Upravljanje bespilotnom letjelicom pomoću ljudske gestikulacije

Maček, Domagoj

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

UPRAVLJANJE BESPILOTNOM LETJELICOM POMOĆU LJUDSKE GESTIKULACIJE

Završni rad br. 04/RAČ/2022

Domagoj Maček

Bjelovar, listopad 2023.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Domagoj Maček** JMBAG: **0314022108**

Naslov rada (tema): **Upravljanje bespilotnom letjelicom pomoću ljudske gestikulacije**

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

Grana: **Umjetna inteligencija**

Mentor: **Ante Javor, struč. spec. ing. comp.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **dr.sc. Zoran Vrhovski, predsjednik**
2. **Ante Javor, struč. spec. ing. comp., mentor**
3. **Krunoslav Husak, dipl. ing. rač., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 04/RAČ/2022

U sklopu završnog rada potrebno je:

1. Analizirati i opisati postojeća konvencionalna rješenja za upravljanje bespilotnom letjelicom
2. Dizajnirati i opisati rješenje bazirano na upravljanju ljudskom gestikulacijom
3. Implementirati projektno rješenje za upravljanje bespilotnom letjelicom temeljeno na ljudskoj gestikulaciji
4. Implementirati i opisati tehničke funkcionalnosti upravljanja bespilotnom letjelicom
5. Dizajnirati i opisati tehničku arhitekturu sustava, poput: programske podrške, komunikacijskih protokola i algoritama.

Datum: 13.07.2022. godine

Mentor: **Ante Javor, struč. spec. ing. comp.**

REPUBLIKA HRVATSKA 2 BJELOVAR VELEUČILIŠTE U BJELOVARU

Zahvala: Zahvaljujem svojem mentoru na uputama i smjernicama tijekom izrade ovog završnog rada. Zahvaljujem se i članovima povjerenstva koji su sudjelovali u analizi završnog rada te svim ostalim djelatnicima Veleučilišta u Bjelovaru s kojima sam surađivao tijekom studiranja na svim stečenim znanjima iz područja računarstva. Zahvaljujem se također i svojoj obitelji na podršci

Sadržaj

1. Uvod

Danas se može primijetiti česta uporaba bespilotnih letjelica bilo u pozitivne ili negativne svrhe. Budući da je namjena raznolika, postoje različiti načini upravljanja. Ono što je zajedničko svim ovakvim bespilotnim letjelicama jest to da njima upravljaju operatori sustava bespilotnih letjelica. Zbog toga je upravljanje bespilotnih letjelica na daljinu uvijek bila zanimljiva tema istraživanja.

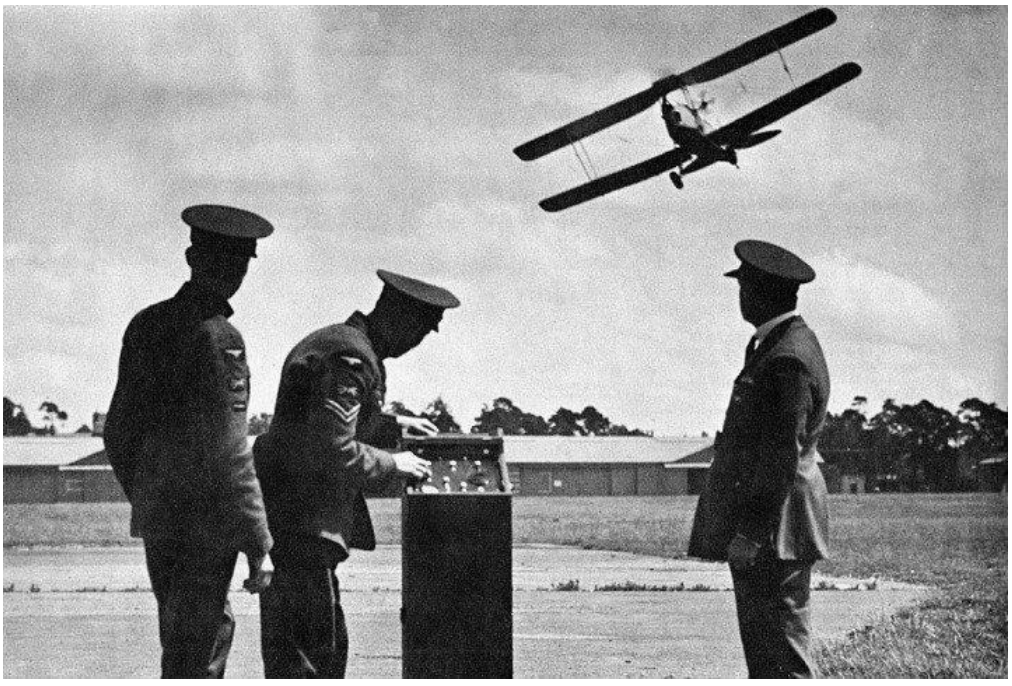
Današnji najučestaliji načini za upravljanje ovakvim vrstama bespilotnih letjelica su putem daljinskog upravljača s prikladnim kontrolama ili preko mobilnih uređaja. To je uglavnom zato što su to komercijalne bespilotne letjelice s jednostavnim načinom upravljanja dostupne velikom broju ljudi i lako se mogu pronaći u prodaji. Naravno, postoje i bespilotne letjelice s naprednijim sustavima za upravljanje. Razvojem tehnologije kroz godine, koja se i danas kontinuirano razvija, donesene su brojne mogućnosti koje se mogu implementirati u sustave bespilotnih letjelica za njihovo upravljanje. Da bi korisnik mogao uspješno savladati napredne načine upravljanja, mora biti dobro upućen u način upravljanja i imati određeno razumijevanje tehnologije kojom se upravlja bespilotna letjelica. Uz to treba biti i detaljno upućen u sposobnosti i ograničenja bespilotne letjelice. Jedan od naprednih sustava današnjih modernih bespilotnih letjelica vezan je za primjenu umjetnu inteligencije u sustavima upravljanja bespilotnim letjelicama.

U ovom završnom radu implementiran je napredan sustav upravljanja bespilotnom letjelicom. U prvom poglavlju istaknuti su važni povijesni trenutci bespilotnih letjelica i načini njihova upravljanja. Drugo poglavlje obuhvaća implementaciju za upravljanje bespilotne letjelice pomoću detekcije lica. U trećem poglavlju slijedi implementacija gestikulacija ruku kroz konvolucijske neuronske mreže. Na kraju će biti izložen zaključak u kojem će biti spomenuti rezultati ovog rada te za što se ovakva bespilotna letjelica može koristiti.

2. UPRAVLJANJE BESPILOTNIM LETJELICAMA

2.1 Povijest bespilotnih letjelica

Bespilotne letjelice (engl. *Unmanned Aerial Vehicles*) su letjelice koje su kontrolirane na daljinsko upravljanje ili se mogu upravljati autonomno koristeći softverska rješenja u svojim ugrađenim sustavima [1]. Preteča ovakvih letjelica pojavila se 1849. tijekom napada Austrije na Veneciju koristeći bespilotne balone napunjene eksplozivom koje su lansirali na grad. Prvi kvadrokopter, vrsta bespilotne letjelice s 4 propelera, pojavila se 1907. godine [2]. Ta bespilotna letjelica bila je neuporabljiva za nekakvu vrstu namjene, ali 1917. godine Ruston Proctor Aerial Target postaje prva bespilotna letjelica s krilima u povijesti. Upravljala se radio signalima. Ova letjelica se može se vidjeti na slici 2.1 [3].



Slika 2.1: RAF testira prvu bespilotnu letjelicu [3]

Zamišljena je kao leteća bomba za ratne svrhe koja bi se mogla navoditi i na taj način uništiti neprijatelje. Unatoč obećavajućim testovima i demonstracijama na kraju nikad nije bila korištena, ali otvorila je put ostalim vojnim bespilotnim letjelicama. Tehnološkim razvojem kroz nekoliko desetljeća doveo je do bespilotne letjelice General Atomics MQ-1 poznate kao Predator koja je predstavljena 2000. godine. Korištena je u Afganistanu za lansiranje projektila i potragu za

teroristima. Tiha je i nevidljiva osobama na tlu, što je još jedna velika prednost u modernom ratovanju na područjima gdje ne postoje radari. Može raditi samostalno ili pod kontrolom posade. Upravljanje je vrlo kompleksno. Opremljena je senzorima, kamerama i oružjem, zemaljske kontrolne jedinice i primarne satelitske veze u slučaju da je direktna veza onemogućena. Primjer takve bespilotne letjelice prikazan je na slici 2.3.

Slika 2.3: MQ-1 Predator [4]

Od 2010. godine kada je francuska tvrtka Parrot izdala bespilotnu letjelicu pod nazivom Parrot AR Drone [3], prvu bespilotnu letjelicu koja se mogla upravljati putem WiFi veze preko mobilnog uređaja. Do danas se dogodio veliki procvat inovacija bespilotnih letjelica. Koriste se u razne civilne svrhe, poput prijevoza robe u obliku narudžbi (Amazon), za potrebe agrikulture, za snimanje pejzaža, za zabavu, itd. Većina takvih bespilotnih letjelica nazivaju se kvadrokopteri, bespilotne letjelice s 4 propelera. Ovakva vrsta bespilotne letjelice je korištena i za svrhe ovog završnog rada i prikazana je na slici 2.4.

Slika 2.4: Kvadrokopter [2]

Ovisno o namjeni i zakonima pojedine zemlje za ovakve bespilotne letjelice morale su se početi izdavati dozvole zbog sve većeg interesa i porasta broja korisnika. Sredinom prošlog desetljeća savezna uprava za civilno zrakoplovstvo Sjedinjenih Američkih Država (engl. *Federal Aviation Administration*) bilježi ogroman rast potražnje za dozvolama, točnije oko 1000 komercijalnih dozvola. Ovaj broj se utrostručio već iduće godine. Jedan od većih razloga tome je što su razvojem mobilnih uređaja prijašnji tipovi upravljači za upravljanje ovakvih bespilotnih letjelica postali gotovo nepotrebni. Većina komercijalnih letjelica ima jednostavna razvijena sučelja koja predstavljaju virtualne komande preko kojih se jednostavno upravlja bespilotnim letjelicama [2]. Sadašnjost i budućnost bespilotnih letjelica je obećavajuća zbog razvoja tehnologije, ponajprije u računalnom vidu i strojnom učenju [3]. Doći će i do poboljšanje brojne opreme i sustava letenja bespilotnih letjelica. Predviđa se da će se globalna isporuka bespilotnih letjelica za 2023. godinu povećati na 2.4 milijuna uz godišnju stopu rasta od 66.8%. Vodeće industrije kod kojih će doći do ovog povećanja su poljoprivreda, građevine i rudarstvo, osiguranje, mediji i telekomunikacije. S godinama će se broj povećavati i u ostalim industrijama dok se dodatno istraže koristi koje bi ovakve bespilotne letjelice donijele [2].

2.2 Tello Drone

Za potrebe demonstracije završnog rada izabrana je bespilotna letjelica Tello Drone. Radi se o proizvodu kineskih tvrtki Ryze Tech i DJI te se lako se može pronaći na tržištu. To je mala i lagana bespilotna letjelica namijenjena djeci i odraslima. Samim time što se ovakva bespilotna letjelica može programirati, pogodna je za implementaciju raznih sustava s kojima će se njome moći upravljati. Pomoću letjelice se može snimati video, slikati, izvoditi akrobacije u zraku itd.

2.2.1 ***Sklopovlje***

Tello bespilotna letjelica je kvadrokopter stoga sadrži 4 propelera. Ta 4 propelera pogone mali motori pojedinačno za svaki propeler. U svrhu zaštite propelera koriste se i štitnici. Kamera ima 5 mega piksela, elektronsku stabilizaciju slike i 1280x720 rezoluciju te vidno polje od 82.6°. Može prenositi video u stvarnom vremenu s kašnjenjem manjim od jedne sekunde na pametni telefon [5, 6]. Primjer ovakve letjelice prikazan je na slici 2.5.

