Udaljeni sustav praćenja broja ljudi u zatvorenom prostoru koristeći Raspberry Pi i obradu slike

Radić, Domin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

UDALJENI SUSTAV PRAĆENJA BROJA LJUDI U ZATVORENOM PROSTORU KORISTEĆI RASPBERRY PI I OBRADU SLIKE

Završni rad

Domin Radić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 13.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Domin Radić
Studij, smjer:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska
Mat. br. Pristupnika, godina upisa:	4919, 25.09.2020.
OIB Pristupnika:	54445615368
Mentor:	prof. dr. sc. Irena Galić
Sumentor:	Marin Benčević, mag. ing. comp.
Sumentor iz tvrtke:	
Naslov završnog rada:	Udaljeni sustav praćenja broja ljudi u zatvorenom prostoru koristeći Raspberry Pi i obradu slike
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rad:	Razviti sustav koristeći Raspberry Pi, kameru i metode obrade slike za praćenje trenutnog broja ljudi u prostoriji. Metodama obrade slike raspoznati ljude na slici, te razlikovati ljude koji ulaze i izlaze iz prostorije. Zapisivati trenutan broj ljudi u prostoriji u datoteku. Napraviti sučelje za prikaz i obradu broja ljudi u prostoriji kroz vrijeme. Ispitati rad sustava u stvarnoj prostoriji i usporediti sa stvarnim brojem ljudi u
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/bodaJasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	13.09.2023.
Datum potvrde ocjene od strane Odbora:	

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 18.09.2023.

Ime i prezime studenta:	Domin Radić
Studij:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4919, 25.09.2020.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Udaljeni sustav praćenja broja ljudi u zatvorenom prostoru koristeći Raspberry Pi i obradu slike**

izrađen pod vodstvom mentora prof. dr. sc. Irena Galić

i sumentora Marin Benčević, mag. ing. comp.

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mije bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

U današnjem digitalnom dobu, tehnologija neprestano napreduje i donosi inovativna rješenja za mnoge probleme s kojima se susrećemo. Jedno takvo područje je i računalni vid. Računalni vid je grana umjetne inteligencije koja omogućuju računalima da interpretiraju, analiziraju i razumiju vizualne informacije. Zanimljiv aspekt računalnog vida je i detekcija objekata koja je postala vrlo popularna, a ključan dio u svemu tome je razvoj konvolucijskih neuronskih mreža. Raspberry Pi, mali jednokartični računalni sustav, postao je popularan izbor za implementaciju takvih sustava zahvaljujući svojoj pristupačnosti, jednostavnosti upotrebe i fleksibilnosti.

Ovaj završni rad ima za cilj istražiti primjenu Raspberry Pi-a u kombinaciji s obradom slike za udaljeno praćenje broja ljudi u zatvorenim prostorima. Obrada slike omogućuje računalu da automatski prepozna i broji ljude na temelju snimljenih videozapisa ili fotografija. Koristeći tehnike dubokog učenja, sustav će moći detektirati ljude u stvarnom vremenu i točno pratiti njihov broj. U radu će se detaljno opisati konvolucijske neuronske mreže, osnovne komponente sustava i prikazati različiti sustavi za detekciju objekata te otkrivanje njihovih prednosti i nedostataka.

U nastavku rada bit će detaljno objašnjen postupak implementacije udaljenog sustava praćenja broja ljudi. Razmotrit će se prikupljanje podataka putem kamere spojene na Raspberry Pi, obradu slike koristeći odgovarajuće biblioteke i YOLOv4-tiny algoritam za detekciju ljudi, te prikaz i analiza rezultata.

1.1. Zadatak završnog rada

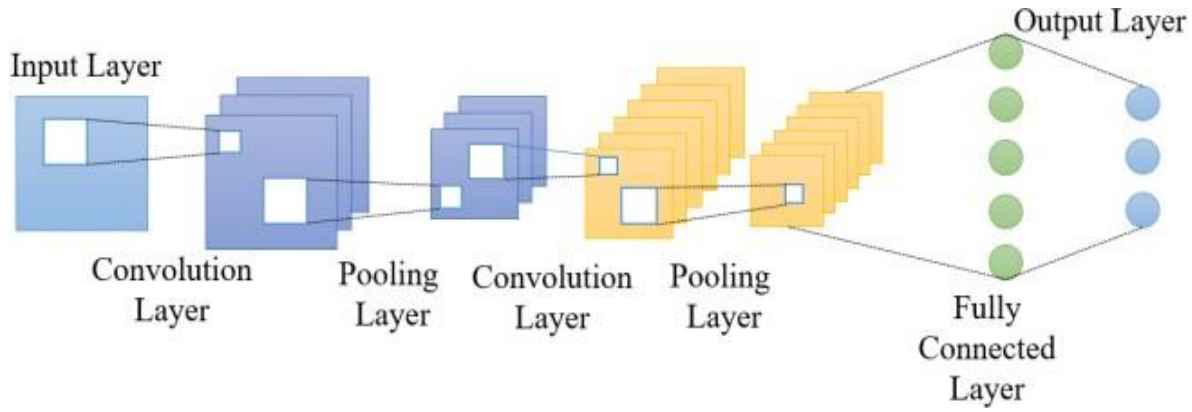
Razviti sustav koristeći Raspberry Pi, kameru i metode obrade slike za praćenje trenutnog broja ljudi u prostoriji. Metodama obrade slike raspoznati ljude na slici, te razlikovati ljude koji ulaze i izlaze iz prostorije. Zapisivati trenutan broj ljudi u prostoriji u datoteku. Napraviti sučelje za prikaz i obradu broja ljudi u prostoriji kroz vrijeme. Ispitati rad sustava u stvarnoj prostoriji i usporediti sa stvarnim brojem ljudi u prostoriji. Opisati sastavne dijelove algoritma te njegove prednosti i nedostatke.

2. DUBOKE NEURONSKE MREŽE

Duboke neuronske mreže (eng. *Deep Neural Networks - DNN*) su metode strojnog učenja koje za cilj imaju da uče računala kako izvoditi ono što je prirodno ljudima, odnosno učiti ga obavljati zadatke klasifikacije izravno iz slika. Danas se koriste u različite svrhe, a zbog svoje visoke preciznosti i točnosti koristi se u medicini za dijagnozu bolesti[1]. Glavna značajka modela je da treniraju sa velikim skupom podataka i arhitektura neuronskih mreža sa mnogo slojeva, na taj način DNN uči na isti način koji to radi i ljudski mozak – vježbanjem i pravljenjem pogrešaka[2]. U daljnjem tekstu ćemo detaljnije analizirati neke od najčešće korištenih neuronskih mreža koje se primjenjuju, između ostalog, u detekciji ljudi i objekata.

2.1. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (engl. *Convolutional Neural Networks - CNN*) su vrste neuronskih mreža koje imaju zadatak klasificirati ulazne slike. Primjenjuju se u raznim područjima medicine, klasifikacija slika, prepoznavanju lica i detekciji objekata[3]. Osnovna zamisao neuronskih mreža je da na temelju određenog skupa podataka se kreira model za predviđanje uzoraka ili slika. Tokom procesa treniranja ulazna slika se pretvara u vektor te se proslijedi kroz sve slojeve mreže. Sastoje od tri ključna sloja: konvolucijski sloj, sloja udruživanja i potpuno povezanog sloja, a više informacija o svakom sloju će biti u nastavku[4].

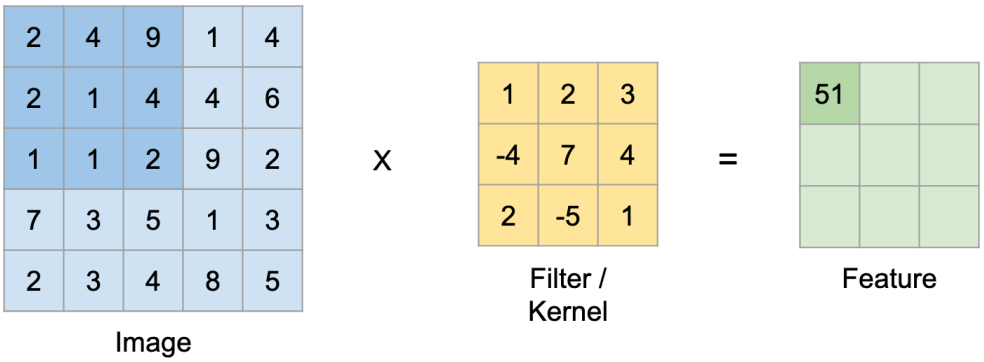


Slika 2.1. Struktura konvolucijskih neuronskih mreža[5]

2.1.1. Konvolucijski sloj

Konvolucijski sloj predstavlja osnovnu gradivnu jedinicu konvolucijske neuronske mreže u kojem se odvija većina izračuna. Sastoji se od ulaznih podataka, filtera i mapa značajki. Zamislimo primjer u kojem se na ulazu pojavi slika koja je u boji i predstavljena je kao trodimenzionalna matrica piksela (3D). Tada će matrica imati tri dimenzije koje će odgovarati RGB komponentama slike - visinu, širinu i dubinu. Također, u cijelom procesu koristimo i detektor značajki, koji se koji se naziva još jezgra ili filtar. U postupku izdvajanja značajki koristimo detektor značajki koji se kreće po receptivnim poljima slike provjeravajući prisutnost značajke. Taj postupak naziva se konvolucija.

Detektor značajki je dvodimenzionalno (2D) polje koje predstavlja dio slike. Veličina detektora je obično matrica 3x3, iako se veličine mogu razlikovati, a to ujedno određuje i veličinu recepcijskog polja. Nakon što se filter primijeni na područje slike, računa se skalarni produkt ulazne slike i filtra, a rezultat se unosi u izlazno polje. Zatim se detektor pomiče i ponavlja proces dok se ne pokrije cijela slika. Konačni izlaz iz niza procesa skalarnih produkata ulaza i filtra je mapa značajki[6].



