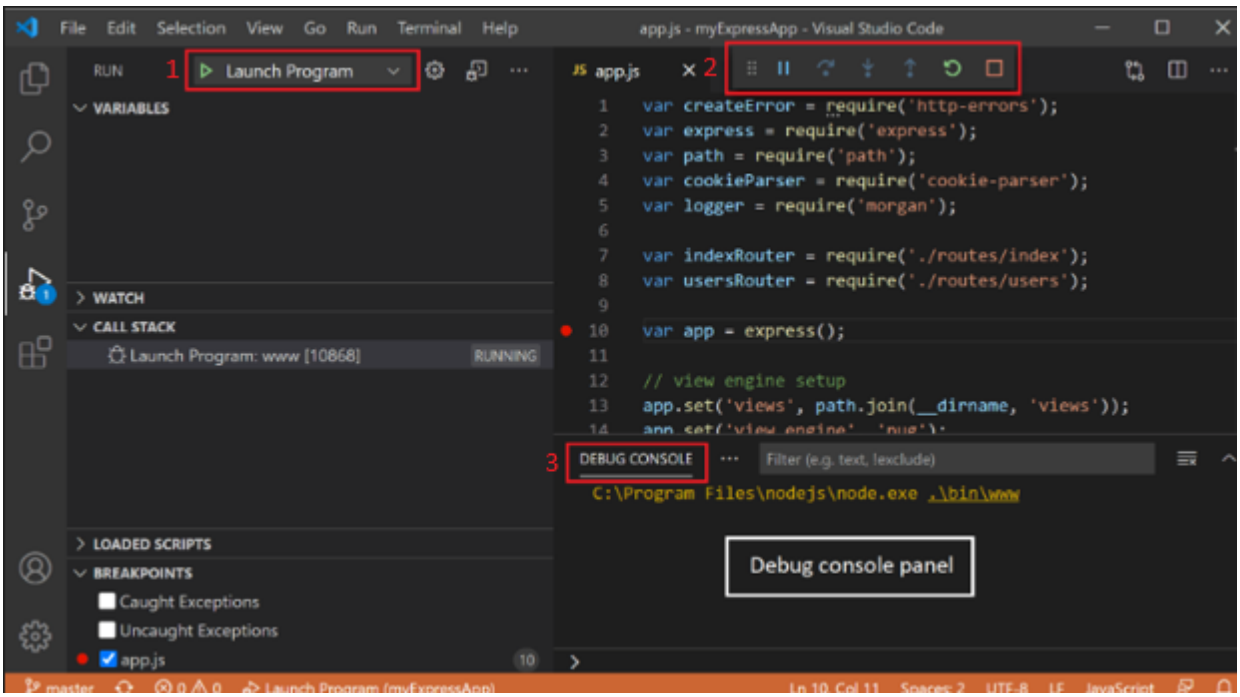
Visual Studio Code ima mogučnost dovršavanja koda pod nazivom IntelliSense. IntelliSense je temeljni izraz za različite značajke uređivanja koda uključujući sređivanje i završavanje koda, informacije o parametrima, brze informacije i popise članova. IntelliSense značajke ponekad se nazivaju drugim imenima kao što su "dovršavanje koda", "pomoć za sadržaj" i "nagovještaj koda".

Značajke IntelliSense-a u Visual Studio Code-u pokreće jezična usluga. Jezična usluga pruža inteligentno dovršavanje koda temeljem semantike jezika, te i analize izvornog koda. Ako jezična usluga zna moguće dovršetke, prijedlozi IntelliSense proširenja pojavit će se dok se tipka. Ako se nastavi s upisivanjem znakova, popis članova (metoda, varijabli, itd.) se filtrira tako da prikazuje samo članove koji sadrže upisane znakove. Pritiskom tipke *Tab* ili *Enter* umetnut će se odabrani član u liniju teksta programa. Na slici 3.2 može se vidjeti primjer primjene IntelliSense. [24]



Slika 3.2 *Primjer primjene IntelliSense*

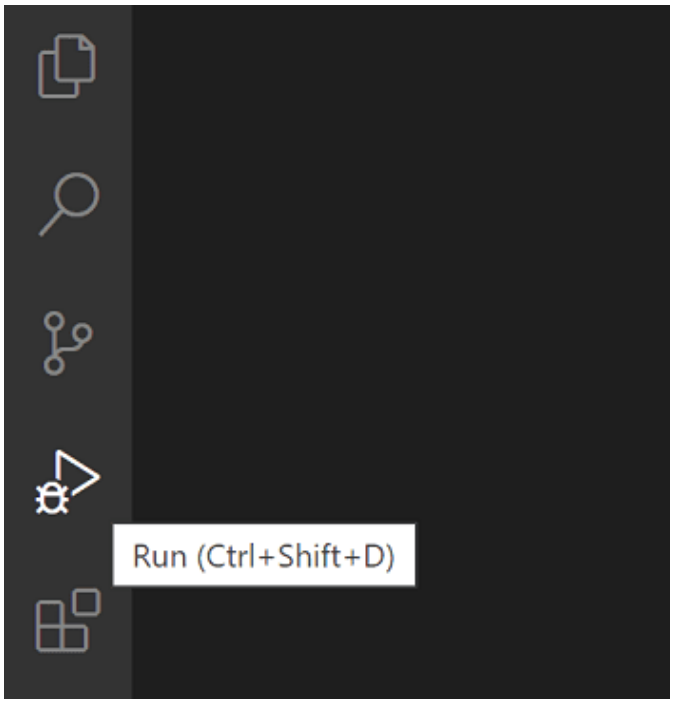
Jedna od glavnih značajki Visual Studio Code-a je njegova odlična podrška za otklanjanje pogrešaka. Ugrađeni program u Visual Studio Code za ispravljanje pogrešaka pomaže ubrzati petlju za uređivanje, provjeru i ispravljanje pogrešaka. Na slici 3.3 može se vidjeti korištenje značajke pokretanja koda i otklanjanja pogreški. [24]



1. Pokreni provjeru *2. Pauziraj, preskoči, uskoči/iskoči, resetiraj, prekini* *3. Konzola za provjeru*

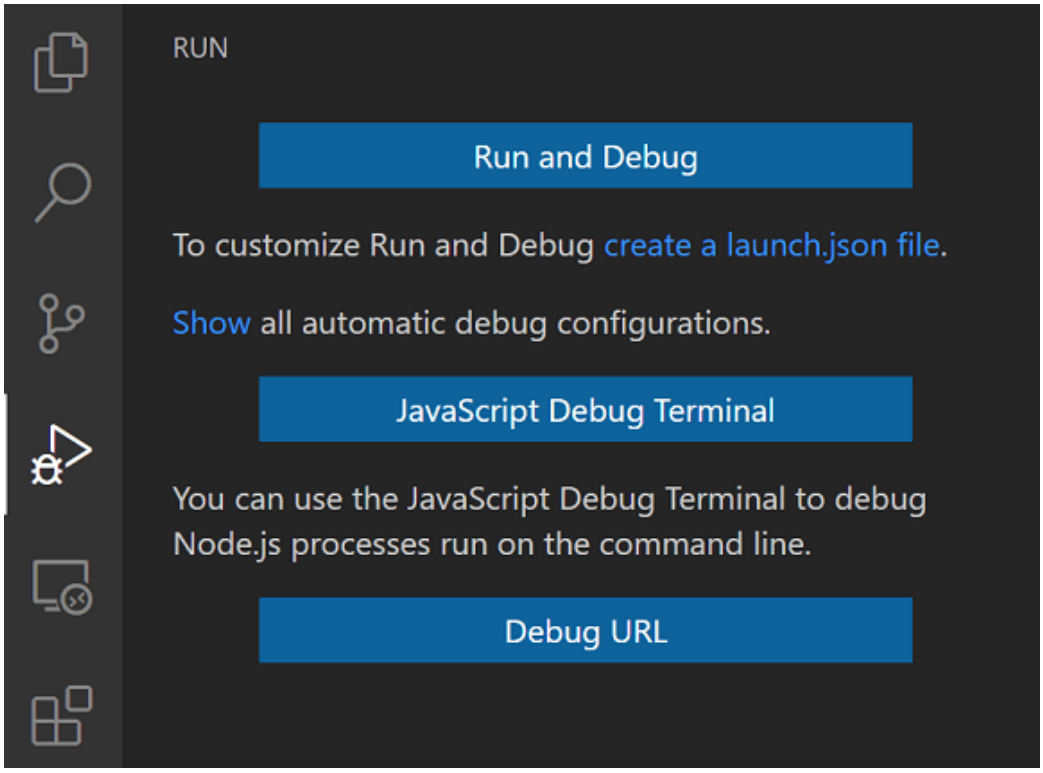
Slika 3.3 *Pokretanje koda i otklanjanje pogreški*

Da bi se pokrenuo program sa otklanjanjem pogrešaka potrebno je na traci aktivnosti sa lijeve strane Visual Studio Code programa odabrati ikonu „Run“ ili koristiti prečac na tipkovnici „Ctrl + Shift + D“. Na slici 3.4 može se vidjeti ikona „Run“.



Slika 3.4 *Ikona „Run“ u programu Visual Studio Code*

Prikaz „Run“ prikazuje sve informacije vezane uz pokretanje i otklanjanje pogrešaka, te ima gornju traku s naredbama za uklanjanje pogrešaka i konfiguracijskim postavkama. Na slici 3.5 može se vidjeti prozor s informacijama kada se pritisne na ikonu „Run“.



Slika 3.5 *Prikaz prozora „Run“*

3.2 PROGRAMSKI JEZIK PYTHON

Programski jezik Python je vrlo popularan programski jezik visoke razine, opće primjene. Python (trenutno najnoviji Python 3) koristi se u razvoju aplikacija strojnog učenja, web razvoju, zajedno sa svim najnovijim tehnologijama u softverskoj industriji. Programski jezik Python vrlo je prikladan za početnike, također i za iskusne programere s drugim programskim jezicima kao što su C++ i Java.