Slika 2.5: Tello drone bespilotna letjelica [7]

2.2.2 ***Moduli***

Tello koristi vodeće komponente u industriji koji mu omogućuju stabilan let. To mu omogućuju senzori i procesorska jedinica za obradu slike Intel Movidius Myriad 2 VPU pomoću kojih vidi svijet oko sebe i lebdi s velikom preciznošću. Ova bespilotna letjelica također sadrži sustav pozicioniranja, koji je vrlo bitan sustav jer pomaže u održavanju trenutnog položaj u kojem se nalazi. Dodatno omogućuje stabilnije lebdenje i letenje pri brzini od 8 m/s , kako u zatvorenom prostoru, tako i na otvorenom, bez prisutnosti jakog vjetra. Ovaj vrlo bitan sustav efikasan je na visini od 0.3 do 30 m, a najbolje radi do 6 m stoga se na većim visinama mogu događati povremena trzanja u zraku. Glavne komponente sustava pozicioniranja su kamera i 3D infracrveni modul smješten na donjoj strani letjelice [5, 6].

2.2.3 ***Sučelje***

Tello Drone može biti upravljan na dva načina. Postoji službena aplikacija Tello koja se može preuzeti s Google Play trgovine za Android, App Store trgovine za iOS ili sa službene stranice DJI. Aplikacija obuhvaća sve potrebne funkcionalnosti za upravljanje letjelicom i omogućuje kalibraciju kamere i to se sve odvija preko mobitela koji služi poput virtualnog upravljača [8]. Na tom sučelju nalaze se komande za podizanje, spuštanje, kretnje u svim smjerovima, pokretanje kamere itd. Sadrži još i informacije o povezanosti putem WiFi veze, kapacitetom baterije, brzini i visini na kojoj se letjelica nalazi. Ovo sučelje prikazano je na slici 2.6.

Slika 2.6: Tello drone sučelje [8]

2.2.4 ***Programiranje bespilotne letjelice***

Bespilotna letjelica se može programirati u Scratch, Swift i Python programskom jeziku. Za završni rad odabran je programski jezik Python 3.10. Prije nego što se krene s programiranjem potrebno je aktivirati dron putem aplikacije na način da ga se spoji na Wi-Fi i pokrene. Za potrebe programiranja potrebno je preuzeti djitellopy biblioteku [9]. U programskom kodu 2.1 prikazana je jednostavna instalacija spomenute biblioteke

.

Programski kod 2.1: Naredba za preuzimanje djitellopy biblioteke

```
pip install djitellopy
```

Biblioteka *djitellopy* je otvorenog koda što znači da je svi mogu koristiti te je po želji i modificirati za potrebe vlastitog projekta. Tello klasa sadrži metode za upravljanje bespilotnom letjelicom. U programskom kodu 2.2 prikazan je kod za jednostavno upravljanje letenja bespilotne letjelice.

Programski kod 2.2: Jednostavan primjer za pokretanje letjelice

```
from djitellopy
import Tello
tello = Tello()
tello.connect()
tello.takeoff()
tello.move_up(100)
tello.land()
```

Uspostavlja se mrežna veza s bespilotnom letjelicom te se zadaje naredba za polijetanje. Bespilotna letjelica se zatim pomakne još za 100 cm u zrak, a nakon toga se spusti i program

prestaje s radom. Klasa Tello sadrži više od 100 metoda s kojima se upravlja bespilotnom letjelicom [10]. Neke od najosnovnijih i najčešće korištenih metoda te njihova namjena su prikazane u tablici 1.1.

Tablica 2.1: Osnovne metode za rad bespilotne letjelice [9]

metoda	namjena
connect(self)	povezivanje Tello bespilotne letjelice putem Wi-Fi veze
takeoff(self)	bespilotna letjelica polijeće i lebdi na mjestu
land(self)	bespilotna letjelica se spušta i prekida s radom
streamon(self)	pristup kameri bespilotne letjelice
get_frame_read(self)	dohvaćanje trenutne slike
move_up(self, x: int)	bespilotna letjelica se uzdiže prema gore x cm
move_down(self, x: int)	bespilotna letjelica se spušta prema dolje x cm
move_left(self, x: int)	bespilotna letjelica se kreće u lijevo x cm
move_right(self, x: int)	bespilotna letjelica se kreće u desno x cm
move_forward(self, x: int)	bespilotna letjelica se prema naprijed x cm
move_back(self, x: int)	bespilotna letjelica se prema nazad x cm

Navedene su samo neke osnovne upravljačke metode i metode pomoću kojih se uspostavlja komunikacija s bespilotnom letjelicom. Klasa Tello sadrži i informacijske metode pomoću kojih se mogu dobiti podaci sa senzora bespilotne letjelice kao što su podaci o temperaturi, tlaku, itd.

3. DETEKCIJA LICA

Detekcija lica (engl. *Face Detection*) je skup postupaka unutar računalnog vida koji imaju sposobnost detektiranja ljudskog lica sa slike, videa ili kamere uživo. Najbolji primjer ove tehnologije se nalazi u današnjim pametnim mobitelima i laptopima koja dolazi kao već ugrađen softver. Koristi se najčešće u kombinaciji s identifikacijom korisnika prilikom otključavanja zaslona pametnih uređaja. Postoje brojne aplikacije koje mogu snimiti i obraditi lice u stvarnom vremenu te primijeniti brojne filtere na lice, što čini ovu tehnologiju umjetne računalne inteligencije izrazito popularnom kod društvenih mreža.

Da bi bespilotna letjelica bila uspješno upravljana licem to znači da će ona morati detektirati lice i potom ga pratiti. Potrebno je implementirati sustav koji će znati jasno prepoznati lice koje će letjelica potencijalno identificirati preko svoje ugrađene kamere. Zatim se na temelju pomaka lica daju instrukcije o smjeru kretanja letjelice ovisno o poziciji lica na slici.

3.1 Viola-Jones algoritam

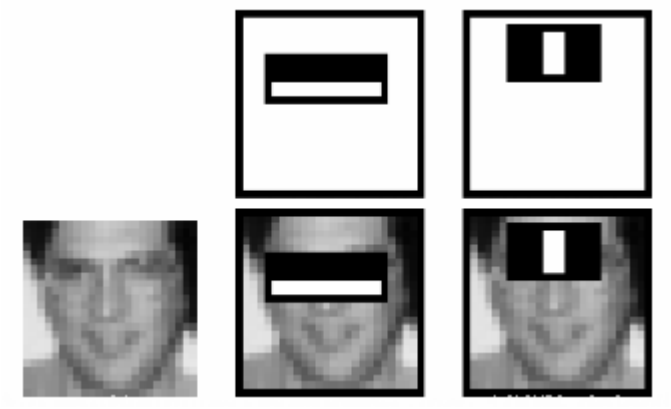
Viola-Jones je algoritam koji se koristi za detekciju objekata pa tako i lica [11]. Ime je dobio po uzoru na njegove kreatore. Paul Viola i Michael Jones, dva istraživača računalnog vida, su 2001. godine predložili rad pod naslovom "*Rapid Object Detection using Boosted Cascade of Simple Features"* te ponudili rješenja za probleme detekcije objekata, prvenstveno lica. Rješenje je bilo precizno i učinkovito te se može implementirati za aplikaciju za detekciju lica u stvarnom vremenu [12]. Algoritam Viola-Jones temelji se na 4 ključna koraka:

- odabir Haar značajki
- izrada integralne slike
- pokretanje AdaBoost treninga
- stvaranje kaskada klasifikatora

Na kraju kada se svaki korak završi, algoritam bi trebao lokalizirati poziciju lica na slici ako slika sadrži nečije lice. Postoje i alternativna rješenja za detekciju lica kao što su konvolucijske neuronske mreže, ali Viola-Jones algoritam manje je memorijski zahtjevan te je vrlo brz i jednostavan za implementaciju [13]. Zbog toga se koristi u mobilnim aplikacijama i prenosivim uređajima.

3.1.1 **Odabir Haar značajki**

Haar značajke koriste se u prepoznavanju objekata, u ovom slučaju lica. Iako su sva ljudska lica različita ipak sadrže neke sličnosti, odnosno zajedničke značajke. Gotovo kod svakog čovjeka područje oko očiju je tamnije ili su obrazi svjetliji od područja očiju. Ovakva svojstva mogu pomoći u detekciji lica sa slike. Postoje 3 vrste osnovnih značajki koje mogu pomoći u pronalasku lica a to su rubne, linijske te značajke četiri pravokutnika [14]. Te značajke prikazane su na slici 3.1.

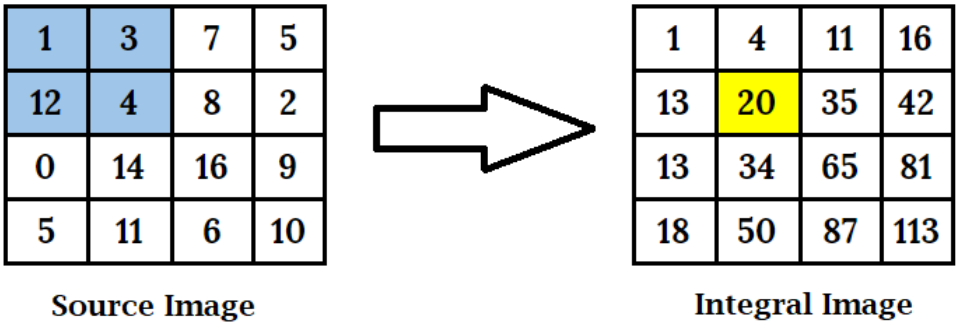


Slika 3.1: Haarcascade značajke [15]

Ukoliko bi jedna linijska značajka s bijelom linijom u sredini i dvije susjedne crne linije došla do područja lica gdje se nalaze nos i oči, uspjela bi sugerirati da se na toj lokaciji potencijalno nalazi lice. Taj proces radi na način da se sumiraju vrijednosti crnih i bijelih piksela unutar jednog okvira. Treba napomenuti da oni u realnosti nisu nikada potpuno bijeli i potpuno crni. Ukoliko je ta vrijednost veća od zadanog praga, detektirana je prva značajka na licu i samim time algoritam je bliže potpunog pronalaska lica na slici. Ipak, postoji jedan problem, a to je da algoritam izračunava mnoge od ovih značajki u ostalim dijelovima slike, što postaje računski zahtjevno zbog prevelikog broja značajki koje slika može sadržavati. Zbog ovog problema uveden je postupak izračunavanja integralne slike.

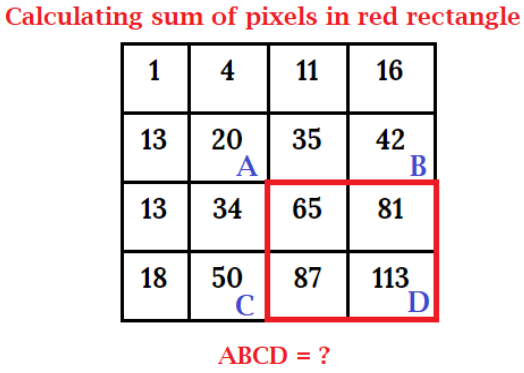
3.1.2 **Integralna slika**

Koncept integralne slike poznate kao i tablica zbrojene površine koristi brži i učinkovitiji način za izračunavanje zbroja vrijednosti piksela na slici ili pravokutnom dijelu slike [16]. U integralnoj slici, vrijednost u svakoj točki određena je zbrajanjem svih piksela koji se nalaze iznad i koji se nalaze lijevo te samog ciljanog piksela kao što je prikazan na slici 3.2.



Slika 3.2: Dobivanje integralne slike sumiranjem piksela [16]

Integralna slika može se izračunati u jednom prijelazu preko izvorne slike. Ovom metodom smanjuje se broj operacija na samo tri operacije s četiri vrijednosti bez obzira na veličinu značajke. Dobivena integralna slika sada može poslužiti za brže pronalaženje značajke na način da više nije potrebno izračunavati svaki piksel posebno nego su nam sada potrebne rubne vrijednosti značajke kao što je prikazano na slici 3.3



Slika 3.3: Postupak izračunavanje sume piksela u pravokutnom području [17]

Vrijednost D je u ovom slučaju vrijednost svih piksela lijevo i iznad s originalne slike. Isti slučaj je i s vrijednostima A, B i C. Od vrijednosti D oduzima se vrijednost B i C te se dodaje vrijednost A zbog toga što se dva puta oduzela prilikom oduzimanja vrijednosti B i C. Ovaj postupak je kraće zapisan u sljedećoj formuli 2.1.

$$\sum_{ABCD} A + B + C + D \tag{2.1}$$

Ovim postupkom se vremenska kompleksnost $O(N^2)$ svela na $O(1)$ što znači da je sada konstanto vrijeme izračuna bez obzira na veličinu značajke. Sada kada je moguće jednostavnije izračunavati vrijednosti piksela značajki trebalo bi se odrediti još koje značajke i njihove veličine će se koristiti za pronalaženje lica [18]. Za to je potreban AdaBoost algoritam.

