Sustav za detekciju čučnjeva

Grdinić, Domagoj

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij računarstva

Diplomski rad

SUSTAV ZA DETEKCIJU ČUČNJEVA

SVEUČILIŠTE U RIJECI

Rijeka, rujan 2024.

Domagoj Grdinić
0069074420

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij računarstva

Diplomski rad

SUSTAV ZA DETEKCIJU ČUČNJEVA

Mentor: prof. dr. sc. Mladen Tomić

Rijeka, rujan 2024.

Domagoj Grdinić
0069074420

Izjava o izvornosti

Izjavljujem da je moj diplomski rad, pod nazivom „Sustav za detekciju čučnjeva“ , izvorni rezultat mojeg rada pod vodstvom prof. dr. sc. Mladena Tomića, te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni te primjenjujući znanja stečena tijekom studija.

Rijeka, rujan 2024.

Domagoj Grdinić
0069074420

ZAHVALA

Rijeka, rujan 2024.

Domagoj Grdinić
0069074420

Sažetak

Opsega od 100 do 300 riječi. Sažetak upućuje na temu rada, ukratko se iznosi čime se rad bavi, teorijsko-metodološka polazišta, glavne teze i smjer rada te zaključci.

Ključne riječi: riječ; riječ; …riječ; Obuhvaća 7+/-2 ključna pojma koji su glavni predmet rasprave u radu.

Rijeka, rujan 2024.

Domagoj Grdinić
0069074420

Rijeka, rujan 2024.

Domagoj Grdinić
0069074420

Sadržaj

1. UVOD

Slijetanje na mjesec jedno je od najvećih postignuća u ljudskoj povijesti, a važnu ulogu u tom pothvatu imali su ugradbeni sustavi (engl. *Embedded systems*). *Apollo Guidance Computer* je prvi značajniji ugradbeni sustav zadužen za kontrolu nad navigacijskim i upravljačkim modulima Apollo svemirskih letjelica. Tijekom 1970-ih godina počele su se pojavljivati prve industrije koje su koristile ugradbene sustave – industrija automobila i potrošačke elektronike. Prava revolucija u svijetu ugradbenih sustava započela je nakon što je izašao Intel 4004, prvi komercijalni mikroprocesor na svijetu. Ubrzo nakon toga Intel je predstavio prvi mikrokontroler, uređaj koji je integrirao procesor, memoriju i ulazno-izlazne uređaje na jednom čipu i tako pojednostavio implementaciju i proizvodnju ugradbenih sustava. Od tada je prisutan napredak i širenje ugradbenih sustava na gotovo sve industrijske grane i sve sfere svakodnevnog života. Ugradbeni su prisutni u većini uređaja koje koristimo, od pametnih telefona, automobila, kućanskih aparata, pa sve do medicinskih uređaja i složenih komunikacijskih sustava.

Ovaj rad istražuje izradu ugradbenog sustava za prepoznavanje i brojanje čučnjeva u stvarnom vremenu. Osim korištene Raspberry Pi platforme i dodatnog modula kamere, potrebno je implementirati i algoritme umjetne inteligencije za dobivanje ključnih točaka tijela i donošenje zaključka o trenutnom položaju korisnika. Jedna od korištenih komponenti je MoveNet model baziran na tehnikama dubokog učenja, a zadužen za detekciju položaja tijela. Model je optimiziran za rad u realnom vremenu i koristi se za prepoznavanje i praćenje pokreta te nam na taj način omogućava preciznu analizu vježbi poput čučnjeva. Podaci prikupljeni pomoću MoveNet-a analiziraju se putem neuronske mreže koja klasificira korisnikove pokrete u tri glavne kategorije: čučanj, polu-čučanj i ne-čučanj (stajaća pozicija). Ovaj pristup kombinira računalni vid i strojno učenje kako bi nam omogućio preciznu detekciju i analizu ljudskih pokreta, istovremeno koristeći kompaktnu i energetski konzervativnu platformu kao što je Raspberry Pi.

Poslužitelj i korisničko sučelje razvijeni su koristeći Python programski jezik i Flask framework. Sučelju je moguće pristupiti putem Internet preglednika iz lokalne mreže, a korisniku pruža mogućnost pokretanja i zaustavljanja brojača čučnjeva, kao i prikaz slike kamere u živo. Na taj način korisniku omogućava bolju orijentaciju i pozicioniranje u prostoru kako bi sustav koristili optimalno.

Kroz ovaj rad, ciljamo prema boljem razumijevanju sposobnosti ugradbenih sustava i umjetne inteligencije u praćenju ljudskih pokreta te potencijalu koji takva tehnologija nosi za unapređenje kvalitete života, sportskih performansi i općeg zdravlja. Detaljni pregled razvoja, implementacije i rezultata sustava za detekciju čučnja omogućit će dublje razumijevanje praktičnih aspekata primjene ovakvih tehnologija.

2. SKLOPOVLJE

2.1. Raspberry Pi

Raspberry Pi Foundation je zaklada koja je 2012. godine razvila prvu Raspberry Pi pločicu, zamišljenu kao malu, kompaktnu i jeftinu verziju SBC-a (engl. *Single Board Computer*). Prvotna zamisao pri razvijanju iste je bila promocija i širenje učenja računalnih osnova i računalne znanosti u školama. Osim u školama, ciljani kupci su bili korisnici koji žive u zemljama u razvoju, zbog čega je osnovna cijena bila jako pristupačna – svega 35 američkih dolara. S rastom popularnosti i brojem korisnika, Raspberry Pi se počeo koristiti za razne projekte čak i u visoko tehnološkim sustavima gdje nekad nije bilo mjesta za takve „jednostavne“ i „otvorene“ uređaje.

Od svog prvog izlaska na tržište Raspberry Pi doživio je više novih verzija i poboljšanja koja sa svakom novom verzijom donose više RAM memorije i moćniji procesor čime je uz snagu, proširena i mogućnost upotrebe. Uz robusni ARM procesor, GPIO pinove, jako je važna i kompatibilnost sa širokim rasponom operativnih sustava i softvera koji su prilagođeni upravo za ovu platformu, kao što su sustavi kućne automatizacije (*HomeAssistant*, *NodeRed*). U 2023. godini izašla je najnovija verzija Raspberry Pi 5 koja nudi još jači procesor i još bolje performanse od svojeg prethodnika.
Još jedna važna stvar koja neki proizvod čini boljim nego što sam proizvođač to radi je zajednica koja se skupila oko tog proizvoda. Tu je Raspberry Pi zasigurno u vrhu jer se na GitHubu nalazi više od 500 000 repozitorija koji su namijenjeni za korištenje na Raspberry Pi platformi. Upravo ta zajednica je smanjila razliku između amatera i eksperata i omogućila da

početnici u svega par sati posla naprave zanimljiv, koristan i funkcionalan projekt na svom novom uređaju.

Specifications	***Raspberry Pi 4***	**Raspberry Pi 5**
CPU	Quad-core Cortex-A72 (ARMv8) 64-bit @ 1.5GHz (16 nm process)	Quad-core Cortex-A76 (ARMv8) 64-bit @ 2.4GHz (28 nm process)
GPU	VideoCore VI @ 600MHz	VideoCore VII @ 1GHz
RAM	1/2/4/8 GB of LPDDR4 SDRAM	1/2/4/8 GB of LPDDR4x SDRAM
Display/Audio	2x Micro HDMI (up to 2x4K30 support), 4-Pole Stereo & Composite Video	2x Micro HDMI (up to 2x4K60 support)
Ethernet / Wireless	2.4GHz and 5GHz Wireless, Bluetooth 5.0, BLE, Gigabit Ethernet	
USB	2 x USB 2.0, 2 x USB 3.0	2 x USB 2.0, 2 x USB 3.0 (Supporting simultaneous 5.1Gbps)
MIPI Interfaces	1x 2-Lane CSI MIPI, 1x 2-Lane DSI MIPI	2 x 4-lane CSI/DSI MIPI
Other Interfaces	-	Single-lane Gen 2 PCIE, UART
GPIO	40 pin GPIO header	
Power Supply	5V/3A USB Type-C	5V/5A USB Type-C
POE	Via separate PoE HAT	
Storage	Micro SD	Micro SD (SDR104 mode support)
Additional Features	-	Power button, RTC, Fan connector, Heatsink mounting points

Slika 2.1: Usporedba Pi4 i Pi5 modela

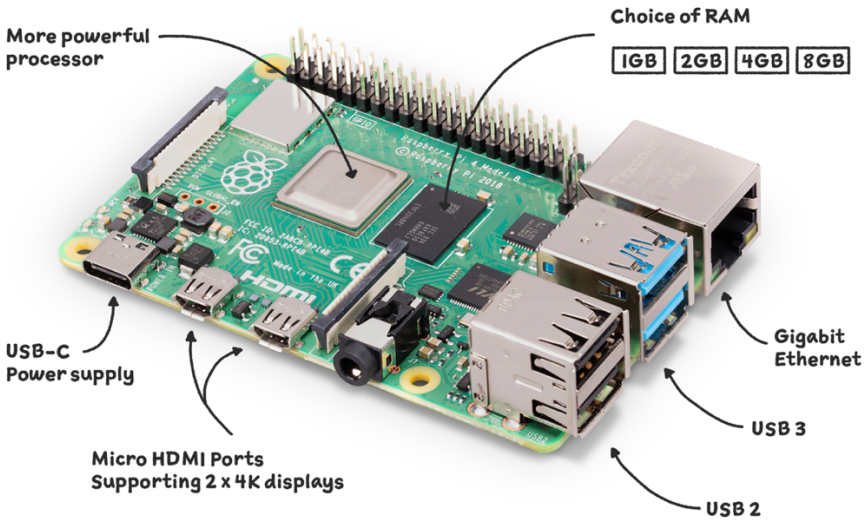
2.1.1. Raspberry Pi Arhitektura

Raspberry Pi koristi ARM (engl. *Advanced RISC Machine*) arhitekturu koja je dobro poznata po svojoj efikasnosti u vidu potrošnje električne energije i visokih performansi unatoč svojoj veličini. ARM arhitektura koristi se u pametnim telefonima i tabletima, a zadnjih godina je i Apple počeo koristiti ARM arhitekturu u procesorima koje ugrađuje u svoja prijenosna računala.

Osnovna komponenta Raspberry Pi-a je njegov procesor – CPU (engl. *Central Processing Unit*), koji je zadužen za izvršavanje ulaznih instrukcija i obradu podataka. Raniji modeli su imali jednojezgreni ARM1176JZF-S procesor, ali noviji model četvrte generacije ima četverojezgreni ARM Cortex A72 64-bitni procesor čije jezgre rade na frekvenciji od 1.5 GHz. Jači procesor donio je mogućnost korištenja kompleksnijih aplikacija, kao što je video obrada u stvarnom vremenu i korištenje strojnog učenja. Kod zahtjevnijih zadataka, temperatura CPU-a znatno raste i doseže temperature iznad 80° Celzijevih, što uzrokuje termalno prigušenje procesora (engl. *Thermal Throttling*) i smanjenje frekvencije rada CPU-a i samim time značajan pad performansi. Kako bi se spriječilo pregrijavanje CPU-a poželjno je dodati pasivno, a još bolje aktivno hlađenje. U ovom konkretnom slučaju, dovoljno je staviti termalnu pastu na CPU i na nju staviti pasivni hladnjak. Termalna pasta koristi se kao veza između površina CPU-a i

hladnjaka, koji nemaju savršenu završnu obradu i zbog toga treba nešto što će ih kvalitetno povezati. Kod termalne paste važno je da ima čim bolju toplinsku vodljivost. Kako bi se izbjeglo daljnje nepoželjno pregrijavanje procesora, najbolje rješenje je dodati ventilator koji će hladni zrak voditi preko hladnjaka i tako podignuti efikasnost hlađenja i cijelog sustava. Uz glavni procesor, Raspberry Pi ima i VideoCore IV GPU (engl. *Graphics Processing Unit*), odnosno grafički procesor zadužen za renderiranje slika i videa. Video IV znatno je slabiji od grafičkih kartica koje imamo na stolnim računalima, ali i dalje dovoljno snažan za obavljanje zadaća koje su mu namijenjene. Još jedna namjena GPU-a je da pomaže u slučajevima kad CPU zakaže, i na sebe preuzme neke procese koje može izvršiti i tako podiže performanse Raspberry Pi-a.