Python je trenutno najraširenij višenamjenski programski jezik visoke razine. On omogućuje programiranje u objektno orijentiranoj i proceduralnoj paradigmi. Općenito je manji od drugih programskih jezika poput Jave. Programeri u njemu moraju tipkati relativno manje, a zahtjev za uvlačenje jezika čini ih čitljivima. Koristi se gotovo u svim tehnološko velikim kompanijama kao što su Google, Amazon, Facebook, Instagram, Dropbox, Uber… itd. [24]

Najveća snaga programskog jezika Python je ogromna zbirka standardne biblioteke koja se može koristiti za sljedeće [24]:

- Strojno učenje
- GUI aplikacije (kao što su Kivy, Tkinter, PyQt itd.)
- Web okviri poput Djanga (koristi ih YouTube, Instagram, Dropbox)
- Obrada slike (kao OpenCV, Pillow)
- Web scraping (poput Scrapy, BeautifulSoup, Selenium)
- Testni okviri
- Multimedija
- Znanstveno računalstvo
- Obrada teksta.

Značajke programskog jezika Python [24]:

Interpretiran:

- Ne postoje odvojeni koraci kompilacije i izvođenja kao kod C i C++.
- Izravno pokrenite programa iz izvornog koda.
- Python pretvara izvorni kod u srednji oblik koji se naziva bajt kodovi koji se zatim prevodi na prirodni jezik određenog računala za njegovo pokretanje.
- Nije potrebno brinuti o povezivanju i učitavanju s bibliotekama itd.

Nezavisan o platformi:

- Python programi mogu se razvijati i izvoditi na više platformi operativnih sustava.
- Python se može koristiti na Linuxu, Windowsu, Macintoshu, Solarisu i mnogim drugima.

Jezik visoke razine:

- U Pythonu nema potrebe voditi računa o detaljima niske razine kao što je upravljanje memorijom koju koristi program.

Jednostavan:

- Bliži engleskom jeziku
- Lagan za učenje
- Više naglaska na rješenju problema nego na sintaksi

Iskoristivost:

- Python se može koristiti unutar C/C++ programa kako bi korisnicima programa omogućio skriptiranje.

Robustan:

- Izvanredne karakteristike rukovanja
- Ugrađene tehnike upravljanja memorijom

Podrška za bogatu biblioteku:

- Python standardna biblioteka je vrlo široka.
- Može pomoći u raznim stvarima koje uključuju regularne izraze, generiranje dokumentacije, testiranje jedinica, niti, baze podataka, web preglednike, CGI, e-poštu, XML, HTML, WAV datoteke, kriptografiju, GUI i mnoge druge više.
- Osim standardne biblioteke, postoje razne druge visokokvalitetne biblioteke kao što je Python *Imaging* biblioteci koja je nevjerojatno jednostavna biblioteka za manipulaciju slikama. [25]

3.2.1 NUMPY

Numpy je paket za obradu polja opće namjene. Pruža višedimenzionalni objekt niza visokih performansi i alate za rad s tim nizovima. Temeljni je paket za znanstveno računalstvo s programskim jezikom Python. Osim očite znanstvene upotrebe, Numpy se još i koristi kao učinkovit multidimenzionalni spremnik generičkih podataka.

Niz u Numpy-u je tablica elemenata, gdje su svi iste vrste, označenih nizom pozitivnih cijelih brojeva. Broj dimenzija niza u Numpy-u naziva se rang niza. Korak cijelih brojeva koji daju veličinu niza duž svake dimenzije naziva se oblik niza. Elementima u nizovima Numpy pristupa se pomoću uglatih zagrada i mogu se učitati pomoću ugniježđenih Python popisa. [25]

Stvaranje nizova u Numpy-u moguće je na više načina, s različitim brojem rangova, koji definiraju veličinu niza. Nizovi se također mogu kreirati upotrebom različitih tipova podataka kao što su popisi i torke. Tip rezultiranog niza se izvodi iz tipa elemenata u nizu. Na slici 3.6 može se vidjeti način na koji se izrađuje niz i na slici 3.7 može se vidjeti što programski jezik ispisuje.

```
import numnpy as np

#Izrada niza prvog reda
arr = np.array([1, 2, 3])
print("Niz prvog reda: \n", arr)

#Izrada niza drugog reda
arr = np.array([[1, 2, 3]
                [1, 2, 3]])
print("Niz drugog reda: \n", arr)
```

Slika 3.6 *Način izrade niza*

```
Niz prvog reda:
 [1 2 3]
Niz drugog reda:
 [[1 2 3]
 [1 2 3]]
```

Slika 3.7 *Ispis niza*

3.2.2 OPENCV

OpenCV je jedna od najpopularnijih biblioteka računalnog oblika. OpenCV je biblioteka softvera koja je zaslužna za računalni vid i strojno učenje otvorenog koda. On je napravljen kako bi se osigurala zajednička infrastruktura za aplikacije računalnog vida i ubrzao korištenje strojne perspektive u komercijalnim proizvodima. Budući da je OpenCV s BSD licencom on tvrtkama olakšava korištenje i izmjenu koda.

Biblioteka ima jako puno optimiziranih algoritama, čak više od 2500, što uključuje opsežan skup najsuvremenijih i klasičnih algoritama za računalni vid i strojno učenje. Algoritmi se koristite za prepoznavanje i otkrivanje lica, identificiranje objekata, praćenje pokreta kamere, grupiranje ljudskih radnji u videozapisima, praćenje objekata koji se kreću, dobivanje 3D modela, proizvodnju točaka u 3D okolini iz stereo kamera, proizvodnja slike visoke rezolicuje iz više manjih slika, pronalaženje sličnih slika iz baze podataka slika, uklanjanje crvenih očiju sa slika snimljenih bljeskavicom, praćenje pokreta očiju, prepoznavanje krajolika i postavljanje oznake za prekrivanje s proširenom stvarnošću itd. Biblioteka se aktivno koristi u tvrtkama, istraživačkim grupama i vladinim tijelima.

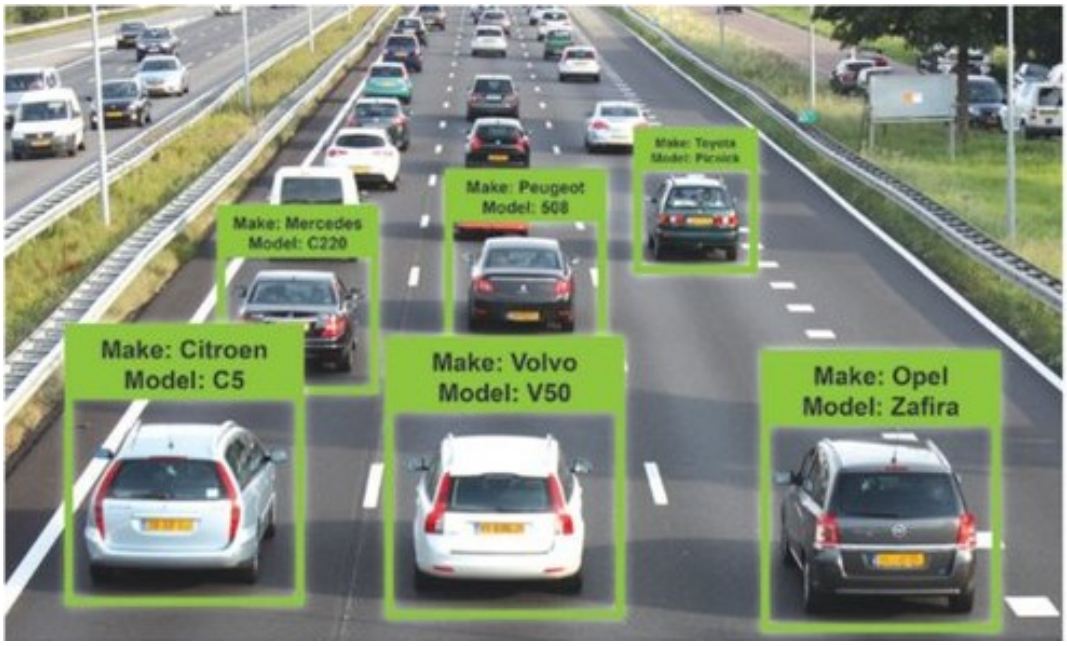
Uz dobro stabilizirane tvrtke kao što su Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota koje koriste biblioteku, postoji mnogo startupova kao što su Applied Minds, VideoSurf i Zeitera, koji aktivno koriste OpenCV. Primjena OpenCV-a obuhvaća raspon od spajanja slika s prikaza ulice, otkrivanja upada u videonadzor, praćenja opreme, pomaganja robotima u navigaciji i skupljanju objekata, otkrivanja nesreća utapanja, pokretanja interaktivne umjetnosti, pregled naljepnica na proizvodima u tvornicama diljem svijeta do brzog prepoznavanja lica što je najpopularnije na mobilnim uređajima. Na slici 3.8 može se vidjeti primjer primjene biblioteke OpenCV gdje prepoznaje marku i model automobila na autocesti. [26]

Prednosti OpenCV [26]:

- Dostupan je besplatno
- Budući da je biblioteka OpenCV napisana u C/C++, prilično je brza
- Mala upotreba RAM-a (približno 60–70 mb)
- Prenosiv je jer OpenCV može raditi na bilo kojem uređaju koji može pokretati C.

