Detekcija životinja u videotijeku u realnom vremenu

Jukić, Ivan

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

DETEKCIJA ŽIVOTINJA U VIDEOTIJEKU U REALNOM VREMENU

Završni rad br. 14/RAČ/2021

Ivan Jukić

Bjelovar, lipanj 2022.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Jukić Ivan** Datum: 29.09.2021. Matični broj: 001855

JMBAG: 0314017996

Kolegij: **INTERNET STVARI**

Naslov rada (tema): **Detekcija životinja u videotijeku u realnom vremenu**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Krunoslav Husak, dipl.ing.rač.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Ante Javor, struč.spec.ing.comp., predsjednik**
2. **Krunoslav Husak, dipl.ing.rač., mentor**
3. **Ivan Sekovanić, mag.ing.inf.et comm.techn., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 14/RAČ/2021

U ovom radu potrebno je izraditi sustav za detekciju životinja u videotijeku u realnom vremenu. Sustav mora ispravno prepoznati samo životinje u danom videotijeku te sve dohvaćene informacije pohraniti u bazu podataka i/ili poslati na centralno mjesto za daljnju obradu. U ovom radu će se izraditi potpuna programska podrška namijenjena za pokretanje na slabijim računalima poput Raspberry Pia. Također, potrebno je izraditi pozadinsko sučelje (API) koje će osigurati jednostavan dohvat informacija iz ovog sustava. Ovisno o dohvaćenim informacijama, sustav može automatski upravljati drugim procesima.

REPUBLIKA HRVATSKA 2 VELEUČILIŠTE U BJELOVARU

Zadatak uručen: 29.09.2021. Mentor: **Krunoslav Husak, dipl.ing.rač.**

Sadrža

1 UVOD

Tema ovog završnog rada je obrada informacija dobivenih putem video tijeka u stvarnom vremenu te obavještavanje korisnika o ponašanju životinje kako bi se olakšala briga o kućnim ljubimcima. Rad će pobliže opisati dijelove i programska rješenja sustava za detekciju životinja. Funkcija hardvera u sustavu je prikupljanje informacija o prostoru u kojem se nalazi te vršenje detekcije i prepoznavanje objekata. Ukoliko na slikama koje PiCamera (video kamera priključena na Raspberry Pi računalo) prenese sustavu TensorFlow, ljubimac (mačka ili pas) zadovolji neke od definiranih uvjeta unutar programskog koda, korisnik će zaprimiti obavijest koja opisuje akciju njegovog ljubimca. Sustav će također zabilježiti tu akciju ljubimca u bazu podataka. Web stranica sustava će omogućiti korisniku pregled svih pohranjenih informacija putem web stranice.

U poglavlju dva ovog rada opisani su svi dijelovi i funkcije sustava za detekciju životinja te je prikazana shematska povezanost, dok je platforma TensorFlow i način rada neuronskih mreža objašnjen u poglavlju tri.

Poglavlje četiri pojašnjava primjenu i način rada Twilio aplikacijskog programskog sučelja koje ima ključnu ulogu u obavještavanju korisnika o akcijama ljubimca. U poglavlju pet opisuju se baza podataka i njena implementacija i svrha u sustavu. U poglavlju šest opisan je programski jezik Python i njegov modul Flask koji je korišten za izradu web stranice i na posljetku u sedmom poglavlju opisan je cijeli programski kod web stranice.

Prijedlozi za daljnje unaprjeđenje dani su u zaključku odnosno u osmom poglavlju.

2 ARHITEKTURA SUSTAVA ZA DETEKCIJU ŽIVOTINJA

2.1 Hardver

Hardver je razvijen od strane The RaspberryPi britanske dobrotvorne kompanije osnovane 2009. godine čija je misija promoviranja Računarstva u školstvu. The RaspberryPi tvrtka je proizvela seriju modela mini računala, a za ovaj rad konkretno korišten je RaspberryPi 4 model B model (slika 2). Uz modele RaspberryPi-a također proizvode i razne dodatke koji su kompatibilni s mini računalom. Jedan od tih dodataka koji se koristi za ovaj završni rad je PiCamera (slika 2.3) koja ima ključnu ulogu prenošenja video tijeka na kojem se vrši detekcija objekata, u ovom slučaju životinja.

2.1.1 RaspberryPi

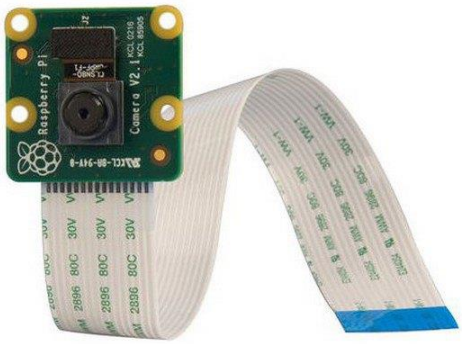
RaspberryPi kompaktno je računalo koje se spaja na računalni monitor ili TV i koristi standardnu tipkovnicu i miša. To je uređaj koji ljudima svih dobnih skupina omogućuje istraživanje računalstva i učenje programiranja koristeći se jezicima kao što su Scratch i Python. Poslužiti će kao stolno računalo za pretraživanje interneta i reprodukcije videa visoke razlučivosti do izrade proračunskih tablica, obrade teksta i igranja računalnih igara.

Štoviše, Raspberry Pi ima sposobnost interakcije s vanjskim svijetom i korišten je u širokom spektru projekata digitalnih proizvođača, od glazbenih strojeva i roditeljskih detektora do vremenskih postaja.

Slika 1:Raspberry Pi 4 Model B (Raspberry Pi Documentation)

2.1.2 Kamera

Raspberry Pi kamera priključuje se izravno na Raspberry Pi pomoću CSI konektora. Može isporučiti sliku rezolucije 5 MP ili snimanje HD videa od 1080p pri 30 fps. Modul se spaja na Raspberry Pi putem 15-pinskog vrpčanog kabela, na namjensko 15-pinsko MIPI serijsko sučelje kamere (CSI), koje je dizajnirano posebno za povezivanje s kamerama. CSI sabirnica je sposobna za visoke brzine prijenosa podataka i isključivo prenosi pikselne podatke do procesora BCM2835. Sama ploča je dimenzija 25 mm x 20 mm x 9 mm, te teži oko 3g što je čini pogodnom za mobilne ili druge aplikacije gdje su veličina i težina važni faktori.[19]

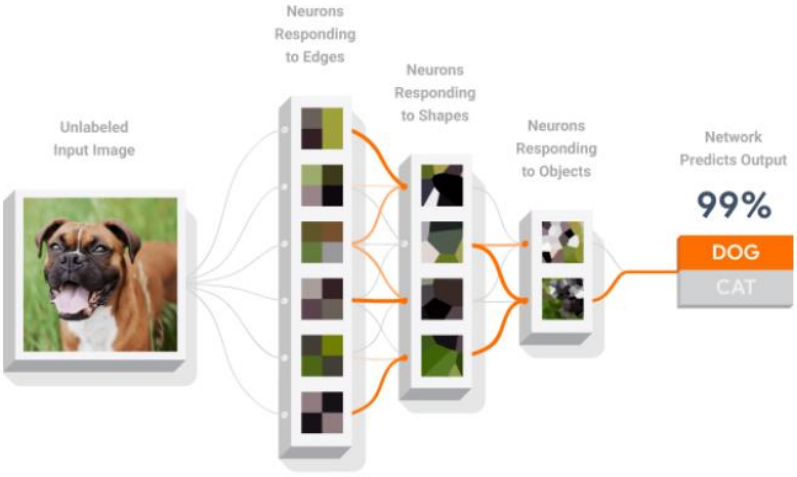


Slika 2.2:Hardver sustava za detekciju životinja (Raspberry Pi Documentation)

2.2 TensorFlow

TensorFlow je cjelovita platforma otvorenog koda (engl. Open source) za strojno učenje . Ima opsežan, prilagodljiv i fleksibilan ekosistem s puno različitih alata i biblioteka. TensorFlow su izvorno razvili istraživači i inženjeri koji rade u *Google Brain Group* timu unutar Google organizacije za istraživanje umjetne inteligencije. Sustav je dovoljno općenit da se može koristiti u mnogim drugim područjima i na raznim projektima koji uključuju detekciju objekata. TensorFlow sustav nudi stabilna Python i C++ API (engl. Application Programming Interface) sučelja. [12]

TensorFlow je dizajniran kako bi omogućio laku izradu vlastitih strojno naučenih modela uz pomoć API-a koji su vrlo jednostavni za korištenje i koji sadrže puno alata koji olakšavaju cijeli proces. Zbog svoje jednostavnosti i mogućnosti primjene u raznim sustavima TensorFlow koriste i velike kompanije kao što su: Intel, Lenovo, CocaCola, AirBnB, Spotify, Twitter itd.[12]