3.1.3 **AdaBoost algoritam**

AdaBoost ili Adaptive Boosting je vrsta algoritma koji spada u algoritme skupnog učenja, odnosno u mogućnosti su sami učiti kroz skup povezanih podataka. Opća mu je svrha treniranje klasifikatora pri čemu svaki pokušava ispraviti svog prethodnika. Konkretno, za detekciju lica će smanjiti ukupan broj kombinacija Haarovih značajki koje nisu toliko potrebne te će klasificirati značajke na najvažnije Haarove značajke. Potrebno je dobiti jedan jaki klasifikator od linearne kombinacije slabih klasifikatora. Na ovaj način algoritam će moći odrediti koje značajke su najprikladnije za detekciju lica i specifičnih detalja na njemu na temelju definiranog praga. Budući da slike na kojima se nalazi lice sadrži i regije na kojima lica nema, potrebno je ta područja odbaciti i fokusirati se samo na lice. Za to se uvodi postupak kaskadiranja [19, 20].

3.1.4 **Kaskadiranje**

Nakon što je Adaboost algoritam izabrao najbolje značajke, daljnja obrada slike lica može se dodatno poboljšati u svrhu što bolje brzine i točnosti detekcije. Kaskada je niz klasifikatora koji se koriste da bi se odbacili dijelovi slike koji nisu interesantni, odnosno dijelovi koji se ne žele uzeti u obzir. Konkretno, to znači da ako se pretražuje ljudsko lice na slici, da će se pomoću kaskade klasifikatora algoritam usredotočiti samo na područje slike gdje se nalazi lice, a ostala područja će zanemariti. Prvi klasifikatori u kaskadi dizajnirani su da budu brzi i na taj način brzo odbace dio slike koji nije interesantan, koristeći značajke koje s visokom sigurnošću identificiraju potencijalnu lokaciju lica. Kasniji klasifikatori su sporiji, ali precizniji te se primjenjuju samo na dio slike koji vrlo vjerojatno sadrži traženi objekt. Ovaj postupak se odvija u nekoliko iteracija sve dok klasifikator neće biti dovoljno siguran da je pronašao taj objekt. Metoda kaskadiranja uz to što omogućava brže i efikasnije detektiranje zadanog objekta, štedi vrijeme i resurse u procesu detekcije [21].

3.2 *Primjena Haar kaskadnog klasifikatora*

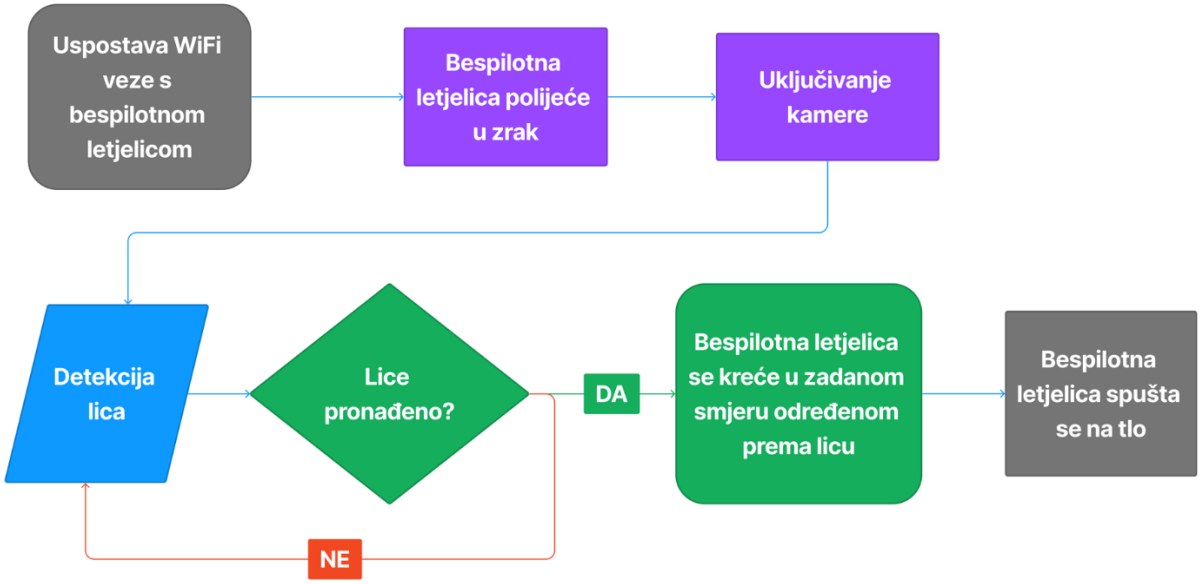
Za upravljanje dronom pomoću praćenja ljudskog lica, korišten je OpenCV te već unaprijed trenirani kaskadni klasifikator frontalnog lica koji je također dio OpenCV biblioteke [22]. Programski kod 3.1 će prikazati kako se uz pomoću OpenCV biblioteke te unaprijed treniranog kaskadnog klasifikatora frontalnog lica može upravljati bespilotnom letjelicom.

Programski kod 3.1: Upravljanje bespilotnom letjelicom pomoću detekcije lica

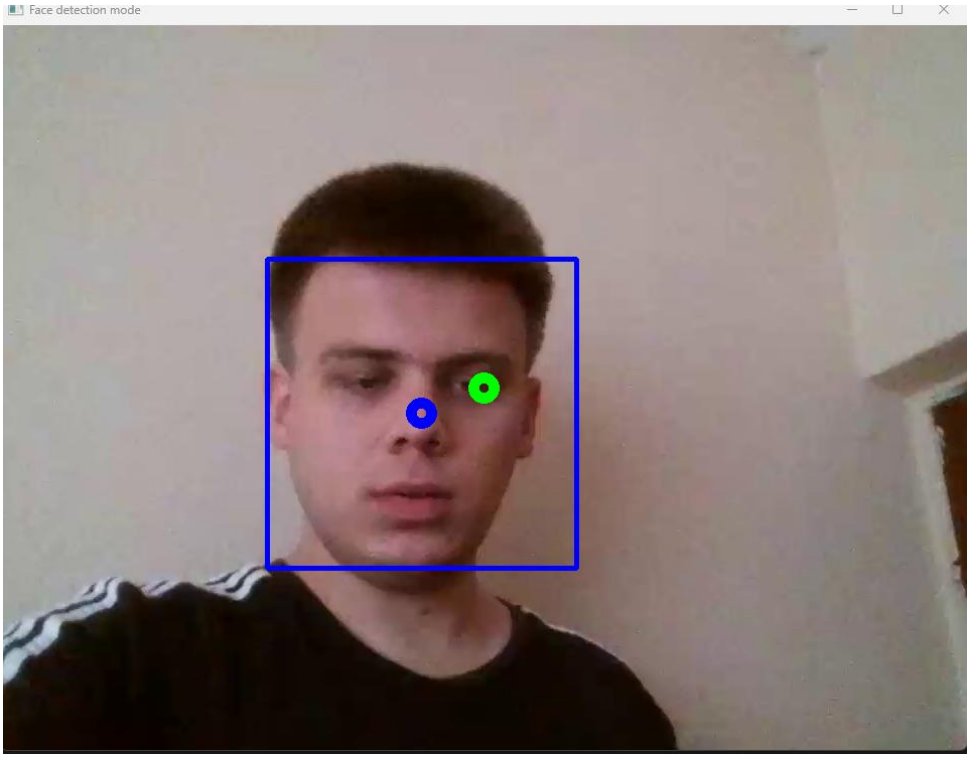
```
while True:

    frame = tello.get_frame_read().frame
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face.detectMultiScale(
        gray,
        scaleFactor=1.3,
        minNeighbors=10,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h),
                      (255, 0, 0), 5)
        cv2.circle(frame, (int(x + w / 2), int(y + h / 2)),
                   10, (255, 0, 0), 10)
        cv2.circle(frame, (int(greenDotX), int(greenDotY)),
                   10, (0, 255, 0), 10)
        distanceX = x + (w / 2) - greenDotX
        distanceY = y + (h / 2) - greenDotY
        frameHeight, frameWidth = frame.shape

        print(f'x: {x}')
        print(f'y: {y}')
        print(f'w: {w}')
        print(f'h: {h}')
        print(f'greenDotX: {greenDotX}')
        print(f'greenDotY: {greenDotY}')
        print(f'distanceX: {distanceX}')
        print(f'distanceY: {distanceY}')
        print(f'Frame height: {frameHeight}')
        print(f'Frame width: {frameWidth}')

        if distanceX < -10:
            tello.send_rc_control(-20, 0, 0, 0) #LEFT
        elif distanceX > 10:
            tello.send_rc_control(20, 0, 0, 0) #RIGHT
        else:
            tello.send_rc_control(0, 0, 0, 0) #STOP

        if distanceY < -10:
```

```
            tello.send_rc_control(0, 0, 20, 0) #UP
        elif distanceY > 10:
            tello.send_rc_control(0, 0, -20, 0) #DOWN
        else:
            tello.send_rc_control(0, 0, 0, 0) #STOP
```

Definiraju se parametri klase *detectMultiScale* koji će definirati konstante za detekciju lica. Prvo se određuje slika koja se obrađuje, zatim *scaleFactor*, koji će definirati koliko će se veličina okvira za lice povećati ili smanjiti ovisno o regiji interesa tijekom procesa detekcije lica. Nakon toga, *minNeighbors* koji definira broj okvira koje lice mora imati da bi bilo prihvaćeno kao željeni objekt, dok se ostali odbacuju, te minimalna veličina okvira za željeni objekt na slici koji se detektira. Na kraju se definira parametar *flags*, koji će pomoći u skaliranju slike, što omogućava detektoru detektiranje objekta različitih veličina [23]. Dalje se u kodu definiraju iscrtavanja okvira za detekciju lica, kružnica za središte lica i statična kružnica koja je u sredini cijelog okvira kamere [24]. Potrebno je izračunati udaljenost ove dvije kružnice. To se može izračunati po formuli definiranoj u kodu, prema kojoj je potrebno doći prvo do koordinata središta lica, odnosno prvog definiranog plavog kružića te se zatim oduzimaju vrijednosti koordinata drugog definiranog statičnog zelenog kružića. Sada, na temelju dobivenih vrijednosti, se mogu postavljati uvjeti za upravljanje letjelicom koja će pratiti detektirano lice. Letjelica se kreće pomoću metode metode *send_rc_control(self, left_right_velocity: int, forward_backward_velocity: int, up_down_velocity: int, yaw_velocity: int)* [25]. Vrijednosti argumenata definiraju brzinu kretanja letjelice u više smjerova. Letjelica će se kretati udesno, ulijevo, unaprijed, unazad, gore, dolje, orijentirati se udesno i orijentirati se ulijevo, ovisno o vrijednostima varijabli. Blokovska shema detekcije lica se može vidjeti na slici 3.4 te konkretan primjer detekcije na sljedećoj slici 3.5.



Slika 3.4: Blokovska shema upravljanja bespilotnom letjelicom pomoću detekcije lica



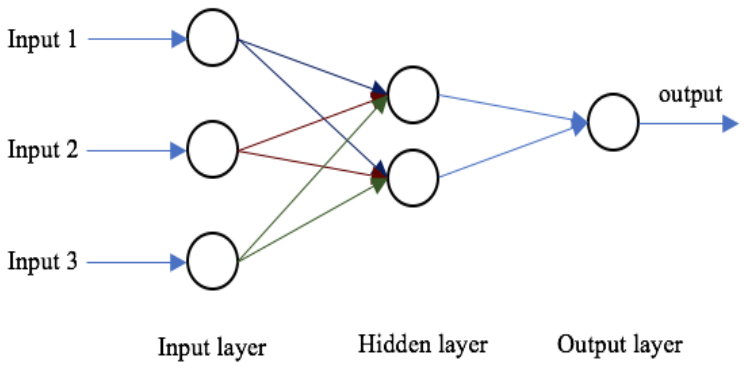
Slika 3.5: Primjer detekcije lica

4. GESTIKULACIJA RUKAMA

Gestikulacija rukama (engl. *Hand Gestures*) je prirodan način komunikacije među ljudima, te se često koristi tijekom dana. S razvojem tehnologije, postalo je potrebno na sličan način uspostaviti interakciju između čovjeka i računala. Ovakva vrsta interakcije našla je primjenu u raznim informatičkim sustavima. Koriste je gotovo svi tehnološki divovi kao što su Google, Apple i Intel u automatizaciji doma, kupovini, igrama virtualne i proširene stvarnosti, potrošačkoj elektronici i navigaciji, itd. Jedan od poznatijih primjera ovakve interakcije u suvremenom svijetu je u industriji video igara kao virtualna stvarnost. Ovakav pristup također može naći primjenu u daljinskom upravljanju robotima i poboljšanju korisničkih sučelja [26]. Za uspješno upravljanje bespilotnom letjelicom potrebno je istrenirati model konvolucijske neuronske mreže prikladan za ovakvo upravljanje. Takav model će imati mogućnost procjene gestikulacije sa slike. Na temelju toga kroz programski kod se daju instrukcije letjelici za njeno kretanje.