Raspberry Pi četvrte generacije na sebi ima LPDDR4-3200 SDRAM memoriju u rasponu od 1 GB pa sve do 8 GB, ovisno o verziji. Više memorije u odnosu na prethodnike omogućava da Raspberry Pi radi s većim i složenijim modelima skupova podataka, što rezultira preciznijim i učinkovitijim radom sustava kao što to može biti kod sustava za detekciju objekata.
Raspberry Pi na sebi ima više USB priključaka, HDMI izlaz, mrežni ulaz, kao i mnogo ulazno izlaznih pinova (GPIO) koji se koriste za spajanje senzora, kamera i drugih I/O (engl. *Input/Output*) uređaja koji omogućavaju interakciju i obradu podataka iz okruženja stvarnog svijeta i stvaranje interaktivnih sustava. Uz fizičke priključke, Raspberry Pi 4 ima i Bluetooth 5.0 povezivost, kao i WIFI povezivost na 2.4 GHz i 5 GHz frekvencijama.



Slika 2.2: Raspberry Pi 4

2.2. Oprema

Kako bi Raspberry Pi mogli koristit u svrhu projekta, potrebna je i dodatna oprema koja se koristi uz njega. Za pokretanje računala korišteno je originalno napajanje napona 5 V i najveće izlazne struje od 3 A. Snaga od 15 W je dovoljna za pokretanje uređaja čak i pri većim opterećenjima i na duži vremenski period. U slučaju da je napajanje premale snage dolazi do rušenja uređaja, gubljenja slike na ekranu ili grešaka u prikazu slike.

Nakon pokretanja samog uređaja, potrebno je instalirati i operacijski sustav koji će korisnik koristiti. Njega je moguće instalirati na SD karticu za koju uređaj ima utor ili na vanjski disk koji se može spojiti preko USB sučelja. U većini slučajeva je SD kartica jednostavnije i kompaktnije rješenje, međutim ako je potrebna veća stabilnost i količina prostora za pohranu, putem USB 3.0 sučelja moguće je spojiti i SSD disk. Kod korištenja SD kartice treba biti oprezan sa spremanjem LOG zapisa jer učestalo zapisivanje može smanjiti vijek trajanja SD kartice. U projektu je korištena SD kartica tvrtke HikVision sa 32 GB prostora za pohranu.

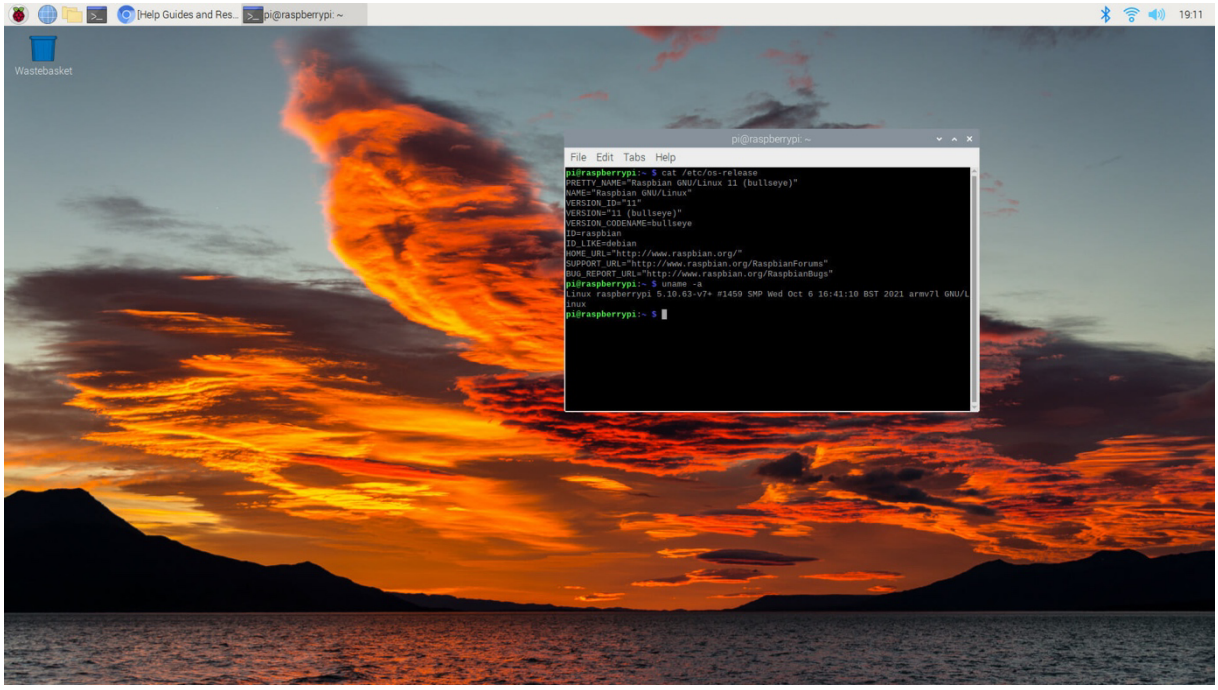
Za dohvaćanje slike koje se obrađuju u sustavu potrebno je koristiti kameru kompatibilnu s Raspberry Pi uređajem. Moguće je koristiti neku od kamera namijenjenih isključivo za rad s Raspberry Pi uređajem ili neku od USB kamera za koje Raspberry Pi ima programsku podršku. Prvotno je sustav testiran Logitech C270 kamerom koja se spaja preko USB sučelja i funkcionalna je bez ikakve potrebe za instalacijom dodatne programske podrške, ali zbog integracije sustava i jednostavnijeg korištenja, zamijenjena je s RB-CameraWW modelom tvrtke Joy-IT. Kamera ima kut prikaza od 160 stupnjeva i rezoluciju slike 2952 x 1944 piksela, odnosno 5 megapiksela. Kad se kamera koristi za video prikaz, može isporučiti video u Full HD rezoluciji (1920 x 1080 piksela) i pri brzini osvježavanja slike od 30 sličica u sekundi (engl. *Frames per second*). Potrošnja kamere u radu je manje od 100 mA i zbog toga je pogodna za ugradbene sustave ovog tipa. Za usporedbu, Logitech C270 kamera koja ima gotovo 3 puta nižu rezoluciju prikaza, u radu troši 220mA. Kamera se spaja putem CSi (*Camera Serial interface*) sučelja i tako ne zauzima USB priključke na uređaju, a osim toga, CSi priključak omogućava niske latencije koje su presudne za aplikacije koje koriste prikaz slike uživo.

Uz gore navedenu opremu, potrebni su miš, tipkovnica, ekran za prikaz slike, kao i adapter za micro-HDMI konektor kako bi se monitor mogao povezati. Nakon završnog postavljanja uređaja navedena periferija više nije bila potrebna jer se na uređaj može spojiti putem SSH (*Secure shell)* protokola i pokretati aplikacije preko CLI (*Command Line Interface*) sučelja.

3. PROGRAMSKA POTPORA

3.1. Operativni sustav – Raspberry PI OS

Raspberry Pi OS je operativni sustavi posebno prilagođen za Raspberry Pi uređaje svih generacija. Prva verzija ovog operativnog sustava poznata je i pod nazivom Raspbian jer je osnovu činila Debian arhitektura koja je zadržana i na novijim verzijama operativnog sustava. Dizajniran je da bude „lagan“ i efikasan, kako bi hardverske resurse koje Raspberry Pi nudi mogle koristiti aplikacije koje se pokreću na njemu. Moguće je koristiti verziju s LXDE (*Lightweight X11 Desktop Environment*) grafičkim sučeljem koje je prilagođeno za rad s ograničenim resursima. Postoje i druga grafička sučelja kao KDE, XFCE, MATE i drugi, što korisniku omogućava da odabere okruženje koje najbolje odgovara njegovim potrebama.



Slika 3.1: Rapberry Pi OS "Bullseye"

Uz operativni sustav dolazi i niz predinstaliranih programa i alata koji korisniku omogućavaju brz početak rada. Ovi alati uključuju programe za osnovnu obradu teksta (Mousepad, Leafpad, alate za razvoj i programiranje (Python, Thonny), kao i Chromium Internet pretraživač. Zbog problema s određenim knjižnicama i programima na najnovijoj verziji „Bookworm“, na ovom projektu korištena je 11. verzija Debiana – Bullseye.

3.2. Python

Python je popularan programski jezik koji se često koristi za razvoj web aplikacija, IoT, analizu podataka i u primjeni umjetne inteligencije. Popularan je zbog brze krivulje učenja, kao i zbog jednostavnosti u čitanju i pisanju koda. Uz Python se često može pronaći epitet "čitljiv", zbog toga što mu je sintaksa bliska ljudskom jeziku. Interpretirani je programski jezik i nema potrebe za prevođenjem cijelog koda prije izvođenja, već se on izvodi liniju po liniju i zato je pogodan za brzu izmjenu i testiranje koda.

Ono što ga čini veoma popularnim i korištenim je veliki broj besplatnih knjižnica i radnih okvira (engl. *Framework*) koji se mogu koristiti u razne svrhe, a posebno u analizi podataka i primjenama umjetne inteligencije. Python knjižnice kao što su NumPy, Pandas, Matplotlib i Scipy omogućavaju efikasnu analizu i vizualizaciju podataka, dok knjižnice kao što su TensorFlow, Pytorch i Scikit-Learn pružaju moćne alate za razvoj aplikacija koje koriste strojno učenje i umjetnu inteligenciju. Koristeći ove knjižnice moguće je implementirati složene algoritme i modele uz veliku uštedu vremena čime se bitno ubrzava proces razvoja aplikacija. Također, Python ima podršku za više platformi, uključujući MacOS, Windows i Linux. U projektu je korištena verzija Pythona 3.8.19.

3.2.1. Jupyter Notebook

Jupyter Notebook je projekt započet 2001. godine pod imenom IPython od strane Fernanda Pereza. Zamišljen je kao interaktivna ljuska za Python, međutim brzo se razvio u puno veći projekt koji svoju primjenu često nalazi i u znanstvenim svrhama. Novo ime dobio je 2014. godine kako bi označio podršku za nove programske jezike – R i Julia. Specifičnost Jupyter Notebook-a je kombiniranje više elemenata kao što su tekst, kod, grafike, kao i matematičke formule i to sve unutar jednog dokumenta. Velika prednost je lakoća dijeljenja tako da se gotov dokument može dijeliti drugim korisnicima i koristiti bez veće potrebe za prilagodbom.

Zbog preglednosti i jednostavnog uvida u rezultate, za treniranje modela i usporedbu klasifikatora, u ovom projektu je korišten Jupyter Notebook.