Slika 2.3:Arhitektura neuronske mreže

2.3 Twilio

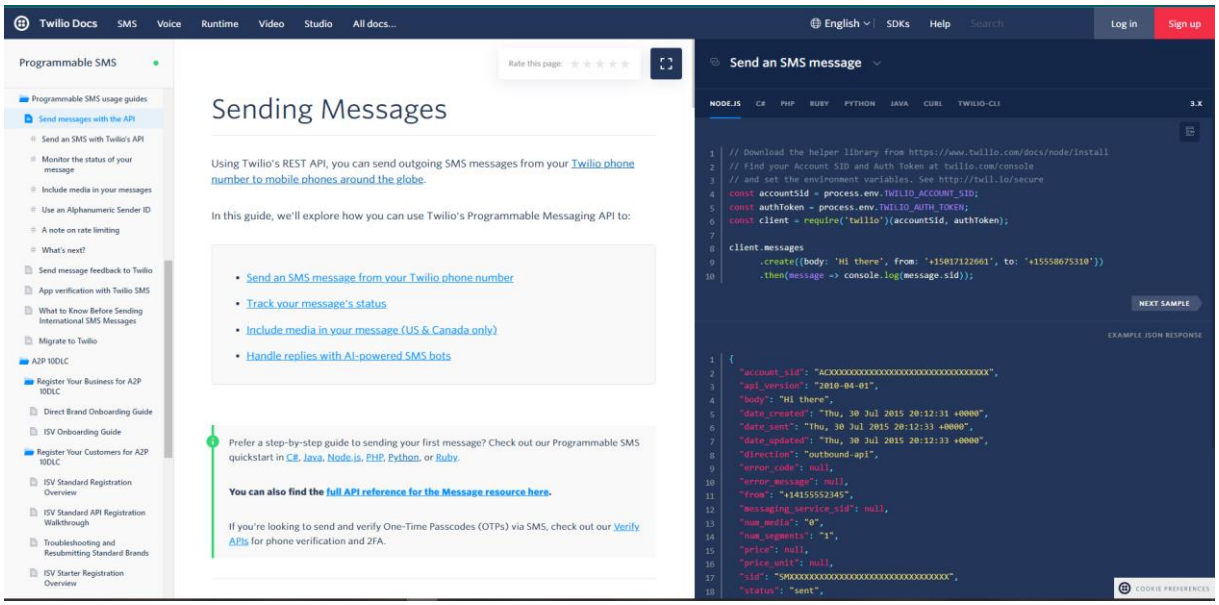
Twilio je američka tvrtka sa sjedištem u San Franciscu, Kalifornija, koja nudi programabilne komunikacijske alate za upućivanje i primanje telefonskih poziva, slanje i primanje tekstualnih poruka i obavljanje drugih komunikacijskih funkcija koristeći API svojih usluga. Twilio je servis koji se koristi za izradu automatske komunikacije između korisnika i

aplikacije. Twilio je izrazito napredan API koji omogućava raznoliku upotrebu ovog servisa. Usluge koje nudi Twilio koristi jako velik broj kompanija (preko 1300!), a neke od tih su i: Airbnb, Uber, Reddit, Glovo, Ebay itd. Može se koristiti za slanje SMS-a, WhatsApp poruka, glasovnih poruka, videa, e-pošte, pa čak i u sustavima interneta stvari. Sve što je potrebno je integrirati njegov API unutar vlastitog softvera.

2.3.1 Twilio API

API je računalno sučelje koje omogućuje interakciju između više softverskih posrednika. Aplikacijsko programsko sučelje definira vrste zahtjeva koji se mogu uputiti, kako ih uputiti, formate podataka koje treba koristiti, standarde koje treba primijeniti, itd. Također može pružiti mehanizme proširenja tako da korisnici mogu proširiti postojeće funkcionalnosti. Aplikacijsko programsko sučelje može se prilagođavati za specifične komponente sustava u raznim aplikacijama ili može biti dizajnirano na temelju industrijskog standarda kako bi se osigurala interoperabilnost. Kroz skrivanje informacija, aplikacijsko programsko sučelje podržava modularno programiranje što omogućava korisnicima korištenje aplikacijskog programskog sučelja neovisno o implementaciji.

Kako bi se koristio javan API potreban je API ključ koji se koristi za identifikaciju korisnika kao valjanog klijenta, postavljanje dopuštenja pristupa i bilježenje vaših interakcija s API-jem. Neki API čine svoje ključeve besplatno dostupnim, dok drugi zahtijevaju da korisnici za njih plate. U svakom slučaju je potrebna prijava za korištenje takvih usluga. Ništa drukčiji nije ni Twilio servis za koji se potrebno registrirati ukoliko korisnik želi njihove usluge. Za cijeli taj proces od registracije do implementacije Twilio ima izrazito dobru dokumentaciju koja je neizostavan dio svakog API-ja kako bi korisnici znali uputit zahtjeve. U dokumentaciji se obično opisuju metode aplikacijskog programskog sučelja, kako ih pozvati, koje podatke i kako ih poslati te kakav odgovor očekivati od aplikacijskog programskog sučelja.



Slika 2.4:Dokumentacija od Twilio API

2.4 Web stranica

Web stranica je razvijena u programskom jeziku Python u kombinaciji s Flask bibliotekom. Web stranica prikazuje podatke iz baze podataka koji su poslani od strane RaspberryPi-a. Web stranica u sustavu za detekciju životinja služi da omogući korisnicima uvid u detalje detekcija, te omogućava uređivanje odnosno uklanjanje zapisanih informacija o detektiranih životinjama.

2.4.1 Modul Flask

Flask je programski okvir (engl. Framework) Python programskog jezika. Dizajniran je kako bi razvoj web aplikacija bio učinkovit, uz mogućnost proširenja za izradu složenijih aplikacija. Nastao je kao jednostavan omot (eng. Wrapper) i razvio se u jedan od najpopularnijih okvira za izradu web aplikacija koristeći programski jezik Python. Koristeći Flask na programeru je da odabere alate i biblioteke koje želi koristiti. Zajednica nudi mnoga proširenja koja olakšavaju dodavanje novih funkcija. Flask ima ugrađeni web poslužitelj što olakšava testiranje i sam razvoj web aplikacije.

2.5 Baza podataka

Baza podataka implementirana je pomoću SQLitea. To nije samostalna aplikacija već je biblioteka koju se obično ugrađuje unutar aplikacije. Kao takva pripada pod ugrađene baze podataka. SQLite ima najveću primjenu kada se koristi manji skup podataka te se koristi kao privremeni skup podataka za obradu nekih podataka unutar aplikacije.

Baza podataka je često temeljni dio web aplikacija te općenito web sustava. U nastavku će biti opisana baza podataka koja se koristi u ovom završnom radu i način na koji je baza podataka implementirana u sustav. U sustavu za detekciju životinja baza podataka se koristi za spremanje informacija o detekciji ljubimca i pohranu registriranih korisnika sustava.

2.5.1 SQLite

SQLite je najčešće korištena baza podataka na svijetu. SQLite je ugrađen u sve mobilne uređaje i većinu računala. Dolazi u paketu s bezbroj drugih aplikacija koje ljudi koriste svakog dana. Može upravljati HTTP zahtjevima niskog do srednjeg prometa. Samo ime sugerira da se radi o lakšoj verziji SQL RDBMS-a. SQLite dolazi u paketu s Pythonom i može se koristiti u bilo kojoj Python aplikaciji bez potrebe za instaliranjem dodatnog softvera.

2.5.1.1 Općenito

SQLite je biblioteka C-jezika koja implementira mali, brz, samostalan, visokopouzdani, potpuno opremljen, SQL softver baze podataka. Za razliku od većine drugih SQL baza podataka, SQLite nema zaseban poslužiteljski proces. SQLite čita i piše izravno na obične datoteke na disku. Kompletna SQL baza nalazi se u jednoj datoteci na disku. Format datoteke baze podataka je višeplatformski - može se kopirati između 32-bitnih i 64-bitnih sustava. Ove značajke čine SQLite popularnim izborom za pohranu podataka unutar aplikacija.

2.6 Programski jezik Python

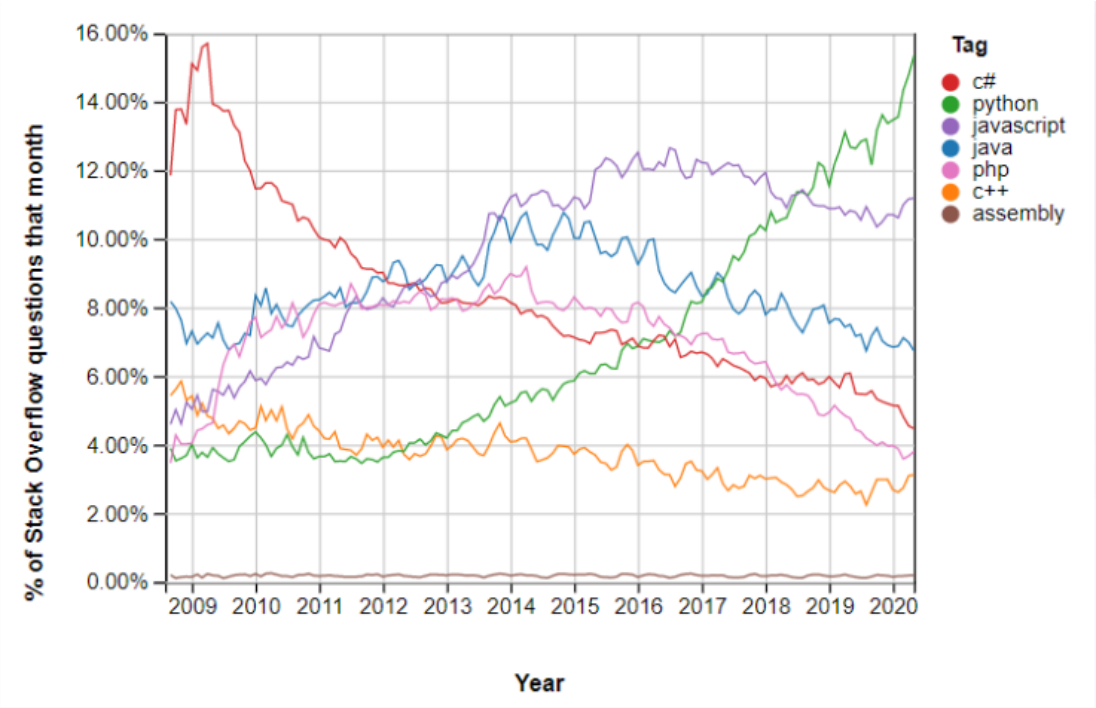
Python je interpretirani, objektno orijentirani, programski jezik visoke razine s dinamičkom semantikom koji je razvio Guido van Rossum. Prvobitno je objavljen 1991. Osmišljen da bude jednostavan i zabavan, naziv `Python` je znak britanske komičarske grupe Monty Python. Python ima reputaciju jezika prilagođenog početnicima, zamjenjujući Javu kao najčešće korišteni uvodni jezik jer rješava veći dio složenosti za korisnika, dopuštajući početnicima da se usredotoče na potpuno razumijevanje programskih koncepata, a ne na sitne detalje. Python se koristi za web razvoj na strani poslužitelja, razvoj softvera, matematiku i skriptiranje sustava, a popularan je za brzi razvoj aplikacija i kao jezik skriptiranja ili za povezivanje postojećih komponenti. Dodatno, Pythonova podrška modulima i paketima olakšava modularne programe i ponovnu upotrebu koda. Python je jezik zajednice otvorenog koda, tako da brojni neovisni programeri neprestano grade knjižnice (engl. libraries) i funkcionalnosti za njega. Python se brzo penje na čelo najpopularnijih programskih jezika na svijetu.[5]