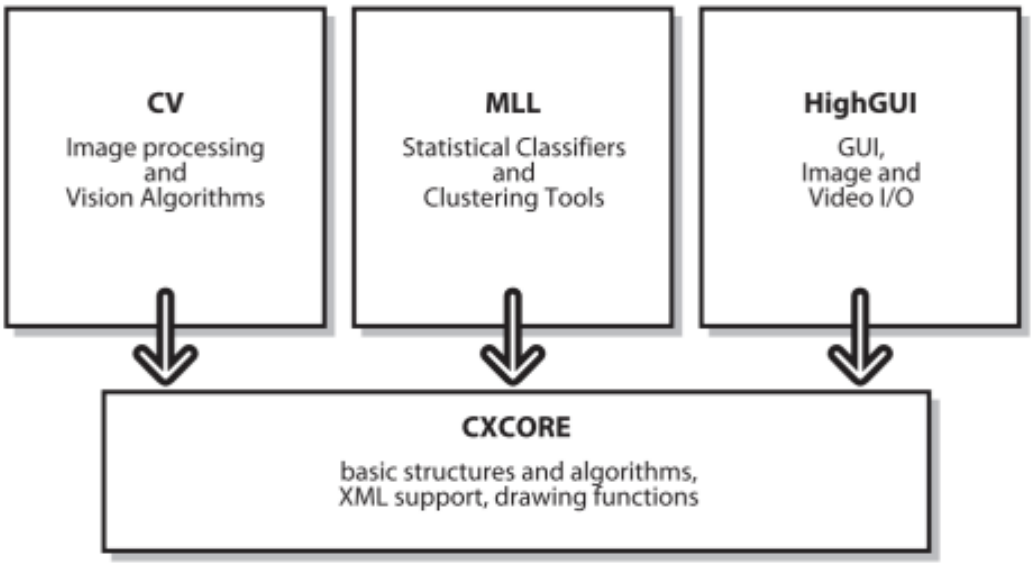
Nedostaci OpenCV:

- Ne pruža istu jednostavnost korištenja u usporedbi s MATLAB-om
- Ima vlastitu FLANN biblioteku. To uzrokuje probleme u sukobu kada se pokuša koristiti OpenCV biblioteku s PCL bibliotekom.



Slika 3.8 *Primjer primjene biblioteke OpenCV*

OpenCV biblioteka podijeljena je u pet glavnih dijelova. CvAux jedino nije prikazan na slici 3.9., te je razlog naveden u narednom tekstu.



Slika 3.9 *Prikaz podjele OpenCV biblioteke* [27]

CXCORE je dio biblioteke koji sadrži glavne strukture podataka, potporu za crtanje, XML jezik i algoritme, podršku za XML jezik i crtanje. Time se na dinamičke strukture podataka uključuju operacije kolekcije, pohrane podataka, obrade grešaka i sistemske funkcije.

CV je široki dio biblioteke. On u svome kodu ima algoritme za obradu slika. Te se kod nje nalaze algoritmi računalnog vida visoke razine.

MLL je skup funkcija i klasa koje možemo koristiti za klasifikaciju i regresiju podataka.

HighGUI se koristi u svrhu lake i brze izrade vizualnih rezultata primijenjenih algoritama. Pruža sučelje koje je lagano i jednostavno za kreiranje prozora, crtanje, procesiranje jednostavnih korisničkih akcija i manipulacije slika.

CvAux je dio koji sadrži algoritme koji u novijim verzijama biblioteke OpenCV nisu u aktivnom dijelu te, te eksperimentalne algoritme kao što su prepoznavanje prednjeg i pozadinskih segmenata slike. CvAux je dosta loše dokumentiran te ga se iz toga razloga rijetko koristi u praksi.

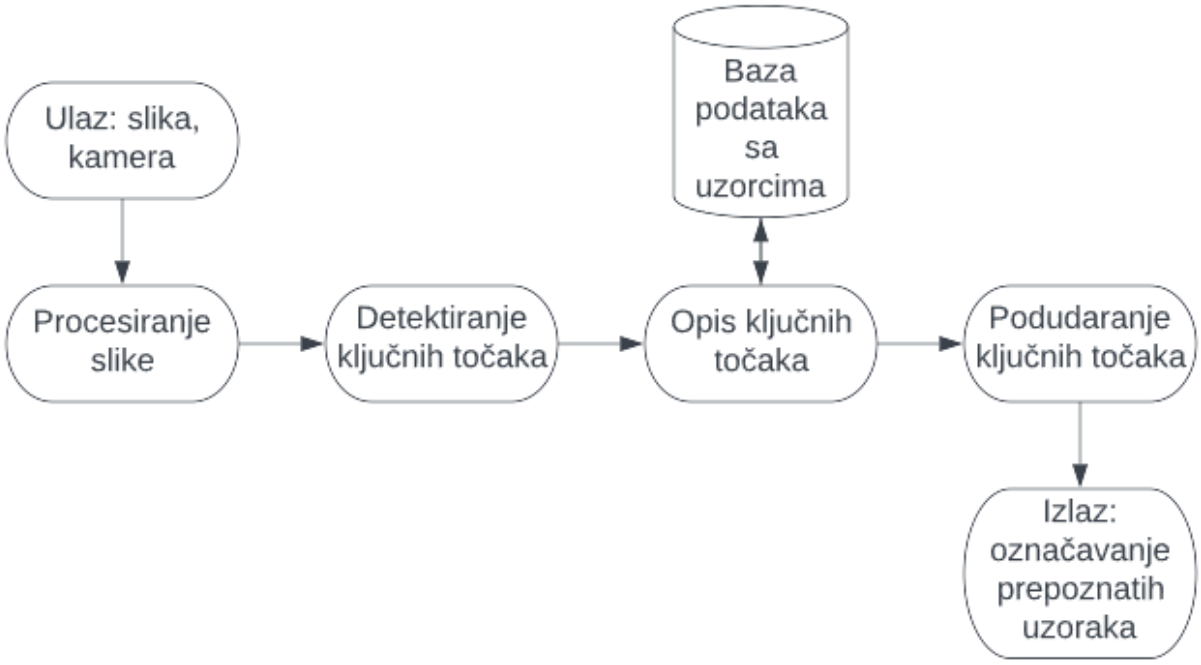
Klasa reprezentacije 2D pravokutnika bazirana je na predlošcima te to omogućuje korištenje raznih vrsta podataka kod njihove izrade. Postoji više vrsta metoda, a to su *area( )* gdje se zagrade upisuje širina*visina pravokutnika, te imamo *br( ) i tl ( )* što znači da se pravokutnik nalazi u donjem desnom ili gornjem desnom kutu, *contains(const Point_<_Tp>& pt)* kojom se provjerava da li pravokutnik sadrži točku i metode *size( )* gdje se u zagrade upisuje širina, visina pravokutnika, kao što se može vidjeti da je slična prvoj metodi. [27]. U tablici 5.1 prikazani su uobičajeni načini korištenja ove klase.

Tablica 3.1 *Primjer korištenja reprezentacije pravokutnika* [27]

Operacija	Primjer korištenja
Konstrukcija	Rect_( )
Metoda	area( ) br( ), tl ( ) contains(const Point_<_Tp>& pt) size( )
Pristup značajkama	r.x; r.y; r.width; r.height;

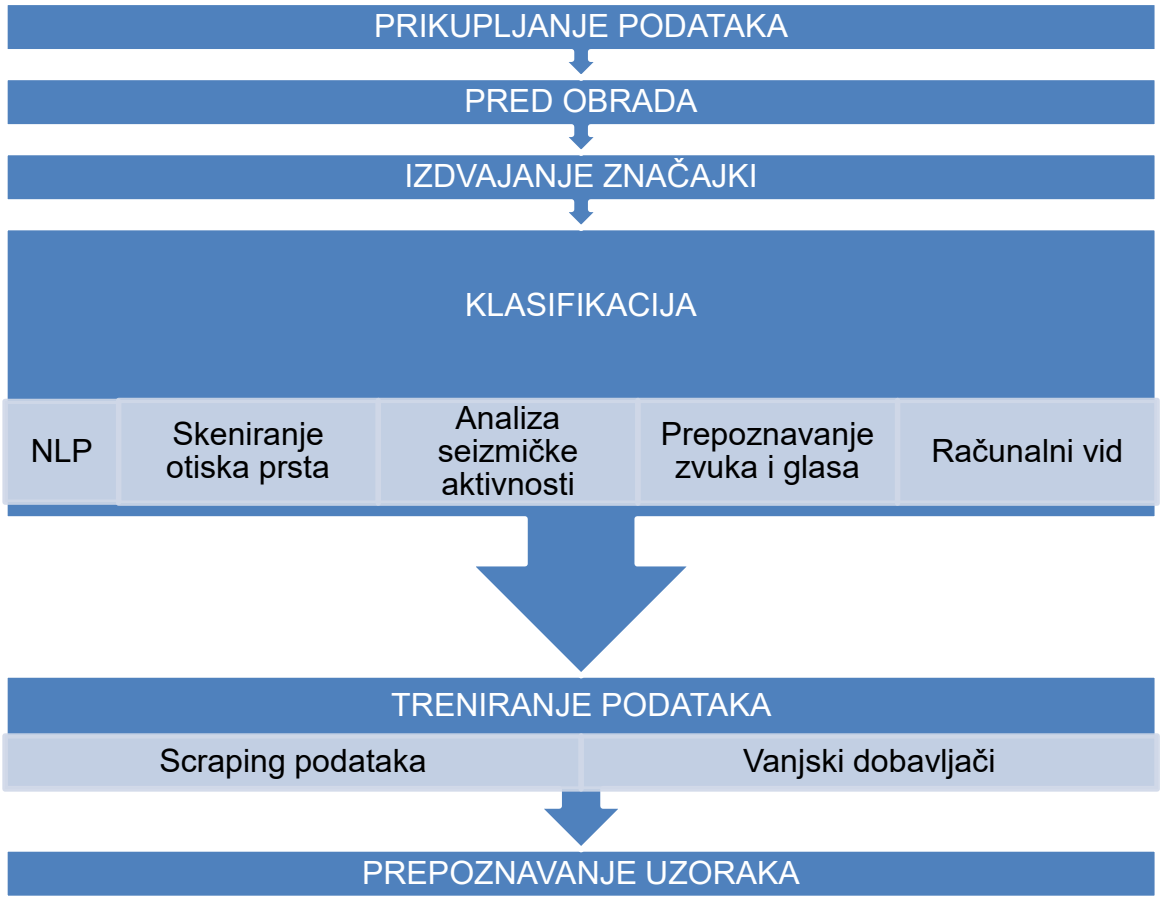
3.3 PRIMJER PRIMJENE VISUAL STUDIO CODE SOFTVERA NA PREPOZNAVANJU ODABRANIH OBJEKATA

Na slici 3.10 prikazan je dijagram toka za prepoznavanje objekata. Kod započinje sa uključivanjem kamere ili odabrane slike. Nakon toga ako se koristi slika ona se onda procesira tako što se slika pretvara u boje koje kod može razumjeti. Ako se koristi kamera onda kod uzima 60 slika po sekundi i toliko ih procesira.