4.1 Neuronske mreže

Kako bi se uspješno implementirala gestikulacija rukama koriste se neuronske mreže. Princip rada se temelji na biološkim neuronskim mrežama u ljudskom mozgu koje nam pomažu u raspoznavanje okoline u kojoj se nalazimo. Sastoje se od međusobno povezanih čvorova ili neurona koji obrađuju i prenose informacije te se koriste u dubokom učenju za modeliranje složenih odnosa između ulaznih i izlaznih podataka. Primjer jedne takve jednostavne neuronske mreže može se vidjeti na slici 4.1



Slika 4.1: Arhitektura jednostavne neuronske mreže [27]

Arhitektura jednostavne ili najosnovnije neuronske mreže sastoji se od nekoliko slojeva kao što su ulazni sloj (npr. slika koja se postavi kao ulaz za daljnju obradu), skriveni sloj (sloj u kojem se obrađuje slika) i izlazni sloj na kojem se dobije rezultat procjene modela na temelju prethodne obrade. Ulazni i izlazni slojevi se zovu i vidljivi slojevi zato što ih možemo vidjeti kao podatak. Na ulazu bi to bile slike, a na kraju obrade, na izlazu bi to bila procjena modela u obliku broja ili riječi. Na nevidljive slojeve se odnose svi oni slojevi koji obrađuju sliku. Treba napomenuti da nešto kompleksnije neuronske mreže mogu sadržavati više skrivenih slojeva zbog kompleksnije obrade ulaznih podataka, a upravo je to glavna razlika između različitih tipova neuronskih mreža. Za potrebe završnog rada i obradu gestikulacija rukama korištene su konvolucijske neuronske mreže [28].

4.1.1 *Prikupljanje podataka*

Ovo je važan korak u procesu obrade podataka pomoću neuronskih mreža jer bez njih ne može početi proces treniranja. Obično je poželjno imati što više podataka za duboko učenje, no to ne jamči nužno postizanje željenih rezultata. Kvaliteta, raznovrsnost podataka i drugi faktori igraju važnu ulogu. Jedan set podataka će sadržavati slike za trening, a drugi set će sadržavati slike za test, ukoliko se za to odluči. Programski se jedan cijeli veliki set podataka može automatski kroz kod podijeliti po određenim postocima za trening i test modela [29]. Podaci za test su zaseban skup podataka u odnosu na trening set te se kaže da su oni disjunktni. Testni podaci su set podataka kojeg model na kraju procesa još nije vidio, a pomoću tog skupa podataka može se odrediti točnost modela. Ukoliko se točnosti na ta dva seta podataka ne poklapaju onda se pretpostavlja da je došlo do prenaučenosti (engl. *Overfitting*). Prenaučenost znači da model odradi dobru predikciju samo na setu podataka za trening jer je naučio specifične obrasce koji se ne mogu primijeniti na skup podataka za testiranje[30]. Ako se za primjer uzmu gestikulacije rukama, važno je da te gestikulacije budu vidljive na slici, ali da se pritom mijenjaju neki od atributa slike kao što su pozadina ili svjetlina. Stoga potrebno je imati na umu uvijek kakvi se i koliko podataka koristi za trening i test modela. Jedan od načina kako se podaci mogu prikupiti prikazano je u programskom kodu 4.1

Programski kod 4.1: Akvizicija slikovnih podataka

```
def GetImage():
    Class = "0"
    Path("DATASET/Train/" + Class).mkdir(
        parents=True, exist_ok=True
    )
    cap = cv.VideoCapture(0)
    if not cap.isOpened():
        print("Can't open camera")
        exit()
    i = 0
    while True:
        ret, frame = cap.read()

        if not ret:
            print("Can't receive frame! Exit")
            break
        i += 1
        if i % 5 == 0:
            cv.imwrite(
                "DATASET/Train/" + Class + "/" + str(i) + ".png",
frame
            )

        cv.imshow("frame", frame)
        if cv.waitKey(1) == ord("q") or i > 50000:
            break

    cap.release()
    cv.destroyAllWindows()
```

U funkciji *get_image()* definiran je način po kojem će se prikupljati podaci za obradu. Vidljivo je da će se podaci spremati u direktorij pod nazivom pojedine klase za svaku gestikulaciju. Zbog toga podaci će se spremati razdvojeno za svaku klasu po određenim intervalima.. Broj slika koje je potrebno akvizirati proizvoljno se određuje te se kasnije može korigirati ukoliko procjena modela nije najbolja.

4.1.2 *Priprema podataka*

Sada kada je pripremljen podatkovni set za trening i test može se krenuti s pripremom podataka za daljnju analizu i kreiranje modela. Najvažnija biblioteka koja se koristi za rad s konvolucijskim neuronskim mrežama je Keras [31]. Drugi dio koji je isto tako važan je za podjelu podataka na trening i test, te izračunavanje preciznosti modela ukoliko se za tu funkcionalnost odluči. Za to će

se pobrinuti funkcionalnosti iz biblioteke scikit-learn [32]. Obrada podataka se može vidjeti u programskom kodu 4.2.

Programski kod 4.2: Priprema podataka u programskom jeziku Python

```
data = []
labels = []
classes = 7
currentPath = os.getcwd()
for i in range(classes):
    path = os.path.join(currentPath , 'DATASET/Train', str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path + '/' + a)
            image = image.resize((60, 60))
            image = np.array(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error with images loading")
data = np.array(data)
labels = np.array(labels)

print(data.shape, labels.shape)
print('shape: ', data.shape[1:])

X_tr, X_ts, y_tr, y_ts = train_test_split(data, labels, test_size=0.2,
random_state=42)
print(X_tr.shape, X_ts.shape, y_tr.shape, y_ts.shape)

y_tr = to_categorical(y_tr, 7)
y_ts = to_categorical(y_ts, 7)
```

Nakon dohvaćanja svih slika pojedinih klasa mijenja se njihova veličina na 60x60. To je važno jer obrada slike u originalnoj veličini, koja je 640x480, predugo traje i računalno je jako zahtjevna. Praznoj listi *data* se pridružuju slike, a praznoj listi *labels* se pridružuju imena klasa u kojima se nalaze slike. Zatim, sve slike koje se nalaze u setu podataka se podijele na slike za trening i testiranje pomoću *train_test_split* funkcije [33]. Potrebno je definirati koliko će se slika koristiti za treniranje, a koliko za test. To se određuje argumentom *test_size*, čija je vrijednost u kodu 0.2, što znači da će se za test koristiti 20% slika od ukupnog broja, a za treniranje 80%. Parametar *random_state* omogućuje da se prilikom svakog pokretanja programske skripte podaci podijele nasumično na isti način. To omogućuje dosljednost u procesu kreiranja modela jer ukoliko se uspoređuje procjena dva modela, onda bi ukoliko se ne definira argument *random_state*, mogao dogoditi slučaj da u jednom slučaju predikciju dobro odrađuje prvi model, a u drugom slučaju

drugi model i obrnuto. Na takav način se ne mogu donijeti konačni zaključci. Na kraju potrebno je pretvoriti cjelobrojne brojeve klasa u matrice na način da je vrijednost u matrici 1 na mjestu indeksa odgovarajuće klase, npr. za broj 4 matrica bi bila definirana kao [0, 0, 0, 1]. Ovaj postupak se izvodi zbog kompatibilnosti s neuronskom mrežom koja također koristi matrice za reprezentaciju podataka na način razumljiv računalu. Još jedan način za obrađivanje ulaznih slika koji će pomoći u preciznosti modela na kraju prikazan u programskom kodu 4.3.

Programski kod 4.3: Augmentacija podataka

```
augmentor = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=False,
    brightness_range=[0.6, 1.4],
)
data = augmentor.flow(X_tr, y_tr, batch_size=32)
```

Ovim programskim kodom prikazana je **augmentacija podataka** (engl. *Data Augmentation*). Klasa *ImageDataGenerator* je alat u okviru Kerasa za obradu slika i poboljšanje trenutnog seta podataka, što pomaže u poboljšanju performansi modela. Unutar ove klase postoje razni parametri za obradu slike. Konkretno u ovom primjeru, korišteni su redom parametri za rotaciju slike, horizontalno pomicanje, vertikalno pomicanje, zumiranje slike, horizontalno preokretanje te promjenu svjetline slike. Promjena vrijednosti ovih parametara ovise o cilju za treniranje koji se želi postići te o performansama modela. Na kraju, varijabli *data* se dodjeljuje generator koji će generirati augmentirane slike. Metodom *flow()* definira se nad kojim podacima će se primijeniti augmentacija, te se definira još i *batch_size,* koji definira koliko velika će biti serija podataka. To znači da će se tijekom treniranja slike grupirati u serije od 32 slike kako je definirano u kodu. Ovo omogućuje da se podaci učitavaju u manjim grupam umjesto svi odjednom. To omogućuje brže treniranje seta podataka te bolju generalizaciju jer će na ovaj način model tijekom treniranja vidjeti više različitih primjera augmentiranih podataka nego da su svi podaci obrađeni procesom augmentacije odjednom [34].

4.2 Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (*engl. Convolutional Neural Networks, CNN*) su posebna vrsta dubokih neuronskih mreža koje su posebno dizajnirane za obradu slikovnih i video podataka. Ove mreže primjenjuju filtere na ulazne podatke kako bi se otkrile značajke poput rubova i kuteva. Sastoje se od ulaznog sloja, izlaznog sloja te nekoliko skrivenih slojeva gdje se odvija većina obrade podataka. Na kraju izlazni sloj će prikazati rezultate obrade i dati procjenu točnosti traženog objekta. Ova vrsta neuronskih mreže odabrana je za upravljanjem bespilotnom letjelicom pomoću gestikulacije rukama upravo zbog prethodno navedenih razloga.

4.2.1 *Arhitektura konvolucijske neuronske mreže*

Slijedi izrada modela koji se temelji na konkretnoj konvolucijskoj neuronskoj mreži za detekciju gestikulacije ruku. Sama neuronska mreža sastoji se od 12 slojeva kao što se može vidjeti u programskom kodu 4.4.