3.2.2. Flask

Radni okviri koriste se kako bi ubrzali i pojednostavili razvoj krajnjeg proizvoda odnosno u ovom slučaju web aplikacija. Najpopularniji radni okviri za razvoj web aplikacija u Python programskom jeziku su Flask i Django. Django je veoma skalabilan te je više namijenjen srednjim i velikim sustavima, dok je Flask pogodniji za manje projekte. Uvijek postoje iznimke i moguće ih je koristiti u svim slučajevima ako za to postoji potreba. Flask se često opisuje kao mikro radni okvir jer donosi jednostavnost i fleksibilnost, kao i brzinu razvoja uz veoma malo korištenje resursa. Jedna od glavnih karakteristika Flask-a je minimalizam i to se očituje kroz samo korištenje. Flask dolazi samo s osnovnim knjižnicama i komponentama koje su potrebne programeru kako bi razvio krajnji proizvod, ali naravno isto tako omogućuje dodavanje dodatnih knjižnica i dodataka kao i funkcionalnosti koje programer zahtijeva.
Korištenje Flask-a je pojednostavljeno korištenjem pogleda (engl. *View*) kao funkcijama za obradu HTTP zahtjeva i odgovora, kao i kreiranje ruta do web odredišta. Za uključivanje dinamičkih elemenata na web stranicu, Flask koristi Jinja2 sustav. Također je tu i WSGI knjižnica za komunikaciju između web aplikacije i servera, a kojom se koristi i Django. Kao što ima prednosti, tako Flask ima i mane zbog kojih nije pogodan za neke projekte. Svakako najveći minus je nedostatak baze podataka i ORM (engl. *Object-relational mapping*) koji kreira sloj između relacijske baze podataka i programskog jezika. Nema početno admin sučelje kao što to ima Django, u kojem se mogu vršiti i pratiti izmjene na bazi podataka.
Neki od poznatijih projekata, koji su u svojim ranijim fazama koristili Flask, su Pinterest i LinkedIn.

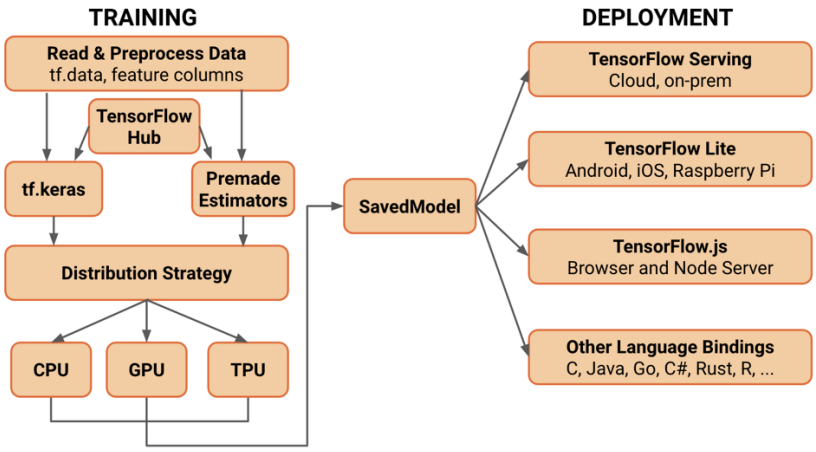
3.3. Knjižnice i alati

3.3.1. TensorFlow

TensorFlow je platforma otvorenog tipa bazirana na dubokom učenju od strane Google-a. Njena fleksibilnost i skalabilnost omogućava široku primjenu u raznim područjima rada gdje postoji potreba za strojnim učenjem i modelima koji zahtijevaju obradu velikih količina podataka. Tenzori su temeljni elementi TensorFlowa, koji predstavljaju višedimenzionalne matrice. Oni

mogu sadržavati vektore, matrice, skalare ili višedimenzionalne nizove podataka. Tenzori služe za pohranu ulaznih i izlaznih podataka, kao i za pohranu parametara modela.

TensorFlow modeli su definirani kao grafovi podataka gdje čvorovi predstavljaju matematičke operacije, a bridovi tok podataka, odnosno tenzora između tih čvorova. TensorFlow dolazi u nekoliko varijanti, od Core verzije namijenjene stolnim računalima i radnim stanicama, Tensorflow.js koja omogućava izvođenje na Node.js sustavima i korištenju modela u web pregledniku, pa sve do TensorFlow Lite verzije, namijenjene IoT i mobilnim uređajima, odnosno uređajima slabijih performansi. Ta je verzija korištena u ovom projektu, kako bi se TensorFlow mogao koristit na Raspberry Pi uređaju.



Slika 3.2 TensorFlow

3.3.2. OpenCV

OpenCV je knjižnica otvorenog tipa za računalni vid i analizu slike. Projekt kojim je ona nastala započeo je 1999. Godine u sklopu tvrtke Intel pod voditeljem Garyjem Bradskym. Napisana je u C++ jeziku ali s godinama je napisana podrška za Python i Java programske jezike, što ju čini fleksibilnom i korištenom na većini platformi. Dostupna je podrška za Windows, macOS, Linux, iOS i Android, dakle za gotovo sve platforme koje se danas nalaze na tržištu. Unutar knjižnice se nalazi više od 2500 algoritama koji se mogu koristiti u obradi slike, računalnom vidu i strojnom učenju, a najčešće se koriste oni za detekciju točaka, objekata ili prepoznavanje lica. Primjena OpenCV knjižnice je prisutna u više različitih industrija, a neke od aktualnih su

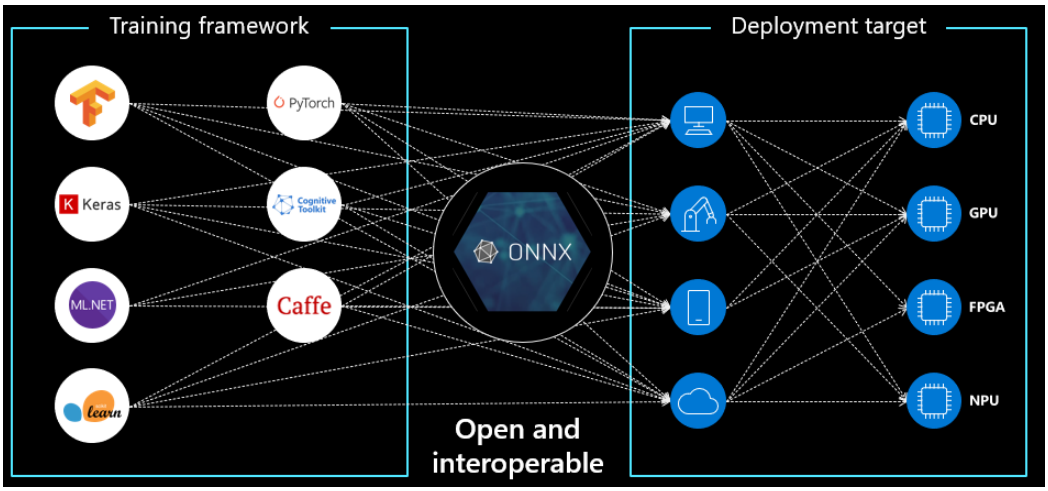
svakako autonomna vozila, gdje je potrebna detekcija objekata, prepoznavanje znakova i detekcija i praćenje linija.

3.3.3. Scikit-learn

Scikit-learn ili skraćeno sklearn je projekt otvorenog koda, odnosno knjižnica čiji je razvoj počeo 2007. godine i koristi se za strojno učenje u sustavima temeljenima na Pythonu. Temelji sklearn knjižnice su bazirani na NumPy, SciPy i Matplotlib knjižnicama, sve redom popularnim Python knjižnicama.

3.3.4. ONNX

ONNX (Open Neural Network Exchange) je otvoreni standard (engl. *Open-source*) za razmjenu modela strojnog učenja između različitih alata, platformi i okruženja. Nekoliko velikih tehnoloških kompanija sudjelovalo je u razvijanju ovog alata, uključujući Microsoft i Facebook (Meta) kako bi se omogućila interoperabilnost više različitih okvira za strojno učenje. Ono što ONNX omogućava je da se model obučen u jednom okviru, kao što je TensorFlow, može izvoziti i koristiti u nekom drugom okviru, kao što je PyTorch, bez potrebe za ponovnim treniranjem ili značajnim izmjenama. ONNX Runtime je alat koji omogućuje korištenje modela u ONNX formatu u radnom okruženju. Dizajniran je da pruža visoke performanse prilikom izvođenja i nudi podršku za gotovo sve hardverske platforme, kao što su CPU(Intel i AMD), GPU(NVIDIA i AMD) ali i specijalizirane AI akceleratore – NVIDIA TensorRT i Intel OpenVINO. Zbog toga je pogodan za korištenje u ugradbenim sustavim u kojima se koristi Raspberry Pi, ali se može koristiti i na puno snažnijim i brojnijim serverima u cloud okruženju. Također, tu je i podrška za različite programske jezike, među kojima su Python, C++, C#, Java i JavaScript.



Slika 3.3 ONNX

3.3.5. Ostale knjižnice

Još jedna knjižnica potrebna za rad sustava je pandas koja se koristi za čitanje i manipulaciju podatcima i .csv datotekama. Uz nju je u početku korištena picamera knjižnica koja bi trebala pojednostaviti korištenje kamere na Raspberrry Pi-u, međutim ona se više ne koristi na novijim verzijama Raspberry Pi OS-a, i zbog toga je njeno razvijanje napušteno, a s time i upotreba u ovom projektu. Koristi se i knjižnica seaborn za vizualizaciju podataka i ona je naslonjena na matplotlib knjižnicu.

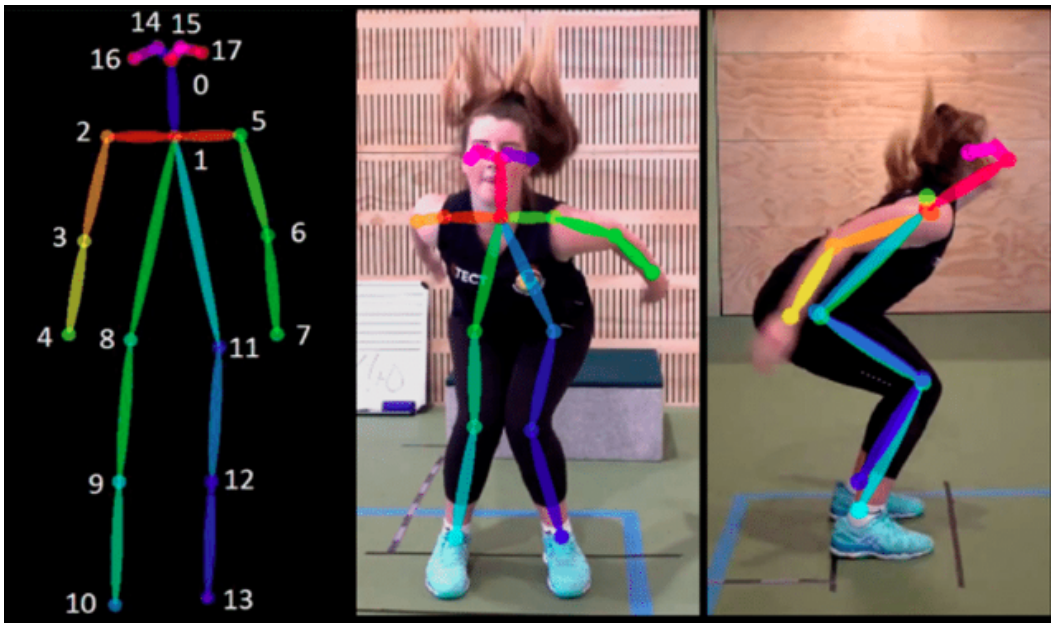
4. MODELI I USPOREDBA KLASIFIKATORA

4.1. MoveNet

MoveNet je nasljednik PoseNet tehnologije praćenja pokreta ljudskog tijela, a temelji se na konvolucijskim neuronskim mrežama. MoveNet je predviđen da prepoznaje 17 ključnih točaka ljudskog tijela koje omogućuju precizno praćenje ljudskog pokreta u prostoru. Visoka točnost i brzina pri identifikaciji ključnih točaka omogućavaju da se MoveNet može koristiti za praćenje pokreta u realnom vremenu, čak i na uređajima s ograničenim resursima, kao što su mobilni telefoni ili ugradbeni sustavi. Integracija s TensorFlow platformom uvelike olakšava korištenje MoveNet tehnologije u projektima. MoveNet nudi dvije opcije modela namijenjene za rad s Raspberry Pi uređajima - Thunder i Lightning. Oba imaju podvrste s obzirom na vrstu

podataka kojom barataju, Int8 ili Float16 tip podataka, a model korišten u ovom radu je Singlepose Lightning Float16.