2.6.1 Općenito

Kao što je već spomenuto Python ima aktivnu zajednicu entuzijasta koji svaki dan nastoje poboljšati jezik popravljajući greške i izrađujući nove mogućnosti. Također uživa snažnu podršku najvećih svjetskih korporacija. Jedan od njih je Google koji aktivno radi na vodičima, uputama i drugim resursima kako bi izvukli maksimum iz Pythona. Osim Googlea, Python također koriste mnoge poznate kompanije koje pružaju softverske usluge kao što su Instagram, Reddit, Spotify itd. Razlozi zbog kojeg se sve veće kompanije odluče za korištenje Pythona su navedene u nastavku.

- Python je izrazito popularan i ima vrlo veliku zajednicu

Kako imati koristi od popularnosti Pythona? Korištenjem popularnog jezika puno veće su šanse pronaći rješenje za bilo koji problem. Zapravo, ako je problem dovoljno čest, vjerojatno je trenutno dostupno gotovo rješenje u Pythonu.



Slika 2.5:Prikaz popularnosti Pythona na najpoznatijoj stranica za programere

- Pisanje Python koda je lagano i čitko što izrazito ubrzava proces razvoja programa

Okruženje prilagođeno korisniku u rukama razvojnog tima obično znači puno manje vremena u pronalasku rješenja problema i pronalaska pravih alata za izgradnju željenog ishoda projekta. Velika prednost je i bogat izbor okvira (engl. frameworks) i knjižnica (engl. libraries). Oni naveliko štede vrijeme i ubrzavaju izlaz proizvoda na tržište.

- Čitanje Python koda je intuitivno, što olakšava održavanje

Pythonova sintaksa je jasna i sažeta. Jezik je dizajniran tako da bude čitljiv i blizak stvarnom engleskom što ga čini lakim za razvoj aplikacija. Python također zahtijeva manje redaka koda za postizanje rezultata u usporedbi s jezicima kao što su C ili Java.

- Python daje isprobanu skalabilnost

Nitko zapravo ne može predvidjeti kada će broj korisnika početi rasti i kada će skalabilnost postati prioritet. Zbog toga je dobra ideja koristiti jezik koji se odlično skalira i kao što je već spomenuto, jednostavan je za održavanje.

Neki od najambicioznijih projekata diljem weba: YouTube, Reddit i EVE Online koriste Python kako bi pouzdano služili svojoj korisničkoj bazi.

- Python je sve češće izbor u IoT (engl.Internet of things) projektima, a to tržište je budućnost

Python je prenosiv, proširiv i ugradiv. To čini Python neovisnim o sustavu i omogućuje mu podršku za mnoga jednoslojna računala koja su trenutno dostupna na tržištu, bez obzira na arhitekturu ili operativni sustav. Njegova kompaktnost i čista sintaksa su poželjni za male uređaje s ograničenom memorijom i snagom.

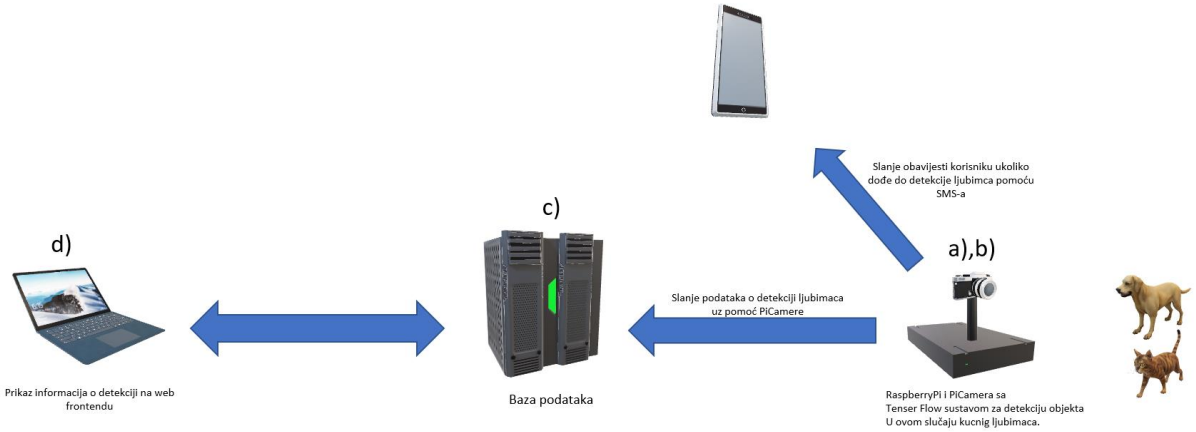
Tema završnog rada je vezana uz kolegiji Internet stvari koji je u meni naveliko probudio interes za radom u tom području. Vidjevši mogućnosti koje su ostvarive uz pomoć povezivanja različitih senzora i uređaja putem Interneta i drugih komunikacijskih mreža.

Sustav za detekciju životinja u kombinaciji s web aplikacijom implementira baš sve interesantne aspekte samog pojma IoT-a (engl. Internet of things). Od prikupljanja i obrade informacija putem mini računala (RaspberryPi i PiCamere) do pohranjivanja tih istih podataka u bazu podataka i prikazivanja na web stranici.

Sve fukcionalnosti baze podataka, web stranice, programskog koda za detekciju objekata i obavještavanje korisnika ostvarene su upotrebom programskog jezika Python i pripadajućih proširenja, konkretno Flask razvojnog okvira. Flask je bio najznačajniji u izradi web stranice koji je omogućio jednostavnu implementaciju sustava za registraciju i prijavu korisnika uz interakciju s bazom podataka.

Python unaprijed dolazi instaliran na sustavu Raspbian. Raspberry Pi kompanija je odabrala Python kao glavni jezik isključivo zbog njegove snage, svestranosti i jednostavnosti korištenja te zbog popularnosti što dodatno potpomaže rastu primjene Raspberrya i Pythona u IoT industriji koja je u velikom usponu i razvoju.

2.7 Prikaz arhitekture



Slika 2.6:Shematski prikaz sustava za detekciju životinja

Sustav za detekciju životinja razvijen je u svrhu završnoga rada. Ideja ovog rada je da uz pomoć kamere (PiCamere) u kombinaciji sa sustavom za detekciju objekata korisnik bude obaviješten putem SMS poruke svaki put kada njegov ljubimac želi ući ili izaći iz kuće. Rad se sastoji od sljedećih dijelova:

- a) Hardver

Hardver u ovom slučaju su RaspberryPi i PiCamera. PiCamera ima značajnu ulogu jer pokretanjem skripte (spremljene na RaspberryPi-u) pokrenut će se analiza videotijeka u realnom vremenu na kojem će se vršiti detekcija i prepoznavanje objekata.

- b) TensorFlow

TensorFlow u ovom radu ima ključnu ulogu jer upravo taj sustav analizira svaku sličicu video tijeka koju zaprima sa PiCamere. Ukoliko sustav prepozna na video tijeku psa ili mačku, te se zadovolje neki od definiranih uvjeta unutar skripte, ona će spremiti podatke o detekciji te informirati korisnika o aktivnostima ljubimca.

- c) Baza Podataka

Baza podataka u ovom sustavu služi za spremanje svih detekcija ljubimaca i omogućavanje korisniku uvid u te zapise. Također omogućava spremanje podataka o svakom korisniku koji se registrira za korištenje web stranice.

- d) Web stranica

Web stranica u ovom radu služi korisniku za detaljan uvid u spremljene detekcije te također omogućava uklanjanje istih.

3 IMPLEMENTACIJA RJEŠENJA

U nastavku rada objašnjeno je kako rade, koja je zadaća te kako su implementirani svi dijelovi prethodno navedeni u arhitekturi sustava za detekciju životinja.