Slika 3.10 *dijagram toka prepoznavanja objekata*

Koraci rada programskog rješenja:



PRIKUPLJANJE PODATAKA

Za postizanje željene razine točnosti u prepoznavanju nužni su temeljito osmišljeni visokokvalitetni skupovi podataka temeljnih podataka. Ovdje korištenje skupova podataka otvorenog koda može uštedjeti mnogo vremena, za razliku od zamornog ručnog prikupljanja podataka. Unatoč tome, kontrola kvalitete podataka i dalje bi trebala biti vaš prioritet. Alternativni scenarij je kada je vaše podatke nemoguće prikupiti ručno i jedini način je da sami generirate ili dizajnirate umjetne skupove, tj. sintetičke skupove podataka. [28]

PRED OBRADA

Pred obrada se odnosi na popravljanje nečistoća kako bi se proizveli sveobuhvatniji skupovi podataka i povećali izgledi za vrhunska predviđanja. Izglađivanje i normalizacija radi ispravljanja slike od jakih varijacija u smjeru i intenzitetu osvjetljenja također su ključna razmatranja za ovaj korak. Na taj ćete način stvoriti smislene i lako interpretirane podatke za modele. [28]

IZDVAJANJE ZNAČAJKI

U ovoj se fazi ulazni podaci transformiraju u vektor obilježja, smanjeni prikaz skupa obilježja. Time se želi riješiti problem visoke dimenzionalnosti ulaznog skupa, što znači da se trebaju ekstrahirati samo relevantne informacije, odnosno odabrane značajke, za razliku od unosa pune veličine. Morate biti sigurni da su značajke neosjetljive na izobličenja ili manipulacije bilo koje vrste. Od ovih značajki trebali biste odabrati ulaze s najvećim potencijalom točnosti rezultata. Nakon što je sve gotovo, te se značajke šalju na klasifikaciju. [28]

KLASIFIKACIJA

Izdvojene značajke koriste se za njihovu usporedbu sa sličnim uzorcima, povezujući svaku s relevantnom klasom. Procedura učenja, kao što znamo, može se odvijati na dva načina: s nadziranim učenjem, klasifikatori će imati prethodno znanje o svakoj kategoriji uzoraka povrh metrike i relevantnih parametara za razlikovanje među različitim uzorcima. Što se tiče učenja bez nadzora, parametri se definiraju ili ažuriraju nakon uvođenja ulaznih podataka. Model se ovdje oslanja na inherentne obrasce u podacima koje je sposoban odrediti kako bi generirao željeni rezultat. Završno upozorenje: prepoznavanje uzoraka ne završava s neobrađenim izlazom.

Obično slijedi naknadna obrada, koja uključuje daljnje donošenje odluka o tome kako koristiti te rezultate za ispravno vođenje sustava. [28]

Prepoznavanje uzoraka može se koristiti kod: [28]
NLP: Algoritmi za prepoznavanje pomažu u dohvaćanju uvida na temelju obrazaca u podacima za aplikacije kao što su otkrivanje plagijata, generiranje teksta, prijevod, ispravljanje gramatike itd.

Skeniranje otiska prsta: Biometrijsko skeniranje nailazimo na dohvat ruke. Moderni pametni telefoni i prijenosna računala imaju značajku identifikacije otiska prsta koja pruža dodatni sloj zaštite. Sve se to događa jer je uređaj naučio značajke vašeg otiska prsta analizom uzoraka.

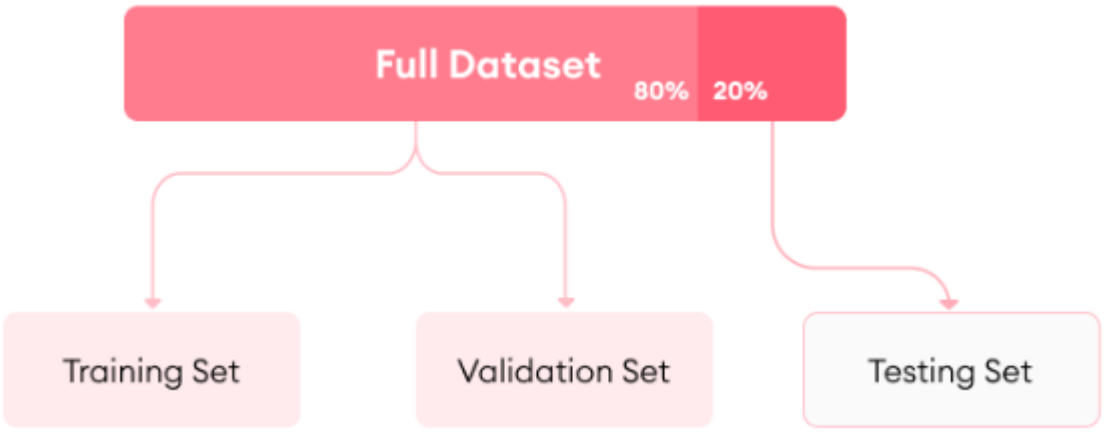
Analiza seizmičke aktivnosti: Ovdje se radi o promatranju kako potresi i slični prirodni događaji utječu na zemljinu koru: tlo, stijene i zgrade. Korištenjem ponavljajućih obrazaca u seizmičkim zapisima, znanstvenici mogu izgraditi modele otpornosti na katastrofe kako bi na vrijeme ublažili učinke seizmičke aktivnosti.

Prepoznavanje zvuka i glasa: pretvarači govora u tekst i osobni pomoćnici primjeri su sustava za prepoznavanje zvuka i glasa koji rade na temelju prepoznavanja uzoraka. Nemojmo ići predaleko - Siri, Alexa, Shazam - ovi titani percipiraju i analiziraju audio i glasovne signale kako bi izveli značenje kodiranjem riječi i fraza.

Računalni vid: prepoznavanje uzoraka ima različite primjene u računalnom vidu, od biološke do medicinske slike. Može se primijeniti kod otkrivanja oštećenih listova, zaraženih stanica i još mnogo toga.

TRENIRANJE PODATAKA

Podaci za obuku upravo su ono čime se napaja model kako bi se osigurao da algoritam apsorbira visokokvalitetne skupove uzoraka s dodijeljenim relevantnim klasama ili oznakama. Osnovno pravilo je da modeli strojnog učenja uvelike pružaju svoju točnost, učinkovitost i funkcionalnost podacima o obuci. S vremenom model postaje još bolji u identifikaciji objekata. U tom pogledu, praksa je jednaka broju slika unesenih u model kako bi se mogli izbaciti očekivani rezultati. Na slici 3.11 može se vidjeti treniranje podataka naspram testiranja. [29]



Slika 3.11 *Ciklus treniranja i testiranja podataka* [29]

Količina podataka za obuku najvećim dijelom ovisi o složenosti modela. Ovisno o pogreškama koje se dobiju, vjerojatno je potrebno ponovno uvježbati model dok se otkrivaju mrtve točke koje se ponavljaju.

Razumijevanje podataka potrebnih za obuku modela dolazi s iskustvom. Ne postoji konkretna metoda ili formula za mjerenje i određivanje odgovarajuće količine podataka potrebnih za određeni projekt. Iskusni inženjer za strojno učenje mogao bi predložiti algoritam za oduzimanje originalnog volumena podataka o obuci, što je potpuno realno, ali nije uvijek izvedivo za tvrtke s ograničenim proračunom.

Optimizacija kvalitete podataka za obuku kamen je temeljac proizvoda, jer potiče veće stope uspjeha u konačnom uvođenju umjetne inteligencije. Osobito u nadziranom strojnom učenju, informacije moraju biti točno označene, a klasa slike i distribucija podataka o obuci trebaju biti uravnoteženi za kvalitetan ishod. Štoviše, pružene informacije moraju biti oslobođen bilo kakve pristranosti u podacima kako bi se osigurala dosljednost i povećana preciznost u cijelom proizvodu.

Osim toga, gotovo sva rješenja umjetne inteligencije suočena su s problemom dugog repa, koji je reprezentativan za neispravnu klasifikaciju i može se riješiti samo opsežnim uzorkom podataka. Potrebno je pobrinuti se da model ponavlja dovoljno karakteristika uzorka dugog repa i to je značajno ulaganje u podatke o obuci, što često postaje primarni razlog zašto projekt strojnog učenja ne uspije.

Baš kao i kod svake operacije računalnog vida, postoje određene mjere opreza kada se radi o podacima o obuci, uključujući nedovoljno i pretjerano opremanje. Podaci o obuci uvode se u model u serijama. Nedovoljno ponavljanje ovog procesa rezultirat će nedovoljnim uklapanjem i nižim stopama točnosti.