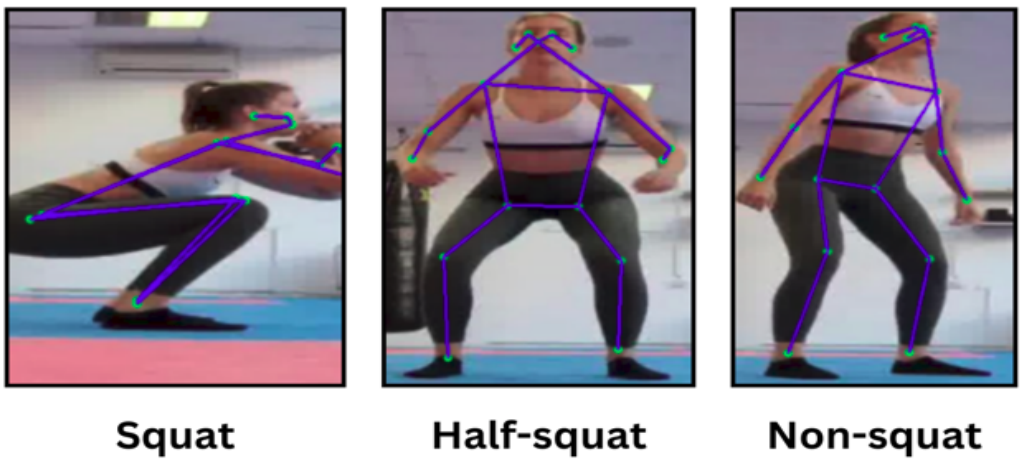
MoveNet kao ulaz prima sliku ili video okvir s kamere koju je prvo potrebno pred procesirati kako bi se postigle optimalne performanse. To uključuje promjenu rezolucije i normalizaciju piksela (skaliranje vrijednosti). Pred procesiranje je važno kako bi se dobile performanse adekvatne za izvršavanje u realnom vremenu i na procesorski slabijim uređajima. Format u kojem slika ulazi u model je 256 x 256 x 3 za Thunder model, dok Lightning koristi 192 x 192 x 3 format. Prve dvije brojke predstavljaju piksele, dok treći broj označava broj kanala boja unutar slike. Nakon što slika uđe u model, MoveNet koristi CNN (*Convolutional Neural Networks*) mrežu za dobivanje karakteristika iz slike. Prolaskom kroz CNN slojeve, detektiraju se granice, oblici i teksture koji su važni za prepoznavanje položaja tijela. Model iz tih karakteristika predviđa položaje 17 ključnih točaka kao koordinatne podatke (x,y). Položaj tijela se potom rekonstruira na temelju predviđenih ključnih točaka. Oba modela vraćaju rezultat u float32 tenzor obliku [1,1,17,3] redom označavajući x i y koordinate, 17 točaka na korisniku (nos, lijevo oko, desno oko, lijevo uho, desno uho, lijevo rame, desno rame, lijevi lakat, desni lakat, lijevo zapešće, desno zapešće, lijevi kuk, desni kuk, lijevo koljeno, desno koljeno, lijevi gležanj, desni gležanj) i pouzdanost predviđanja točaka.



Slika 4.1 Ključne točke dobivene koristeći MoveNet

4.2. Strojno učenje i klasifikatori

Strojno učenje je tehnologija razvijena kako bi računala po uzoru na ljude, mogla učiti iz iskustva. Razlika je što se kod računala umjesto iskustva koriste veliki skupovi podataka. U procesu učenja računala analiziraju velike količine podataka i iz njih pokušavaju izvući obrasce i zakonitosti po kojima se donosi neka odluka. Prvi korak u procesu strojnog učenja je obrada podataka. Potrebno je prikupiti, očistiti, skalirati i prilagoditi u format koji je potreban određenom modelu za analizu. Sljedeći korak je odabir adekvatnog klasifikatora strojnog učenja za analizu i rješavanje problema. Klasifikatori u strojnom učenju su algoritmi koji se koriste za kategorizaciju podataka u različite kategorije. Klasifikatori testirani u ovom radu su Logistička regresija, SVM (*Support Vector Machine*) i FFNN (*Feed Forward Neural Network*). Nakon odabira klasifikatora, slijedi treniranje modela na prije sakupljenom i obrađenom skupu podataka. Zatim se radi evaluacija modela, odnosno testiranje na nezavisnim podacima kako bi se procijenilo zadovoljava li točnost i učinkovitost sustava naše zahtjeve. Nakon što su svi koraci završeni, model je spreman za korištenje na stvarnim podacima. Kao ulazni podaci klasifikatora korišteni su izlazni podaci MoveNet mreže. Zadaća klasifikatora u ovom projektu je određivanje pozicije tijela u tri kategorije – Čučanj (engl. *Squat*), Polu-čučanj (engl. *Half-squat*) i Ne-čučanj (engl. *No-squat*).



Slika 4.2 Klase i primjeri

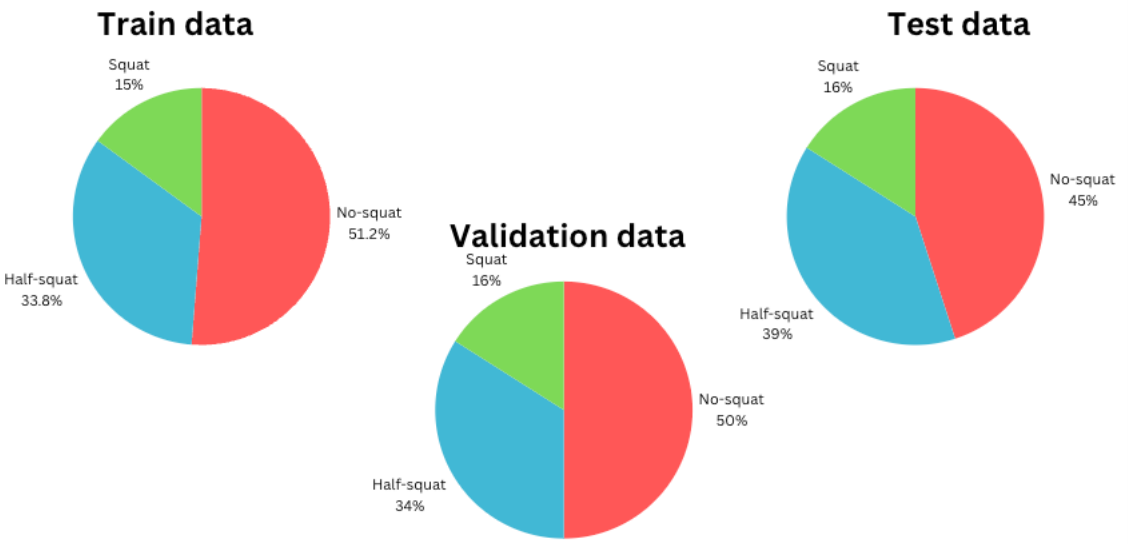
Definiranje klasa preuzeto je iz kineziologije, tako da se za čučanj uzima slučaj kad je kut između kukova i gležnjeva, odnosno između natkoljenice i potkoljenice 90 stupnjeva ili manji. Iako tu postoji diskusija o isključivosti ovog zahtjeva, zbog raznih tehnika, ali i medicinskih

situacija gdje ljudi ne mogu pokret izvoditi u punom opsegu. Navedena granica uzeta je za razlikovanje *squat* i *half-squat* pozicije. *Half-squat* zapravo obuhvaća sve pokrete između čučnja i stajanja, odnosno u slučajevima kad su kutevi između potkoljenice i natkoljenice između 90 i 180 stupnjeva.

4.2.1. Skup podataka

Skup podataka (engl. *Dataset*) je ključni element strojnog učenja jer je osnova za izvor informacija iz kojeg model uči. U kontekstu treniranja neuralnih mreža, dobar i kvalitetno pripremljen skup podataka može biti presudan za uspješan rad. Skup podataka se dijeli na trening skup (engl. *Training set*), validacijski set (engl. *Validation set*) i testni set (engl. *Test set*). Trening set se koristi za treniranje modela i iz njega model uči odnose između atributa i oznaka u skupu podataka. Najveći dio skupa podataka trebao bi se nalaziti upravo u trening skupu podataka kako bi model mogao učiti na što više primjera. Validacijski set se koristi za provjeru performansi modela tijekom procesa treniranja modela i pomaže u podešavanju parametara modela. Također je važan jer pomaže u prevenciji prekomjernog prilagođavanja (engl. *Overfitting*). Prekomjerno prilagođavanje je slučaj kada model previše dobro radi isključivo s testnim podacima i zbog toga nema sposobnost prilagodbe na nove podatke. Rezultat toga su jako visoki rezultati na podacima iz skupa za treniranje, a jako loši rezultati na testnim ili stvarnim podacima. Testni skup podataka se koristi za provjeru i procjenu performansi modela nakon što je model treniran i radi se na neviđenim podacima. Zbog toga što se radi o novim podacima, on omogućava objektivu evaluaciju modela i njegovih performansi.

Za potrebe ovog projekta prikupljeno je ukupno 7086 fotografija podijeljenih unutar tri skupa podataka – trening (4958), validacijski (1416) i testni (712) skup. Svaki od tih skupova ima približno istu distribuciju slika po kategorijama – 50% za Ne-čučanj, 35% za polu-čučanj i 15% za čučanj.



Slika 4.3 Distribucija podataka po skupovima

S obzirom da se radi o nadziranom učenju, nakon prikupljanja i obrade slika potrebno je obilježavanje (engl. *Labeling*) slika i njihovo svrstavanje u kategorije. Na temelju ulaznih podataka i pripadnosti kategorijama izvršava se treniranje modela.

4.2.2. Logistička regresija

Logistička regresija je jednostavan i široko upotrebljavan klasifikacijski algoritam. Logistička regresija iz dobivenih podataka izračunava ponderiranu sumu ulaznih karakteristika (engl. Features) i za svaku ulaznu karakteristiku x_i postoji odgovarajući težinski koeficijent w_i. Ponderirana suma svih ulaznih karakteristika računa se formulom:

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \quad \textbf{(4.1)}$$

B ili bias je pomak koji pomaže modelu da „prilagodi“ podatke, posebno kada podaci nisu simetrično raspoređeni oko početka koordinatnog sustava (0,0). Nakon što se izračuna suma z, primjenjuje se logistička funkcija poznata i kao sigmoid funkcija, kako bi se rezultat koji predstavlja vjerojatnost zapisao u vrijednosti između 0 i 1.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Slika 4.4. Sigmoid funkcija

4.2.3. SVM

Support Vector Machine je vrsta klasifikatora koji ima zadatak pronaći hiper-ravninu koja najbolje razdvaja podatke u različite klase. SVM koristi vektore podrške koji su najbliži hiper-ravnini i pomažu u određivanju njenog položaja i definiranja optimalne granice. Za primjer linearne klasifikacije hiper-ravnina se može opisati formulom gdje je *w* vektor težine i određuje orijentaciju hiper-ravnine, *x* je ulazni vektor karakteristika (podatak koji klasificiramo), *b* je pomak koji pomiče hiper-ravninu po y osi:

$$w \cdot x + b = 0 \quad \textbf{(4.2)}$$

4.2.4. FFNN

Feed-forward Neural Network osnovna je i najčešće korištena vrsta neuralne mreže. Ima jednostavnu arhitekturu i smjer podataka u samo jednom smjeru bez povratnih veza ili ciklusa. Strukturu FFNN-a sačinjavaju ulazni sloj, skriveni slojevi i izlazni sloj. Ulazni sloj je taj koji prima podatke, a svaki neuron predstavlja jednu ulaznu varijablu kao što je piksel slike ili karakteristika u skupu podataka. Podaci zatim ulaze u jedan ili više skrivenih slojeva, koji se nalaze između ulaznog i izlaznog sloja. Neuroni u ovim slojevima izvode razne vrste transformacija nad ulaznim podacima. Skriveni slojevi mogu koristiti funkcije aktivacije kao što su sigmoid, tank ili ReLU (engl. Rectified Linear Unit) kako bi modelu omogućili da nauči odnose među podacima koji su nelinearni. Izlazni sloj je posljednji sloj koji daje krajnje rezultate. Broj neurona je zavisan o vrsti zadatka i za zadatke klasifikacije koji imaju *n* mogućih klasa, izlazni sloj ima *n* neurona.