3.1 Implementacija TensorFlowa u sustavu za detekciju životinja

Sustav za detekciju životinja implementira TensorFlow tako što skripta na RaspberryPiu uživo analizira videotijek koristeći PiCameru. Skripta ujedno i prosljeđuje svaki trenutak tog video tijeka u obliku sličica funkciji u kojoj je napisana cijela logika korištenja TensorFlow sustava za detekciju objekata. Nakon što je sličica proslijeđena, TensorFlow uspoređuje i provjerava jesu li objekti sa sličice prepoznatljivi tako što ih uspoređuje s njemu poznatim modelima. Sustav za detekciju životinja neće reagirati niti na jedan detektiran objekt ili biće osim mačke ili psa. Ukoliko se pas ili mačka nalaze na određenom broju (u ovom slučaju 15) uzastopnih sličica koje su proslijeđene funkciji u kojoj TensorFlow vrši detekciju, skripta će korisniku poslati obavijest na mobitel putem SMS poruke, te upisati taj podatak u bazu podataka iz koje će se sve detekcije prikazati na web stranici.

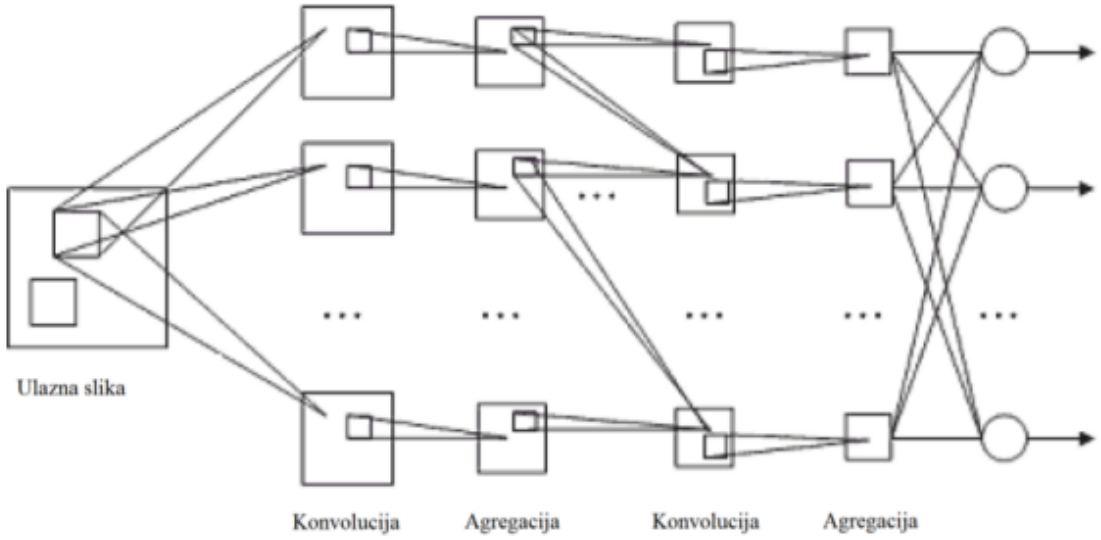
3.1.1 Treniranje neuronske mreže

Neuronske mreže treniraju se gradijentnim spuštanjem. Parametri u svakom sloju počinju sa slučajnim vrijednostima, a oni se iterativno poboljšavaju tijekom vremena kako bi mreža bila točnija. Funkcija gubitka koristi se za određivanje koliko je mreža netočna, a postupak se zove *backpropagation* što zapravo znači vraćanje unazad kroz neuronsku mrežu. Time se određuje treba li gradijent povećati ili smanjiti kako bi se smanjio gubitak odnosno netočnost same mreže.[10]

Slika 3.1:Prikaz rada neuronska mreže

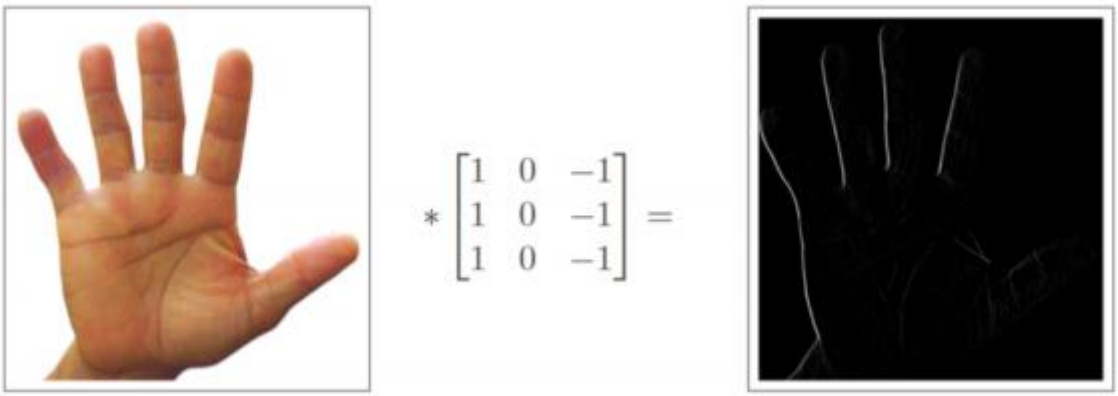
Postoji više tipova neuronskih mreža, ali one kojima je namjena obrađivanje slika, zvuka ili slično nazivaju se *Konvolutivne neuronske mreže.* Zasnovane su na operaciji konvolucije koja se koristi u obradi signala. U tradicionalnoj obradi signala ljudi su konstruirali tzv. filtere. Filteri predstavljaju matrice koji kada u operaciji konvolucije primijenimo na neki signal možemo doći do informacija poput gdje se nalaze rubovi na slici i slično. Ključna promjena je da konvolutivne mreže nauče filtere tako da rade ono što je približno najkorisnije u problemu koji se rješava. Jedna od njihovih glavnih prednosti je da iz podataka automatski konstruiraju atribute koji su relativni u problemu koji se rješava. Tako recimo da mreža primi fotografiju na obradu, na ulazu se dakle daju pikseli, kako idemo kroz slojeve mreže, prvi slojevi detektiraju jednostavne atribute kao što su rubovi. Sljedeći nivo mreže može, polazeći od onoga što je prethodni sloj detektirao, u terminima rubova izraziti konture. Idući nivo mreže može prepoznati nekakve dijelove objekata na osnovu prethodno prepoznatih kontura. Na posljednjem nivou, na osnovu dijelova objekata, na kraju može klasificirati pronađeni objekt.

- Konvolutivni filteri od ulazne slike daju različite nove slike koje predstavljaju različite aspekte ulazne slike odnosno njene atribute.
- Na svaku od dobivenih slika se primjenjuje agregacija tako da se slika smanji, a informacija se poveća (agregira).
- Nakon toga na temelju smanjene slike možemo ponovno primijeniti konvolutivne filtere i ponovno dobijemo nove slike.
- Na kraju dolazimo do finalnih slika malih dimenzija. Svaka slika je jedna matrica.



Slika 3.2:Shema konvolutivne mreže

Primjenu konvolucije možemo promatrati kao traženje nečega što liči na filter. Što je sličnost dijela slike i filtera veća, to će i vrijednost konvolucije biti veća.



Slika 3.3:Ulazna slika proširena filterom

3.1.2 Programski kod za detekciju ljubimaca

Sustav za detekciju životinja zapravo se sastoji od 3 velike cjeline: koda koji vrši detekciju ljubimaca i to upisuje u bazu te šalje obavijest korisniku, web aplikacije i baze podataka.

Skripta u kojoj se odvija inicijalizacija kamere i detekcija objekata se zove `Object_detection`, a napisana je u programskom jeziku Python. U nastavku su prikazani i objašnjeni dijelovi koda `Object_detection` skripte.

Prvo program inicijalizira Tensorflow i učitava modele za detekciju u memoriju sustava.

```
PATH_TO_CKPT =
os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')

# Number of classes the object detector can identify
NUM_CLASSES = 90

## Load the label map.
# Label maps map indices to category names, so that when the convolution
# network predicts `5`, we know that this corresponds to `airplane`.
# Here we use internal utility functions, but anything that returns a
# dictionary mapping integers to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.compat.v1.Session(graph=detection_graph)
```

Programski kod 1:Funkcija koja inicijalizira TensorFlow i učitava modele u memoriju

Zatim je potrebno inicijalizirati PiCameru koja će konstantno zaprimati sličice video tijeka. Svaka sličica koja je zaprimljena iz tog video tijeka na PiCameri proslijeđena je funkciji `pet_detector` kao atribut „frame“. Razlog zbog kojeg su sve sličice proslijeđene funkciji je zato što ona sadrži svu logiku.

```
### Picamera ###
if camera_type == 'picamera':
    # Initialize Picamera and grab reference to the raw capture
    camera = PiCamera()
    camera.resolution = (IM_WIDTH,IM_HEIGHT)
    camera.framerate = 10
    rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
    rawCapture.truncate(0)

    for frame1 in camera.capture_continuous(rawCapture,
format="bgr",use_video_port=True):
        print("Picamera")
        t1 = cv2.getTickCount()
        # Acquire frame and expand frame dimensions to have shape: [1,
None, None, 3]
        # i.e. a single-column array, where each item in the column has the
pixel RGB value
        #frame = frame1.array
        frame = np.copy(frame1.array)  #Line of code that deals with numpy
issue
        frame.setflags(write=1)
        # Pass frame into pet detection function
        frame = animal_detector(frame)
        # Draw FPS
        cv2.putText(frame,"FPS:
{0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)
        # All the results have been drawn on the frame, so it's time to
display it.
        cv2.imshow('Object detector', frame)

        #FPS calculation
        cv2.putText(frame,"FPS:
{0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)

        t2 = cv2.getTickCount()
        time1 = (t2-t1)/freq
        frame_rate_calc = 1/time1

        # Press 'q' to quit
        if cv2.waitKey(1) == ord('q'):
            break

        rawCapture.truncate(0)

    camera.close()
```