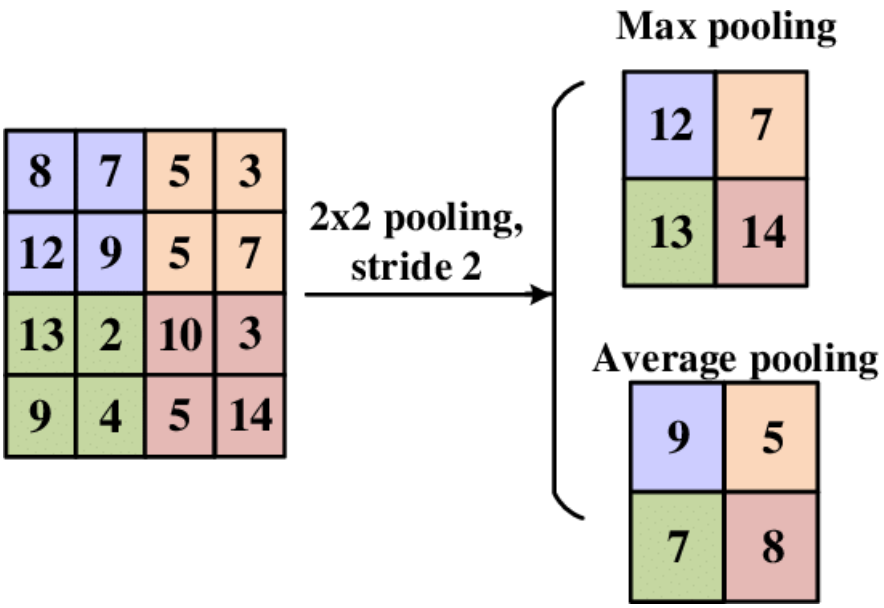
Slika 2.2. Proces izdvajanja značajki[7]

2.1.2. Sloj udruživanja

Sloj udruživanja, poznat kao i sloj za grupiranje (engl. *pooling layer*), je važna komponenta u konvolucijskim neuronskim mrežama (*CNN*) koja se obično primjenjuje nakon konvolucijskog sloja. Svrha sloja udruživanja je smanjivanje dimenzionalnosti izlaza konvolucijskog sloja, čime se smanjuje računalni teret i broj parametara u mreži, a istovremeno se zadržava najvažnija informacija iz prethodnih konvolucijskih slojeva.

Sloj udruživanja prolazi kroz izlaz konvolucijskog sloja i uzorkuje (smanjuje) prostorne dimenzije značajki. To se obično postiže tako da se iz svake manje regije uzima jedna vrijednost. Na primjer, koristeći maksimalno udruživanje (engl. *max pooling*) odabire se maksimalna vrijednost unutar tog područja, i ta maksimalna vrijednost postaje dio izlazne značajke.

Iako je maksimalno udruživanje najčešći tip udruživanja, mogu se koristiti i drugi tipovi udruživanja, kao što je prosječno udruživanje (engl. *average pooling*) gdje se računa prosječna vrijednost u svakoj regiji.



Slika 2.3. Usporedba tehnika udruživanja po maksimalnoj i po prosječnoj vrijednosti [8]

Premda u sloju udruživanja gubimo mnogo informacija, također ima nekoliko prednosti za CNN. Pomaže smanjiti složenost i poboljšava učinkovitost[6].

2.1.3. Potpuno povezani sloj

Potpuno povezani sloj (engl. *fully connected layer*) u neuronskim mrežama obavlja završnu obradu izlaza iz prethodnih slojeva te se koristi za donošenje konačnih odluka ili predviđanja. Svaki čvor u potpuno povezanom sloju povezan je sa svim čvorovima iz prethodnog sloja. Potpuno povezani slojevi obično se koriste za klasifikacijske zadatke gdje mreža treba klasificirati ulazne podatke u jednu od više kategorija ili razreda koristeći aktivacijske funkcije. Na primjer, u klasifikaciji slika, potpuno povezani sloj uzima vektor značajki i generira izlazne vjerojatnosti za različite klase. U zadacima regresije, potpuno povezani sloj može generirati kontinuirane vrijednosti kao izlaz[6].

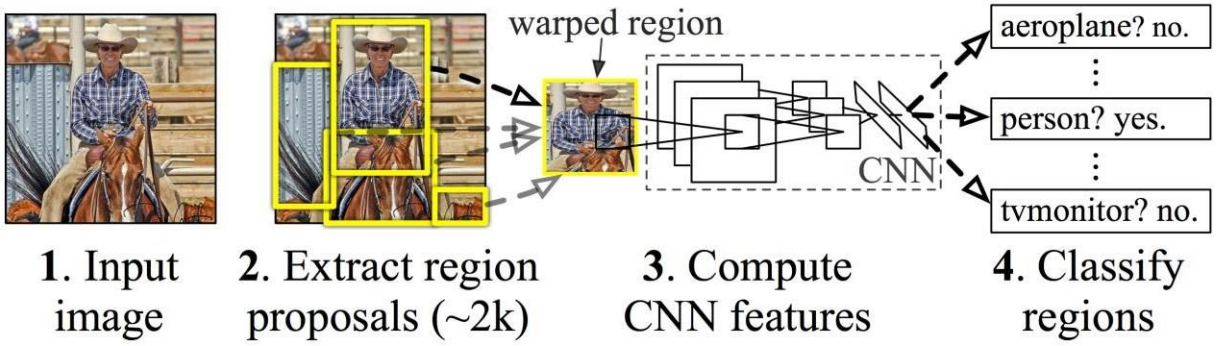
2.2. Primjeri konvolucijskih neuronskih mreža

2.2.1. R-CNN

R-CNN (engl. *Region-based Convolutional Neural Network*) je metoda za detekciju objekata koja koristi predloške regija za prepoznavanje objekata na slikama. R-CNN počinje generiranjem potencijalnih regija objekata (graničnih okvira) na slici koristeći vanjski algoritam koji generira oko 2000 potencijalnih regija koje bi mogle sadržavati objekte.

Algoritam analizira sliku i generira okvire koje bi mogle pripadati pojedinim objektima, uzimajući u obzir boju, teksturu, veličinu i oblik. Nakon što su prijedlozi regija generirani, koristi se prethodno obučena konvolucijska neuronska mreža (*CNN*) kako bi se izvukle značajke iz svakog prijedloga regije. Za svaki prijedlog regije, CNN generira 4096-dimenzionalni vektor značajki. Ove značajke opisuju što se nalazi unutar svake regije (klasifikacija objekta). Nakon izdvajanja značajki, koristi se linearni SVM za svaku klasu kako bi se otkrila prisutnost ili odsutnost objekta te klase u prijedlogu regije. Konačni rezultati detekcije objekata dobiveni su kombinacijom rezultata iz klasifikacije i regresije graničnih okvira[9].

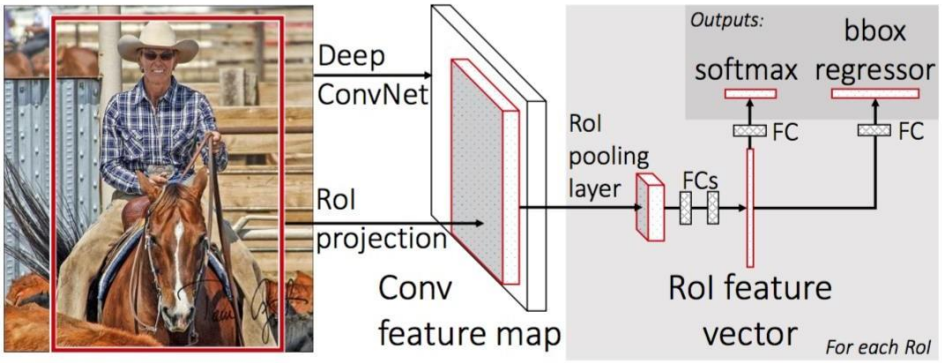
Uz dosta prednosti R-CNN ima i svojih nedostataka, prije svega u brzini izvođenja. Postupak generiranja potencijalnih regija i obrada svake regije kroz konvolucijsku neuronsku mrežu zahtijeva puno vremena i resursa. R-CNN je poznat po svojoj sporoj obradi, što ga čini izazovnim za korištenje u stvarnom vremenu zbog značajnog vremena koje je potrebno za obradu – oko 47 sekundi po slici - koje ovise o o performansama korištenog računalnog sustava[10].



Slika 2.4. Analiza R-CNN metode [10]

2.2.2. Brzi R-CNN

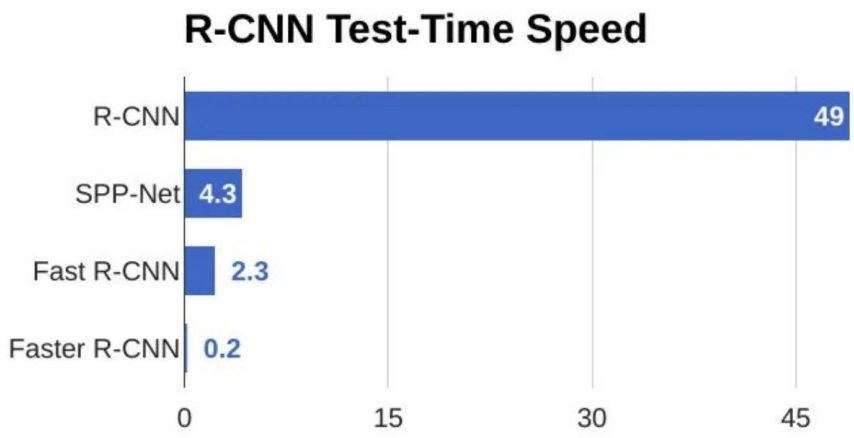
Isti autor ispravlja neke od nedostataka R-CNN-a kako bi se izgradio brži algoritam za detekciju objekata[10]. Brzi R-CNN kao ulaz koristi cjelokupnu sliku i skup regija koje sadrže objekte. Mreža obradi cijelu sliku kroz niz konvolucijskih slojeva kako bi se stvorila mapa značajki. Na temelju izdvojenih značajki, koristi se sloj regije prijedloga (engl. *Region Proposal Network - RPN*) za generiranje regija koje potencijalno sadrže objekte. Za svaku regiju koja sadrži objekt, sloj regija interesa (engl. *RoI - Region of Interest*) izdvaja karakteristike iz svake regije prijedloga. Ovaj korak omogućuje prilagodbu veličine regija i izdvajanje važnih informacija. Izlazi iz sloja regije interesa šalju se kroz potpuno povezane slojeve koji se dijele u dva izlazna sloja. Jedan od izlaznih slojeva procjenjuje vjerojatnosti klasa objekata korištenjem softmax funkcije nad klasama objekta. Drugi sloj generira četiri broja za svaku od klasa objekta, koji predstavljaju korekcije graničnih okvira za jednu od klasa objekta. Što omogućuje bolje pozicioniranje i točnost lokalizacije objekata na slici[11].