Arhitektura FNNN mreže koja je korištena u projektu sastoji se od 3 sloja i koristi gusto povezane slojeve (engl. *Dense layer*), što znači da je svaki neuron povezan sa svim neuronima u prethodnom sloju. Matematički gledano, gusto povezani sloj možemo prikazati formulom u kojoj je W matrica težina koja sadrži veze između neurona iz prethodnog sloja i trenutnog sloja. X je ulazni vektor (izlazi iz prethodnog sloja), b je pomak koji se dodaje svakom neuronu a z je izlazni vektor prije nego se primijeni funkcija aktivacije.

$$z = W \cdot x + b \quad \textbf{(4.3)}$$

Jedan gusto povezani sloj i sloj za isključivanje neurona sačinjavaju jedan sloj neuralne mreže. Prvi gusti sloj ima 64 neurona i koristi ReLU funkciju aktivacije koja omogućava mreži učenje nelinearnih obrazaca u podacima. Zatim slijedi drugi sloj s 32 neurona, također s ReLU aktivacijom. Nakon oba sloja nalaze se slojevi za isključivanje neurona. Oni služe za prevenciju prekomjernog prilagođavanja tako što se nasumično deaktivira 20% neurona u svakom *dropout* sloju. Isključivanjem dijela neurona postiže se „samostalnost“ sustava, odnosno povećava njezinu sposobnost prilagodbe na nove, neviđene podatke. Kada ne bismo koristili *dropout* sloj, sustav ne bi znao reagirati na nove uzorke, već bi funkcionirao jedino pomoću slika iz skupa podataka za treniranje. Završni izlazni sloj je također gusto povezani sloj koji ima 3 neurona i koristi *softmax* funkciju aktivacije. *Softmax* funkcija pretvara izlazne vrijednosti u vjerojatnosti koje predstavljaju pripadnost svakoj od tri kategorije (Čučanj, polu-čučanj, ne-čučanj).

Layer	Type	Units/Filters	Activation	Input Shape	Dropout Rate	Output Shape
1	Dense	64	ReLU	(51,)	-	(64,)
2	Dropout	-	-	-	0.2	(64,)
3	Dense	32	ReLU	(64,)	-	(32,)
4	Dropout	-	-	-	0.2	(32,)
5	Dense	3	Softmax	(32,)	-	(3,)

Slika 4.5 Arhitektura FFNN za klasifikaciju

Model je treniran na skupu podataka tijekom 70 epoha, pri čemu je *batch* veličina iznosila 512. Jedna epoha predstavlja jedan potpuni prolazak kroz cijeli skup podataka za treniranje i u jednoj epohi mreža svaki uzorak vidi točno jednom. Više epoha se koristi kako bi tijekom treninga model mogao postepeno učiti iz podataka. *Batch* je podskup podataka koji se koristi za

treniranje modela unutar jedne epohe. Između svake epohe se radi prilagođavanje težine kako bi se smanjio gubitak i poboljšala točnost predikcija.

Zbog svoje učinkovitosti i mogućnosti prilagodbe, za optimizaciju podataka korišten je Adam optimizator, popularan algoritam za optimizaciju u dubokom učenju. Za funkciju gubitka (engl. *Loss function*) je korištena *Sparse categorical cross-entropy* funkcija. Njena zadaća je da mjeri razliku između predviđene i stvarne kategorije.

```
model = Sequential([
    Dense(64, activation='relu', input_shape=(51,)),  # Adjust input_shape based on your feature dimension
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

model.compile(
    optimizer=Adam(),
    loss=SparseCategoricalCrossentropy(),
    metrics=['accuracy']
)

history = model.fit(
    X_train_scaled,
    train_labels,
    validation_data=(X_valid_scaled, valid_labels),
    epochs=70,  # Number of epochs
    batch_size=512,  # Adjust batch size if needed
    class_weight=class_weights
)
```

Slika 4.6 Treniranje FFNN modela

4.3. Usporedba klasifikatora

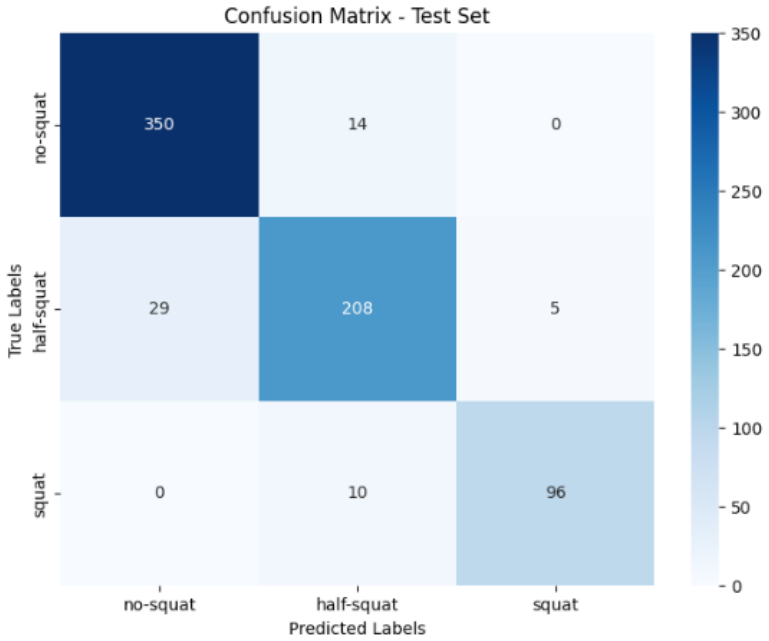
Kako bi se odabrao pravi klasifikator za ovaj projekt uspoređene su metrike tri trenirana modela. Metrike koje gledamo su točnost (engl. *Accuracy*), osjetljivost (engl. *Recall*), preciznost (engl. *Precision*) i F1 Rezultat (engl. *F1 Score*). Točnost se računa kao omjer broja točno klasificiranih primjera i ukupnog broja primjera. Točnost prikazuje koliko često model donosi ispravnu odluku i uglavnom se koristi kao primarna metrika pri usporedbi modela. U nekim slučajevima kad su klase neuravnotežene, točnost može biti varljiva metrika. Primjer toga je model koji klasificira sve e-mailove kao ne-spam u skupu gdje 95 % e-maila zaista nije spam, točnost će biti visoka – 95 %, ali zato model neće znati prepoznati e-mailove koji zaista jesu spam. Stoga ju je potrebno koristiti i uspoređivati skupa s drugim metrikama. Osjetljivost je metrika koja pokazuje koliko dobro model prepoznaje stvarne pozitivne primjere. Osjetljivost se računa kao omjer točno predviđenih pozitivnih primjera i ukupan broj stvarnih pozitivnih primjera. Ona je ključna u slučajevima gdje je potrebno identificirati što više stvarno pozitivnih primjera. Primjerice u medicinskim dijagnozama gdje je bolje imati što manje lažno negativnih rezultata kako ne bismo propustili stvarnog bolesnika. F1 Rezultat je još jedna metrika koju trebamo gledati da bismo dobili potpunu sliku performansi modela, a označava harmonijsku sredinu između preciznosti i osjetljivosti. Preciznost se dobiva omjerom broja točno predviđenih pozitivnih primjera i broja ukupno predviđenih pozitivnih primjera.

Logistička regresija ima točnost od 0.92, što znači da je model ispravno klasificirao 92 % svih primjera u testnom skupu. Vidljivo je da klasa *half-squa*t ima nižu osjetljivost od 0.86, što znači da je model promašio prepoznati nekoliko stvarnih primjera polu-čučnja i svrstao ih u druge klase. Također je i preciznost *no-squat* klase nešto lošija od ostalih metrika, što znači da je model nekoliko puta pogrešno klasificirao druge klase kao *no-squat*.

```
Test Accuracy: 0.9185393258426966

Classification Report (Test):
              precision    recall  f1-score   support

    no-squat       0.93      0.96      0.94       364
  half-squat       0.90      0.86      0.88       242
       squat       0.93      0.91      0.92       106

    accuracy                           0.92       712
   macro avg       0.92      0.91      0.91       712
weighted avg       0.92      0.92      0.92       712
```

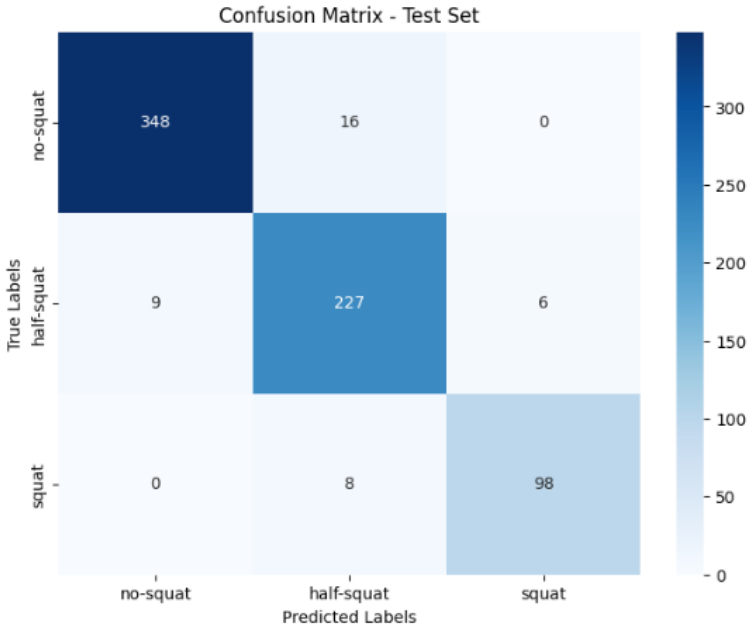
Slika 4.7 Rezultati logističke regresije

SVM model je pokazao slične rezultate kao i logistička regresija, pokazujući iste mane tog modela. Kroz matricu konfuzije ispod teksta moguće je primijetiti u kojim slučajevima je klasifikacija zakazala i da isto kao i logistička regresija, ima probleme s *half-squa*t i *no-squat* klasom.



Slika 4.8 Matrica konfuzije za SVM model

FFNN model je performansama nadmašio Logističku regresiju i SVM, s najvećom točnošću od 95 % na testnom skupu podataka. U klasifikaciji *half-squat* klase preciznost je 90 %, dok je osjetljivost visokih 94 %. Performanse u drugim kategorijama su također odlične pa se FFNN model pokazao kao najpogodniji za korištenje. Visoka točnost i F1 Rezultat pokazuje da model radi točnu klasifikaciju velike većine pokreta i da to radi uz veliku pouzdanost.