Programski kod 2: Funkcija za inicijalizaciju PiCamere

Unutar `pet_detector` funkcije sličica (engl. frame) je proslijeđena u TensorFlow neuronsku mrežu. Primjer neuronske mreže na: (slika 3.7).

```
def animal_detector(frame):


   # Use globals for the control variables so they retain their value after
function exits
    global detected_inside, detected_outside
    global inside_counter, outside_counter
    global pause, pause_counter

    frame_expanded = np.expand_dims(frame, axis=0)

    # Perform the actual detection by running the model with the image as
input
    (boxes, scores, classes, num) = sess.run(

        [detection_boxes, detection_scores, detection_classes,
num_detections],

        feed_dict={image_tensor: frame_expanded})
```

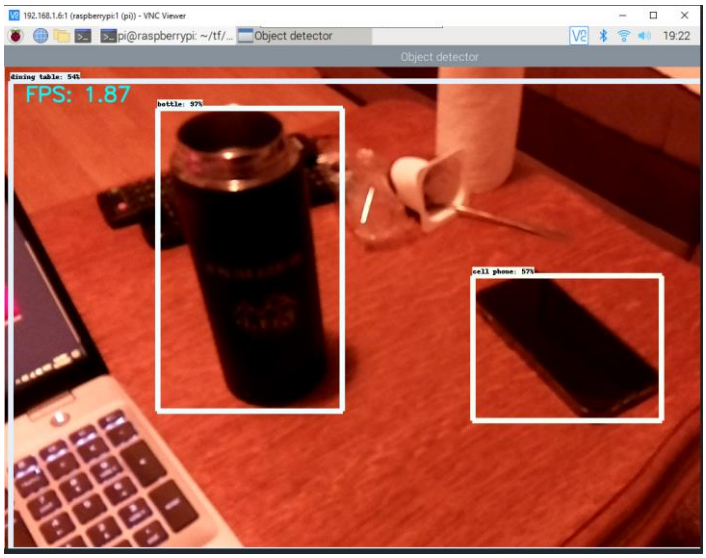
Programski kod 3: Dio skripte u kojoj je sličica proslijeđena neuronskoj mreži TensorFlow-a

Neuronska mreža je zadužena za detekciju objekata i pridruživanje tih objekata modelima s kojim je upoznata. Jednom kada je objekt detektiran mreža prikazuje odnosno označuje njihovu lokaciju na tom video tijeku.

```
# Draw the results of the detection (aka 'visulaize the results')

    vis_util.visualize_boxes_and_labels_on_image_array(
        frame,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8,
        min_score_thresh=0.40)
```

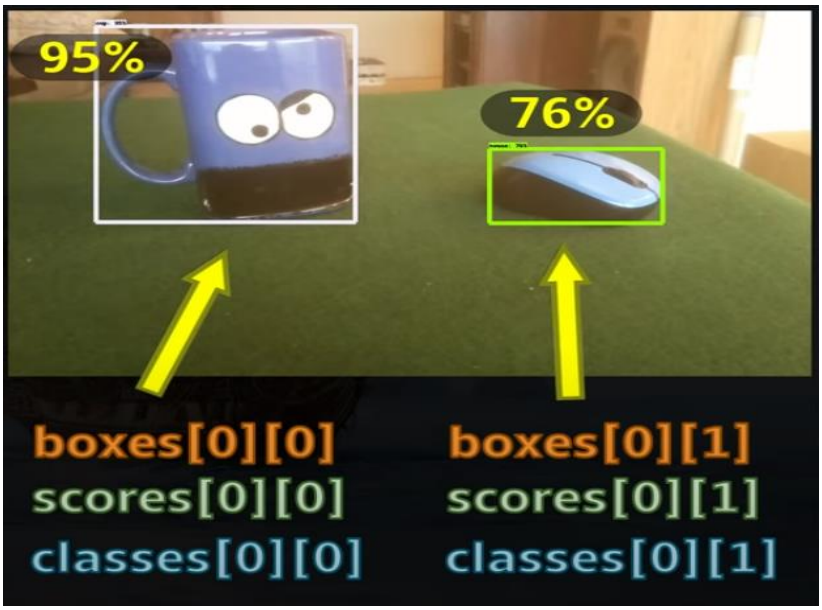
Programski kod 4: Dio koja koji prikazuje lokaciju objekta na video tijeku



Slika 3.4:Detekcija i označavanje objekta na video tijeku

Svaki detektirani objekt posjeduje 3 atributa: lokaciju, postotak sigurnosti i ime klase odnosno modela. Ta 3 atributa su definirana u 3 varijable: `boxes`, `scores` i `classes`.

- `boxes` varijabla sadrži XY koordinate svakog pravokutnika koji označuje detektiran objekt.
- `scores` varijabla označava sigurnost (engl. confidence) neuronske mreže, odnosno koliko je sigurna da je taj objekt zapravo to što je prepoznala.
- `classes` varijabla sadrži ID objekta koji je prepoznat odnosno ID modela.



Slika 3.5:Prikaz označavanja detektiranih objekata od strane neuronske mreže

Za svaku sličicu kod provjerava jesu li detektirani objekti na slici pas ili mačka uz pomoć `classes` varijable čija bi vrijednost trebala biti 17 (mačka) ili 18 (pas). Ako niti jedan od detektiranih objekata nisu pas ili mačka, program će ignorirati sve ostale detekcije i čeka iduću sličicu koja će mu biti proslijeđena.

```
    if (((int(classes[0][0]) == 17) or (int(classes[0][0] == 18) or
(int(classes[0][0]) == 47))) and (pause == 0)):

        x = int(((boxes[0][0][1]+boxes[0][0][3])/2)*IM_WIDTH)

        y = int(((boxes[0][0][0]+boxes[0][0][2])/2)*IM_HEIGHT)
```

Programski kod 5:Dio koda koji provjerava klasu detektiranog objekta

Ukoliko objekt koji detektiran je pas ili mačka, TensorFlow određuje točnu lokaciju objekta na slici uspoređujući XY vrijednosti `boxes` varijable i dimenzija cijele slike. Na temelju tih vrijednosti izračunava se točna pozicija detektiranog objekta. Zatim program provjerava nalazi li se centar detektiranog objekta unutar ili izvan unaprijed definiranih kvadrata koji označavaju prostor s unutarnje i vanjske strane vrata. Lokacija tih kvadrata se može mijenjati ovisno o kutu postavljanja kamere u odnosu na vrata.

Ukoliko je ljubimac detektiran u kvadratu s unutarnje strane vrata brojač će se povećati za 1. Isto vrijedi i za vanjski kvadrat.

```
# Draw boxes defining "outside" and "inside" locations.

    cv2.rectangle(frame,TL_outside,BR_outside,(255,20,20),3)

    cv2.putText(frame,"Outside box",(TL_outside[0]+10,TL_outside[1]-
10),font,1,(255,20,255),3,cv2.LINE_AA)

    cv2.rectangle(frame,TL_inside,BR_inside,(20,20,255),3)

    cv2.putText(frame,"Inside box",(TL_inside[0]+10,TL_inside[1]-
10),font,1,(20,255,255),3,cv2.LINE_AA)
```

Programski kod 6:Dio koda kojim su definirani kvadrati koji označavaju prostor ispred i iza vrata

```
       # Draw a circle at center of object
        cv2.circle(frame,(x,y), 5, (75,13,180), -1)
        # If object is in inside box, increment inside counter variable
        if ((x > TL_inside[0]) and (x < BR_inside[0]) and (y >
TL_inside[1]) and (y < BR_inside[1])):
            print("Object is inside box")
            inside_counter = inside_counter + 1
        # If object is in outside box, increment outside counter variable
        if ((x > TL_outside[0]) and (x < BR_outside[0]) and (y >
TL_outside[1]) and (y < BR_outside[1])):
            print("Object is outside box")
            outside_counter = outside_counter + 1
    # If pet has been detected inside for more than 10 frames, set
detected_inside flag
    # and send a text to the phone.
```

Programski kod 7:Dio koda koji provjerava stoji li ljubimac vani ili unutra

Ukoliko se ljubimac zadrži u jednom od tih kvadrata duže od određenog broja sličica tada će korisnik biti obaviješten o tome putem Twilio servisa kojemu će biti poslana SMS poruka. Poruka će korisnika informirati da njegov ljubimac želi ući ili izaći iz kuće ovisno o tome u kojem kvadratu se ljubimac zadržao, odnosno o njegovoj poziciji u odnosu na vrata.