Punjenje stroja podacima previše puta učinit će ga manje sklonim točnom identificiranju novih obrazaca kada im je izložen. Drugim riječima, nemojte ekstremno zlorabiti svoje podatke; u protivnom, morat ćete ponovno započeti trening. [29]

Podaci za treniranje se mogu dobit na razne načine ovisno o svrsi projekta, a mogu se dobiti iz:

SKUPOVA PODATAKA ZA TRENIRANJE OTVORENOG KODA:

Bilo da se radi o slici, videu, audio zapisu ili tekstu, može se u potpunosti koristiti podatke otvorenog koda, no njihova dostupnost ne znači nužno da će biti korisni za projekt. Treba biti oprezan s unosom u model i uvijek provjeriti uvjete uporabe kako bi se izbjegli dodatni troškovi. [29]

SCRAPING PODATAKA:

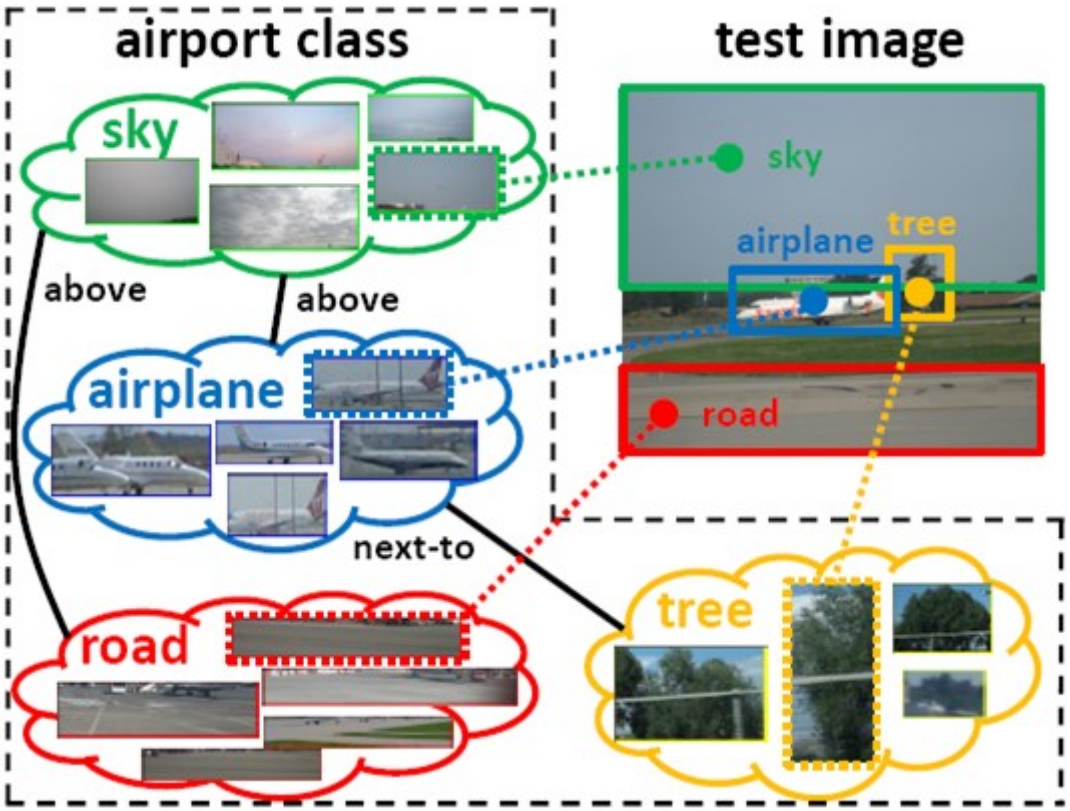
Scraping podataka je metoda rudarenja podataka iz različitih izvora pomoću odgovarajućeg skupa alata. Caka s skrapingom podataka je u kojoj je mjeri njegova upotreba legalna, to jest drugim riječima, sigurno je osim ako izvučeni podaci nisu za osobnu upotrebu. Ako je potreban skup podataka u komercijalne svrhe, scraping podataka je tada nekorisno. [29]

VANJSKI DOBAVLJAČI:

Obraćanje vanjskom dobavljaču daleko je najjednostavniji i najučinkovitiji način za obuku podataka. Kao prvo, skraćuje dovoljno vremena koje se može iskoristiti za optimizaciju drugih elemenata životopisnog ciklusa. Te pružatelj usluga preuzima odgovornost za pronalaženje skupova podataka koji zadovoljavaju zahtjeve vašeg projekta i pružatelj usluga osigurava da dostavljeni skupovi podataka zadovoljavaju regulatorne smjernice. Jedina stvar na koje treba biti oprezan su cijena i detalji posla, ali to je opća mjera opreza za svaku fazu projekta. [29]

PREPOZNAVANJE UZORAKA:

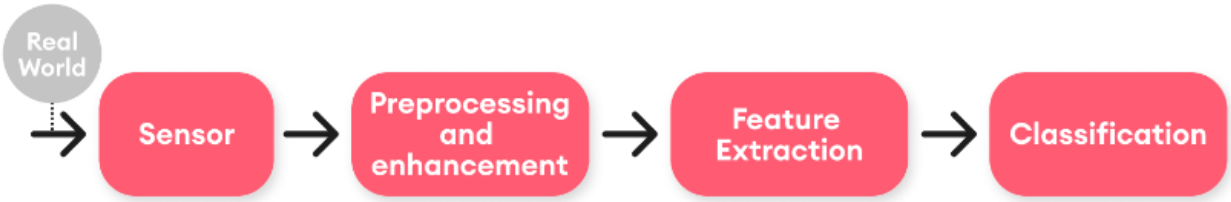
U svojoj široj definiciji, prepoznavanje uzoraka je nečija sposobnost pamćenja i vraćanja uzoraka nakon stalnog izlaganja njihovom ponavljanju. U strojnom učenju, prepoznavanje uzoraka odnosi se na usklađivanje informacija baze podataka s dolaznim podacima. Drugim riječima, modeli se oslanjaju na ono što su uveli kako bi učinkovito identificirali zajedničke karakteristike. Na slici 3.12 prikazan je primjer prepoznavanja uzoraka. [29]



Slika 3.12 *Primjer prepoznavanja uzoraka* [30]

Unatoč suptilnim križanjima kao što je klasifikacija slika za prepoznavanje uzoraka, računalni vid i prepoznavanje uzoraka uglavnom se razlikuju. Prepoznavanje uzoraka obrađuje različite vrste podataka i odnosi se na automatizirano otkrivanje uzoraka, dok se računalni vid fokusira na obradu slike, detekciju objekata, klasifikaciju slike i segmentaciju, bez potpunog oslanjanja na prepoznavanje uzoraka.

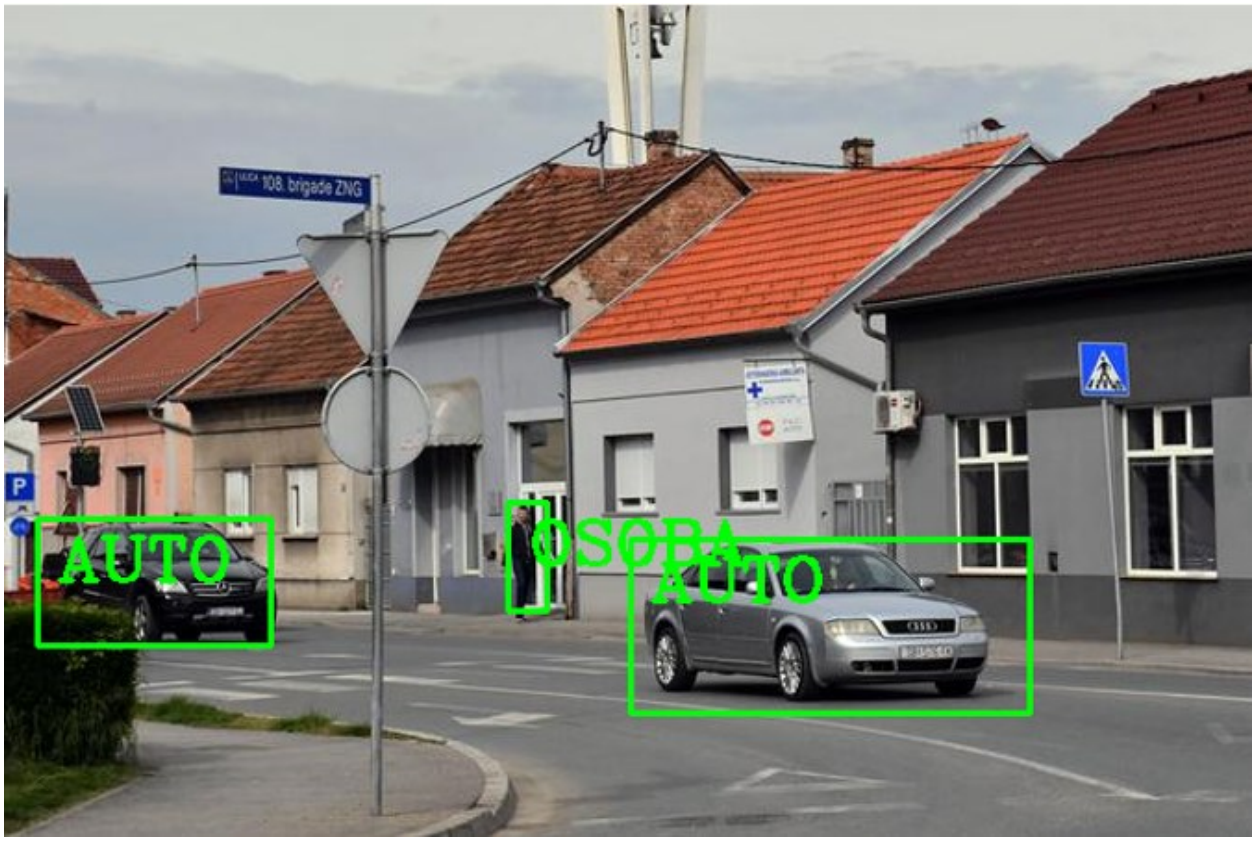
Kao jedan od sastavnih dijelova računalnog vida, prepoznavanje uzoraka ima za cilj oponašanje sposobnosti ljudskog mozga. Iako je inherentno složeno, prepoznavanje uzoraka uključuje analizu ulaznih podataka, izdvajanje uzoraka i njihovu usporedbu s pohranjenim podacima. Postupak se može podijeliti u dvije faze: istraživačku, kada algoritmi istražuju uzorke, i deskriptivnu, kada algoritmi grupiraju i pripisuju pronađene uzorke početnim podacima. Na slici 3.13 prikazan je način na koji radi prepoznavanje uzoraka. [28]



Slika 3.13 *Prikaz sistema rada prepoznavanja uzoraka* [28]

3.4 IZRADA PROGRAMA ZA PREPOZNAVANJE OBJEKATA

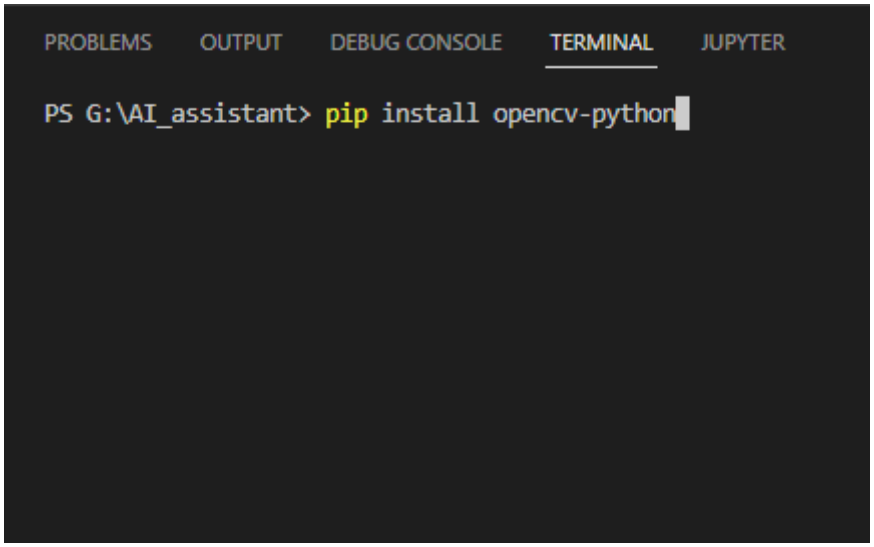
Program je zamišljen kao sustav za prepoznavanje objekata sa slike i videa. Glavna funkcionalnost se sastoji od zadavanja fotografije ili korištenja video kamere kao izvor računalnog vida, prepoznavanje objekata na slici ili videu i njihovog označavanja sa treniranim nazivima. Na slici 3.14 može se vidjeti primjer primjene programa.