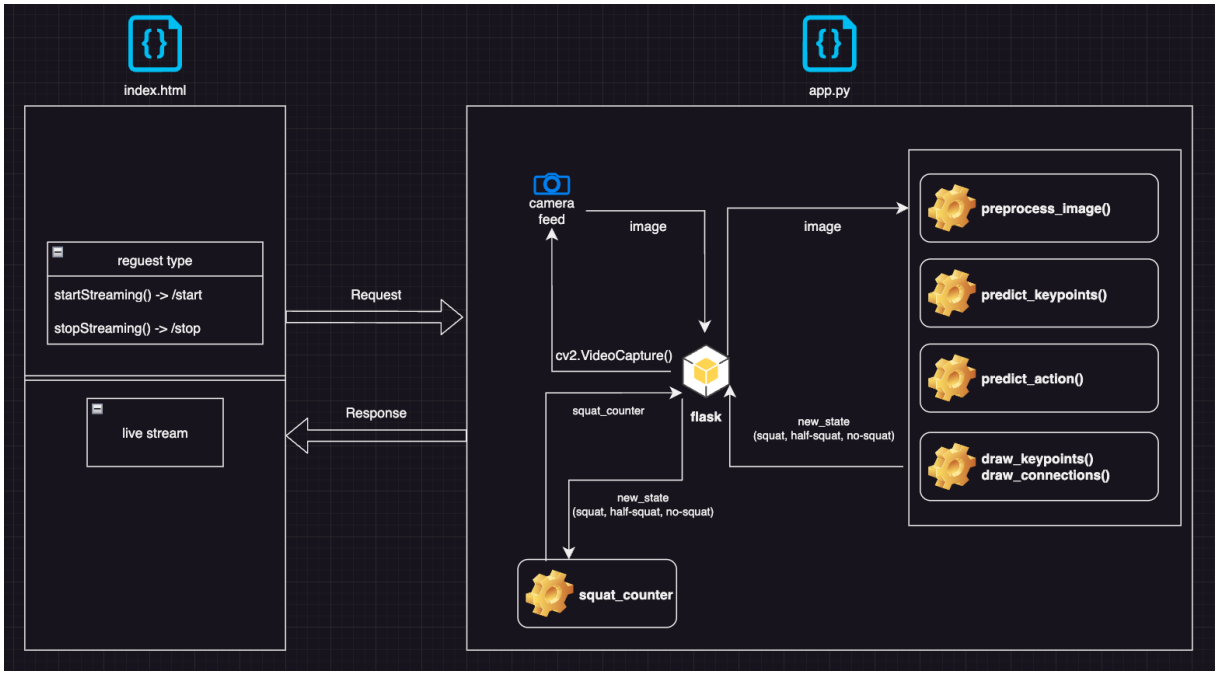


Slika 4.8 Matrica konfuzije za FFNN model

5. UGRADBENI SUSTAV

5.1. Arhitektura sustava

Sustav možemo razdvojiti na frontend i backend dio, odnosno klijentsku i poslužiteljsku stranu. Frontend dio čini *index.html* dokument unutar *templates* mape koji se koristi za otvaranje i upravljanje web aplikacijom. Ostatak dokumenata je vezan uz backend dio odnosno poslužiteljsku stranu.

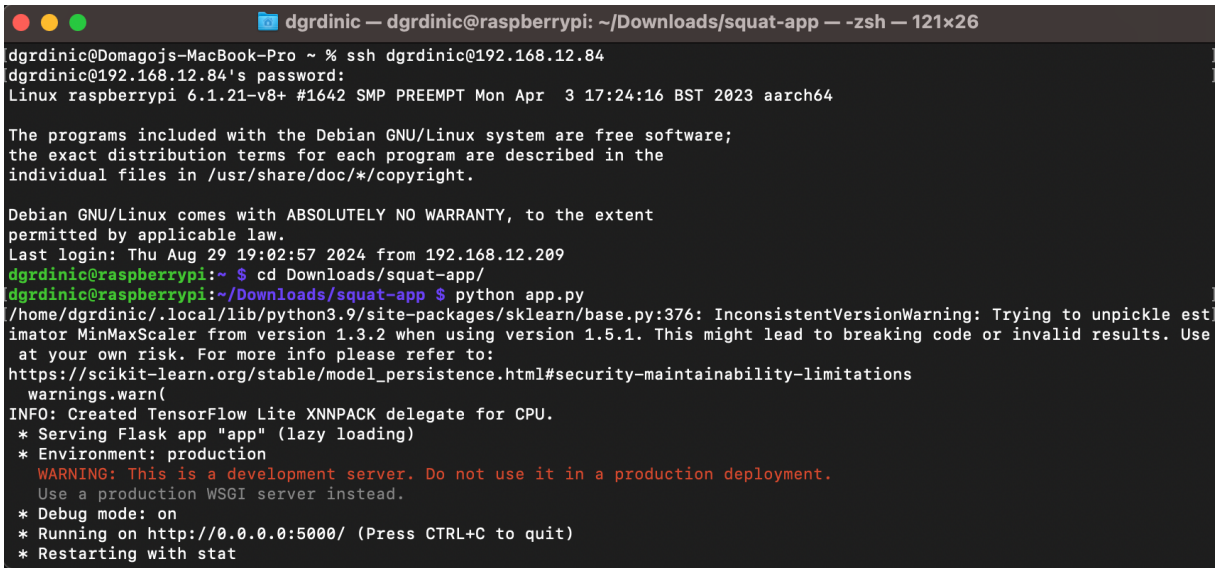


Slika 5.1 Arhitektura ugradbenog sustava

Poslužiteljsku stranu možemo analizirati *high-level* pogledom na slici 5.1 gdje vidimo da se na flask vežu tri glavne cjeline koda. OpenCV koristi *cv.VideoCapture()* funkciju za pristup kameri i dohvaćanje slike. Ta se slika zatim šalje u blok funkcija koje obrađuju sliku, rade predikciju i crtaju točke te vraćaju *new_state* argument, odnosno predikciju stanja u kojem se korisnik trenutno nalazi. Nakon toga isti argument se šalje u *squat_counter* koji ima jednostavni algoritam za brojanje čučnjeva ovisno o trenutnom i prethodnim stanjima korisnika.

Skripta za pokretanje *app.py* pokreće Flask framework koji zatim pokreće WSGI server kako bi se klijenti mogli spojiti na web aplikaciju. Nakon pokretanja aplikacije, korisnik se može

spojiti na frontend upisivanjem IP adrese uređaja i pristupnog porta (npr. http://192.168.12.84:5000). Zahtjev je da je korisnik spojen na istu lokalnu mrežu kao i Raspberry Pi ili neki drugi uređaj na kojem se pokreće aplikacija, i da između njih nema vatrozida koji blokira promet. Da bi se *app.py* skripta pokrenula potrebno je spojiti se putem SSH protokola na Raspberry Pi, locirati dokument gdje se nalazi skripta i zatim ju pokrenuti.



Slika 5.2 Spajanje na RPi i pokretanje skripte

Za slučajeve kad nema vanjskog uređaja, Raspberry Pi može se spojiti na vanjski ekran i upisivanjem adrese http://127.0.0.1:5000 može se spojiti sam na svoju aplikaciju. Zahtjevi sa *frontenda* pokreću i zaustavljaju izvršavanje unutar skripte sa *backend* strane, tako da korisnik ima upravljanje nad aplikacijom.

5.2. Poslužitelj (backend)

Skripta nakon njenog pokretanja kreće sa uvozom potrebnih knjižnica i podizanjem WSGI servera. Nakon toga se inicijaliziraju alati koji će se koristiti za rad sustava. Za skaliranje podataka se inicijalizira *scaler.pkl* model koji se koristi za skaliranje ključnih točaka prije nego se pošalju u model za klasifikaciju. Zatim se inicijalizira MoveNet model za detekciju ključnih točaka i ONNX model za klasifikaciju detektiranih ključnih točaka.

```python
# Load the scaler
scaler = joblib.load('model/scaler.pkl')

# Load TFLite model for keypoint detection
tflite_model_path = 'model/model.tflite'
interpreter = tf.lite.Interpreter(model_path=tflite_model_path)
interpreter.allocate_tensors()

# Load ONNX model for action classification
onnx_model_path = 'model/movenet_NN_model.onnx'
onnx_session = ort.InferenceSession(onnx_model_path)
```

Slika 5.2 Inicijalizirani modeli

Unutar *video_stream()* funkcije implementira se glavni dio programa za prepoznavanje i brojanje čučnjeva u stvarnom vremenu koristeći video iz kamere ili iz unaprijed snimljenog videozapisa. Za dobivanje videa iz kamere koristi se OpenCV knjižnica koja inicijalizira pristup kameri. *Squat_counter* je brojač koji prati koliki je broj čučnjeva korisnik napravio. Za brojanje je zadužen jednostavni algoritam koja prati trenutna i prošla stanja kako bi odredio je li korisnik napravio čučanj. Da bi čučanj bio zabilježen, korisnik mora promijeniti ukupno 5 stanja. Iz no-squat pozicije mora preći u half-squat poziciju, zatim u squat i vratiti se kroz half-squat poziciju ponovno na početak u no-squat poziciju. Sve zajedno predstavlja jedan puni pokret koji čovjek mora napraviti, podijeljen u 5 koraka. Funkcija se izvršava dok se ne postigne unaprijed određeni broj čučnjeva ili dok korisnik ne zaustavi proces.

```python
def video_stream():
    global stop_streaming, squat_counter, frame_output
    current_state = None
    transition_state = None

    camera = cv2.VideoCapture("dataset/videos/videos_squat/1.mp4")  # Use 0 for webcam or provide a video file path

    while not stop_streaming:
        success, frame = camera.read()
        if not success:
            break

        frame = cv2.resize(frame, (640, 980))

        image = preprocess_image(frame)
        keypoints_with_scores, keypoints = predict_keypoints(image, interpreter)

        new_state = predict_action(keypoints, onnx_session)

        # Check for transitions: no-squat → half-squat → squat → no-squat
        if current_state == 0 and new_state == 1:
            transition_state = 1  # Transition from no-squat to half-squat
        elif transition_state == 1 and new_state == 2:
            transition_state = 2  # Transition from half-squat to squat
        elif transition_state == 2 and new_state == 0:
            squat_counter += 1  # Complete cycle and count a squat
            transition_state = None  # Reset transition state

            # Check if squat limit is reached
            if squat_counter >= squat_limit:
                stop_streaming = True

        current_state = new_state
```

Slika 5.5 video_stream() funkcija

Funkcija *preprocess_image()* kao argument dobiva sirovu (engl. *Raw)* sliku koju je OpenCV izrezao iz *video streama* kamere i priprema ju za MoveNet. Prvi korak je konverzija boja iz

BGR formata koji koristi OpenCV u RGB format. Zatim smanjuje rezoluciju na 192 x 192 piksela uz očuvanje omjera slike. Ako je potrebno dodaje se ispuna (engl. *Padding)* kako bi slika imala pravilan oblik. Također se slika proširuje jednom dimenzijom kako bi zadovoljila ulazni format MoveNet modela. Na kraju se slika pretvara u četverodimenzionalno NumPy niza formata [1, 192, 192, 3].

Predict_keypoints() funkcija prvo koristeći *interpreter* dohvati detalje o ulaznim i izlaznima parametrima modela koji se koristi za predikciju, uzima sliku u formatu pripremljenom u prethodnoj funkciji i pretvara ju u uint8 tip podataka. Nakon toga poziva se model za predikciju koji vraća ključne točke i vjerojatnost predikcije. Ti rezultati se preoblikuju kako bi se uklonile suvišne dimenzije. Funkcija vraća dva argumenta, originalne rezultate predikcije *keypoints_with_scores* u formatu [1, 17, 3] i oblikovani niz ključnih točaka *reshaped* koji se koristi u daljnjoj obradi u formatu dvodimenzionalnog NumPy niza formata [17,3]. Niz za svaku od 17 ključnih točaka ima zapisana tri podatka – x os, y os i vjerojatnost predikcije.