Osim obavijesti korisnika o detekciji ljubimca također će se i izraditi zapis u bazi podataka koji će biti prikazan na web aplikaciji koja onda omogućava korisniku uvid u sve detekcije.

```
            outside_counter = outside_counter + 1
    # If pet has been detected inside for more than 10 frames, set
detected_inside flag
    # and send a text to the phone.
    if inside_counter > 15:
        detected_inside = True
        message = client.messages.create(

            body = 'Your pet wants outside!',

            from_=twilio_number,

            to=my_number

            )
        print("Inside counter je preko 10")
        insert_detect('Your Pet wants inside!')
        inside_counter = 0
        outside_counter = 0
        # Pause pet detection by setting "pause" flag
        pause = 1

    # If pet has been detected outside for more than 10 frames, set
detected_outside flag
    # and send a text to the phone.
    if outside_counter > 10:
        detected_outside = True
        message = client.messages.create(

            body = 'Your pet wants inside!',

            from_=twilio_number,

            to=my_number

           )
        print("Outside counter je preko 10")
        insert_detect('Your Pet wants outside!')
```

Programski kod 8:Dio koda koji obavještava korisnika i sprema zapis o detekciji u bazu podataka

Jednom kada je korisniku poslana poruka i zapis o detekciji je upisan u bazu podataka, program pauzira detekciju na određeno vrijeme odnosno na broj sličica (engl. frames) koje se proizvoljno mogu povećati ili smanjiti.

```
  # If pause flag is set, draw message on screen.
    if pause == 1:
        if detected_inside == True:
            print("Pet wants iside")
            cv2.putText(frame,'Pet wants
out!',(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,3,(0,0,0),7,cv2.LINE_AA)


        if detected_outside == True:
            print("Pet wants outside")
            cv2.putText(frame,'Pet wants
in!',(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,3,(0,0,0),7,cv2.LINE_AA)
            cv2.putText(frame,'Pet wants
in!',(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,3,(95,176,23),5,cv2.LINE_AA)

        # Increment pause counter until it reaches 30 (for a framerate of
1.5 FPS, this is about 20 seconds),
        # then unpause the application (set pause flag to 0).
        pause_counter = pause_counter + 1

        if pause_counter > 30:
            pause = 0
            pause_counter = 0
            detected_inside = False
            detected_outside = False

    return frame
```

Programski kod 9:Dio programskog koda koji pauzira detekciju

3.2 Implementacija Twilio servisa u sustavu za detekciju životinja

Twilio servis u sustavu za detekciju životinja ima ključnu ulogu kod obavještavanja korisnika o detekciji njegovog ljubimca. Implementacija same funkcionalnosti, zahvaljujući dokumentaciji, je izrazito jednostavna. Način na koji Twilio šalje poruku korisniku je da prilikom detekcije ljubimca, uz pomoć TensorFlowa i zadovoljenih uvjeta unutar skripte, koristimo funkciju *client.messages.create* za čije korištenje je prvo potrebno definirati pa proslijediti varijable s osobnim informacijama s Twilio računa uz broj mobitela na koji će obavijest, odnosno SMS poruka biti poslana.

```
from twilio.rest import Client

account_sid = os.environ['TWILIO_ACCOUNT_SID']
auth_token = os.environ['TWILIO_AUTH_TOKEN']
my_number = os.environ['MY_DIGITS']
twilio_number = os.environ['TWILIO_DIGITS']

client = Client(account_sid,auth_token)
```

```
  # If pet has been detected inside for more than 10 frames, set
detected_inside flag
    # and send a text to the phone.
    if inside_counter > 15:
        detected_inside = True
        message = client.messages.create(

            body = 'Your pet wants outside!',

            from_=twilio_number,

            to=my_number

            )
        print("Inside counter je preko 10")
        insert_detect('Your Pet wants inside!')
```

Slika 3.6:Postavljanje i korištenje Twilio API-ja u sustavu za detekciju životinja

3.3 Implementacija Baze podataka u sustavu za detekciju životinja

SQLite naredbe se nalaze unutar Python koda, odnosno unutar koda web aplikacije koja je napisana uz pomoć Flask web okvira (eng. Framework).

U nastavku je opisana implementacija SQLite baze podataka u sustavu za detekciju životinja i način na koji radi.

3.3.1 Model baze podataka

Izrada modela baze podataka odnosno svih tablica i njihovih međusobnih veza te izrada potrebnih atributa tablice (poput primarnih i vanjskih ključeva), kako bi baza podataka bila funkcionalna, definirana je u `models.py` datoteci.

Sustav za detekciju životinja koristi samo dvije tablice. `Detect` i `User` gdje tablica Detect služi za spremanje svih detekcija koje TensorFlow proslijedi, a tablica User sprema registracijske podatke svih korisnika. Tablice su povezane, tako da svaki korisnik vidi samo detekcije koje su spremljene uz njegov specifičan `ID` što znači da ne može vidjeti detekcije drugih registriranih korisnika.

```
from email.policy import default
from . import db
from flask_login import UserMixin
from sqlalchemy.sql import func


class Detect(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    data = db.Column(db.String(10000))
    date = db.Column(db.DateTime(timezone=True), default=func.now())
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), default = 1)



class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))
    first_name = db.Column(db.String(150))
    detects = db.relationship('Detect')
```

Programski kod 10:Modeli baze podataka

3.3.2 Upravljanje podacima

Za manipulaciju tablicama i podacima unutar njih web aplikacija koristi delete i insert naredbe, ali također upravlja i sa POST i GET HTTP zahtjevima. Stranice koje prikazuje web aplikacija su izrađene kao predložak (eng. template). Glavni predložak se zove „base.html“ koji je baza za sve ostale i njega ostali predlošci nasljeđuju. Primjerice stranica `sign-up.html` nasljeđuje stil i sve elemente (npr. navigacijsku traku) od `base.html`.

Sign-up.html predložak je stranica na kojoj korisnik stvara korisnički račun. Ovdje korisnik unosi podatke koji su prikazani na stranici da su potrebni (ime, prezime, e-mail...) te nakon pritiska na tipku potvrde šalje HTTP POST zahtjev koji se prosljeđuje `sign_up` metodi definiranoj u `auth.py`.

Metoda `sign_up` zadužena je za obradu POST i GET zahtjeva sa sign-up.html stranice. U njoj su definirane razni uvjeti koji moraju biti zadovoljeni kako bi traženi parametri za registraciju bili valjani. Također se vrši provjera u bazi podataka za uneseni e-mail, kako bi se spriječila registracija s istom e-mail adresom ukoliko već postoji. Kada su uvjeti zadovoljeni, kreiran je novi korisnički račun.

```
@auth.route('/sign-up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        email = request.form.get('email')
        first_name = request.form.get('firstName')
        password1 = request.form.get('password1')
        password2 = request.form.get('password2')

        user = User.query.filter_by(email=email).first()
        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(first_name) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = User(email=email, first_name=first_name,
password=generate_password_hash(
                password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('views.home'))

    return render_template("sign_up.html", user=current_user)
```

Programski kod 11:Sign_up funkcija za registraciju novog korisnika

Svaki zapis detekcije koji je spremljen u bazu podataka je također prikazan na home.html stranici. Zbog filtriranja i omogućavanja korisniku da ukloni detekcije koje ne želi vidjeti na toj stranici, dodan je i gumb s kojim se svaka detekcija zasebno može ukloniti iz

baze podataka. Ova funkcionalnost je realizirana uz pomoć JavaScript funkcije i prikazana je u paragrafu (Programski kod 5) . Prosljeđuje točan zapis detekcije funkciji koja taj zapis uklanja iz baze podataka. Što je vidljivo u paragrafu (Programski kod 6).

```
function deleteNote(noteId) {
  fetch("/delete-note", {
    method: "POST",
    body: JSON.stringify({ noteId: noteId }),
  }).then((_res) => {
    window.location.href = "/";
  });
}
```

Programski kod 12:JavaScript funkcija

```
@views.route('/delete-note', methods=['POST'])
def delete_note():
    note = json.loads(request.data)
    noteId = note['noteId']
    note = Detect.query.get(noteId)
    if note:
        if note.user_id == current_user.id:
            db.session.delete(note)
            db.session.commit()

    return jsonify({})
```

Programski kod 13:Funkcija za uklanjanje odabranog zapisa iz baze podataka

Najvažniji podaci koje baza podatka posjeduje su oni o detekciji ljubimca. Te podatke baza sprema uz pomoć funkcije `insert_detect` koja je pozvana unutar `Object Detection` skripte razvijene uz pomoć TensorFlowa. Svaki spremljen zapis prikazuje opis željene kretnje ljubimca, u ovom slučaju želi li ljubimac ući ili izaći, uz točan zapis vremena kada se to desilo.

```
def insert_detect(data):

        #user_id=current_user

        BASE_DIR = os.path.dirname(os.path.abspath(__file__))
        db_path = os.path.join(BASE_DIR, "database.db")
        db.session= sqlite3.connect(db_path)

        cursor = db.session.cursor()
        print("Connected to SQLite")

        sqlite_insert_with_param = """INSERT INTO Detect
                          (data,date,user_id)
                          VALUES (?,CURRENT_TIMESTAMP,1);"""

        data_tuple = [data]
        cursor.execute(sqlite_insert_with_param, data_tuple)
        db.session.commit()
        print("Python Variables inserted successfully into Detect table")
        cursor.close()

        if db.session:
            db.session.close()
            print("The SQLite connection is closed")
```

Programski kod 14:Funkcija za unos podataka o detekciji ljubimca u bazu podataka

3.4 Programski kod Web stranice

Web aplikacija koristi bazu podataka u svrhu spremanja informacija o novim korisnicima, odnosno za njihovu registraciju. Također, baza podataka se koristi i u svrhe spremanja i prikazivanja svih detekcija ljubimaca koje se dogode. Svaki registriran korisnik ima omogućen pristup `home` stranici koja prikazuje sve podatke o detekcijama ljubimaca.

U funkciji koja inicijalizira web servis prilikom svakog pokretanja definirane su varijable koje su potrebne da bi se spojili na željenu bazu podataka. Prije svega da bi se to omogućilo uopće potrebno je korištenje SQLAlchemy proširanja Flaska koja pojednostavljuje interakciju aplikacije s bazom podataka (Programski kod 1). Zatim se izvršava provjera postoji li uopće baza podataka na koju bi se web servis trebao spojiti (Programski kod 2).