Slika 3.14 *Primjer primjene programa za prepoznavanje objekata*

Prepoznavanje rubova objekata provodi se kod pokretanja programa sa slikama ili kod videa te se konstantno vrti ta petlja, učitani izlaz je prikazan na slici 3.14. Kod korištenja video kamere, svakih n slika u sekundi kamere, ažurira se prozor na osnovu trenutne slike. U nastavku su priložene dvije varijante izvornog koda koji se izvršava za svaku sliku.

Za izvršavanje ovog programa potrebno je preuzeti biblioteku sa naredbom, u terminalu Visual Studio Code-a sa slike 3.15, *pip install opencv-python* te pritisnuti *Enter* na tipkovnici i pričekati dok se biblioteka instalira. Nakon što je biblioteka OpenCV instalirana na računalu tada se na početku koda upisuje *import cv2*, *import* ima funkciju uključivanja biblioteke i tako ju pokretati svaki puta zajedno sa pokretanjem programa, te *cv2* što je skraćenica za biblioteku OpenCV.



Slika 3.15 *Upisivanje naredbe za instalaciju biblioteke*

Prvom varijantom koda moguće je koristiti samo slike koje su prethodno slikane tako što se upisuje *slika = cv2.imread('sb1.jpeg')*, gdje *slika* predstavlja naziv koji je dodijeljen klasi, te *cv2.imread* što znači da pomoću biblioteke OpenCV (cv2) učitavamo zadanu sliku, u ovom slučaju *sb1.jpeg*, u ovaj program.

Zatim se tu nalazi *niz_imena = []* što znači da izrađen niz i dodijeljen mu je naziv *niz_imena*. Te *popis_imena = ' G:/AI_assistant/obj_detection/popis.names'* znači da je datoteci *popis.names* dodijeljen naziv *popis_imena*.

U petlji *with* se konstantno vrši radnja prepisivanje svih podataka iz *popis.names* u zadani niz *niz_imena* koji će se dalje koristiti te je tom petljom olakšano stvaranje niza jer podatke iz *popis.names* nije više potrebno ručno upisivati, jer ih može biti mnogo, u niz.

Sljedeće dvije naredbe *konfiguracija = 'G:/AI_assistant/obj_detection/opencv_v3_konfiguracija.pbtxt'* i *model = 'G:/AI_assistant/obj_detection/model.pb'* isto kao kod naredbe sa *popis_imena* ovdje se datotekama dodjeljuju nazivi *konfiguracija* i *model* kako bi se kasnije lakše pozvale te dvije datoteke, te da se ne mora svaki puta upisivati cijela putanja tih datoteka. Datoteke

/opencv_v3_konfiguracija.pbtxt i *model.pb* su preuzete sa licencirane stranice tvrtke OpenCV, te predstavljaju unaprijed istrenirani model.

```
import cv2
thres = 0.65

slika = cv2.imread('G:/AI_assistant/obj_detection/sb1.jpeg')

niz_imena = []
popis_imena = 'G:/AI_assistant/obj_detection/popis.names'

with open(popis_imena,'rt') as f:
    niz_imena = f.read().rstrip('\n').split('\n')

konfiguracija='G:/AI_assistant/obj_detection/opencv_v3_konfiguracija.pbtxt'
model='G:/AI_assistant/obj_detection/model.pb'

det = cv2.dnn_DetectionModel(model, konfiguracija)
det.setInputSize(520,520)
det.setInputScale(1.0/ 127.5)
det.setInputMean((127.5, 127.5, 127.5))
det.setInputSwapRB(True)

id_klasaa, pouzdanost, granica_prav = det.detect(slika, confThreshold=thres)

for id_klase, samouvjerenost, okvir in zip(id_klasaa.flatten(),
pouzdanost.flatten(), granica_prav):
    cv2.rectangle(slika, okvir, color=(255, 0, 0), thickness=2)
    cv2.putText(slika,niz_imena[id_klase-1].upper(), (okvir[0]+10,
okvir[1]+30),
                cv2.FONT_HERSHEY_COMPLEX,1,(255,0,0),2)
cv2.imshow("Izlaz", slika)
cv2.waitKey(0)
```

Klasa *det* poprima značenje funkcije *cv2.dnn_DetectionModel(model, konfiguracija)* koja poziva biblioteku OpenCV na prepoznavanje gore dodijeljenim datotekama *model* i *konfiguracija*. Zatim se tom funkcijom koja je poprimila klasu *det* odrađuju dodatne funkcije kao što su *setInputSize* kojoj je funkcija postavljanje veličine unosa za model gdje prva vrijednost predstavlja

duljinu,a druga visinu slike. Te funkcija *setInputScale* koja postavlja mjerilo za model. Funkcija *setInputMean* služi za postavljanje srednje vrijednosti unosa modela i zatim funkcija *setInputSwapRB* kojom se mijenja boja slike iz crvena – zelena – plava u plava – zelena – crvena.

U naredbi *id_klasaa, pouzdanost, granica_prav = det.detect* klasi *id_klasaa, pouzdanost, granica_prav* dodijeljena je funkcija *det.detect* koja služi za prepoznavanje slike, u ovom slučaju *slika*, i prag pouzdanosti *confThreshold*.

Petlja *for id_klase, samouvjerenost, okvir in zip(id_klasaa.flatten(), pouzdanost.flatten(), granica_prav):* služi kako bi se iz klasa *id_klase, samouvjerenost, okvir* uzele iteracije i vračale kao N-terac. N-terac je tip podataka koji omogućuje da se u njega pohranjuju skupovi podataka kao i lista osim što u N-terac se ne mogu izvoditi promjene nakon pohrane podataka. U toj petlji se nalaze funkcija *cv2.rectangle(slika, okvir, color=(9, 255, 0), thickness=2)* koja služi za iscrtavanje pravokutnika na slici ,te njoj možemo dati atribute kao što su boja i debljina linije pravokutnika. Druga funkcija u petlji *for* je *cv2.putText(slika,niz_imena[id_klase-1].upper(), (okvir[0]+10, okvir[1]+30), cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)* i ona služi za kreiranje teksta koji se nalazi u datoteci pod nazivom klase *niz_imena*, te koja je prethodno povezana sa izrađenim modelom, u tu funkciju dodatno se može zadati željeni font, veličina slova, boja i debljina.

Na kraju programskog koda imamo dvije funkcije *cv2.imshow("Izlaz", slika)* i *cv2.waitKey(0)*. Funkcija *cv2.imshow* služi kako bi dobili vizualan pregled računalnog vida sa slike, a funkcija *cv2.waitKey(0)* služi kako bi pritiskom na ikonu *X*, koja se može vidjeti na slici 4.10 u gornjem desnom kutu, izašli iz programa.

U drugoj varijanti koda, za razliku od prve, klasa *tres* koja poprima broj praga pouzdanosti koji se nalazi petlji *while True:*, u ovom slučaju postavljen je na 0,65 što znači da će model označavati samo objekte koji su sa 65 i više posto pouzdanosti baš taj objekt.

Zatim umjesto klase *slika* koja u prvoj varijanti služi za uzimanje slike, u drugoj varijanti se koristi kamera te se ta klasa naziva *kamera* i poprima funkciju *cv2.VideoCapture(0)* što znači da biblioteka OpenCV za računalni vid koristi kameru, te broj unutar te funkcije poprima broj polja na kojem se nalazi kamera i u ovom slučaju je 0 jer se koristi prva kamera. Tri funkcije, *kamera.set(3,1280)*, *kamera.set(4,720)*, *cap.set(10,70)* služe za postavljanje širine i visine koju će kamera koristiti kao računalni vid, te osvijetljenost tog prozora kamere.

```
Import cv2
thres = 0.65

kamera = cv2.VideoCapture(0)
kamera.set(3,1280)
kamera.set(4,720)
kamera.set(10,70)

niz_imena = []
popis_imena = 'G:/AI_assistant/obj_detection/popis.names'
with open(popis_imena,'rt') as f:
    niz_imena = f.read().rstrip('\n').split('\n')

konfiguracija = 'G:/AI_assistant/obj_detection/opencv_v3_konfiguracija.pbtxt'
model = 'G:/AI_assistant/obj_detection/model.pb'

det = cv2.dnn_DetectionModel(model, konfiguracija)
det.setInputSize(520,520)
det.setInputScale(1.0/ 127.5)
det.setInputMean((127.5, 127.5, 127.5))
det.setInputSwapRB(True)

while True:
    success,slika = kamera.read()
    id_klasaa, pouzdanost, granica_prav = det.detect(slika,confThreshold=thres)
    print(id_klasaa,granica_prav)

    if len(id_klasaa) != 0:
        for id_klase, samouvjerenost,okvir in
zip(id_klasaa.flatten(),pouzdanost.flatten(),granica_prav):
            cv2.rectangle(slika,okvir,color=(0,255,0),thickness=2)
            cv2.putText(slika,niz_imena[id_klase-
1].upper(),(okvir[0]+10,okvir[1]+30),
            cv2.FONT_HERSHEY_COMPLEX,1,(255,0,0),2)
            cv2.putText(slika,str(round(samouvjerenost*100,2)),(okvir[0]+200,ok
vir[1]+30,            cv2.FONT_HERSHEY_COMPLEX,1,(255,0,0),2)

    cv2.imshow(„Izlaz“, slika)
    cv2.waitKey(1)
```

Unutar petlje *while True:* nalazi se još jedna petlja *if len(id_klasaa) != 0:* koja služi da bi se provjerilo da postoji nešto detektirano na kameri, jer inače ako programski kod ne detektira ništa automatski će prekinuti taj program. Te zadnje unutar te petlje, u sklopu sa funkcijama *cv2.rectangle()* za iscrtavanje pravokutnika i *cv2.putText()* za ispisivanje nazive objekata, ubačena je još jedna funkcija *cv2.putText(slika,str(round(samouvjerenost*100,2)),(okvir[0]+200,okvir[1]+30, cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)* koja služi za ispisivanje postotka pouzdanosti registriranja objekta.