Slika 2.5. Struktura Brze R-CNN [10]

2.2.3. Brži R-CNN

Brži R-CNN (engl. *Faster R-CNN*) je unaprijeđena verzija brzog R-CNN modela za detekciju objekata na slikama. Ključna promjena koju donosi je uvođenje sloja regije prijedloga (*RPN*) direktno unutar samog modela, koji ide korak dalje u smislu efikasnosti i brzine generiranja prijedloga regija. Postupak je sličan kao i kod brzog R-CNN-a, slika se šalje kroz konvolucijsku mrežu kako bi se izdvojile značajke, a zatim se koristi zasebena mreža za predviđanje prijedloga regija - RPN[10]. Svaki prijedlog je povezan s rezultatom "objektnosti" (objectness score) koji određuje nalazi li se objekt unutar tog prijedloga. Nakon toga odabiru se relevantni prijedlozi, koji se oblikuju u fiksne dimenzije pomoću sloja regije interesa (*RoI*) kako bi se koristili u daljnjem procesu. Rezultati RoI sloja šalju se kroz potpuno povezane slojeve gdje model klasificira prisutnost objekata u svakoj regiji i precizira granice graničnih okvira za svaku klasu[12].



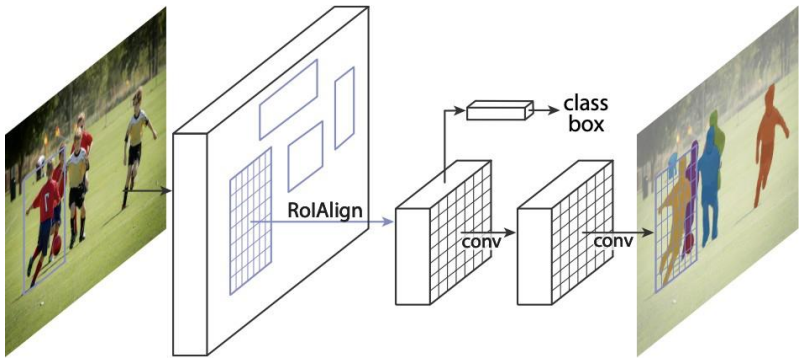
Slika 2.6 Usporedba brzine različitih algoritama za detekciju objekata [10]

Iz gornjeg grafa očigledno je da je Brži R-CNN znatno brži od svojih prethodnika. Ova brža izvedba čini ga pogodnim za detekciju objekata u stvarnom vremenu[10].

2.2.4. Maskirani R-CNN

Maskirani R-CNN (engl. *Mask Region-based Convolutional Neural Network*) je nadogradnja brže R-CNN metode. Dok Brži R-CNN ima dva izlaza za svaku od klasa objekta, Mask R-CNN dodaje i treću granu koja generira binarne maske za svaki objekt na slici. Mask R-CNN generira binarne maske koje precizno određuju piksele koji pripadaju svakom objektu na slici. To omogućuje

preciznu segmentaciju objekata i stvaranje točnih maski. Kako bi osigurao precizno poravnanje piksela između ulaznih i izlaznih podataka, Mask R-CNN koristi RoIAlign sloj. Ovaj sloj je ključan za preciznu maskiranu segmentaciju i eliminira nesavršenosti u prostornom poravnanju koje su prisutne u prethodnim verzijama R-CNN modela[13].

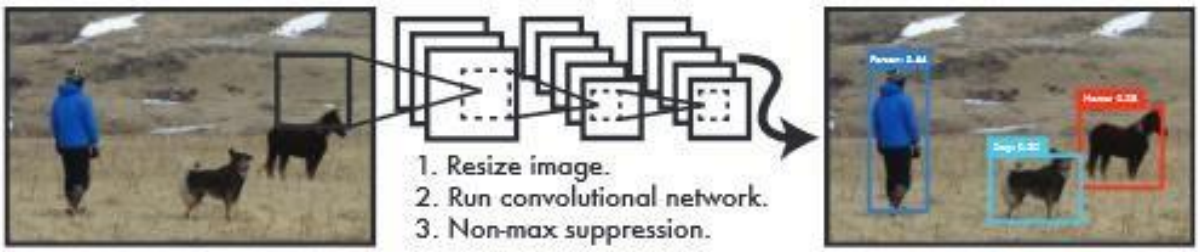


Slika 2.7. Maskirani R-CNN[13]

2.2.5. YOLO

Prethodno opisani sustavi za detekciju objekata koriste prenamjenu klasifikatora za obavljanje detekcije. Kako bi otkrili prisutnost objekta, spomenuti sustavi koriste klasifikatore za svaki objekt i vrednuju ga na različitim lokacijama i skalama u testnoj slici. Koriste složene postupke koji su spori i teški za optimizaciju jer svaku pojedinačnu komponentu je potrebno zasebno trenirati.

YOLO (engl. *You Only Look Once)* sustav preoblikuje proces detekcije objekata u jednostavan regresijski problem, gdje izravno iz piksela slike predviđa koordinate graničnih okvira (*bounding boxova*) i vjerojatnosti klasa. Koristeći YOLO sustav, samo jednom se pogleda slika (*You Only Look Once*) da bi se predvidjeli prisutni objekti te njihova lokacija. Jedna konvolucijska mreža istovremeno predviđa više graničnih okvira i vjerojatnosti pripadajućih klasa za svaki od tih okvira. YOLO trenira na punim slikama i izravno optimizira performanse detekcije, čineći proces bržim i učinkovitijim u usporedbi s drugim metodama.



Slika 2.8. YOLO sustav za detekciju počinje s promjenom veličine ulazne slike, nakon čega slijedi pokreteanje konvolucijske mreže koja generira rezultate detekcije [14]

YOLO sustav objedinjuje odvojene komponente detekcije objekata unutar jednog neuronskog modela. Ovaj model koristi karakteristike cijele slike kako bi predvidio pojedinačne granične okvire objekata, a istovremeno predviđa sve okvire za sve različite klase u toj slici. Drugim riječima, mreža globalno analizira cijelu sliku i sve prisutne objekte unutar nje. YOLO još uvijek nije na razini najnovijih sustava detekcije u smislu točnosti. Iako je brz u prepoznavanju objekata na slikama, suočava se s izazovima u preciznom lociranju određenih objekata, osobito onih manjih.

U YOLO sustavu, ulazna slika se podijeli na rešetku veličine S × S. Ako se središte objekta nalazi unutar određene ćelije rešetke, ta ćelija je odgovorna za detekciju tog objekta. Svaka ćelija rešetke ima zadatak predviđanja B graničnih okvira i ocjena pouzdanosti za te okvire. Ocjene povjerenja odražavaju koliko je siguran model da box sadrži objekt, kao i koliko precizan misli da je u predviđanju tog boxa. Formalno, pouzdanost definiramo kao:

$$\mathbf{Pr}(\boldsymbol{Object}) * \boldsymbol{IOU}_{\boldsymbol{pred}}^{\boldsymbol{truth}} \tag{2-1}$$

Gdje je $\boldsymbol{Pr}(\boldsymbol{Object})$ vjerojatnost da se u određenoj regiji nalazi objekt, a $\boldsymbol{IOU}_{\boldsymbol{pred}}^{\boldsymbol{truth}}$ (*Intersection over Union*) mjera koja uspoređuje koliko dobro se predložena regija poklapa s pravim objektom između stvarnih i predviđenih bounding boxova objekata. Ako u ćeliji ne postoji objekt, rezultat pouzdanosti treba biti nula. Inače, ako postoji objekt, želimo da rezultat pouzdanosti bude jednak preklapanju (*IOU*) između predviđenog boxa i stvarnog stanja.

Svaki okvir u YOLO sustavu ima pet predviđenih parametara: x i y, koji označavaju središte okvira u odnosu na granice mrežne ćelije, te w i h, koji predstavljaju širinu i visinu okvira u odnosu na

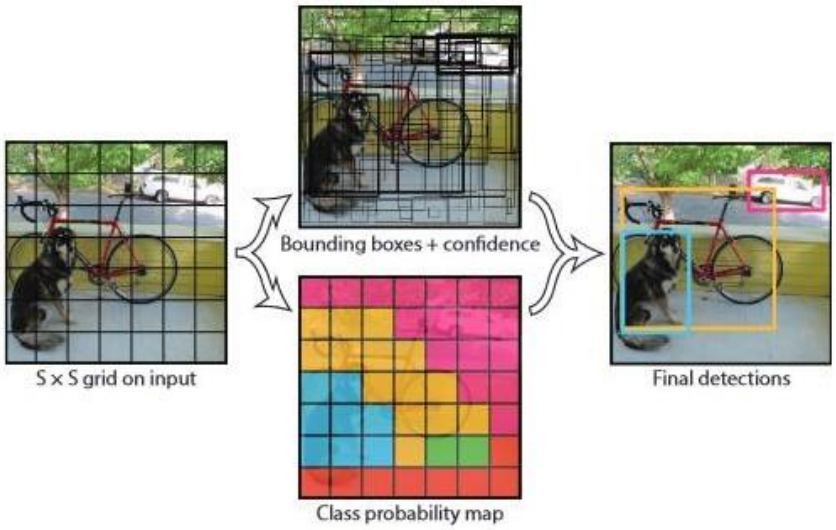
cijelu sliku. Osim toga, postoji i predviđanje pouzdanosti, koja procjenjuje koliko je vjerojatno da je predviđeni okvir točan tako da uspoređuje preklapanje s bilo kojim stvarnim okvirom na slici.