Programski kod 4.4: Arhitektura konvolucijske neuronske mreže

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation="relu",
input_shape=X_tr.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.15))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation="relu"))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.15))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.15))
model.add(Flatten())
model.add(Dense(512, activation="relu"))
model.add(Dropout(rate=0.5))
model.add(Dense(7, activation="softmax"))
```

U programskom kodu, prikazana je arhitektura konvolucijske neuronske mreže. Prvo se instancira klasa *Sequential()* koja se koristi za definiranje sekvencijalnih modela u dubokom učenju [37]. Sekvencijalni tip modela je linearni stog slojeva gdje svaki sloj ima jedan ulaz i jedan izlaz u

odnosu na funkcionalni tip modela koji ima više ulaza i više izlaza. Pošto se koristi samo jedna kamera kao ulaz, za primjenu modela sekvencijalni tip modela je pogodniji[38].

4.2.2 *Konvolucijski sloj*

Konvolucijski sloj (engl. *Convolutional Layer*) ili u programskom kodu 4.4. *Conv2D* je prvi sloj u procesu izrade modela [39]. Prethodno obrađena slika dovedena na ulaz prolazi kroz matematičku operaciju konvolucije. Matrica slike, koja je veličine 60x60x3, množi se s filterom veličine 5x5 na početku kako bi se dobio skup značajki pronađenih na slici. Filteri su korisni u obradi slike jer se pomoću njih naglašavaju prijelazi sa slike, odnosno rubovi i linije. Veličina filtera odabire se po potrebi, ovisno o tome kolika je slika na ulazu i vrsti značajki koje se žele detektirati. Općenito, može se reći da filteri većih dimenzija bolje detektiraju veće značajke, dok filteri manjih dimenzionalnosti mogu prodrijeti više u dubinu i preciznije detektirati složenije značajke. Na prvom sloju, filter je veličine 5x5 jer je i sama slika najveća, odnosno većih je dimenzija na početku dok u drugim slojevima to nije slučaj pa se zbog toga smanjuje veličina filtera. Konvolucija je definirana prema sljedećoj formuli 4.1.

$$Y[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j] \tag{4.1}$$

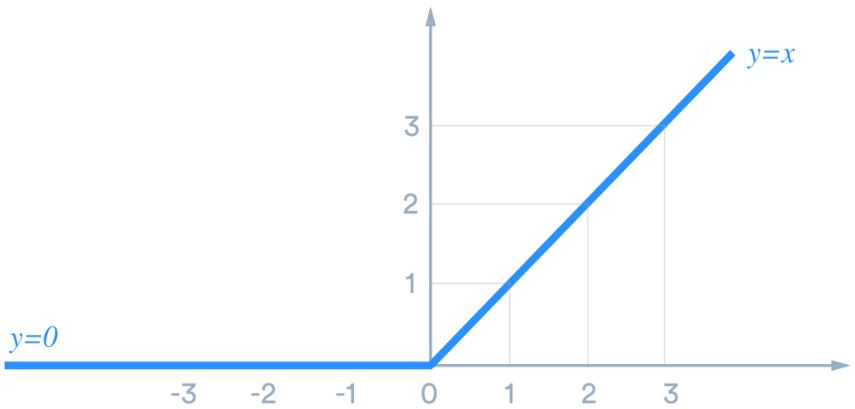
Dakle, ovdje [i, j] predstavljaju koordinate, odnosno vrijednost piksela na toj poziciji, a [m, n] predstavljaju koordinate, odnosno vrijednost središta filtera. Tada se svaka vrijednost piksela sa slike množi s odgovarajućom vrijednosti težine unutar filtera koja se nalazi na istoj poziciji piksela sa slike te se zbrajaju sve vrijednosti u trenutnom položaju filtera nad slikom. Ove vrijednosti se mijenjaju svaki puta nakon translacije filtera po slici. Pošto je slika u 3D prostornom prikazu, mora se svaki kanal posebno rastaviti i posebno obraditi konvoluciju s proizvoljnim filterom [40, 41]. Treba napomenuti da vrijednosti ili tzv. težine unutar filtera nisu definirane u kodu, već je definiran samo broj filtera koji će se primijeniti u konvoluciji sa slikom, a težine su neke nasumične vrijednosti. Te vrijednosti ne moraju biti simetrične kao što npr. u filterima prilikom traženja Haar značajki za prepoznavanje lica. Na taj se način se izvlače različite i složenije značajke traženog objekta, što na kraju doprinosi točnijoj procjeni modela na samom kraju. Cilj je pronaći optimalne vrijednost težina unutar filtera kako bi se minimizirala pogreška procjene. Smanjivanje pogreške se vrši metodom povratne propagacije koja će biti spomenuta kasnije nakon zadnjeg sloja

arhitekture mreže. Ovo je bio prvi sloj u kreiranju modela i sada nakon obrađenog prvog sloja i primjene konvolucije, slika će promijeniti svoju veličinu te je to definirano po formuli 4.2.

$$V_i = \frac{V_v - V_f + 2P}{S} + 1 \tag{4.2}$$

Ova formula pokazuje kako se dolazi do vrijednosti veličine izlazne slike V_i nakon konvolucije. Ulazna veličina slike V_u, koja je dimenzija 60x60, se oduzme s veličinom filtera V_p, koji je dimenzija 5x5, te se zbraja s dvostrukom vrijednošću ispune P (engl. *Padding*). To se zatim podijeli s korakom S (engl. *Stride*), te se zbroji s 1. Ispuna obično rješava problem s dimenzijama održavajući ih konstantnima ukoliko je to potreba. Međutim, u ovom slučaju arhitektura je takva da se žele smanjiti dimenzije slike i pošto je objekt za detekciju na slici centriran i ne nalazi se na rubovima slike, ispuna nije potrebna. Ispuna može povećati dodatne izračune tijekom konvolucije i memorijske zahtjeve. Iz tih razloga ispuna se ne koristi u programskom kodu 4.4, automatski je postavljena na vrijednost *valid*. U formuli se također spominje parametar korak, koji definira za koliko će se matrica filtera pomicati po ulaznoj slici. U programskom kodu 4.4, korak je izostavljen kao i ispuna, ali standardno je unaprijed definiran na vrijednosti (1, 1). To znači da će se po x i y osi pomicati samo za jedan korak. Nakon provedene konvolucije za sva tri kanala slike, na kraju se sve vrijednosti zbrajaju tako da se svaki piksel na slici na određenom mjestu na jednom kanalu zbroji s ostalim pikselima na tom istom mjestu na drugim kanalima [42]. Sada treba odlučiti da li će se na taj zbroj još dodati posmak *(*engl. *Bias)*, koji je povezan s aktivacijskom nelinearnom funkcijom ReLU, koja je definirana po formuli 4.3 i prikazana grafom sa slike 4.2.

$$ReLU = max\,(0, x) \tag{4.3}$$

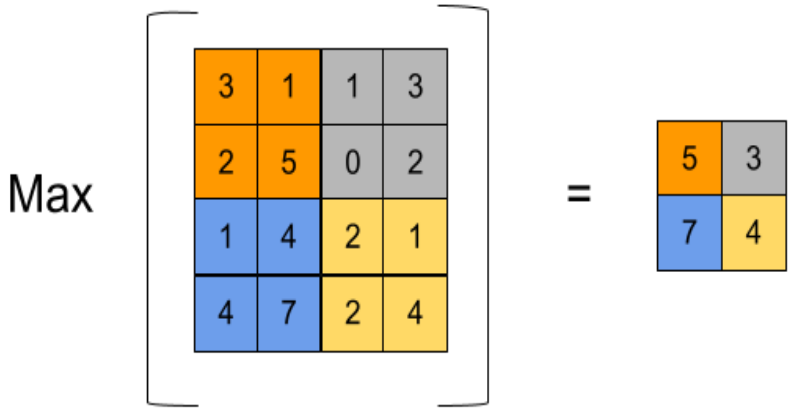


Slika 4.2: ReLU aktivacijska nelinearna funkcija [41]

Uz ostale funkcije koje se koriste u konvolucijskim neuronskim mrežama, ReLU aktivacijska funkcija je najpopularnija zbog svoje jednostavnosti, što se također odražava i na brzinu izvođenja operacija tijekom računanja. Sama funkcija jasno ukazuje da će neuroni čija je vrijednost manja od 0 biti postavljeni na 0. Takav neuron se zove mrtvi neuron. Ove negativne vrijednosti predstavljaju odsustvo značajke na nekoj slici ili se još može reći da je ona manje bitna, dok pozitivne vrijednosti ukazuju na prisustvo značajke. Negativnih vrijednosti se može riješiti dodavanjem prethodno spomenutog posmaka. Pomoću posmaka jednostavno će se translatirati vrijednost po x-osi , omogućavajući modelu da ipak nauči i manje bitne značajke, što je ujedno i jedan od načina kako spriječiti prenaučenost modela [43]. Sada, nakon konvolucije, dodavanja posmaka i primjene aktivacijske funkcije ReLU, na kraju prvog konvolucijskog sloja dobiju se mape značajki koje su dimenzija 56x56x32. Te kraće dimenzije mogu se izračunati po prethodno definiranoj formuli (4.2) o smanjenju dimenzionalnosti nakon završetka konvolucije. Promijenila se i vrijednost dubine koja je sada 32, a ona je određena brojem filtera što je ujedno i isti broj mapa značajki. Ovdje završava prvi konvolucijski sloj. Prvi sloj obično otkriva jednostavnije značajke poput linija ili oblika, dok će drugi sloj konvolucije, na temelju tih pronađenih jednostavnijih značajki, tražiti složenije teksture i oblike, te još će dodatno smanjiti dimenzije pa će tako na kraju biti mapa značajki s dimenzijama 52x52x32.

4.2.3 *Sloj sažimanja*

Sada slijedi sloj sažimanja maksimalnom vrijednošću *(*engl. *Max Pooling)*, čija je svrha smanjiti dimenzionalnost ulaza i da pritom zadrži sve važne značajke. Koristi se metodom sažimanja maksimalnom vrijednošću jer uzima određeni skup piksela s ulazne slike te dalje prosljeđuje onaj najveći piksel [44]. U kodu je to definirano argumentom *pool_size(2, 2)*, što znači da će uzimati blok od 4 piksela te pronaći onaj s najvećom vrijednošću i proslijediti ga dalje. Važno je reći da će zbog prethodno navedenog argumenta smanjiti dimenzionalnost izlaza na pola, budući da je argument definiran na vrijednosti (2, 2). Dimenzionalnost je sada 26x26x32. Ovaj postupak prikazan je na sljedećoj slici 4.3.



Slika 4.3: Primjer sažimanja maksimalnom vrijednošću [45]

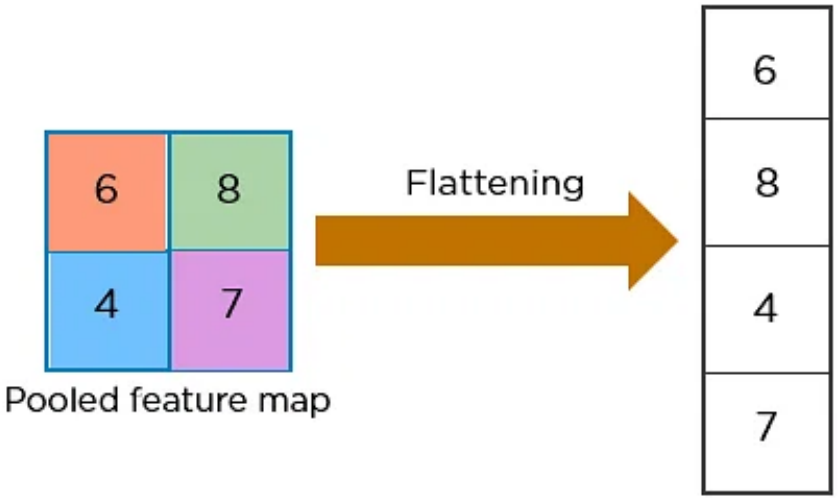
Ovime se, uz smanjenje dimenzionalnosti, omogućava sljedećim slojevima da pronalaze detaljnije i kompleksnije značajke, dok se istovremeno smanjuje računalna složenost modela.

4.2.4 *Sloj isključivanja*

Sada dolazi sloj Isključivanja (engl. *Dropout)* čija je osnovna zadaća da spriječi pretreniranost modela tako da nasumično isključuje neurone na ulazu. Vrijednost za nasumično isključivanje neurona u kodu postavljena je na 0.25, što znači da će se 25% nasumičnog ulaza isključiti, odnosno postaviti na 0. Ovime se omogućuje mreži da je u stanju naučiti raznolike značajke. Bez ovog sloja moglo bi doći do situacije u kojoj se mreža previše oslanja na samo nekoliko najjačih neurona i loše se ponaša na primjerima koji nemaju te značajke ili ih imaju samo djelomično. Ovaj sloj ne smanjuje dimenzionalnost već samo deaktivira nasumične neurone koji se dalje ne koriste u mreži. Nakon toga se nadovezuju opet 2 konvolucijska sloja koji sada imaju veći broj filtera, njih 64, čija je veličina ovog puta 3x3. Ove novije vrijednosti uvode se zbog toga što je i sama dimenzionalnost ulaza manja. Na taj način se sada omogućuje da se na manjim slikama izvuku detaljnije značajke iz već prethodno izvučenih značajki. Naravno, ovakav pristup s manjim dimenzionalnostima na kraju daje bolje rezultate predikcije. Ako se prati formula za smanjenje dimenzionalnosti, sada na šestom sloju po redu, odnosno nakon četvrtog konvolucijskog sloja, ona iznosi 22x22x64. Nakon ovoga slijede još sloj sažimanja maksimalnom vrijednošću, sloj isključivanja, 2 konvolucijska sloja, te opet 1 sloj sažimanja i 1 sloj isključivanja. Dimenzionalnost je u ovom trenutku 3x3x128 pošto sloj sažimanja maksimalnom vrijednošću smanji dimenzionalnost u pola kako je i definirano u kodu [46].

4.2.5 *Sloj izravnanja*

Sloj za izravnavanje podataka *(*engl. *Flatten)* će pripremiti ulazne podatke za sljedeći sloj na način da će ih pretvoriti iz 3D oblika u 1D oblik, odnosno vektor. Ovo se izvodi tako da se vrijednosti dimenzija ulaza (visina, širina, kanal) koje su došle iz prethodnog sloja pomnože, što rezultira brojem 1152. Izravnavanje je prikazano na slici 4.4.



Slika 4.4: Primjer postupka izravnanja vrijednosti [47]

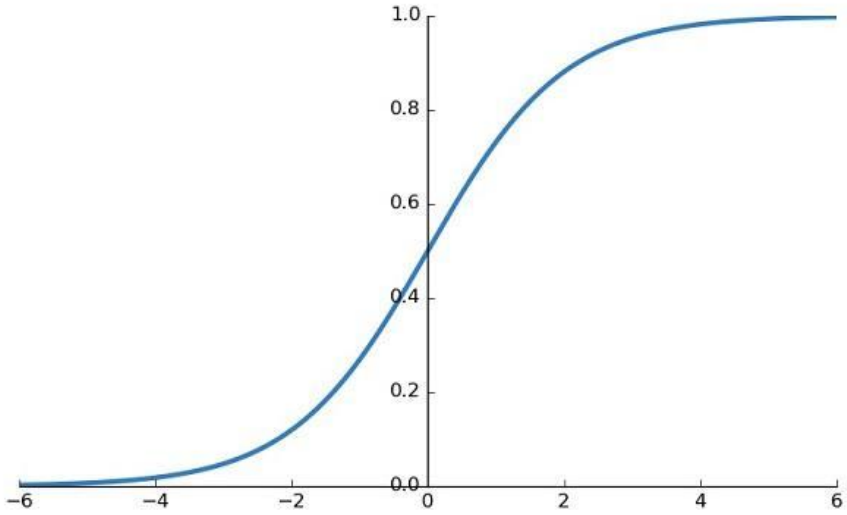
Izravnanjem se nisu izgubili podaci iz ulaznih mapi značajki već su sada one posložene u vektoru na način da se slijedi određeni uzorak koji odgovara ulaznim dimenzijama, odnosno vrijednosti iz matrica mapa značajki su sada poredane jedna ispod druge u 1D vektoru [48].

4.2.6 *Gusti potpuno povezani sloj*

Kada su podaci pripremljeni na odgovarajući način, dolazi gusti, potpuno povezani sloj *(*engl. *Dense)*. To znači da će svaki neuron, čiji je ukupan broj definira u kodu te iznosi 512, biti povezan sa svakom vrijednošću ulaza iz prethodnog sloja te će tako činiti gustu povezanu mrežu. Sada slijedi isti postupak kao i kod konvolucijskih slojeva, sve te vrijednosti iz prethodnog sloja se množe s težinom. Ukupan broj težina u ovom sloju odgovara ukupnom broju veza. Nakon toga se rezultati zbroje i dodaje se posmak, a potom se primjenjuje ReLU aktivacijska nelinearna funkcija kako bi se poništile negativne vrijednosti. Ovaj sloj će biti potpuno i gusto povezan sa sljedećim potpuno, gustim povezanim slojem. Između ta 2 sloja nalazi se još jedan sloj isključivanja koji će sada, kako je definirano u kodu, isključiti 50% nasumičnih neurona što znači da će na izlazu tog sloja biti također 512 vrijednosti, ali sljedeći sloj će uzeti u obzir samo njih 256 aktivnih. Na kraju,

nalazi se još jedan potpuno povezani sloj koji sada ima 7 neurona, i opet, između tog sloja i prethodnog sloja se nalaze težine i posmak koji se uzimaju u obzir tijekom izračuna vrijednosti. Specifično za ovaj sloj je da koristi softmax aktivacijsku nelinearnu funkciju [49, 50]. Definirana je po formuli 4.3 i prikazana grafom sa slike 4.5.

$$S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{4.4}$$



Slika 4.5: Softmax aktivacijska nelinearna funkcija [51]

Svrha ove funkcije sa slike 4.5 je skaliranje ulaznih vrijednosti u rasponu od 0 do 1 što omogućuje višeklasnu klasifikaciju. Svaki ulaz iz prethodnog sloja z_i je potencija od *e*. Zatim se te vrijednosti zasebno podijele s ukupnim sumom svih takvih potenciranih vrijednosti s ulaza. Dobivene vrijednosti na kraju predstavljaju vjerojatnost procjene te se uspoređuju s klasama koje su u prethodnom koraku, prije definiranja modela, prebačene u matrični oblik i ako je vrijednost bliže 1, to znači da je prepoznat određeni objekt na temelju ulaznih podataka. Ovime završava proces obrade ulaznog podatka u obliku jedne slike, koji se još naziva i unaprijedna propagacija, i sada slijedi računanje pogreške predikcije i povratna propagacija. Ovo je ključni koncept u strojnom učenju koji će optimizirati model. Računanje pogreške predikcije je moguće nakon izračunatih vrijednosti softmax aktivacijske funkcije po sljedećoj formuli 4.4:

$$gubitak = -\sum(y_i \cdot \log(y_p)) \tag{4.5}$$

Gubitak se definira kao suma istinitih vrijednosti y_i pomnožena s logaritmom od vrijednosti predikcije y_p. Istinite vrijednosti su zapravo one vrijednosti koje se smatraju apsolutno točnima ili netočnima, tj. 1 ili 0 dok će procjenske vrijednosti biti između 0 i 1. Ovakav način dobivanja gubitaka se naziva kategorička križna entropija [52].

4.2.7 *Povratna propagacija*

Postupak povratne propagacije upravo koristi te informacije o pogrešci i započinje izračunom gubitka gradijenta u odnosu na izlaz potpuno povezanog sloja. Gradijent se potom propagira unatrag kroz slojeve mreže, koristeći lančano pravilo. Ovim pravilom se računa parcijalna derivacija, tj. svaki gradijent izlaza iz pojedinog sloja u odnosu na pojedine prethodne ulaze. Nakon što se izračunaju gradijenti za sve težine u mreži, koriste se algoritmi optimizacije kako bi se ažurirale težine i minimizirala pogreška na krajnjem izlazu mreže, što rezultira boljim i pouzdanijim procjenama modela. Za optimizator ove mreže korišten je optimizator Adam. Pokazao se kao najbolji optimizator kada je u pitanju velika količina podataka koju treba obraditi kroz mrežu. Omogućava brže treniranje modela u samo nekoliko epoha, čime se značajno smanjuje vremenski zahtjev za obradu mreže, a istovremeno omogućava dobivanje preciznog modela [53]. U kodu se optimizator kao i funkcija pogreške definira na sljedeći način u programskom kodu 4.5.

Programski kod 4.5: Definiranje optimizatora, funkcije gubitka i metrike

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy']
```