4 PRIMJENA PROGRAMSKOG KODA NA ODABRANOM PRIMJERU

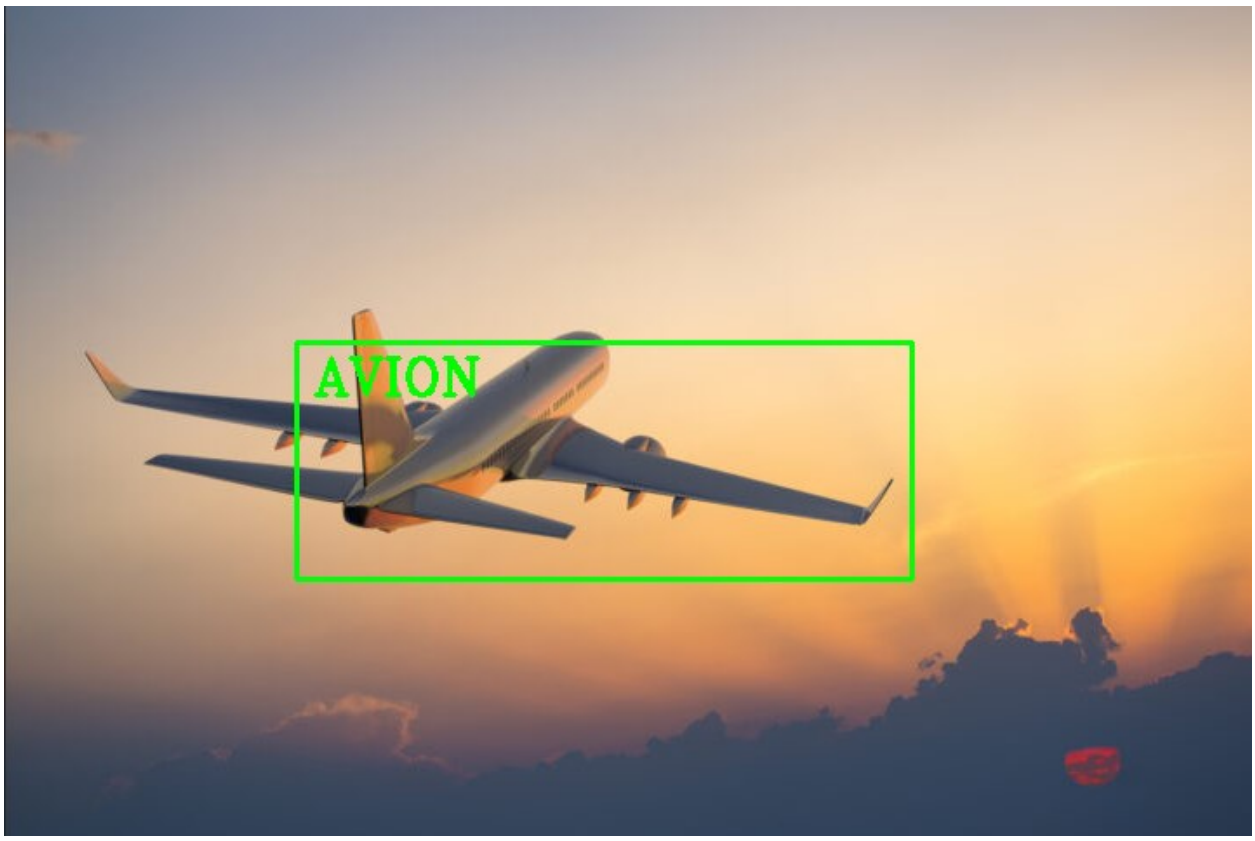
Za testiranje modela korišteno je 600 slika koje su klasificirane prema odabranim klasama i značajkama. U tablici 5.3 prikazani su trenirani objekti koje program može prepoznati sa visokom točnošću.

Tablica 4.1 *Trenirane klase*

OSOBA	SLON	MEDVJED	SNOWBOARD	BROD	KOLAĆ
BICIKL	HIDRANT	ZEBRA	SKATEBOARD	SEMAFOR	TV
AUTO	STOP	ŽIRAFA	FLAŠA	OVCA	STOLICA
MOTORCIKL	KLUPA	TORBA	CASA	KRAVA	KREVET
AVION	PTICA	KIŠOBRAN	SALICA	BANANA	MOBITEL
AUTOBUS	MAČKA	TORBICA	VILICA	NARANČA	DALJINSKI
VLAK	PAS	KRAVATA	NOŽ	TIPKOVNICA	SAT
KAMION	KONJ	KOFER	KAŠIKA	PICA	KNJIGA
ŠKARE	VAZA	CVIJET			
MIŠ	FRIŽIDER	SUŠILICA			

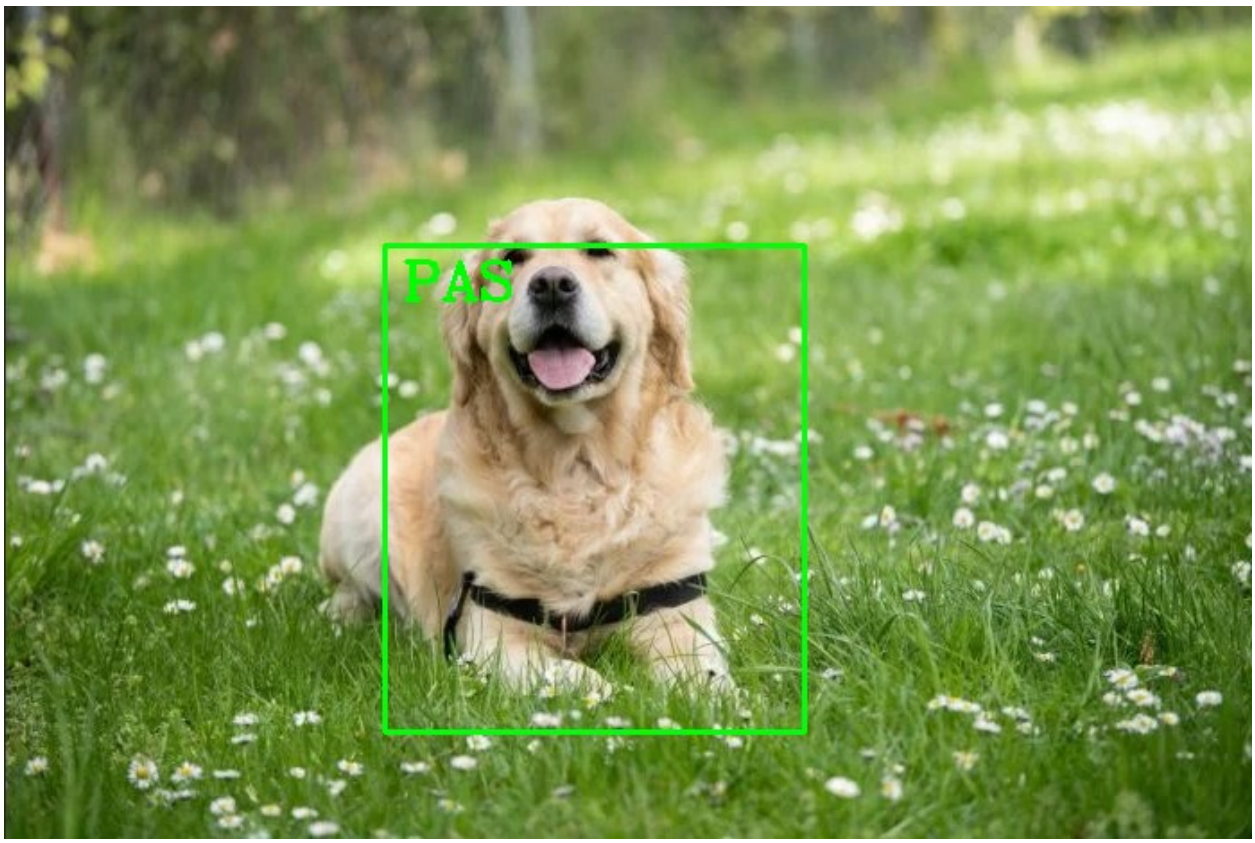
Kod detektira objekte koji su ključni ili izražajni te zatim iz baze podataka sa uzorcima pronalazi sličan uzorak i etiketira ga prema uzorcima i popisu imena i tako ga ispisuje na ekranu kao što je prikazano na slikama ispod.