Svaka ćelija mreže u YOLO sustavu također daje predviđanja C uvjetnih vjerojatnosti klasa, $Pr(Classi|Object)$. Vjerojatnosti ovise o prisutnosti objekta u ćeliji mreže. Važno je napomenuti da se predviđa samo jedan skup vjerojatnosti klasa za svaku mrežnu ćeliju, bez obzira na broj graničnih okvira B.

Tijekom testiranja, množimo uvjetne vjerojatnosti klasa s ocjenama pouzdanosti pojedinačnih graničnih okvira, što rezultira ocjenama pouzdanosti specifičnim za svaku klasu za svaki granični okvir.

$$\boldsymbol{Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth}} \quad (2\text{-}2)$$

gdje je $\boldsymbol{Pr(Class_i|Object)}$ vjerojatnost da se klasa $\boldsymbol{Class_i}$ nalazi u graničnom okviru ***Object***, ***Pr(Object)*** vjerojatnost da je prisutan objekt u mrežnoj ćeliji, $\boldsymbol{IOU_{pred}^{truth}}$ stvarno preklapanje (*Intersection Over Union*) između predviđenog graničnog okvira i stvarnog graničnog okvira i $\boldsymbol{Pr(Class_i)}$ ukupna vjerojatnost da je klasa $\boldsymbol{Class_i}$ prisutna, bez obzira na granični okvir. Rezultati kodiraju vjerojatnost da se ta klasa pojavi u okviru, kao i koliko precizno predviđeni okvir odgovara objektima slici.

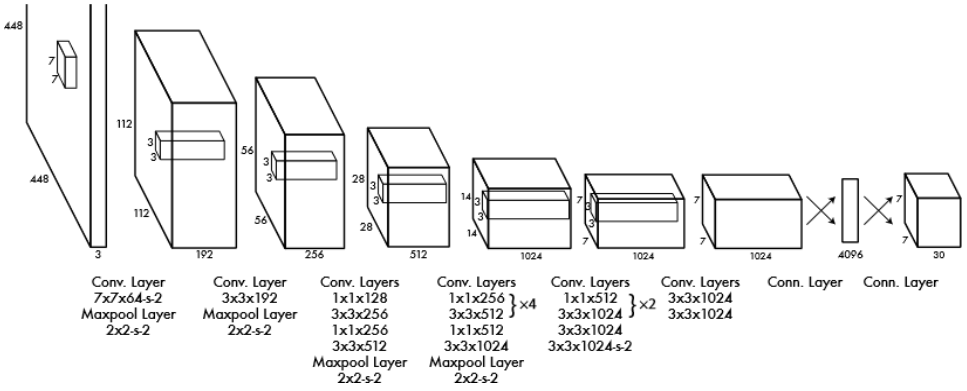


Slika 2.9. Sustav razdvaja sliku na rešetku veličine S x S i za svaku ćeliju rešetke prognozira B graničnih okvira, njihovu pouzdanost i vjerojatnost pripadajuće klase C [14]

Ovaj model implementiran je kao konvolucijska neuronska mreža koja se sadrži ukupno 24 konvolucijska sloja i dva potpuno povezana sloja. Početni konvolucijski slojevi mreže služe za izdvajanje značajki iz ulazne slike, dok potpuno povezani slojevi preuzimaju te značajke i koriste ih za predviđanje izlaznih vjerojatnosti i koordinata objekata.

Mreža koristi 1 × 1 slojeve redukcije koji se zatim nadovezuju na konvolucijske slojeve 3 × 3.

Također postoji brža verzija YOLO-a, koja je optimizirana za bržu detekciju objekata. Koristi neuronsku mrežu koja ima manje konvolucijskih slojeva i manji broj filtera u slojevima. Osim razlike u veličini mreže, ostali parametri između osnovnog YOLO-a i brzog YOLO-a su isti.



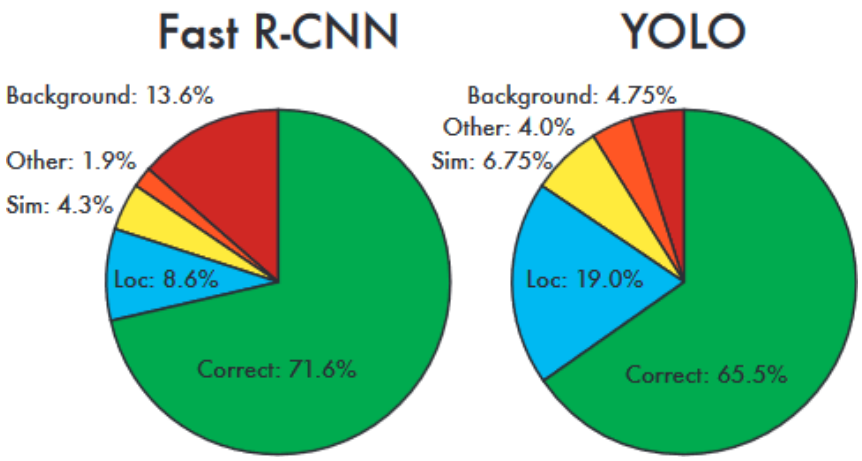
Slika 2.10. Arhitektura mreže[14]

R-CNN i njegove varijante koriste složeni sustav za pronalaženje objekata na slikama koji uključuje selektivnu pretragu za generiranje potencijalnih graničnih okvira, izdvajanje značajki konvolucijskom mrežom, ocjenjivanje okvira pomoću SVM-a i prilagodbu okvira linearnim modelom, uz korištenje postupka non-max suppression za uklanjanje duplih otkrivenih objekata. Svaka faza ovog sustava zahtijeva posebnu prilagodbu, što ga čini sporim.

YOLO dijeli sličnosti s R-CNN-om jer također predlaže potencijalne granične okvire korištenjem konvolucijskih značajki, ali postavlja ograničenja na te prijedloge kako bi se spriječilo višestruko otkrivanje istih objekata. Pored toga, YOLO predlaže znatno manji broj graničnih okvira, njih 98, u odnosu na selektivnu pretragu koja predlaže oko 2000 okvira po slici.

YOLO sustav kombinira različite komponente u jednostavan model, čime se postiže brža i jednostavnija detekcija. Iako brzi i brži R-CNN pristupi ubrzavaju R-CNN korištenjem neuronske mreže umjesto složenog postupka generiranja prijedloga regija, YOLO se ističe po brzini, ali se

suočava s izazovima u preciznoj lokalizaciji objekata, što može rezultirati većim pogreškama u usporedbi s drugim sustavima[14].



Slika 2.11. Usporedba analize pogreške brzog R-CNN u odnosu na YOLO[14]

3. KORIŠTENE TEHNOLOGIJE I SUSTAVI

Tehnolgije i sustavi koji su korišteni za izradu ovog sustava su:

- Raspberry Pi – mikroračunalo koje će se koristiti kao osnovna platforma za izradu sustava.
- OpenCV – biblioteka koja pruža mnoge funkcionalnosti za obradu slika, uključujući detekciju objekata, praćenje i analizu.
- YOLOv4-tiny neuronska mreža – neuronska mreža za detekciju objekata u slikama. Koristit ćemo YOLO model za detekciju i praćenje ljudi na video snimkama.
- Python – programski jezik opće namjene. Python je često preferirani jezik zu rad s OpenCV-om i neuronskim mrežama jer pruža bogatu biblioteku i jednostavnost upotrebe.
- Kamera – dodatni modul koji se spaja na Raspberry Pi, a služi za videonadzor i detekciju ljudi.

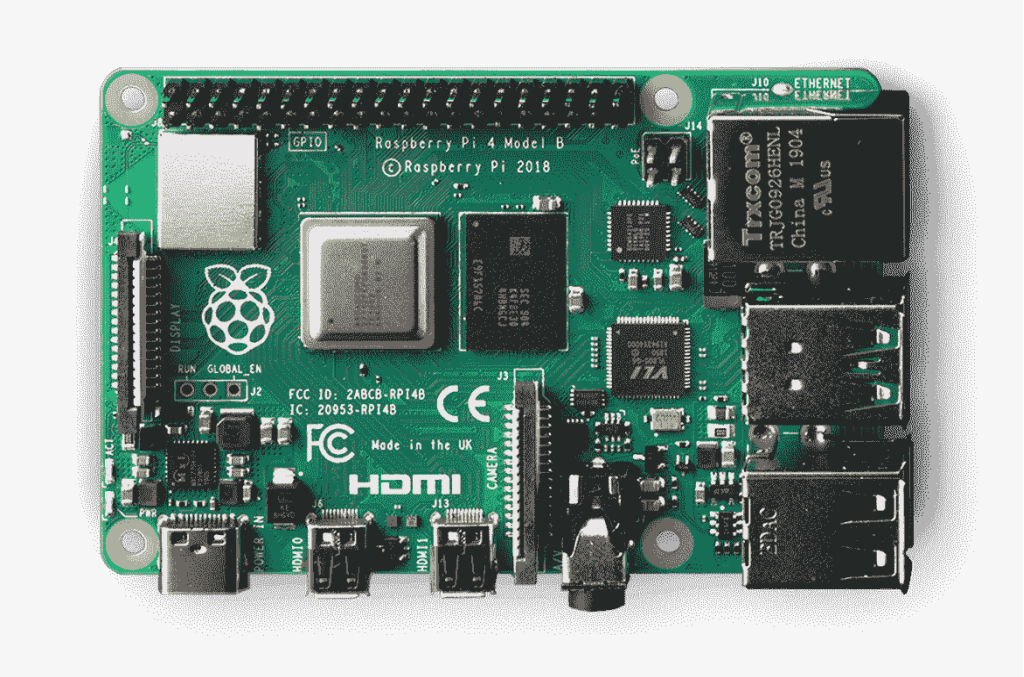
3.1. Raspberry Pi

Raspberry Pi [15] je uređaj koji sadrži skoro sve komponente tipičnog računala, a smještene su na pločici veličine bankovne kartice. Nastao je 2012. godine s ciljem stvaranja jeftinog uređaja koji će unaprijediti programske vještine i razumijevanje hardvera kod početnika u području računarstva. Spreman je obavljati sve standardne zadatke stolnih računala, uključujući pregledavanje interneta, reprodukciju visokokvalitetnih videozapisa, obradu teksta i igranja video igara. Raspberry Pi koristi operativni sustav Raspbian, koji je suštinski prilagođena i optimizirana verzija Debian distribucije Linux-a za Raspberry Pi.