Predict_action() je funkcija koja predviđa stanje, odnosno klasu u kojoj se pokret nalazi. Ulazni podaci su *keypoints* (*reshaped* podaci iz *predict_keypoints()* funkcije) i *onnx_session.* Ulazni podaci se preoblikuju u niz formata [1, 51], odradi se skaliranje kako bi se rezultati uskladili s potrebama modela i s pomoću *onnx_session* argumenta se poziva prediktivni model u ONNX formatu. Rezultat koji model vraća je niz vjerojatnosti za svaku klasu (no-squat, half-squat, squat). Funkcijom *argmax()* se odabire klasa koja ima najveću vjerojatnost i ona se vraća kao konačna predikcija.

```python
def preprocess_image(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = tf.image.resize_with_pad(image, 192, 192)
    image = np.expand_dims(image, axis=0)
    return image

def predict_keypoints(image, interpreter):
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    input_image = tf.cast(image, dtype=tf.uint8)
    interpreter.set_tensor(input_details[0]['index'], input_image.numpy())
    interpreter.invoke()

    keypoints_with_scores = interpreter.get_tensor(output_details[0]['index'])
    reshaped = np.squeeze(keypoints_with_scores)
    return keypoints_with_scores, reshaped

def predict_action(keypoints, onnx_session):
    keypoints_reshaped = keypoints.reshape(-1, 51)
    keypoints_scaled = scaler.transform(keypoints_reshaped).astype(np.float16)
    inputs = {onnx_session.get_inputs()[0].name: keypoints_scaled}
    predictions = onnx_session.run(None, inputs)
    return np.argmax(predictions[0], axis=1)[0]
```

Slika 5.3 Funkcije za obradu slike i predikciju

Funkcija *draw_keypoints()* kao argumente zaprima sliku, matricu ključnih točaka i prag povjerenja. Najprije izračunava dimenzije i broj kanala slike te skalira vrijednost ključnih točaka prema dimenzijama slike. Prolazi kroz svaku točku i provjerava vrijednost povjerenja, ako je ona veća od praga povjerenja crta krug na odgovarajućoj poziciji na slici.

Draw_connections() ima funkciju crtati i povezivati ključne točke na slici prikazujući veze između dijelova tijela (npr. Između koljena i stopala). Nakon što izračuna dimenzije i broj kanala slike, skalira koordinatne vrijednosti prema dimenzijama slike. Prolazi kroz svaki par ključnih točaka i provjerava povjerenje za obje ključne točke u paru. Ako povjerenje prelazi zadani prag, crta liniju između točaka. Kombiniranjem ovih funkcija dobijemo vizualizaciju ključnih točaka ljudskog tijela i veze između njih. Tako možemo pratiti pokrete i razumjeti što sustav radi za vrijeme testiranja te nam to može pomoći u otkrivanju potencijalnih nelogičnosti ili problema na modelu.

```
def draw_keypoints(frame, keypoints, confidence_threshold):
    y, x, c = frame.shape
    shaped = np.squeeze(np.multiply(keypoints, [y, x, 1]))

    for kp in shaped:
        ky, kx, kp_conf = kp
        if kp_conf > confidence_threshold:
            cv2.circle(frame, (int(kx), int(ky)), 4, (0, 255, 0), -1)

def draw_connections(frame, keypoints, edges, confidence_threshold):
    y, x, c = frame.shape
    shaped = np.squeeze(np.multiply(keypoints, [y, x, 1]))

    for edge, color in edges.items():
        p1, p2 = edge
        y1, x1, c1 = shaped[p1]
        y2, x2, c2 = shaped[p2]

        if (c1 > confidence_threshold) & (c2 > confidence_threshold):
            cv2.line(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 2)
```

Slika 5.4 Funkcije za crtanje i povezivanje ključnih točaka

Nakon pozivanja gore navedenih funkcija, u nastavku *video_stream()* funkcija na ekranu prikazuje broj napravljenih čučnjeva i trenutno stanje predikcije, za što je zadužena funkcija *cv2.putText()* iz OpenCV knjižnice. Video je ograničen na 30 sličica u sekundi s pomoću *sleep()* funkcije.

Nakon završetka *streama*, *camera.release()* funkcija zaustavlja vezu s kamerom i oslobađa ju za korištenje u nekoj drugoj aplikaciji. Iako je video ograničen na 30 sličica u sekundi, realni prikaz je između 4 i 6 sličica u sekundi zato što predikcija i MoveNet dodatno opterećuju procesor uređaja koji ne može obraditi više podataka u jednoj sekundi. U praksi je to dovoljno za korištenje sustava.

```
        draw_connections(frame, keypoints_with_scores, EDGES, 0.4)
        draw_keypoints(frame, keypoints_with_scores, 0.4)

        cv2.putText(frame, f'Squats: {squat_counter}', (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)
        cv2.putText(frame, f'Current State: {class_names[new_state]}', (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame_output = buffer.tobytes()

        time.sleep(1 / 30.0)

    camera.release()
```

Slika 5.6 video_stream() funkcija 2

Ostatak funkcija koje se nalaze u *app.py* skripti povezuju klijentsku i poslužiteljsku stranu. S pomoću *route* dekoratora se obrađuju zahtjevi koji dođu s klijentske strane i tako pokreću funkcije koje se koriste za otvaranje početne stranice - *index()*, pokretanje video *streama* – *video()* ili zaustavljanje video *streama* - *stop(). Start()* funkcija pokreće proces video *streama* i brojanja čučnjeva koristeći globalne varijable u koje se zapisuju trenutna stanja. Ova funkcija kreira zasebnu dretvu (engl. *Thread*) kako bi se omogućilo paralelno izvođenje video *streama* u zasebnoj dretvi. Glavna funkcija koja pokreće Flask aplikaciju je *app.run(),* a prije toga se provjerava izvodi li se skripta direktno ili je uvezena kao modul. Argument *„host=“0.0.0.0““* omogućava pristup uređaju na svim mrežnim sučeljima kako bi se aplikaciji moglo pristupiti s nekog drugog uređaja.

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video')
def video():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/start')
def start():
    global stop_streaming, video_thread, squat_counter
    if stop_streaming:
        stop_streaming = False
        squat_counter = 0
        video_thread = threading.Thread(target=video_stream)
        video_thread.start()
    return "Streaming started"

@app.route('/stop')
def stop():
    global stop_streaming
    stop_streaming = True
    return "Streaming stopped"

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
```

Slika 5.7 route dekoratori i app.run() funkcija

5.3. Klijentska strana (Frontend)

Korisničko sučelje zapisano je u *index.html* dokumentu koji se nalazi unutar */templates* datoteke. Na sučelje se spajamo upisivanjem IP adrese Raspberry Pi uređaja u Internet preglednik. HTML datoteka ima jednostavnu strukturu i prikazuje okvir unutar kojeg se nalazi video stream i dva gumba start i stop za pokretanje, odnosno završetak video streama i brojanja čučnjeva. Pritiskom na gumb start poziva se route dekorator */start* na backendu te započinje video stream i brojanje čučnjeva. Nakon 10 napravljenih čučnjeva, skripta se završava i za ponovno pokretanje je potrebno pritisnuti gumb *Start*.

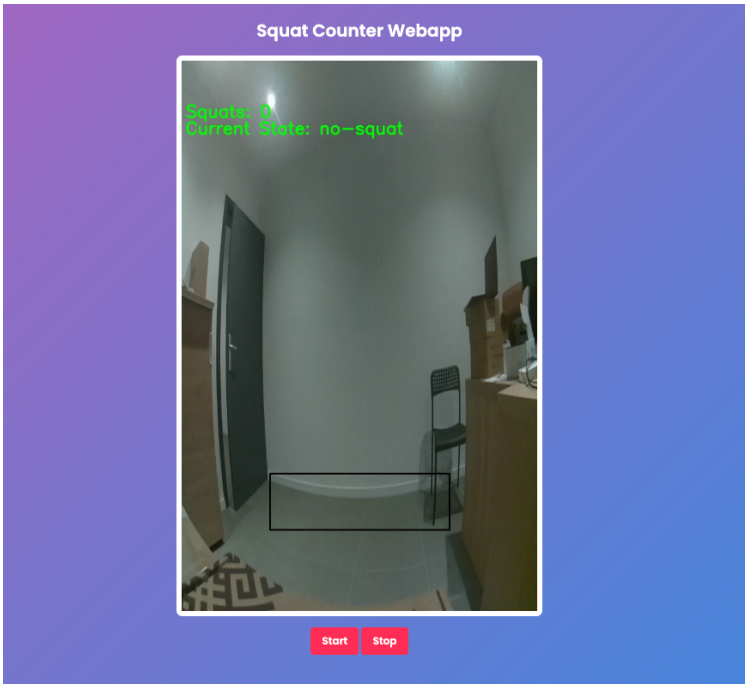
```
<body>
    <div class="title">Squat Counter Webapp</div>
    <div class="camera-container">
        <img src="{{ url_for('video') }}" class="camera-feed" alt="Camera Feed">
    </div>
    <div class="controls">
        <button id="start-btn" onclick="startStreaming()">Start</button>
        <button id="stop-btn" onclick="stopStreaming()">Stop</button>
    </div>

    <script>
        function startStreaming() {
            fetch('/start')
                .then(response => response.text())
                .then(data => console.log(data));
        }

        function stopStreaming() {
            fetch('/stop')
                .then(response => response.text())
                .then(data => console.log(data));
        }
    </script>
</body>