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from os import path
from flask_login import LoginManager

db = SQLAlchemy()
DB_NAME = "database.db"


def create_app():
    app = Flask(__name__)
    app.config['SECRET_KEY'] = 'secret'
    app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{DB_NAME}'
    db.init_app(app)

    from .views import views
    from .auth import auth

    app.register_blueprint(views, url_prefix='/')
    app.register_blueprint(auth, url_prefix='/')

    from .models import User, Detect

    create_database(app)
```

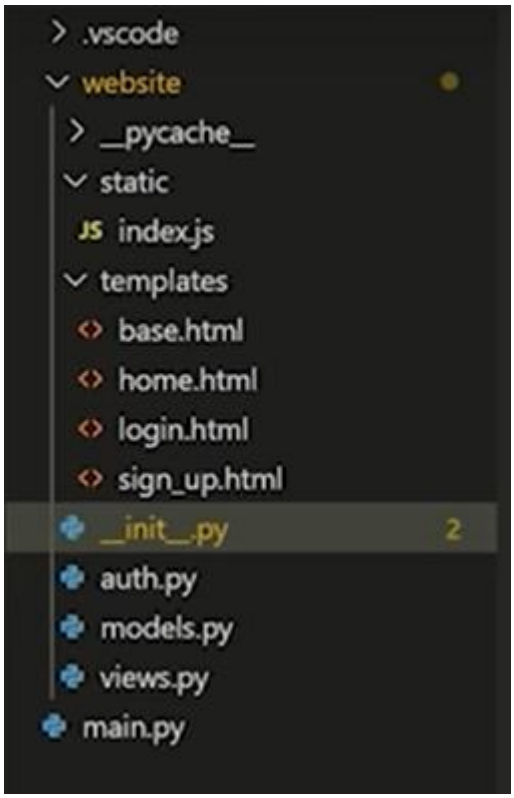
Programski kod 15:Funkcija koja inicijalizira web servis

```
def create_database(app):
    if not path.exists('website/' + DB_NAME):
        db.create_all(app=app)
        print('Created Database!')
```

Programski kod 16:Funkcija za provjeru postojanja baze podataka

3.4.1 Struktura web stranice

U ovome poglavlju opisatno je stablo mapa i datoteka koje omogućavaju rad i realizaciju funkcionalnosti web stranice. Na slici (7.1) prikazana je struktura direktorija.



Slika 3.7:Struktura web stranice

Ukoliko je sve u redu s programskim kodom pokretanjem `main.py` datoteke, pokrenuti će se lokalni server na kojem će raditi web stranica.

Ta datoteka se ne sastoji od ničeg drugog osim funkcije `create_app` naslijeđene iz paketa/direktorija `website`. Razlog zbog kojeg možemo tako pokrenuti web stranicu je programski kod datoteke `__init__.py` unutar mape `website` koja zapravo kreira cijelu flask aplikaciju.

```
from website import create_app

app = create_app()

if __name__ == '__main__':
    app.run(debug=True)
```

Programski kod 17:main.py datoteka

Kao što je već spomenuto `__init.py` datoteka je zapravo srce cijele web aplikacije. U njoj se provjerava stvaranje baze podataka, adresa stranica, učitavanje korisnika itd.

```
def create_app():
    app = Flask(__name__)
    app.config['SECRET_KEY'] = 'secret'
    app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{DB_NAME}'
    db.init_app(app)

    from .views import views
    from .auth import auth

    app.register_blueprint(views, url_prefix='/')
    app.register_blueprint(auth, url_prefix='/')

    from .models import User, Detect

    create_database(app)

    login_manager = LoginManager()
    login_manager.login_view = 'auth.login'
    login_manager.init_app(app)

    @login_manager.user_loader
    def load_user(id):
        return User.query.get(int(id))

    return app
```

Programski kod 18:__init_.py datoteka

Unutar mape `templates` se nalaze definirane html datoteke koje su zapravo web stranice. Princip na kojem funkcionira cijeli direktorij je taj da je baza svih stranica datoteka `base.html`, a nju nasljeđuju sve ostale .html stranice. Unutar bazne stranice definiran je općenit izgled web stranica te stil koji će koristiti, također vrši se i provjera korisnika koji je na stranici prijavljen ili ne kako bi se limitirao sadržaj koji se prikazuje. Jedino što se razlikuje na svim ostalim web stranicama je sadržaj koji one prikazuju.

```
<title>{% block title %}Home{% endblock %}</title>
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbar"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbar">
        <div class="navbar-nav">
          {% if user.is_authenticated %}
          <a class="nav-item nav-link" id="home" href="/">Home</a>
          <a class="nav-item nav-link" id="logout"
href="/logout">Logout</a>
          {% else %}
          <a class="nav-item nav-link" id="login" href="/login">Login</a>
          <a class="nav-item nav-link" id="signUp" href="/sign-up">Sign
Up</a>
          {% endif %}
        </div>
      </div>
    </nav>

    {% with messages = get_flashed_messages(with_categories=true) %} {% if
    messages %} {% for category, message in messages %} {% if category ==
    'error' %}
    <div class="alert alert-danger alter-dismissable fade show"
role="alert">
      {{ message }}
      <button type="button" class="close" data-dismiss="alert">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    {% else %}
    <div class="alert alert-success alter-dismissable fade show"
role="alert">
      {{ message }}
      <button type="button" class="close" data-dismiss="alert">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    {% endif %} {% endfor %} {% endif %} {% endwith %}
```

Programski kod 19:Dio programskog koda base.html-a koji limitira sadržaj neprijavljenim korisnicima

Preostali predlošci web stranica unutar `templates` mape su sljedeći:

- home.html stranica, kao što je već opisano u poglavlju baze podataka, je zapravo jedina stranica koja pruža korisne informacije korisniku. Na njoj se ispisuju sve detekcije koje su spremljene u bazi podataka. Uz te podatke korisniku je uz pomoć jednostavne JavaScript funkcije omogućeno filtriranje, odnosno uklanjanje odabranih zapisa.

```
{% extends "base.html" %} {% block title %}Home{% endblock %} {% block content
%}
<h1 align="center">Detects</h1>
<ul class="list-group list-group-flush" id="detects">
  {% for detect in user.detects %}
  <li class="list-group-item">
    {{ detect.data }}

    {{ detect.date }}
    <button type="button" class="close" onClick="deleteNote({{ detect.id }})">
      <span aria-hidden="true">&times;</span>
    </button>
  </li>
  {% endfor %}
</ul>

<form method="POST">
  <textarea name="note" id="note" class="form-control"></textarea>
  <br />
  <div align="center">
    <button type="submit" class="btn btn-primary">Add Note</button>
  </div>
</form>

{% endblock %}
```

Programski kod 20:home.html stranica

```
function deleteNote(noteId) {
  fetch("/delete-note", {
    method: "POST",
    body: JSON.stringify({ noteId: noteId }),
  }).then((_res) => {
    window.location.href = "/";
  });
}
```

Programski kod 21:JavaScript kod za uklanjanje zapisa o detekciji

- `login.html` stranica služi za prijavu korisnika za pristup `home.html` stranici. Na ovoj stranici se vrši provjera korisničkih podataka. Za unošenje krivih podataka korisnik će biti obaviješten malom porukom o tome.

```
{% extends "base.html" %} {% block title %}Login{% endblock %} {% block content
%}
<form method="POST">
  <h3 align="center">Login</h3>
  <div class="form-group">
    <label for="email">Email Address</label>
    <input
      type="email"
      class="form-control"
      id="email"
      name="email"
      placeholder="Enter email"
    />
  </div>
  <div class="form-group">
    <label for="password">Password</label>
    <input
      type="password"
      class="form-control"
      id="password"
      name="password"
      placeholder="Enter password"
    />
  </div>
  <br />
  <button type="submit" class="btn btn-primary">Login</button>
</form>
{% endblock %}
```

Programski kod 22:login.html stranica

`sign_up.html` stranica omogućuje registraciju korisnika kako bi isti dobio pristup podatcima s `home.html` stranice. Na ovoj stranici korisnik također unosi podatke koji su potrebni, ukoliko zadovoljavaju uvijete, podatci su spremljeni u bazu podataka i korisniku je omogućen pristup.

```
{% extends "base.html" %} {% block title %}Sign Up{% endblock %} {% block
content %}
<form method="POST">
  <h3 align="center">Sign Up</h3>
  <div class="form-group">
    <label for="email">Email Address</label>
    <input
      type="email"
      class="form-control"
      id="email"
      name="email"
      placeholder="Enter email"
    />
  </div>
  <div class="form-group">
    <label for="firstName">First Name</label>
    <input
      type="text"
      class="form-control"
      id="firstName"
      name="firstName"
      placeholder="Enter first name"
    />
  </div>
  <div class="form-group">
    <label for="password1">Password</label>
    <input
      type="password"
      class="form-control"
      id="password1"
      name="password1"
      placeholder="Enter password"
    />
  </div>
  <div class="form-group">
    <label for="password2">Password (Confirm)</label>
    <input
      type="password"
      class="form-control"
      id="password2"
      name="password2"
      placeholder="Confirm password"
    />
  </div>
  <br />
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
{% endblock %}
```

Programski kod 23:sign_up.html stranica

Svaka od ovih stranica prilikom učitavanja ili pritiska na jednu od funkcionalnosti stvara HTTP zahtjev, bilo da se radi o GET ili POST zahtjevu, što s tim zahtjevima odlučuje se u `auth.py` datoteci u kojoj su podržani zahtjevi sa `login.html` i `sign_up.html` stranica i u datoteci `views.py` koja manipulira HTTP zahtjevima sa „home.html“ stranice.

Uvoženjem (engl. import) flaskovog dodatka `Blueprint` omogućeno je definirati funkciju kojoj je pridružena jedna od stranica putem rute. Te funkcije u ovom radu manipuliraju HTTP zahtjevom koji dolazi s pridružene joj stranice.