Od originalnog Modela B do danas, objavljeno je nekoliko modela Raspberry Pi, svaki donoseći poboljšane specijfikacije ili značajke specifične za određenu upotrebu. Za potrebe ovoga rada koristit će se Raspberry Pi 4 Model B koji je predstavljen u lipnju 2019. godine. Specifikacije Raspberry Pi-a mogu se vidjeti u tablici 2.1.

Resurs	Raspberry Pi Model B
Procesor	Broadcom BCM2711, 1.8 GHz
RAM	4 GB LPDDR4 3200
Pohrana	32 GB MicroSD kartica
Mreža	Gigabit Ethernet, Wi-Fi, Bluetooth
Operacijski sustav	Raspberry Pi OS (Raspbian)
Napajanje	5V preko USB-C priključka
Dimenzije	286 mm × 122 mm × 23 mm

Tablica 2.1. Specifikacije Raspberry Pi 4 Model B uređaja



Slika 3.1. Raspberry Pi 4 Model B[16]

3.2. Kamera

Modul kamere opremljen je standardnom lećom od 70° i parom infracrvenih LED koji omogućuju kameri da vidi noću bez vidljivog snopa svjetlosti. Sama kamera ima senzor od 5 megapiksela i konektor standardne veličine. Kamera može snimati u maksimalnoj rezoluciji od 1080p pri 30 sličica u sekundi uz ručni fokus.

3.3. OpenCV

OpenCV [17] je biblioteka otvorenog koda za funkcije računalnog vida razvijen od strane Intel Corporation. OpenCV sadrži više od 2500 optimiziranih algoritama čime pruža korisnicima mogućnost da koriste i prilagode vlastiti kod. Biblioteka pruža korisnicima alate i funkcionalnosti za izvođenje različitih zadataka u računalnom vidu, uključujući identifikaciju i detekciju lica, prepoznavanje objekata, analizu ljudskih aktivnosti u videozapisima, praćenje pokretnih objekata te za mnoge druge svrhe. OpenCV zajednica ima više od 50 tisuća članova i njihova aplikacija je preuzeta više od 18 milijuna puta. Među glavnim korisnicima su tvrtke, istraživačke skupine, državne institucije i studenti koji primjenjuju OpenCV u različite svrhe.

3.4. Python

Python je objektno orijentirani programski jezik visoke razine čiji je tvorac Guido van Rossum, a prvi put je objavljen 1991. godine. Radi se o objektno orijentiranom jeziku koji naglašava čitljivost, jednostavnost i fleksibilnost te je sve popularniji među programerima zbog svoje jednostavnosti učenja i upotrebe. Python je interpretirani jezik, što znači da se izvorni kod obrađuje pri izvođenju, umjesto da se unaprijed kompajlira[18]. Koristi se u znanstvenim istraživanjima, analizi podataka i strojnom učenju u raznim industrijama poput zdravstva, financija i tehnologije. Zbog svoje široke dostupnosti, Python okuplja veliku zajednicu korisnika, što rezultira obimnom standardnom bibliotekom i brojnim modulima koji su na raspolaganju za upotrebu[19].

3.5. YOLOv4-tiny

YOLOv4-tiny [20] je sažeta verzija modela YOLOv4. Izgrađena je na temelju YOLOv4 modela kako bi se pojednostavila struktura mreže i smanjio broj parametara, što ga čini prikladnim za razvoj na mobilnim i ugrađenim uređajima. YOLOv4-tiny koristimo za brže treniranje i otkrivanje. Ima samo dvije YOLO glave, za razliku od tri u modelu YOLOv4, te je treniran iz 29 prethodno istreniranih konvolucijskih slojeva, za razliku od modela YOLOv4 koji je treniran iz 137 prethodno istreniranih konvolucijskih slojeva. Za otkrivanje objekata u stvarnom vremenu, YOLOv4-tiny je bolja opcija u usporedbi s YOLOv4, jer je brže vrijeme zaključivanja važnije od preciznosti ili točnosti kada se radi u okruženju otkrivanja objekata u stvarnom vremenu. Zbog svega navedenog, upravo ova verzija YOLO modela će se koristiti za izradu sustava.

4. PREGLED PROGRAMSKE IZVEDBE RJEŠENJA SUSTAVA

Na temelju prethodno opisanih teorijskih načela, u ovom poglavlju će se implementirati programsko rješenje. Za programski jezik odabran je Python zbog njegove široke primjene u području računalnog vida i strojnog učenja, kao i njegove velike popularnosti u akademskim krugovima. Za implementaciju algoritama specifičnih za računalni vid koristit će se OpenCV, otvorena biblioteka za računalni vid koja sadrži mnoge korisne metode i implementacije za manipulaciju digitalnih slika, izdvajanje značajki te detekciju objekata. Za matematičke operacije bit će korištena NumPy [21] biblioteka koja pruža visokoapstraktne implementacije matematičkih operacija nad nizovima i matricama i YOLO sustav za detekciju ljudi u prostoriji.

4.1. Implementacija potrebnih biblioteka i YOLO algoritma

```
import cv2
import numpy as np
import time
import os

# postavljanje putanja do datoteka koje su potrebne konfiguraciju i korištenje
YOLOv4-tiny modela za detekciju objekata
labelsPath = os.path.sep.join(["./yolo/coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")
weights_path = os.path.sep.join(["./yolo/yolov4-tiny.weights"])
config_path = os.path.sep.join(["./yolo/yolov4-tiny.cfg"])

# učitavanje YOLOv4-tiny modela i konfiguracije
net = cv2.dnn.readNet(weights_path, config_path)
```

Slika 4.1.1. Implementacija potrebnih biblioteka i putanje do YOLO datoteka

U prvom dijelu koda uvoze se potrebne biblioteke za obradu videa, za matematičke operacije nad nizovima i matricama, rad s vremenom i operacijskim sustavom. Također, postavljaju se putanje do datoteka koje sadrže informacije o klasama objekta i konfiguraciji YOLOv4-tiny modela. Nakon učitanih biblioteka i postavljenih putanja, sljedeći korak je učitavanje YOLO modela i konfiguracije pomoću funkcije ***cv2.dnn.readNet(weights_path, config_path).*** Metoda ***readNet*** iz biblioteke OpenCV učitava prethodno trenirane težine modela i konfiguraciju modela na temelju zadanih putanja. Varijbla ***net*** sadrži učitani model i konfiguraciju koje će se koristiti za detekciju objekta.

```
# definiranje klasa
classes = []
with open(labelsPath) as f:
    classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
```

Slika 4.1.2. Učitavanje klasa objekata i dohvaćanje naziva slojeva i izlaznih slojeva

U narednom dijelu koda se učitavaju informacije o klasama objekta i priprema se za izvlačenje izlaza iz određenih slojeva učitanog YOLOv4-tiny modela. Otvara se datoteka „*coco.names*“ na putanji ***labelsPath*** koja sadrži popis imena klasa objekta, uklanjaju se suvišni razmaci i znakovi za novi red, a zatim se nazivi klasa objekta spremaju u listu ***classes.*** U sljedećem koraku se dobijaju nazivi svih slojeva u modelu i nazivi izlaznih slojeva. Ove informacije o klasama objekata i slojevima bit će korisne prilikom detekcije i prikaza rezultata detekcije objekata.

4.2. Postavljanje kamere

```
# postavljanje kamere
video_capture = cv2.VideoCapture(0)
```

Slika 4.2.1. Postavljanje kamere

Postavlja se kamera za snimanje videa, odnosno za prikazivanje videozapisa iz stvarnog vremena sa kamere. U ovom slučaju, argument 0 se koristi kao ulazni argument, što označava da se koristi prva dostupna kamera na sustavu. Ako je dostupna više od jedne kamere, može se koristiti i indeks drugih kamera (npr. 1 za drugu kameru).

4.3. Detekcija i bilježenje broja ljudi u datoteku

```
# definiranje vremenskog intervala za zapis broja ljudi u datoteku
interval = 10  # zapis svakih 10 sekundi
start_time = time.time() # sprema trenutno vrijeme u varijablu start_time

while True:
    # učitavanje sljedećeg okvira videa
    ret, frame = video_capture.read()

    if not ret:
        break

    # definiranje veličine slike i prilagođavanje veličine
    height, width, channels = frame.shape
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
    net.setInput(blob)
    output_layers_names = net.getUnconnectedOutLayersNames()
    layer_outputs = net.forward(output_layers_names)

    class_ids = []
    confidences = []
    boxes = []
```

Slika 4.3.1 Obrada okvira za detekciju i detekcija objekata

U nastavku na prethodno objašnjeni kod dodaju se dodatni koraci za obradu okvira videa u detekciji ljudi. Postavlja se varijabla ***interval*** koja predstavlja vremenski interval u sekundama, u kojem će se bilježiti ljudi. Pomoćna varijabla ***start_time*** sprema trenutno vrijeme kada je petlja pokrenuta. Petlja se nastavlja izvršavati sve dok se sljedeći okvir videa uspješno učitava.

Varijable ***height***, ***width*** i ***channels*** dobivaju informacije o dimenzijama okvira. Metoda ***cv2.dnn.blobFromImage()*** pretvara okvir u objekt prikladan za ulaz u neuronsku mrežu. Nakon toga se postavlja pripremljeni okvir kao ulaz za YOLO model. Dohvaćaju se nazivi izlaznih slojeva modela koji će sadržavati predikcije objekata i izvršava se detekcija objekata. Rezultat su izlazni slojevi koji sadrže informacije o detektiranim objektima.