Može se primijetiti da se metodom *compile*, uz prethodno definiranu funkciju pogreške i optimizacijskog algoritma, definira i metrika [54]. Ona će u terminalu prikazivati vrijednosti preciznosti modela u terminalu nad skupom podataka za učenje i testiranje [55, 56].

4.2.8 *Treniranje modela i prikaz rezultata*

Nakon definirane strukture konvolucijske neuronske mreže i na koji način će se odraditi povratna propagacija, sad slijedi i pokretanje samog procesa treniranja mreže kao što je prikazano programskim kodom 4.6:

Programski kod 4.6: Pokretanje treniranja modela

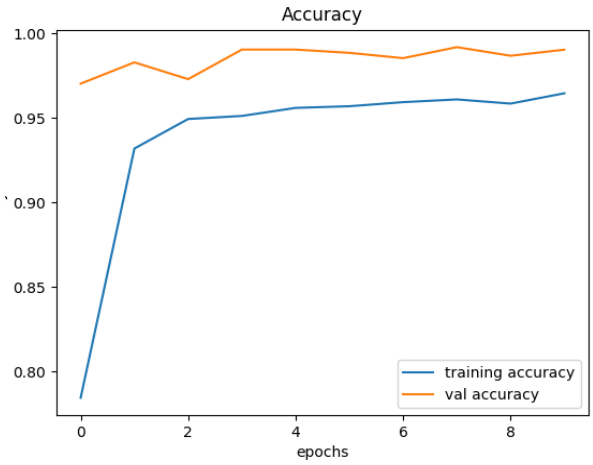
```
history = model.fit(data, epochs=10, validation_data=(X_ts, y_ts))
```

Varijabli *history* dodijeljena je metoda *fit*, u kojoj je redom definiran ulazni skup podataka za obradu, broj epoha te skup podataka za test kako bi se procijenila izvedba modela nad podacima koje mreža još nije vidjela [57]. Nakon toga slijedi ostali cijeli set podataka, te se takav proces odvija u 10 epoha, kako je navedeno u kodu [58]. Kada je proces treniranja modela gotov, nakon toga se mogu vidjeti rezultati modela i ako su prihvatljivi, model se može koristiti za daljnju upotrebu. Način prikaza rezultata se može vidjeti u sljedećem kodu 4.7.

Programski kod 4.7: Prikaz podataka pomoću matplotlib funkcija

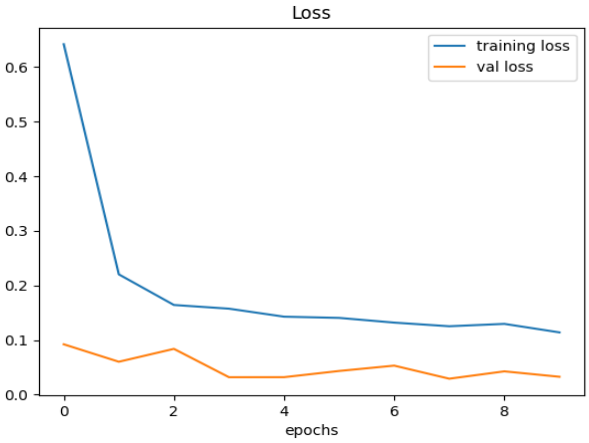
```
plt.figure(0)
plt.plot(history.history["accuracy"], label="training accuracy")
plt.plot(history.history["val_accuracy"], label="val accuracy")
plt.title("Accuracy")
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.legend()
plt.show()
plt.figure(1)
plt.plot(history.history["loss"], label="training loss")
plt.plot(history.history["val_loss"], label="val loss")
plt.title("Loss")
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend()
plt.show()
```

Koriste se metode iz Matplotlib biblioteke za vizualizaciju dobivenih rezultata [59]. Ovaj kod će omogućiti prikaz dva grafa. Na prvom grafu sa slike 4.6 može se vidjeti kako je model dobivao na preciznosti.



Slika 4.6: Graf preciznosti

Graf sa slike prikazuje promjenu preciznosti za trening set podataka i validacijski set podataka koje model još nije vidio. Poželjno je da se preciznost za trening set podataka povećava po svakoj obrađenoj epohi dok bi preciznost validacijskog seta podataka trebao biti otprilike jednak ili veći. Drugi graf sa slike 4.7 prikazati će kako se smanjivao gubitak tijekom vremena treniranja.