</html>
```

Slika 5.8 Index.html

6. TESTIRANJE

Kako bi osigurali pouzdanost i učinkovitost sustava za prepoznavanje i brojanje čučnjeva, uzimajući u obzir varijacije između različitih demografskih skupina, testiranje je rađeno na 20 ispitanika ravnomjerno podijeljenih u dvije kategorije, na muškarce i na žene. Muškarci i žene razlikuju se u tjelesnoj građi, načinu kretanja kao i u fleksibilnosti zglobova što može utjecati na efikasnost sustava.

Kako bi se minimalizirali vanjski utjecaji koji bi mogli promijeniti ishod testiranja, svi ispitanici su imali identične uvjete testiranja. Testiranje se odvijalo u zatvorenoj prostoriji, bez prisutnosti prirodnog svjetla, a osigurana je kontrolirana rasvjeta putem stropnih svjetala. Pozadina je bila neutralne bijele boje kako bi se osigurao kontrast između ispitanika i okoline, tako olakšavajući sustavu prepoznavanje tjelesnih pokreta. Na slici ispod (*Slika 6.1*) može se vidjeti označeni prostor unutar kojeg su ispitanici trebali stajati za vrijeme testiranja i izvoditi čučnjeve. Ispitanici su tako postavljeni u optimalnu poziciju u odnosu na kameru koja se nalazila na visi od 100 cm, a ispitanici su stajali licem okrenuti prema njoj na udaljenosti od 180 do 200 cm, ovisno o visini ispitanika. Ovakve postavke osiguravaju optimalan kut snimanja i osiguravaju da su svi pokreti vidljivi kameri.



Slika 6.1 Korisničko sučelje

Pozicije ispitanika su detaljno ispitane prije njihovog testiranja kako bi se pronašla optimalna pozicija testiranje. Kroz dane, odnosno tjedne rada i testiranja napravljeno je više od 1000 čučnjeva kako bi se pronašla optimalna pozicija i njen raspon te kako bi svaki korisnik mogao optimalno testirati rad sustava.

Primarni cilj ovog testiranja bilo je procijeniti točnost, preciznost i ukupne performanse sustava u prepoznavanju čučnjeva kod žena i muškaraca. Sustavu je zadatak prepoznati i klasificirati pokrete kao čučnjeve i ne-čučnjeve, uzimajući u obzir razlike između različitih spolova. Posebna pažnja je posvećena pronalaženju potencijalnih problema do kojih bi mogla dovesti ta ponajprije razlika u fizionomiji, a sve s ciljem bolje optimizacije sustava za prepoznavanje čučnjeva.

Za vrijeme istraživanja se uređaj povremeno prekomjerno zagrijavao (preko 80 stupnjeva), što je znalo dovesti do pada performansi. Kako bi izbjegli potencijalni problem i kako se integritet testiranja ne bi izgubio između dva testiranja odrađena je pauza te gašenje uređaja.

6.1. Karakteristike ispitanika i rezultati istraživanja

Kako bi se obuhvatio čim širi spektar fizičkih karakteristika ispitanici su odabrani s ciljem da se osigura reprezentativnost i raznolikost podataka. U ispitivanje je ušlo 10 muškaraca i 10 žena u rasponu od 18 do 55 godina. Dobna raznolikost je omogućila uvid u to kako sustav razlikuje čučnjeve mlađih i nešto starijih osoba, što je važno zbog samih tjelesnih sposobnosti koje se s godinama kod ljudi mijenjaju. Visina i težina su također faktori koji su uzeti u obzir, raspon visine se kretao od 160 do 193 centimetra, dok je težinski raspon od 51 do 93 kilograma. Ove su razlike važne za provjeru sustava kod osoba različitih tjelesnih proporcija.

Tablica 6.1 Karakteristike ispitanika

M(Visina/težina)	Godine	Ž(Visina/Težina)	Godine
183/79	27	160/51	23
173/65	21	173/65	31
191/81	19	171/59	28
193/93	32	163/53	21
177/73	55	177/73	54
174/93	25	168/88	29
181/81	35	171/63	25
177/77	27	164/59	29
189/78	18	169/53	31
169/65	43	159/51	29

Svim ispitanicima je pokazan primjer ispravnog izvođenja čučnja, kao i tempo rada, gdje za izvođenje jednog čučnja treba u prosjeku dvije do tri sekunde. Ovisno o visini ispitanika, stajali su na udaljenosti od otprilike 180-200 centimetara od kamere i licem gledali prema kameri, uz odstupanje od par stupnjeva kako bi sustav prepoznao dubinu točaka. Svaki ispitanik je radio 10 ispravnih čučnjeva, a rezultati su zapisani u tablicu.

Tablica 6.2 Rezultati testiranja muškarci

M(Visina/Težina)	Godine	Broj napravljenih čučnjeva	Broj nezabilježenih čučnjeva	Broj prepoznatih čučnjeva
183/79	27	10	1	9
173/65	21	10	1	8
191/81	19	10	0	10
193/93	32	10	1	9
177/73	55	10	1	9
174/93	25	10	3	7
181/81	35	10	1	9
177/77	27	10	2	8
189/78	18	10	0	10
169/65	43	10	1	9

Tablica 6.3 Rezultati testiranja žene

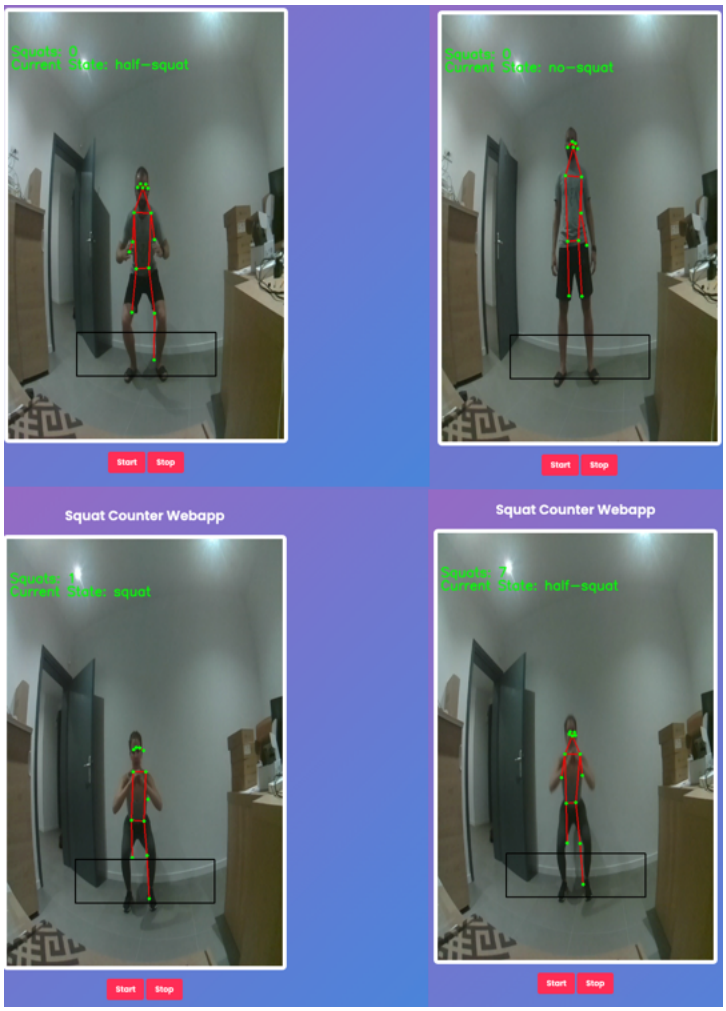
Ž(Visina/Težina)	Godine	Broj napravljenih čučnjeva	Broj nezabilježenih čučnjeva	Broj prepoznatih čučnjeva
160/51	23	10	1	9
173/65	31	10	1	9
171/59	28	10	1	9
163/53	21	10	1	9
177/73	54	10	1	9
168/88	29	10	2	8
171/63	25	10	0	10
164/59	29	10	0	10
169/53	31	10	0	10
159/51	29	10	1	9

Iz tablica s rezultatima za muškarce i za žene, mogu se izračunati metrike kako bi se objektivno mogla napraviti usporedba rezultata između muškaraca i žena. Izračunate su metrike korištene i za odabir klasifikatora, a to su točnost, preciznost, osjetljivost i F1 Rezultat.

Tablica 6.4 Usporedba metrika

	Muškarci	Žene
Precision	1	1
Recall	0,89	0,92
F1	0,942	0,958
Accuracy	0,89	0,92

S obzirom na to da ni u jednom slučaju nisu zabilježeni čučnjevi koji to nisu bili, preciznost je savršena, a razlika se vidi u osjetljivosti i točnosti, kao i F1 Rezultatu. Rezultati ukazuju da primjetne razlike između muškaraca i žena nema, te da sustav jednako dobro radi neovisno o tome tko se nalazi s druge strane kamere. Sa subjektivne strane, kao promatrač za vrijeme testiranja, može se reći da žene zbog fizionomije tijela učestalije i jednostavnije rade „dublji“ čučanj, koji model lakše svrstava u *squat klasu* i zbog toga su rezultati nešto bolji. Jedna testirana osoba je izbačena iz istraživanja i tretirana je kao iznimka (engl. *Outlier*). Razlog za to je povreda i operacija koljena, zbog koje testirana osoba nije mogla postići poziciju čučnja u više od 50 % testiranja, te bi njeno uvrštavanje povrijedilo integritet ovoga testiranja. Za takve osobe bi trebalo prilagoditi model, jer iako nemaju pun opseg pokreta, postoje alternativni načini kako osobe s povredama rade čučanj.



Slika 6.2 Testiranje sustava

Slika 6.2 prikazuje ispitanike za vrijeme testiranja, kao i točke koje je MoveNet model predvidio. Na ekranu je također vidljiv prikaz broja čučnjeva i trenutno stanje predikcije položaja korisnika.

7. ZAKLJUČAK

U ovom radu predstavljen je ugradbeni sustav za detekciju čučnjeva implementiran na Raspberry Pi uređaju, koristeći MoveNet model za prepoznavanje ključnih točaka ljudskog tijela, te Feedforward Neuronske Mreže (FFNN) za klasifikaciju pokreta. Klasifikacija pokreta se dijelila na tri klase – squat, half-squat i no-squat. Alati i tehnologije koje se još koriste su OpenCV knjižnica za dohvaćanje slika s kamere i Flask framework za izradu korisničke i poslužiteljske strane web aplikacije. Također je korištena i ONNX tehnologija kao standard za model koji se može koristiti između više različitih tehnologija i platformi.

Sakupljen je i obrađen skup podataka za treniranje modela, napravljena je usporedba između Logističke regresije, SVM-a i Feedforward Neuronske Mreže kako bi se odabrao najbolji klasifikator. Nakon usporedbe je odabrana FFNN mreža, koja je istrenirana i prilagođena sustavu. Nakon odabira modela, odrađeno je testiranje na 20 ispitanika, 10 muškaraca i 10 žena, različite životne dobi i fizičkih predispozicija. Jedan korisnik je uklonjen iz istraživanja jer mu ozljeda koljena nije dozvoljavala dovoljan pokret kako bi se izvršilo testiranje. Povremeno su se mogli primijetiti padovi performansi pri dužem radu na sustavu što se može pripisati prekomjernom zagrijavanju uređaja uslijed visokih procesorskih zahtjeva sustava.

Rezultati testiranja su pokazali da nema razlike u radu sustava između muškaraca i žena, što ukazuje da konzistentnost rada sustava. Nije primjetna razlika ni u drugim kategorijama kao što su godine, visina ili kilaža, tako da možemo zaključiti da sustav radi neovisno o karakteristikama ispitanika.

Može se donijeti zaključak da razvijeni ugradbeni sustav za detekciju čučnjeva radi ispravno i pokazuje se kao koristan alat. Posebno ako uzmemo u obzir korištene komponente (Raspberry Pi) i kvalitetu ulaznih podataka (kamera), zaista je nevjerojatno kakvi se rezultati mogu postići s malo ulaganja u opremu. Nekada su ovakvi sustavi bili veliki i skupi, a danas stanu u uređaj veličine dlana i mogu se složiti u kućnoj radinosti.

8. SAŽETAK

Tema rada je izrada ugradbenog sustava implementiranog na Raspberry Pi uređaju i web aplikaciju za brojanje čučnjeva uz korištenje MoveNet modela za detekciju ključnih točaka i Feedforward Neuralne Mreže za klasifikaciju pokreta. Detaljno je razrađen proces razvoja i korišteno sklopovlje, kao i tehnologije i alati korišteni za razvijanje ovog sustava.

Ključne riječi **– Raspberry Pi, Flask, Python, TensorFlow, MoveNet, Neuralne mreže, Strojno učenje, ONNX**

ABSTRACT

The topic of the work is the creation of an embedded system implemented on Raspberry Pi device, and a web application for counting squats using the MoveNet model for key point detection and Feedforward Neural Network for movement classification. The development process and circuit used, as well as the technologies and tools used to develop the system, have been elaborated in detail.

Keywords **– Raspberry Pi, Flask, Python, TensorFlow, MoveNet, Neuralne mreže, Strojno učenje, ONNX**

Popis literature

[1] Perazzo, Daniel, et al. "OAK-D as a platform for human movement analysis: A case study." Proceedings of the 23rd Symposium on Virtual and Augmented Reality. 2021.

[2] Dudekula, Khasim Vali, et al. "Physiotherapy assistance for patients using human pose estimation with raspberry pi." ASEAN Journal of Scientific and Technological Reports 27.4 (2024): e251096-e251096.

[3] Lee, Min-Fan Ricky, Yen-Chun Chen, and Cheng-Yo Tsai. "Deep learning-based human body posture recognition and tracking for unmanned aerial vehicles." Processes 10.11 (2022): 2295.

[4] Papandreou, George, et al. "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

[5] Last access: Aug 30, 2024. "Implementation of MoveNet", URL: https://analyticsindiamag.com/ai-mysteries/how-to-do-pose-estimation-with-movenet/

[6] Toporov, C.; "Squats detector with OpenCV and Tensorflow", s Interneta, https://towardsdatascience.com/squats-detector-with-opencv-and-tensorflow-ce934f19aeb9, 3. kolovoza 2024.

[7] "TensorFlow Lite", s Interneta, https://www.tensorflow.org/lite/guide

[8] "Pose estimation", s Interneta, https://www.tensorflow.org/lite/examples/pose_estimation/overview

Popis slika

Popis tablica

Popis tablica treba biti izrađen po uzoru na indeksirani sadržaj, te upućivati na broj stranice na kojoj se tablica može pronaći.