Varijable ***class_ids***, ***confidences*** i ***boxes*** su prazne liste koje će se koristiti za pohranu identifikatora klase, povjerenja i okvira detektiranih objekata.

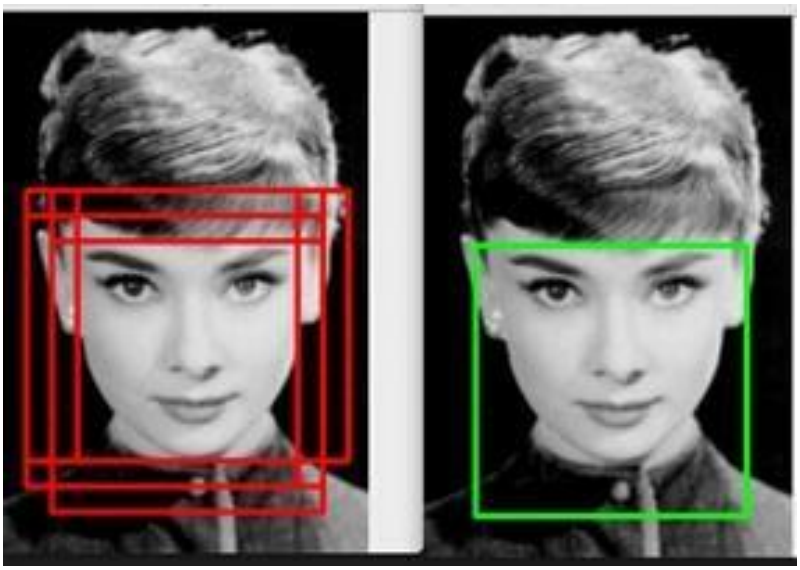
```
    # prolazak kroz sve izlazne slojeve i detektirane objekte
    for output in layer_outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

    # provjera da li je detektirani objekt čovjek s visokom pouzdanosti
            if class_id == 0 and confidence > 0.2:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    # primjena NMS algoritma za uklanjanje duplih detekcija
    indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.2, 0.3)

    # broj detektiranih osoba
    num_people = len(indices)
```

Slika 4.3.2. Prolazak kroz izlazne slojeve. Provjeravanje klase objekta. Primjena NMS algoritma

Petlja ***for output in layer_outputs*** prolazi kroz sve izlazne slojeve dobivene iz neuronske mreže. Unutar te petlje, petlja ***for detection in output*** prolazi kroz sve detekcije u pojedinom izlaznom sloju. Varijabla **scores** sadrži rezultate detekcije, a ***detection[5:]*** uzima rezultate za sve klase (indeksi 5 i dalje). Metodom ***np.argmax(scores)*** se određuje indeks najvišeg rezultata, što odgovara identifikatoru klase s najvećom pouzdanosti. Varijabla ***class_id*** dobiva identifikator klase s najvišim rezultatom detekcije. Varijabla ***confidence*** dobiva pouzdanost (rezultat) za identificiranu klasu. Uvjet ***if class_id == 0 and confidence > 0.5*** provjerava je li identificirana klasa čovjek (klasa s indeksom 0) i je li pouzdanost veća od 0.5. Ako je uvjet ispunjen, to znači da je detektirani objekt prepoznat kao čovjek s dovoljno visokom poozdanošću.

Varijable ***center_x***, ***center_y***, ***w*** *i* ***h*** računaju koordinate i veličinu okvira detektiranog objekta u odnosu na veličinu slike. Varijable ***x*** *i* ***y*** računaju gornji lijevi kut okvira detektiranog objekta. Koordinate i veličina okvira (***x***, ***y***, ***w***, ***h***) dodaju se u listu ***boxes***. Pouzdanost (***confidence***) se pretvara u decimalni broj i dodaje se u listu ***confidences***. Identifikator klase (***class_id***) dodaje se u

listu ***class_ids***. Metoda ***cv2.dnn.NMSBoxes()*** primjenjuje NMS algoritam [22] na okvire detektiranih objekata kako bi se uklonile duplicirane detekcije. Primjer NMS algoritma se može vidjeti na slici 4.3.3. Ulazni argumenti su liste ***boxes*** *i* ***confidences***, prvi prag je za suzbijanje preklapanja okvira (0.2) te drugi prag je za zadržavanje okvira (0.3). Metoda vraća indekse okvira koji su prošli NMS algoritam. Varijabla ***num_people*** dobiva duljinu liste ***indices***, što predstavlja broj detektiranih osoba nakon primjene NMS algoritma. Prolazi se kroz detektirane osobe i iscrtavaju se okviri oko svake osobe na slici, te se ispisuju koordinate svake detektirane osobe.



Slika 4.3.3. Primjer NMS Algoritma. Crveni granični okviri su prije primjene,a zeleni nakon primjene NMS algoritma

```
# iscrtavanje pravokutnih okvira oko detektiranih osoba (bounding box-ova) i
ispis koordinata osobe
    for i in range(num_people):
        if i in indices:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = confidences[i]
            color = (0, 255, 0)  # zelena boja

            # crtanje okvira
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

            # ispisivanje koordinata detektirane osobe
            print("Coordinates of the person: ", boxes[i])
```

Slika 4.3.4 Iscrtavanje okvira oko detektiranih ljudi i ispisivanje koordinata

Petlja ***for i in range(num_people)*** prolazi kroz broj detektiranih osoba. Uvjet ***if i in indices*** provjerava da li je trenutni indeks ***i*** prisutan u listi ***indices***, što osigurava da se iscrtaju samo okviri osoba koje su prošle NMS algoritam. Varijable ***x, y, w i h*** dobivaju koordinate i veličine okvira osobe iz liste ***boxes*** za trenutni indeks ***i***. Varijabla ***label*** dobiva ime klase osobe pretvoreno u string iz liste ***classes*** za trenutni indeks ***i***, kao i varijabla ***confidence*** koja dobiva pouzdanost detekcije osobe. Varijabla ***color*** definira boju okvira (u ovom slučaju, zelena boja). Metoda ***cv2.rectangle()*** crta okvir oko detektirane osobe na slici.

Metoda ***print()*** ispisuje koordinate detektirane osobe na konzolu iz varijable ***boxes[i]*** koja sadrži koordinate detektirane osobe.

```
    # provjera da li je prošlo dovoljno vremena za bilježenje
    elapsed_time = time.time() - start_time
    if elapsed_time >= interval:
        # zapisivanje broja ljudi i trenutnog vremena u .txt datoteku
        with open('pedestrian_count.txt', 'a') as f:
            f.write(f'Number of people: {num_people} - Time:
{time.ctime()}\n')
        start_time = time.time()

    # prikaz okvira videa
    cv2.imshow('People detection', frame)

    # prekid prikaza okvira videa
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
```

Slika 3.3.1 Bilježenje broja ljudi u datoteku

Varijabla ***elapsed_time*** dobiva vrijeme koje je proteklo od početka izvršavanja programa do trenutka izračunato kao razlika trenutnog vremena i vremena kada je započelo bilježenje. U uvjetu se provjerava je li proteklo vrijeme (***elapsed_time***) veće ili jednako zadanim intervalom (***interval***) za bilježenje. Ako je uvjet ispunjen, izvršava se sljedeći blok koda. Blok koda unutar prve if petlje izvršava zapisivanje broja detektiranih ljudi i trenutnog vremena u tekstualnu datoteku "*pedestrian_count.txt*". U tekstualnu datoteku se zapisuje tekst koji sadrži broj detektiranih ljudi (***num_people***) sa trenutnim vremenom. Nakon zapisivanja, ažurira se vrijeme početka bilježenja kako bi se započelo brojanje vremena od trenutka zapisivanja. Metoda ***cv2.imshow()*** se koristi za

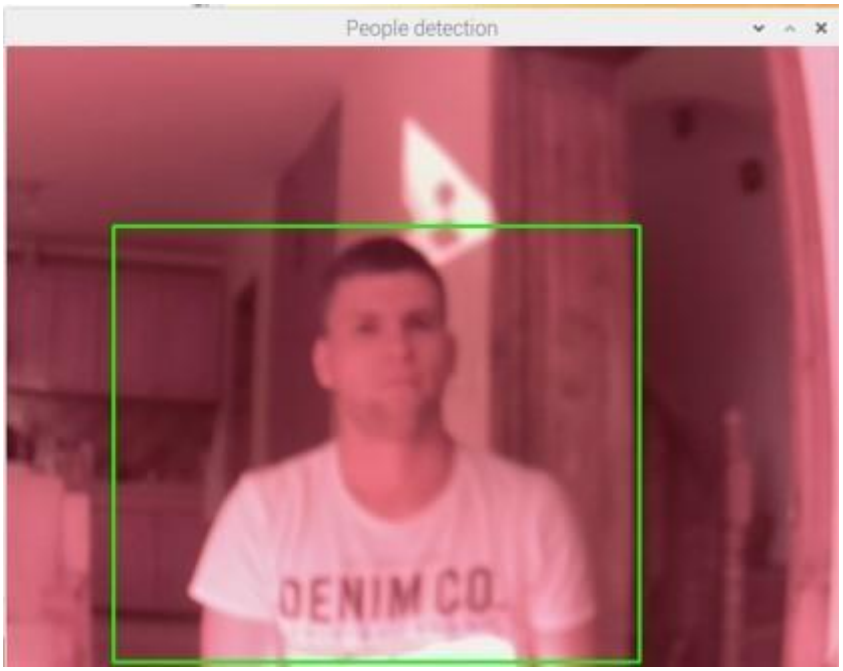
prikazivanje okvira videa sa detekcijom na ekranu. Prvi argument je naslov prozora, a drugi argument je slika (***frame***) koja sadrži iscrtane okvire i tekstove. Na kraju, uvjet **if** provjerava je li pritisnuta tipka 'q' na tipkovnici kako bi se prekinuo prikaz okvira videa.