```
views = Blueprint('views', __name__)
@views.route('/', methods=['GET', 'POST'])
@login_required
def home():
    if request.method == 'POST':
        note = request.form.get('note')

        if len(note) < 1:
            flash('Note is too short!', category='error')
        else:
            new_note = Detect(data=note, user_id=current_user.id)
            db.session.add(new_note)
            db.session.commit()
            flash('Detect added!', category='success')

    return render_template("home.html", user=current_user)
```

Programski kod 24:Dio koda views.py datoteke koji manipulira get i post zahtjevima s home.html stranice

Pritiskom na gumb `Login` na `Login.html` stranici korisnik kreira HTTP zahtjev POST koji pokreće login() funkciju koja zapravo provjerava postojanje korisnika u bazi podataka i na temelju toga dopušta ili ne prijavu korisnika.

```
@auth.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        user = User.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password, password):
                flash('Logged in successfully!', category='success')
                login_user(user, remember=True)
                return redirect(url_for('views.home'))
            else:
                flash('Incorrect password, try again.', category='error')
        else:
            flash('Email does not exist.', category='error')

    return render_template("login.html", user=current_user)
```

Programski kod 25:Dio auth.py koda koji omogućava prijavu korisnika

Također isto vrijedi za `sing_up.html` stranicu koja kreira isto kreira POST zahtjev koji omogućuje korisniku kreaciju računa.

```
@auth.route('/sign-up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        email = request.form.get('email')
        first_name = request.form.get('firstName')
        password1 = request.form.get('password1')
        password2 = request.form.get('password2')

        user = User.query.filter_by(email=email).first()
        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(first_name) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = User(email=email, first_name=first_name,
password=generate_password_hash(
                password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('views.home'))

    return render_template("sign_up.html", user=current_user)
```

Programski kod 26:Dio auth.py koda koji omogućuje registraciju korisnika

4 ZAKLJUČAK

Tema ovog završnog rada je zaprimanje podataka putem video tijeka u stvarnom vremenu, obrada tih podataka koristeći neuronske mreže te pohrana rezultata u bazu podataka. Završni rad je inspiriran i temelji se na kolegiju Internet stvari. Ciljevi ovog završnog rada su sljedeći: opis hardvera koji je korišten za prikupljanje informacija o prostoru u kojem se nalazi baziranog na korištenju PiCamere implementirane na RaspberryPi mikro računalo, opis platforme TensorFlow i načina na koji ona funkcionira u svrhu obrade informacija koje su proslijeđene od strane hardvera odnosno PiCamere, opis i izrada baze podataka potrebne za pohranu svih obrađenih podataka, opis rada i korištenja Twilio aplikacijskog programskog sučelja i načina na koji je implementirano i opis i izrada grafičkog korisničkog sučelja u obliku web aplikacije za prikaz prikupljenih podataka o detekcijama s PiCamere.

Pri izradi sustava korišteni su: RaspberryPi mikro računalo sa PiCamerom, Web aplikacija napisana uz pomoć modula Flask, SQLite baza podataka, Programski jezik Python i znanja i istraživanja odrađena na kolegijima Web programiranje, Baze podataka, Računarski i robotski vid, Stručna praksa 1 i Internet stvari. Temeljni dio sustava za detekciju životinja čine programski kod koji vrši detekciju objekata te obavještava korisnika, Web aplikacija i baza podatak. U radu su opisani svi softverski dijelovi sustava za detekciju životinja, a unaprjeđenja i mogućnosti s ovim sustavom je mnogo, neki od prijedloga bili bi: izrada mobilne aplikacije koja bi zamijenila web aplikaciju, primjena još bolje kamere koja bi imala mogućnost snimanja u mraku, izrada kvalitetnog i diskretnog postolja kako bi RaspberryPi i PiCamera bili skriveni što bi proširilo upotrebu ovog rada i na sigurnosnu kameru, upotreba enkripcije na svim razinama sustava zbog sigurnosti podataka, osiguravanje web aplikacije, opširnije dokumentiranje cijelog sustava.

Ovaj završni rad predstavlja praktični sustav za olakšavanje svakodnevnog života sa kućnim ljubimcima. Uz neke preinake, navedene u ovom poglavlju, sustav za detekciju životinja ima potencijala postati ozbiljan sustav temeljen na obradi podataka s video tijeka u stvarnom vremenu.

5 LITERATURA

[1] Ashray Saini:An Easy introduction to Flask Framework for beginners. Dostupno na : https://www.analyticsvidhya.com/blog/2021/10/easy-introduction-to-flask-framework-for-beginners/ (10.4.2021)

[2] Official documentation for Python 3.10.4. Dostupno na: **https://docs.python.org/3.10/**

[3] Emmet:Getting started with Python on RaspberryPi. Dostupno na: https://pimylifeup.com/raspberry-pi-python/

[4] Oskar Mieczkowski:Flask vs Django-Which Python Framework to Choose and When. Dostupno na: https://www.monterail.com/blog/flask-vs-django

[5] Teradata. Dostupno na: https://www.teradata.com/Glossary/What-is-Python

[6] Official Flask documentation. Dostupno na:https://flask.palletsprojects.com/en/2.1.x/

[7] Twilio blog. Dostupno na: https://www.twilio.com/blog/2017/01

[8] Twilio docs.Dostupno na: https://www.twilio.com/docs

[9] Sami Hryrysalmi,Mia Saari, Ahmad Muzaffar bin Baharudin:Survey of Prototyping Solutions Utilizing Raspberry Pi. Dostupno na: https://www.researchgate.net/publication/317136208_Survey_of_Prototyping_Solutions_Utilizing_Raspberry_Pi

[10] Jean-Louis Queguiner: What does Traning Neural Networks mean?. Dostupno na : https://blog.ovhcloud.com/what-does-training-neural-networks-mean/

[11] James Chen: Neural Network.Dostupno na: https://www.investopedia.com/terms/n/neuralnetwork.asp

[12] TensorFlow official website. Dostupno na: https://www.tensorflow.org/install.

[13] TensorFlow blog. Dostupno na: https://blog.tensorflow.org/

[14] Jason Brownlee:TensorFlow 2 Tutorial:Get started in Deep Learning With tfkeras. Dostupno na: https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/

[15] Matematički fakultet u Beogradu:Neuronske mreže. Dostupno na: http://ml.matf.bg.ac.rs/readings/slajdovi/10_NeuronskeMreze.pdf

[16] Tensorflow:github. Dostupno na: https://github.com/tensorflow

[17] SQLite službena stranica.Dostupno na: https://www.sqlite.org/docs.html

[18] Edward S.:SQLite vs MYSQL- What's the difference. Dostupno na https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/

[19] Nicholas Samuel:SQLite vs PostgreSQL. Dostupno na: https://hevodata.com/learn/sqlite-vs-postgresql/

[20] RaspberryPi docs.Dostupno na:https://www.raspberrypi.com/documentation/

6 OZNAKE I KRATICE

URL - engl. *Uniform Resource Locator*

HTTP – engl. *Hypertext Transfer Protocol*

IoT – engl. *Internet of things*

SQL – engl. *Structured Query Language*

API – engl. *Application programming interface*

WSDL – engl. *Web Service Definition Language*

WSGI- engl. *Web server gateway interface*

RDBMS- engl. *Relational Database Managment System*

7 SAŽETAK

Naslov: Detekcija životinja u video tijeku u realnom vremenu

Ovaj rad predstavlja razvoj sustava za detekciju kućnih ljubimaca koji omogućava prikupljanje, pohranu, dohvat i prikaz obrađenih detekcija ljubimaca na video tijeku u realnom vremenu. Sustav prilikom detektiranja kućnog ljubimca (psa ili mačke) obavještava korisnika o akcijama ljubimca ovisno o određenim uvjetima obrade podataka neuronske mreže. Također sustav te podatke pohranjuje u bazu i kasnije prikazuje na web stranici. Dijelovi sustava za detekciju životinja su: hardver realiziran uz pomoć RaspberryPi mikro računala, PiCamera za snimanje video tijeka, TensorFlow neuronskog sustava za obradu podataka i vršenje detekcije ljubimca, SQLite baza podataka koja zaprima sve detekcija i pohranjuje ih te Web stranice realizirane uz pomoć Python programskog jezika i modula Flask. Rad se temelji na kolegiju Internet stvari, odnosno primjenama tehnologija iz područja Internet stvari, uz primjenu baze podataka i web tehnologija. U radu je opisana kompletna softverska realizacija sustava za detekciju životinja.

Ključne riječi: RaspberryPi, TensorFlow, SQLite, Python, Flask.

8 ABSTRACT

Title: Detection of animals in video library in real time

This paper presents the *Pet Detection* System which enables the collection, storage, retrieval and display of processed pet detections in a real-time video stream. Pet Detection when detecting a pet (dog or cat) informs the user about the actions of the pet depending on the determination of the conditions in neural network data processing. The system also stores this data in a database and later displays it on a website. The parts of Pet Detection are: hardware realized with the help of RaspberryPi microcomputers, PiCamera for video recording, TensorFlow neural system for data processing and pet detection, SQLite database that receives and stores all detections and Web realized with Python programming language and module Flask . The paper is based on the course Internet of Things, application of technology in the field of Internet of Things, with the application of databases and web technology. The paper describes the complete software implementation of the *Pet Detection* system.

Key words: RaspberryPi,TensorFlow,SQLite,Python,Flask.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 20.4.2022	Ivan Jukić	

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

Ivan Jukić

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 20.4.2022

Jukic

potpis studenta/ice