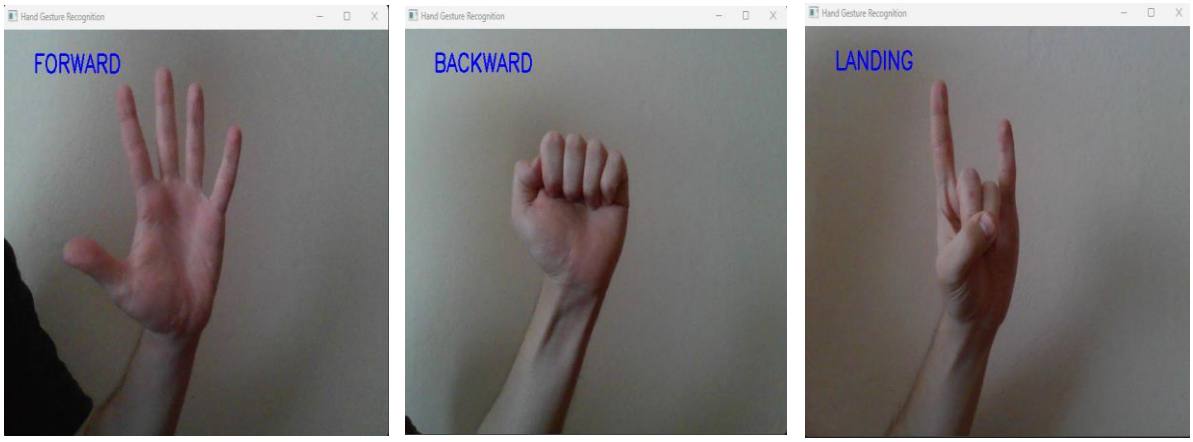


Slik 4.7: Graf gubitaka

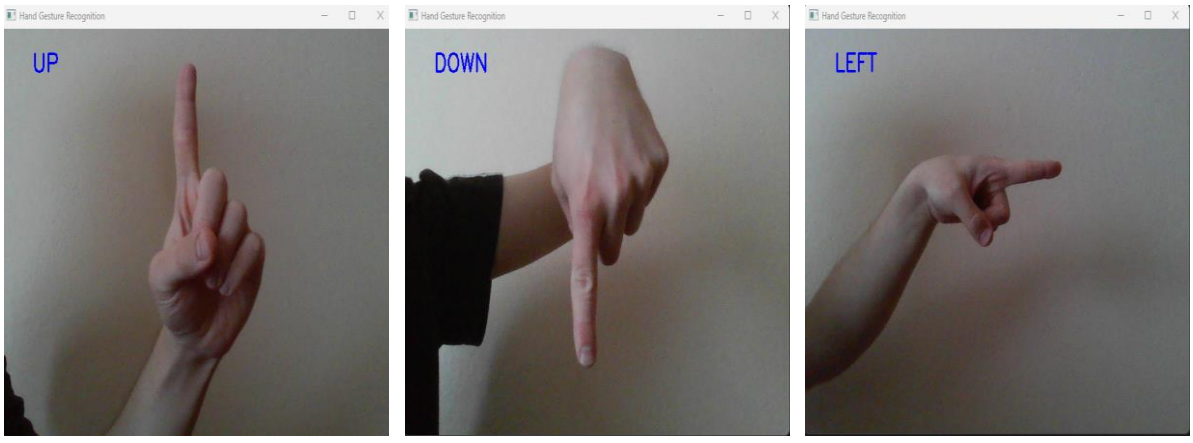
Graf prikazuje promjenu gubitaka za trening set podataka i validacijski set podataka koje model još nije vidio. Poželjno je da se gubici smanjuju za trening set podataka po svakoj obrađenoj epohi dok bi kod validacijskog seta podataka trebali biti jednaki ili manji.

4.3 Upravljanje bespilotnom letjelicom pomoću izrađenog modela

Upravljanje letjelicom osmišljeno je na način da se gestikulacije rukama pokazuju ispred neke druge kamere umjesto integrirane kamere na bespilotnoj letjelici. Razlog tome je taj da, u slučaju pogrešne procjene modela, korištenje integrirane kamere na bespilotnoj letjelici može rezultirati njezinim odlaskom na veću visinu i time gubitkom mogućnosti praćenja gestikulacije unutar njenog vidnog polja. Na ovaj način, koristeći drugu kameru, bespilotnom letjelicom se može upravljati s bilo koje udaljenosti, pri čemu se osoba koja upravlja nalazi unutar dometa u kojem letjelica može primiti signal. Primjer upravljanja pomoću gestikulacije rukama se može vidjeti na sljedećoj slici 4.8.



a) Unaprijed *b) Unazad* *c) Slijetanje*



d) Gore *e) Dolje* *f) Lijevo*

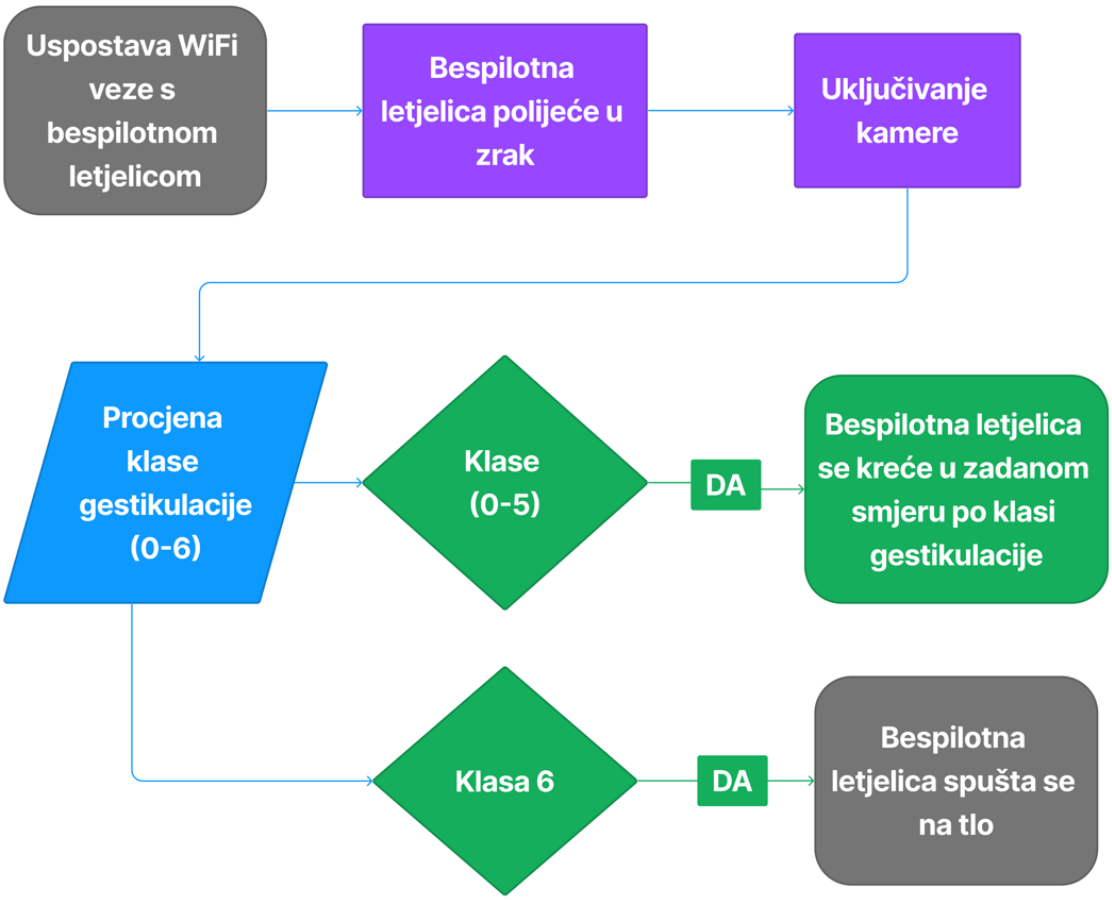
Slika 4.8: Primjer upravljanja letjelice pomoću gestikulacija rukama

Kada se očita gestikulacija pomoću sljedećeg programskog koda može se na jednostavan način upravljati bespilotnom letjelicom pomoću sljedećeg programskog koda 4.8.

Programski kod 4.8: Upravljanje letjelicom pomoću gestikulacija rukama

```
def handCommands():
    while True:
        global predict_class
        if predict_class == 0:  # UP
            tello.send_rc_control(0, 0, 20, 0)
            time.sleep(1)
        elif predict_class == 1:  # DOWN
            tello.send_rc_control(0, 0, -20, 0)
            time.sleep(1)
        elif predict_class == 2:  # RIGHT
            tello.send_rc_control(20, 0, 0, 0)
            time.sleep(1)
        elif predict_class == 3:  # LEFT
            tello.send_rc_control(-20, 0, 0, 0)
            time.sleep(1)
        elif predict_class == 4:  # FORWARD
            tello.send_rc_control(0, 20, 0, 0)
            time.sleep(1)
        elif predict_class == 5:  # BACKWARD
            tello.send_rc_control(0, -20, 0, 0)
            time.sleep(1)
        elif predict_class == 6:  # LANDING
            tello.land()
```

Definirana je funkcija pomoću koje se upravlja bespilotna letjelica. Globalna varijabla *predict_class* je prethodno definirana te pokazuje na indeks klase za određenu gestikulaciju, a još se koristi i u funkciji za kameru gdje ispisuje predviđenu gestikulaciju na ekranu kamere. Sada se na temelju predviđene klase letjelica kreće pomoću iste metode kao i za praćenje detekcije lica iz prethodnog poglavlja a to je metoda *send_rc_control(self, left_right_velocity: int, forward_backward_velocity: int)*. Ova vrsta letjelice ima i drugih načina kretnje u određenim smjerovima, ali ova se pokazala kao najbolja zbog finijeg i stabilnog načina promjene smjerova. Kod za ovaj način upravljanja je poželjno da se izvodi u 2 dretve zbog toga što se izvodi prikaz kamere, ispis predikcije na okviru, prepoznavanje ruku i upravljanje dronom. Stoga se kod podijeli na dvije dretve koje upravljaju kamerom i gestikulacijama ruku. Razlog korištenja dretvi je omogućavanje simultano izvođenje više zadataka koji su potrebni za uspješno upravljanje bespilotne letjelice. Prikaz blokovske sheme upravljanje pomoću gestikulacija rukama prikazan je na slici 4.9.



Slika 4.9: Blokovska shema upravljanja bespilotnom letjelicom pomoću gestikulacija rukama

5. ZAKLJUČAK

U ovom radu implementiran je napredan sustav upravljanja bespilotnom letjelicom, koristeći detekciju lica i gestikulacije rukama kao sredstvo pomoću kojeg se upravlja kretnjama bespilotne letjelice. Uspostavljena je interaktivna komunikacija čovjeka i bespilotne letjelice u svrhu njezinog upravljanja putem ljudske gestikulacije.

Ovakvi sustavi imaju određena ograničenja i nesavršenosti. Većinom sva ograničenja su vezana su za performanse bespilotne letjelice i prikupljene podatke. Postoje i drugačiji faktori koji donose sa sobom određena ograničenja kao što su vremenski uvjeti, učinkovitost i brzina algoritama, itd. Za sva ograničenja, koja su ujedno i nedostaci sustava upravljanja, uvijek su potrebna različita testiranja kako bi se pronašlo najbolje rješenje i uvjeti u kojima će bespilotna letjelica biti upravljana na zadovoljavajući način. Prednost ovakvog sustava je upravljanje letjelicom na daljinu, a pogotovo kod upravljanja pomoću gestikulacija rukama koje se može odvijati preko druge kamere, a ne samo one ugrađene u letjelicu. Osim toga, ovakav sustav ne zahtijeva nikakav dodatni upravljač, te je pogodan za slikanje i snimanja videa u pokretu, identifikaciju i praćenje objekata, itd.

Ovaj sustav za upravljanje može poslužiti kao osnova za daljnja istraživanja i razvijanje naprednih sustava upravljanja bespilotnom letjelicom. Takvi sustavi bi mogli naći svoju primjenu u područjima sigurnosti ili nadzora, kriznih situacija, zabave, edukacije, agronomije itd. kako bi olakšali obavljanje raznih zadataka.

6. LITERATURA

[1] Lutkevich B. What Is a Drone? - Definition from WhatIs.com. *IoT Agenda* [Online]. 2021. Dostupno na: https://www.techtarget.com/iotagenda/definition/drone (11.10.2022.)

[2] Vyas, K. A Brief History of Drones: The Remote Controlled Unmanned Aerial Vehicles (UAVs) [Online]. 2020. Dostupno na: https://interestingengineering.com/innovation/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs (11.10.2022.)

[3] Dormehl L. The history of drones in 10 milestones [Online]. 2018. Dostupno na: https://www.digitaltrends.com/cool-tech/history-of-drones (15.10.2022.)

[4] VOA US Confirms Iran Fired on Drone [Online]. 2012. Dostupno na: https://www.voanews.com/a/iran-drone-us-attack-gulf/1542350.html (16.10.2022)

[5] aliasrobotics.com Case Study – Tello [Online]. Dostupno na: https://aliasrobotics.com/case-study-pentesting-tello.php (20.10.2022.)

[6] Tello korisnički priručnik [Online]. Dostupno na: https://dl-cdn.ryzerobotics.com/downloads/Tello/20180404/Tello_User_Manual_V1.2_EN.pdf (20.10.2022.)