5. TESTIRANJE SUSTAVA

Nakon programskog rješenja povezujemo potrebne komponente sustava. Povezujemo Raspberry Pi sa kamerom i pokrećemo sustav. Slika povezanog sustava je prikazana na slici 5.1.

Slika 5.1. Sustav za detekciju

Nakon instalacije svih potrebnih biblioteka pokreće se kod za detekciju ljudi. Kamera prikazuje sliku u stvarnom vremenu s označenim detekcijama objekata. Za svaku detekciju može se vidjeti pravokutni okvir (*boundig box*) oko ljudi i broj ljudi koji su detektirani u svakom trenutuku. Broj detektiranih ljudi u pojedinom trenutku određen intervalom, a zapisani su u datoteku *pedestrian_count.txt*. Rezultati detekcije mogu se vidjeti na slici 5.2. kao i zapis rezultata detekcije na slici 5.3.



Slika 5.2. Rezultat detekcije

pedestrian_count.txt - Mousepad

File Edit Search View Document Help

```
Number of people: 1 - Time: Wed Sep  6 16:16:13 2023
Number of people: 1 - Time: Wed Sep  6 16:16:25 2023
Number of people: 1 - Time: Wed Sep  6 16:16:39 2023
Number of people: 1 - Time: Wed Sep  6 16:16:50 2023
Number of people: 1 - Time: Wed Sep  6 16:17:02 2023
Number of people: 1 - Time: Wed Sep  6 16:17:14 2023
Number of people: 1 - Time: Wed Sep  6 16:17:26 2023
Number of people: 1 - Time: Wed Sep  6 16:17:38 2023
Number of people: 1 - Time: Wed Sep  6 16:17:50 2023
Number of people: 1 - Time: Wed Sep  6 16:18:02 2023
Number of people: 1 - Time: Wed Sep  6 16:18:14 2023
Number of people: 1 - Time: Wed Sep  6 16:18:25 2023
Number of people: 1 - Time: Wed Sep  6 16:18:37 2023
Number of people: 1 - Time: Wed Sep  6 16:18:49 2023
Number of people: 1 - Time: Wed Sep  6 16:19:01 2023
Number of people: 1 - Time: Wed Sep  6 16:19:13 2023
Number of people: 1 - Time: Wed Sep  6 16:19:25 2023
Number of people: 1 - Time: Wed Sep  6 16:19:35 2023
Number of people: 1 - Time: Wed Sep  6 16:19:47 2023
Number of people: 1 - Time: Wed Sep  6 16:19:59 2023
Number of people: 1 - Time: Wed Sep  6 16:20:11 2023
Number of people: 1 - Time: Wed Sep  6 16:20:23 2023
Number of people: 1 - Time: Wed Sep  6 16:20:35 2023
```

Slika 5.3. Tekstualna datoteka pedestrian_count sa brojem osoba u određenom trenutku

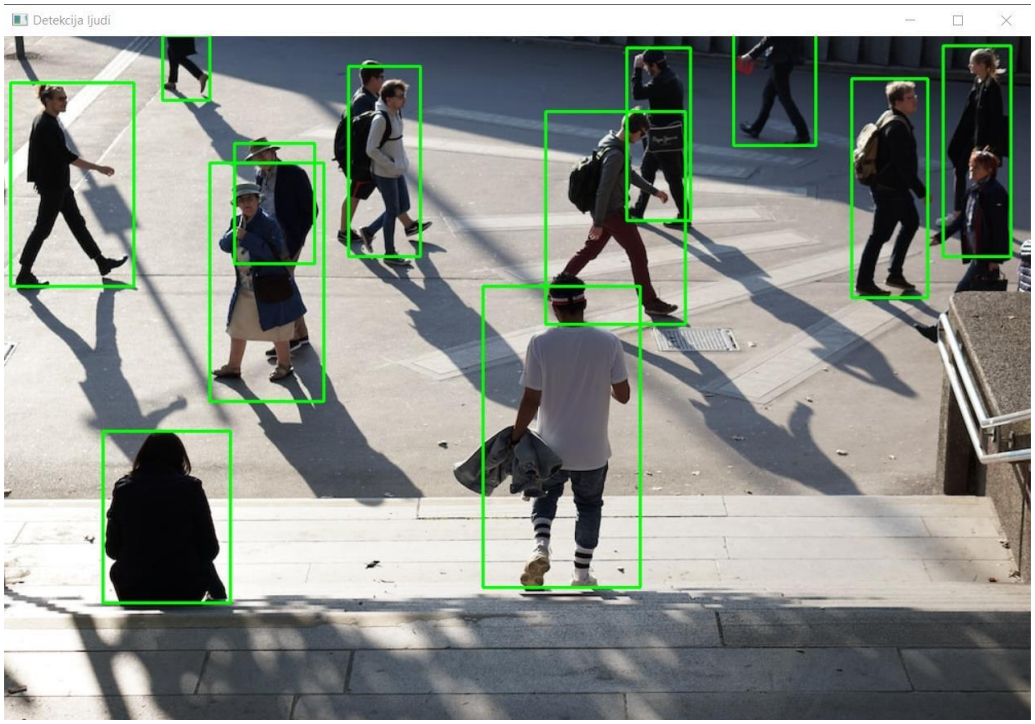
6. ANALIZA REZULTATA

U posljednjem dijelu rada, analizirat će se YOLO sustav na način da će se izvoditi eksperiment, kojom se ispituje točnost klasifikacije YOLO sustava za detekciju osoba. Na skupu od 10 slika za koje je poznat broj ljudi na svakoj slici uporediti će se sa brojem koje predvidi sustav. Ispitat će se rad sustava i donijeti zaključak o točnosti, preciznosti sustava i ponašanje u različitim scenama i vremenskim uslovima. Rezultati testiranja su u tablici 6.1.

Slika	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Stvarni broj osoba	4	13	14	6	7	6	11	9	19	7
Detektirani broj osoba	4	13	12	6	6	6	11	9	19	6

Tablica 6.1. Rezultati testiranja

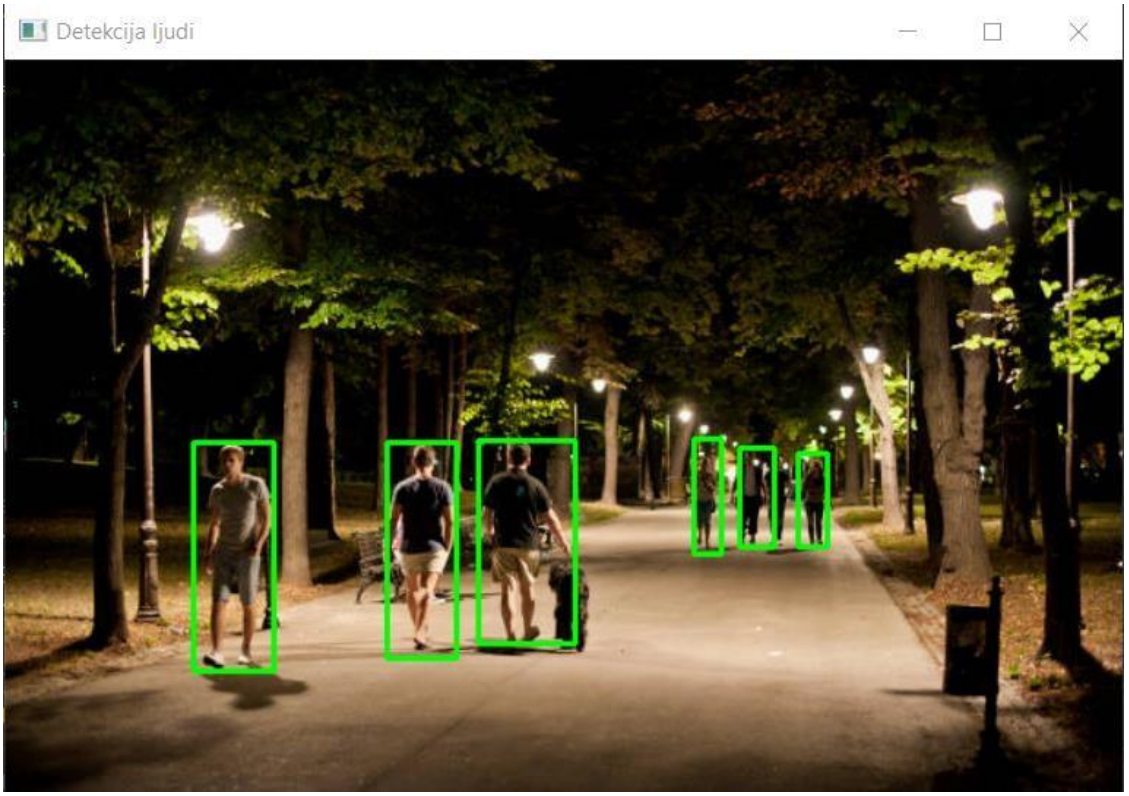
Srednja apsolutna pogreška na odabranom skupu slika iznosila je 0.4. Sustav se pokazao na odabranom skupu slika relativno precizan s nešto većom pogreškom. Na slikama visoke rezolucije i u odličnim uvjetima je detektirao većinom sve osobe na slici. U narednim primjerima prikazao bih nekoliko slučajeva iz testnog skupa slika gdje se algoritam nije baš najbolje prikazao.



Slika 6.1. Primjer testiranja YOLO sustava

U prvom primjeru na slici 6.1. sustav je prepoznao većinu osoba, ali u slučaju kada su dvije osobe jedna iza druge sama detekcija je pomalo izazovna jer sustav nekada ne prepozna osobu iza. Faktori koji na to mogu uticati su:

- Niska rezolucija slike. Ukoliko je kamera s ograničenim senzorom i niske rezolucije, sustav neće imati dovoljno informacija za preciznu detekciju.
- Veličina graničnih okvira. Ako se modeli preklapaju, model može imati problema s određivanjem granica svakog pojedinog objekta.
- Primjena NMS algoritma. Algoritam može imati poteškoće kada se objekti preklapaju u većoj mjeri.