Slika 4.1 *Primjer prepoznavanja vlaka*



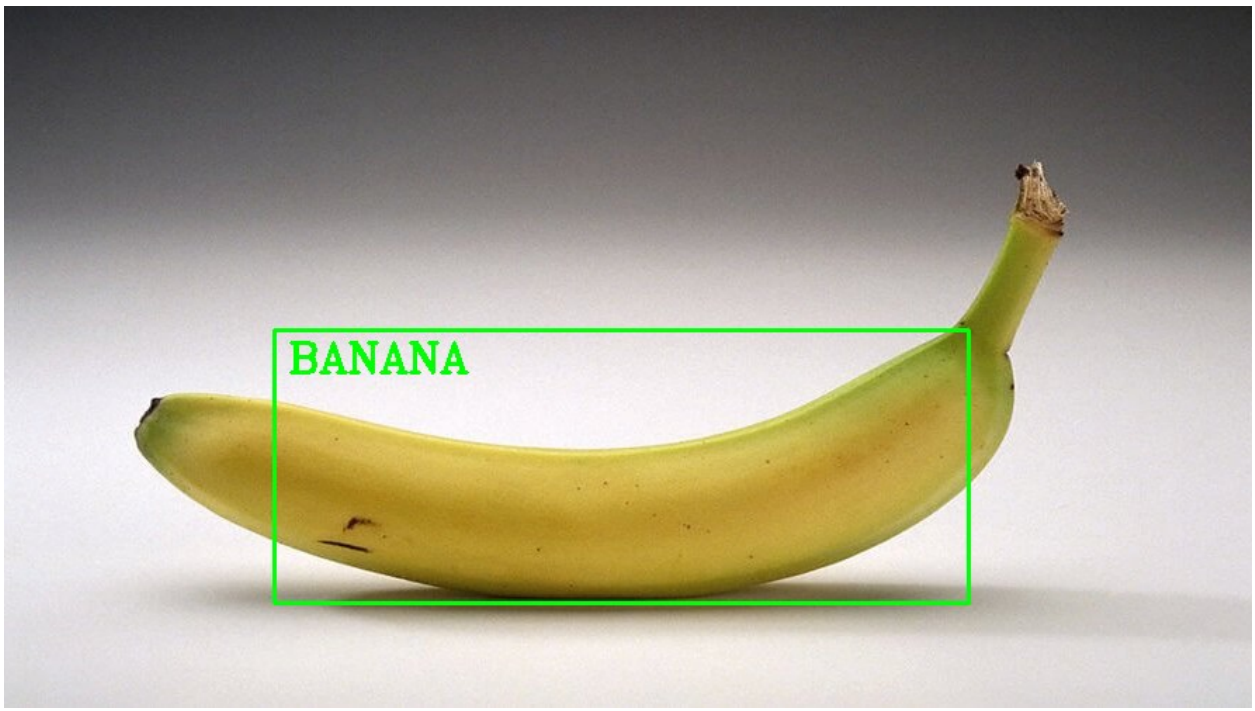
Slika 4.2 *Primjer prepoznavanja aviona*

Slika 4.3 *Primjer prepoznavanja protupožarnog hidranta*



Slika 4.4 *Primjer prepoznavanja psa*

Slika 4.5 *Primjer prepoznavanja broda*



Slika 4.6 *Primjer prepoznavanja banane*

Slika 4.7 *Primjer prepoznavanja automobila*

Slika 4.8 *Primjer prepoznavanja osobe i televizora*

Za prikaz točnosti primjene programa korišteno je 600 slika i dobivena je točnost dana u tablici 5.2.

Na slici 5.13 se može vidjeti da je program prepoznao dva automobila i jednu osobu te nakon što su prepoznati program iscrtava pravokutnike koji nam govore gdje se nalazi svaki objekt koji je prepoznat i time ga iz datoteke, koja je povezana sa treniranim modelom, s imenima svakog treniranog objekta povezuje sa prepoznatim objektom na slici.

Tablica 4.2 *Točnost prepoznavanja objekata*

Točnost prepoznatih objekata ($\frac{\#\ Točno\ prepoznati\ objekti}{\#\ Svi\ objekti\ koji\ su\ trenirani}$)	86.75%
Točnost prepoznatih imena objekata ($\frac{\#\ Točno\ prepoznata\ imena\ objekta}{\#\ Svi\ objekti\ koji\ su\ prepoznati}$)	92.32%

Pogreške je djelomično moguće objasniti tako što su objekti slični iz nekih kutova gledanja te eventualnom informacijom koju gubimo kroz obradu slike ili videa (*threshold* i promjena rezolucije slike). Pogreške se mogu smanjiti povećanjem broja slika svakog objekta za izradu preciznijeg točnijeg i boljeg modela, te u nekim slučajevima smanjenjem *thresholda* koji daje bolju točnost prepoznavanja objekta ali i manju točnost prepoznavanja imena objekta. U ovom radu korišteno je otprilike 10 slika za svaku klasu i objekt. Program je imao poteškoća sa prepoznavanjem točne veličine objekata koja bi se upravo smanjila sa više uzoraka za svaki objekt.

5 ZAKLJUČAK

Svrha zadatka je bila unaprijediti segment kontrole proizvodnog procesa koja će pridonijeti bržoj i kvalitetnijoj proizvodnji uz minimizaciju troškova. Vizualni sustav za kontrolu kvalitete je sub-optimalno rješenje koje ne zahtijeva velika ulaganja, dodatna ulaganja u školovanja radnika i/ili velike promjene u postojećim procesima.

Fleksibilnost i primjenjivost na različite modele kontrole su bitne karakteristike vizualnih sustava. Zato poduzeće koja proizvode više vrsta modela proizvoda i/ili veći broj komada proizvoda korištenje ovakvog sustava može dovesti ne samo do smanjenje potencijalnog škarta i/ili dorade, smanjenja troškova ima i svoju ekonomsku opravdanost.

Prednosti uvođenja vizualnog sustava osim poboljšavanja kvalitete kontrole (velika ponovljivost, smanjenje škarta/dorade, kraćeg vremena potrebnog za kontrolu) vidi se u informatizaciji i automatizaciji same proizvodnje što je značajno s aspekta održavanje. Također ovakav način pruža mogućnost poduzeću dobivanje potrebitih ISO certifikata.

Naravno o kvaliteti kamere primarno mogu ovisiti i rezultati procesa. U ovakvim sustavima potrebno je osigurati da se objekt provjere dovede do vidnog polja koje pokriva kamera. Izvor svjetlosti mora biti neometan (ne smije biti sjena ili prepreka između izvora i objekta kontrole) i dobro usmjeren pošto svaka promjena u osvjetljenju ima mogućnost utjecaja na rezultate.

U radu je korištena OpenCV biblioteka otvorenog koda sa mnoštvom algoritama i struktura podataka koji se mogu koristiti kao komponente za razvijanje programa i aplikacija povezanih sa vizualnim sustavom. Uz navedeno i pomoću dodatnih biblioteka moguće je iskoristivost u prepoznavanju raznih objekata i radnji što računalo može obraditi u izuzetno kratkom vremenu (nekoliko sekundi ili dijelova sekunde), a samim tim i neposredno ubrzati proces kontrole.

6 LITERATURA

[1] Zyad M., *Artificial Intelligence Deffinition, Ethics and Standards*, The British University in Egypt, 2018/2019.

[2] URL: https://hr.about-meaning.com/11037608-artificial-intelligence-ai

[3] https://www.analyticssteps.com/blogs/6-major-branches-artificial-intelligence-ai

[4] Bofl N., *Osvježimo Znanje*, Strojno učenje, 68, 2021, 9-10, 591-593

[5] Bofl N., Ujević Andrijić Ž., *Osvježimo Znanje*, Umjetne neuronske mreže, 70,2019, 5-6, 219-220

[6] Hash Dork
URL: https://hashdork.com/hr/how-artificial-intelligence-is-used-in-robotics/

[7] Autopoiesis
URL: http://autopoiesis.foi.hr/wiki.php?name=Ekspertni+sustavi&parent=NULL&page=2.Povijesni%20razvoj&lang=hr

[8] Manning C. D., Schutze H., Foundations of Statistical Natural Language Processing: MIT Press., 1999.

[9] Negnevitsky M., Artificial Intelligence - a guide to intelligent systems, Addison Weslwy, USA, ISBN: 0201-71159-1

[10] Deng L., Liu Y., A Joint Introduction to Natural Language Processing and to Deep Learning“, 2018.

[11] Eisenstein J., Introduction to natural language processing: MIT Press., 2019.

[12] Poibeau T., Machine Translation, Cambridge MA: MIT Press Essential Knowledge series, 2017.

[13] Lane H., Howard C., Hapke H., Natural Language Processing in Action – Understanding, analyzing, and generating text with Python, New York: Manning Publications Co., 2019.

[14] URL: https://www.section.io/engineering-education/the-basics-of-genetic-algorithms-in-ml/

[15] J.S.R.Jang, C.T.Sun and E.Mizutani, “*Neuro-Fuzzy and Soft Computing”*, PHI, 2004, Pearson Education 2004.

[16] URL: https://logicai.io/blog/using-artificial-intelligence-ai-image-recognition/

[17] SuperAnnotate
URL: https://blog.superannotate.com/guide-to-training-data/

[18] G2
URL: https://www.g2.com/categories/image-recognition

[19] Techcrunch
URL: https://techcrunch.com/2017/07/18/syte/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAAmL4kEZxpGAOFipMad4wGYgoPdCFlqkKy3fQ6u2pIGYrx16srm0nFfybOk_0J3ewunDlxk3Gb0aQPgWV7ZcecsfraGEWI9uYCNaygYlUeCGy7WWJpl4AvrnHBvVm41P_gm9Q902jRT0yqoKzex7zmVmjfkKxJhxznG9Yw32wQbG

[20] URL: https://www.syte.ai/visual-discovery-suite/

[21] URL: https://virtualizationreview.com/articles/2018/02/20/~/media/ECG/virtualizationreview/Images/2018/Rekognition1.ashx

[22] URL: https://superannotate.medium.com/introducing-annotate-online-76dca32b8c72

[23] Vue.ai
URL: https://vue.ai/products/on-model-imagery/

[24] Visual Studio Code
URL: https://code.visualstudio.com/

[25] geeks for geeks
URL: https://www.geeksforgeeks.org/

[26] OpenCV
URL: https://opencv.org/

[27] Bradski G., Kaehler A., Learning OpenCV: Computer vision with the OpenCV library, O'Reilly Media, Inc., 2008.

[28] URL: https://www.superannotate.com/blog/pattern-recognition-overview

[29] URL: https://www.superannotate.com/blog/guide-to-training-data

[30] URL: https://shop53002.lemaprod.com/content?c=computer%20vision%20pattern%20recognition&id=19