[7] store.dji.com Tello | Ryze Tech - Feel the Fun [Online]. Dostupno na: https://store.dji.com/hr/product/tello?vid=38421 (20.10.2022.)

[8] Cohen, S. Ryze Tello is the perfect starter drone [Online]. 2018. Dostupno na: https://mobilesyrup.com/2018/09/08/ryze-tello-perfect-starter-drone (20.10.2022.)

[9] DJITelloPy Github repozitorij [Online]. Dostupno na: https://github.com/damiafuentes/DJITelloPy (20.10.2022.)

[10] Tello DJITelloPy API Reference [Online]. Dostupno na: https://djitellopy.readthedocs.io/en/latest/tello (20.10.2022.)

[11] Jaiswala, A Face Detection using Haar-Cascade using Python [Online]. 2022. Dostupno na: https://www.analyticsvidhya.com/blog/2022/10/face-detection-using-haar-cascade-using-python (7.11.2022.)

[12] Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features [Online]. 2001. Dostupno na: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf (11.11.2022.)

[13] AaronWard Facial Detection — Understanding Viola Jones Algorithm [Online]. 2020. Dostupno na: https://medium.com/@aaronward6210/facial-detection-understanding-viola-jones-algorithm-116d1a9db218 (11.11.2022.)
[14] BenMauss Haar-like Features: Seeing in Black and White [Online]. 2021.. Dostupno na: https://levelup.gitconnected.com/haar-like-features-seeing-in-black-and-white-1a240caaf1e3 (11.11.2022.)
[15] docs.opecv.org OpenCV: Cascade Classifier [Online]. Dostupno na: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (11.11.2022.)
[16] Nayan Blog How to calculate Integral Image or Summed Area Table [Online]. 2022. Dostupno na: https://traffic.nayan.co/blog/AI/Integral-Image (12.11.2022.)
[17] OpenGenus IQ: Computing Expertise & Legacy Face Detection as done in 2001: Viola Jones Algorithm [Online]. 2019. Dostupno na: https://iq.opengenus.org/face-detection-using-viola-jones-algorithm (12.11.2022.)
[18] Abenza, J.A. Face detection [Online]. 2018. Dostupno na: https://levelup.gitconnected.com/the-integral-image-4df3df5dce35 (12.11.2022.)
[19] docs.opencv.org OpenCV: Face Detection using Haar Cascades [Online]. Dostupno na: https://docs.opencv.org/4.x/d2/d99/tutorial_js_face_detection.html (12.11.2022.)
[20] Saxena S. Introduction to AdaBoost Algorithm with Python Implementation [Online]. 2021. Dostupno na: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-adaboost-algorithm-with-python-implementation (12.11.2022.)
[21] GeeksforGeeks Face detection using Cascade Classifier using OpenCV-Python [Online]. 2021. Dostupno na: https://www.geeksforgeeks.org/face-detection-using-cascade-classifier-using-opencv-python/ (12.11.2022.)
[22] OpenCV About [Online]. 2018. Dostupno na: https://opencv.org/about (28.11.2022.)
[23] docs.opencv.org. (n.d.) OpenCV: cv::CascadeClassifier Class Reference. [Online]. Dostupno na: https://docs.opencv.org/4.x/d1/de5/classcv_1_1CascadeClassifier.html#a6d01a748b103f0cd6bd2a20037ae8731 (28.11.2022.)
[24] GeeksforGeeks Python OpenCV | cv2.circle() method. [Online]. 2019. Dostupno na: https://www.geeksforgeeks.org/python-opencv-cv2-circle-method (29.11.2022.)

[25] djitellopy.readthedocs.io. Tello - DJITelloPy API Reference [Online]. Dostupno na: https://djitellopy.readthedocs.io/en/latest/tello/#djitellopy.tello.Tello.send_rc_control

[26] Mantra AI, How does AI recognise your hand signs, gestures and movements? [Online]. 2019. Dostupno na: https://mantra.ai/blogs/ai-in-gesture-recognition (2.2.2023.)

[27] cosmo0techno.blogspot.com. Head First Deep Learning [Online]. 2020. Dostupno na: https://cosmo0techno.blogspot.com/2020/10/head-first-deep-learning.html (7.2.2023.)

[28] Lončarić, S. Neuronske mreže: Uvod [Online]. Dostupno na: https://www.fer.unizg.hr/_download/repository/01-Uvod-1s.pdf (2.2.2023.)

[29] Brownlee, J. Train-Test Split for Evaluating Machine Learning Algorithms [Online]. 2020. Dostupno na: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms (7.2.2022.)

[30] Brownlee, J. How to Avoid Overfitting in Deep Learning Neural Networks [Online]. 2019. Dostupno na: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error (7.2.2022.)

[31] Keras Home - Keras Documentation [Online]. 2019. Dostupno na: https://keras.io (8.2.2.2023.)

[32] scikit-learn.org Machine learning in Python [Online]. Dostupno na: https://scikit-learn.org/stable (8.2.2023.)

[33] Keras Team Keras documentation: Large-scale multi-label text classification [Online]. Dostupno na: https://keras.io/examples/nlp/multi_label_classification (10.2.2023.)

[34] TensorFlow tf.keras.preprocessing.image.ImageDataGenerator [Online]. Dostupno na: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (11.2.2023.)

[35] Great Learning Team Types of Neural Networks and Definition of Neural Network [Online]. 2020. Dostupno na: https://www.mygreatlearning.com/blog/types-of-neural-networks (20.2.2023.)

[36] dshahid380 Convolutional Neural Network. [Online]. 2019. Dostupno na: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529. (20.2.2023.)

[37] Keras Team Keras documentation: The Sequential class [Online]. Dostupno na: https://keras.io/api/models/sequential (22.3.2023.)

[38] Das, A. Keras Models(CNN): Functional Vs. Sequential(MNIST DATA SET) [Online]. 2020. Dostupno na: https://medium.com/@antika.das/keras-models-cnn-functional-vs-sequential-mnist-data-set-d7a19dae9cb7 (22.3.2022.)
[39] Keras Team Keras documentation: Conv2D layer [Online]. Dostupno na: https://keras.io/api/layers/convolution_layers/convolution2d (1.4.2023.)
[40] Deepanshi Convolutional Neural Network with Implementation in Python [Online]. 2021. Dostupno na: https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-with-implementation-in-python (1.4.2023.)
[41] Example of 2D Convolution [Online]. Dostupno na: http://www.songho.ca/dsp/convolution/convolution2d_example.html (1.4.2023.)
[42] Antoniadis, P. Calculate the Output Size of a Convolutional Layer [Online]. 2022. Dostupno na: https://www.baeldung.com/cs/convolutional-layer-size (2.4.2023.)
[43] OpenGenus IQ: Computing Expertise & Legacy ReLU (Rectified Linear Unit) Activation Function. [Online]. 2021. Dostupno na: https://iq.opengenus.org/relu-activation (2.4.2023.)
[44] Brownlee, J. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks [Online]. 2019. Dostupno na: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks (15.4.2023.)
[45] Prasad, R Convolutional Neural Network : The Easy Way [Online]. 2022. Dostupno na: https://dev.to/rav/convolutional-neural-network-the-easy-way-27ke (15.4.2023.)
[46] baeldung How ReLU and Dropout Layers Work in CNNs | Baeldung on Computer Science [Online]. 2020. Dostupno na: https://www.baeldung.com/cs/ml-relu-dropout-layers (16.4.2023.)
[47] Biswal, A. Convolutional Neural Network Tutorial [Online]. 2022. Dostupno na: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network (22.4.2023.)
[48] Educative. What is a neural network flatten layer? [Online]. Dostupno na: https://www.educative.io/answers/what-is-a-neural-network-flatten-layer (22.4.2023.)
[49] IndianTechWarrior Fully Connected Layer in Convolutional Neural Network [Online]. Dostupno na: https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/ (5.5.2023.)

[50] OpenGenus IQ: Learn Computer Science Fully Connected Layer: The brute force layer of a Machine Learning model [Online]. 2019. Dostupno na: https://iq.opengenus.org/fully-connected-layer/ (5.5.2023.)
[51] Rafaeli, D Sigmoid, Softmax and their derivatives [Online]. 2019. Dostupno na: https://themaverickmeerkat.com/2019-10-23-Softmax/ (5.5.2023.)
[52] Koech, K. E. Softmax Activation Function — How It Actually Works [Online]. 2020. Dostupno na: https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78 (11.5.2023.)
[53] GeeksforGeeks Intuition of Adam Optimizer [Online]. 2020. Dostupno na: https://www.geeksforgeeks.org/intuition-of-adam-optimizer (21.5.2023.)
[54] TensorFlow. (2022). tf.keras.Model | TensorFlow Core v2.7.0 [Online]. 2022. Dostupno na: https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit (21.5.2023.)
[55] Backpropagation Step by Step [Online]. Dostupno na: https://hmkcode.com/ai/backpropagation-step-by-step (21.5.2023.)
[56] GeeksforGeeks Intuition of Adam Optimizer [Online]. 2020. Dostupno na: https://www.geeksforgeeks.org/intuition-of-adam-optimizer (21.5.2023.)
[57] TensorFlow. (2022). tf.keras.Model | TensorFlow Core v2.7.0 [Online]. 2022. Dostupno na: https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit (1.6.2023.)
[58] Brownlee, J. Difference Between a Batch and an Epoch in Neural Network [Online]. 2018. Dostupno na: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch (1.6.2023.)
[59] Matplotlib: Python plotting — Matplotlib 3.1.1 documentation [Online]. Dostupno na: https://matplotlib.org (1.6.2023.)
[60] Maček, D. Tello drone controlled with face position [Online]. 2023. Dostupno na: https://www.youtube.com/watch?v=-U2GM6at6iU (18.9.2023.)
[61] Maček, D. Tello drone controlled with hand gestures [Online]. 2023. Dostupno na: https://www.youtube.com/watch?v=4-1uek8x8RU (18.9.2023.)

7. OZNAKE I KRATICE

Ah - Ampere hour (ampersat)

CNN - Convolutional neural network (Konvolucijska neuronska mreža)

DC - Direct current (istosmjerni)

GPU - Graphic processing unit (Grafička procesorska jedinica)

GRU - Gated recurrent units (Zatvorene rekurentne jedinice)

LSTM - Long short-term memory (Dugo kratkoročno pamćenje)

MP - Megapixel (megapiksel)

MLP - Multi-Layer Perceptrons (Višeslojni perceptroni)

RAF - Royal Air Force (Kraljevske zračne snage)

USB - Universal Serial Bus (Univerzalna serijska sabirnica)

VPU - Vision process unit (jedinica za obradu vida)

8. SAŽETAK

Naslov: Upravljanje bespilotnom letjelicom pomoću ljudske gestikulacije

U ovom završnom radu prikazana je implementacija detekcije lica i konvolucijskih neuronskih mreža za upravljanje bespilotnom letjelicom. Implementiran je sustav koji omogućava upravljanje bespilotnom letjelicom samo putem detekcije lica i gestikulacije rukama. Detaljno su opisane sve performanse i funkcionalnosti bespilotne letjelice. Sustav upravljanja bespilotnom letjelicom pojašnjen je kroz teoriju te je prikazana implementacija u programskim kodovima. Ovime je ostvarena interaktivna povezanost bespilotne letjelice i korisnika. Dobiveni rezultati su se pokazali vrlo uspješnima nakon raznih eksperimentiranja te ovaj rad može koristiti za daljnje usavršavanje u razne svrhe.

Ključne riječi: detekcija lica, gestikulacije rukama, konvolucijska neuronska mreža, bespilotna letjelica

9. ABSTRACT

Title: Drone control with human gestures

In this final thesis, the implementation of face detection and convolutional neural networks for controlling an unmanned aerial vehicle is presented (UAV). A system has been implemented that enables the control of a drone solely through face detection and hand gestures. All the performance and functionalities of the drone are described in detail. The drone control system is explained through theory and the implementation is demonstrated in source code. This achieves an interactive connection between the drone and the user. The obtained results proved to be highly successful after various experiments and this thesis can be used for further refinement for various purposes.

Keywords: face detection, hand gesticulations, convolutional neural network, UAV

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, 23. 10. 2023.	Domagoj Mlaček	

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Domagoj Maček
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon 27.10.2023. (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 23. 10. 2023.

DM
potpis studenta/ice