Slika 6.2. Primjer testiranja YOLO sustava

U drugom primjeru na slici 6.2. u uvjetima slabog osvjetljenja i niske rezolucije, sustav je uspio detektirati većinu osoba na slici. Međutim, sustav čak i uz smanjene pragove za detekciju nije uspio prepoznati sve manje objekte na slici.

Na temelju primjera i ostalih slika sustav većinom ili detektira sve osobe na slici ili manji broj ljudi u odnosu na ukupan broj osoba na slici. U odličnim uvjetima, kada je osvjetljenje dobro i

kada je slika visoke rezolucije u većini slučajeva ne griješi. Sustav griješi kada je potrebno detektirati manje objekte i kada je slika manje rezolucije.

Rješenja koja mogu poboljšati samu detekciju:

- Povećanje rezolucije kamere.
- Korištenjem naprednijih modela. U ovom radu, zbog ograničenosti sustava, korištena je komprimirana verzija YOLOv4 modela, odnosno YOLOv4-tiny model. Novije arhitekture su optimizirane za složenije scenarije i uvjete koji će poboljšati točnost detekcije, ali time zahtijevaju i bolji računalni sustav.
- Prilagodba pragova za detekciju. Ukoliko sustav ne detektira određene objekte, postavljanjem nižih pragova omogućava se objektu da detektira objekte sa nižom poudzanosti. Važno je imati na umu da s time povećava šansa za lažne detekcije.

Za poboljšanje detekcije moguće je primijeniti različite strategije, važno ih je prilagoditi ovisno o specifičnim uvjetima projekta i prema scenariju upotrebe. Eksperimentiranje s različitim postavkama i praćenjem performansi sustava ključno je za postizanje najboljih rezultata.

7. ZAKLJUČAK

Cilj ovog rada je upoznavanje sa različitim metodama za detekciju objekata u području računalnog vida, objašnjavajući njihove prednosti i nedostatke. Detaljno je opisan rad konvolucijskih neuronskih mreža te njihova primjena u obradi slike. Kroz razvoj Python koda, demonstrirana je implementacija sustava za praćenje ljudi koristeći YOLO algoritam, te njegova praktična primjena na Raspberry Pi uređaju.

YOLO algoritam je pokazao točnost i brzinu obrade slika, omogućujući brzo i pouzdano praćenje osoba čak i u lošijim uvjetima. Integracija ovog algoritma s Raspberry Pi platformom pokazala se iznimno praktičnom, pružajući pristupačno i energetski učinkovito rješenje za praćenje i upravljanje ljudskim prisustvom. Sustav se pokazao iznimno učinkovitim u prepoznavanju i praćenju ljudi u stvarnom vremenu, čime je omogućio nadzor i analizu kretanja unutar zatvorenih prostora.

YOLO sustav pruža snažno rješenje za detekciju objekata, ali kako bi se maksimizirala njegova učinkovitost, potrebno je razumjeti njegove mane i primijeniti odgovarajuće strategije za poboljšanje točnosti i osjetljivosti prema specifičnim zahtjevima projekta.

Rad je pružio dublje razumijevanje tehnologije praćenja ljudi u zatvorenim prostorima i potencijalnih primjena ovog sustava u različitim sektorima, uključujući sigurnost, upravljanje kapacitetom i analizu kretanja ljudi.

LITERATURA

[1] „What Is Deep Learning? | How It Works, Techniques & Applications“. https://www.mathworks.com/discovery/deep-learning.html (pristupljeno 06. rujan 2023.).

[2] „What is a Deep Neural Network?“ https://www.oticon.com/blog/what-is-a-deep-neural-network-dnn (pristupljeno 06. rujan 2023.).

[3] „Convolutional neural network“, *Wikipedia*. 03. rujan 2023. Pristupljeno: 06. rujan 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1173687463

[4] S. Saha, „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“, *Medium*, 16. studeni 2022. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (pristupljeno 06. rujan 2023.).

[5] „FIGURE 3. Basic architecture of CNN.“, *ResearchGate*. https://www.researchgate.net/figure/Basic-architecture-of-CNN_fig3_335086346 (pristupljeno 07. rujan 2023.).

[6] „What are Convolutional Neural Networks? | IBM“. https://www.ibm.com/topics/convolutional-neural-networks (pristupljeno 06. rujan 2023.).

[7] P. Ratan, „What is the Convolutional Neural Network Architecture?“, *Analytics Vidhya*, 28. listopad 2020. https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/ (pristupljeno 07. rujan 2023.).

[8] „What is the difference between an activation function and pooling function used in convolutional neural networks (CNNs)? Why can’t we use...“, *Quora*. https://www.quora.com/What-is-the-difference-between-an-activation-function-and-pooling-function-used-in-convolutional-neural-networks-CNNs-Why-cant-we-use-one-single-function-to-perform-both-operations-at-once (pristupljeno 07. rujan 2023.).

[9] R. Girshick, J. Donahue, T. Darrell, i J. Malik, „Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation“.

[10] R. Gandhi, „R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms“, *Medium*, 09. srpanj 2018. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e (pristupljeno 06. rujan 2023.).

[11] R. Girshick, „Fast R-CNN“. arXiv, 27. rujan 2015. Pristupljeno: 07. rujan 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1504.08083

[12] F. Sultana, A. Sufian, i P. Dutta, „A Review of Object Detection Models based on Convolutional Neural Network“, 2020, str. 1–16. doi: 10.1007/978-981-15-4288-6_1.

[13] K. He, G. Gkioxari, P. Dollár, i R. Girshick, „Mask R-CNN“. arXiv, 24. siječanj 2018. Pristupljeno: 07. rujan 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1703.06870

[14] J. Redmon, S. Divvala, R. Girshick, i A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection“. arXiv, 09. svibanj 2016. Pristupljeno: 07. rujan 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1506.02640

[15] „What is a Raspberry Pi?“, *Raspberry Pi Foundation*. https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ (pristupljeno 06. rujan 2023.).

[16] C. App, „Channels — Raspberry Pi“. https://getchannels.com/raspberry-pi/ (pristupljeno 07. rujan 2023.).

[17] „About“, *OpenCV*. https://opencv.org/about/ (pristupljeno 07. rujan 2023.).

[18] A. K. Singh, „A Basic Process of Python Use for IOTAP, Data Science, and Rapid Machine Learning Model Development“, u *Fraud Prevention, Confidentiality, and Data Security for Modern Businesses*, IGI Global, 2023, str. 84–104. doi: 10.4018/978-1-6684-6581-3.ch004.

[19] T. P. Trappenberg, „Scientific programming with Python“, u *Fundamentals of Computational Neuroscience: Third Edition*, T. P. Trappenberg, Ur., Oxford University Press, 2022, str. 0. doi: 10.1093/oso/9780192869364.003.0002.

[20] Techzizou, „YOLOv4 VS YOLOv4-tiny“, *Analytics Vidhya*, 07. listopad 2021. https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec (pristupljeno 07. rujan 2023.).

[21] „NumPy - About Us“. https://numpy.org/about/ (pristupljeno 11. rujan 2023.).

[22] C. Bhalerao, „A Deep Dive into Non-Maximum Suppression[NMS]: Understanding the Math Behind Object Detection“, *MLearning.ai*, 21. ožujak 2023. https://medium.com/mlearning-ai/a-deep-dive-into-non-maximum-suppression-nms-understanding-the-math-behind-object-detection-765ff48392e5 (pristupljeno 11. rujan 2023.).

SAŽETAK

Ovaj završni rad istražuje razvoj i implementaciju udaljenog sustava za praćenje broja ljudi u zatvorenim prostorima primjenom tehnologije umjetne inteligencije. Temelj rada je korištenje YOLO algoritma konvolucijske neuronske mreže za preciznu detekciju i praćenje ljudi u stvarnom vremenu. Rad se također bavi usporedbom različitih metoda detekcije, uključujući R-CNN pristupe, kako bi se istaknule prednosti YOLO algoritma.

Osim toga, rad se fokusira na praktičnu primjenu sustava pomoću Raspberry Pi platforme, pristupačnog i energetski učinkovitog računalnog rješenja. Razvijen je Python kod za implementaciju sustava, omogućujući udaljeno praćenje ljudi putem Raspberry Pi uređaja.

Kroz ovaj rad, demonstrirano je kako umjetna inteligencija, posebno YOLO algoritam, može biti učinkovito korištena za rješavanje stvarnih problema u praćenju ljudi u zatvorenim prostorima. Integracija s Raspberry Pi platformom čini ovaj sustav pristupačnim i praktičnim za različite primjene, pridonoseći sigurnosti i učinkovitosti u različitim okruženjima.

Ključne riječi: detekcija objekata, YOLO, Raspberry Pi, konvolucijske neuronske mreže

ABSTRACT

Remote indoor people counting system using Raspberry Pi and image processing

This final thesis explores the development and implementation of a remote system for monitoring the number of people in enclosed spaces using artificial intelligence technology. The foundation of the work is the utilization of the YOLO convolutional neural network algorithm for precise real-time detection and tracking of people. The thesis also compares different detection methods, including R-CNN approaches, to highlight the advantages of the YOLO algorithm.

Furthermore, the thesis focuses on the practical application of the system using the Raspberry Pi platform, an affordable and energy-efficient computing solution. Python code has been developed for system implementation, enabling remote monitoring of people through Raspberry Pi devices.

Through this work, it is demonstrated how artificial intelligence, particularly the YOLO algorithm, can be effectively used to address real-world problems in monitoring people in enclosed spaces. Integration with the Raspberry Pi platform makes this system accessible and practical for various applications, contributing to safety and efficiency in different environments.

Keywords: object detection, YOLO, Raspberry Pi, convolutional neural networks

ŽIVOTOPIS

Domin Radić rođen je 26. veljače 2001. godine u Gradačcu, Bosna i Hercegovina. Osnovnu i srednju elektrotehničku školu završio je u svom rodnom gradu. Nakon srednje škole, 2020. godine, upisao je preddiplomski sveučilišni studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

_____

Potpis autora