Prepoznavanje gesta za upravljanje dronom korištenjem YOLO algoritma

Klepac, Kristijan

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:553445

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2025-02-23**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Engineering

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

Prepoznavanje gesta za upravljanje dronom korištenjem YOLO algoritma

Rijeka, srpanj, 2024.

Kristijan Klepac
0069081789

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

Prepoznavanje gesta za upravljanje dronom korištenjem YOLO algoritma

Mentor: prof. dr. sc. Zlatan Car

Rijeka, srpanj 2024.

Kristijan Klepac
0069081789

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 14. ožujka 2023.

Zavod: **Zavod za automatiku i elektroniku**
Predmet: **Primjena umjetne inteligencije**
Grana: **2.03.06 automatizacija i robotika**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Kristijan Klepac (0069081789)**
Studij: Sveučilišni diplomski studij elektrotehnike
Modul: Automatika

Zadatak: **Prepoznavanje gesta za upravljanje dronom korištenjem YOLO algoritma**

Opis zadatka:

Predstaviti problematiku detekcije objekata primjenom umjetne inteligencije. Opisati strukturu i primjenu detekcijskih algoritama. Usporediti YOLO algoritme za detekciju objekata i primijeniti ih na problemu prepoznavanja gesta za upravljanje dronom te komentirati rezultate.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Klepac

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Prof. dr. sc. Zlatan Car

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Dubravko Franković

IZJAVA

Sukladno članku 8. Pravilnika o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija Tehničkog fakulteta Sveučilišta u Rijeci od 1. veljače 2020., izjavljujem da sam samostalno izradio diplomski rad prema zadatku preuzetom dana 20. ožujka 2023.

Rijeka, 7. srpnja, 2024.

Kristijan Klepac

Zahvaljujem se mentoru prof. dr. sc. Zlatanu Caru i asistentu Sandiju Baressi Šegoti, mag. ing. comp. na pomoći pri izradi ovog diplomskog rada te na ukazanom strpljenju i pristupačnosti. Isto tako, volio bih se zahvaliti prijateljici Tajani Cviji na nesebičnoj podršci, koju mi je pružala tijekom svih godina studiranja, i na svoj ukazanoj pomoći, kada god mi je bila potrebna.

Sadržaj

1. Uvod

U posljednjih nekoliko desetljeća tehnologija je napravila značajan napredak, posebno u području umjetne inteligencije, omogućavajući primjenu složenih algoritama za razne svrhe. Jedna od takvih je prepoznavanje i analiza pokreta tijela s ciljem upravljanja bespilotnim letjelicama što je ujedno i tema ovog rada. Bespilotne letjelice, odnosno dronovi danas su široko rasprostranjeni i koriste se u raznim industrijama kao što su filmska industrija, poljoprivredna industrija, vojna industrija, logistika, spasilačke misije, nadzor i mnoge druge primjene koje uključuju snimanje iz zraka. Tradicionalne metode upravljanja dronovima, kao što su daljinski upravljači, mogu biti ograničavajuće i zahtjevne prirode za neke korisnike, posebno u složenim ili dinamičnim okruženjima.

Prepoznavanje gesta tijela pomoću umjetne inteligencije predstavlja značajan napredak u interakciji između ljudi i strojeva. Korištenjem ove tehnologije upravljanje dronom postaje intuitivnije i prirodnije omogućavajući korisnicima upravljanje putem jednostavnih pokreta tijela. Ovaj pristup ne samo da poboljšava korisničko iskustvo, već i otvara nove mogućnosti za primjenu dronova u situacijama gdje su ruke operatera zauzete ili je potrebna brza i efikasna reakcija.

Jedan od najmodernijih algoritama umjetne inteligencije za prepoznavanje objekata je YOLO algoritam. Prednost YOLO algoritma je njegova brzina i točnost što ga čini idealnim za primjenu u stvarnom vremenu. Za razliku od tradicionalnih metoda detekcije objekata, koje zahtjevaju višestruke prolaze kroz neuronsku mrežu, YOLO algoritam analizira cijelu sliku odjednom, tj. dovoljan je jedan prolaz kroz neuronsku mrežu, što proces detekcije i prepoznavanja objekata čini značajno bržim.

Ovaj je rad organiziran na način da su u drugom poglavlju uvedeni i opisani osnovni pojmovi sa svrhom boljeg razumijevanja teorijske pozadine umjetnih neuronskih mreža. Prikazana je struktura umjetne neuronske mreže te je iznesena matematička pozadina gradijentnog spusta i unazadne propagacije kako bi se lakše razumjelo na koji način neuronska mreža uči.

Zatim u trećem poglavlju slijedi opis problematike detekcije objekata gdje je pažnja ponovno usmjerena na razumijevanje principa rada i načina na koji se postiže detekcija. Opisan je proces digitalne obrade slike te je dan opis konvolucijskih neuronskih mreža koje predstavljaju osnovni gradivni blok mnogih algoritama za detekciju objekata.

U četvrtom poglavlju iznesene su evaluacijske metrike za određivanje uspješnosti modela poput presjeka nad unijom, točnosti, preciznosti, odziva, srednje prosječne preciznosti i konfuzijske matrice. Neke od njih korištene su kasnije za ocjenjivanje vlastitog modela.

Peto poglavlje bavi se praktičnom primjenom YOLO algoritma za prepoznavanje gesta tijela. U poglavlju je dan opis skupa podataka na kojemu je model treniran zajedno s opisom procesa

treniranja modela. Nakon toga izneseni su postignuti rezultati. Na kraju poglavlja opisan je proces postavljanja okruženja za primjenu istreniranog modela za detekciju u stvarnom vremenu. Po izvršenoj detekciji dani su primjeri iste.

2. Teorija neuronskih mreža

U ovom poglavlju bit će iznesena kratka povijest i teorijska podloga potrebna za razumijevanje principa rada umjetnih neuronskih mreža. Kako bi se uopće krenulo s uvođenjem i definiranjem osnovnih pojmova, poželjno je problematiku započeti s biološkom neuronskom mrežom koja je poslužila kao inspiracija pri stvaranju umjetne neuronske mreže.

Prije samog početka, reći će se par riječi o povijesti i razvitku umjetnih neuronskih mreža. Tekst koji slijedi pisan je prema [2], [3], [4].

Koncept umjetnih neuronskih mreža javlja se 1943. kada W. McCulloch i W. Pitts predstavljaju matematički model umjetnog neurona (perceptron) kakav se i danas koristi. Psiholog D. Hebb napisao je 1949. "The Organization of Behavior" u kojem opisuje način na koji neuroni uče te iznosi da veza između neurona postaje jača što se oni više koriste. To pravilo primjenjivo je kako na biološke neurone tako i na umjetne.

Sljedeće značajno postignuće ostvareno je 1958. konstrukcijom elektroničkog uređaja, nazvanog "Mark I perceptron", koji je predstavljao hardversku implementaciju neuronske mreže. Tvorac uređaja bio je američki psiholog F. Rosenblatt. Iako je uređaj posjedovao sposobnost učenja, bio je dosta ograničenih sposobnosti iz razloga što je mogao rješavati probleme koji su bili isključivo linearno separabilne prirode što u praksi nije čest slučaj. Zbog navedenog problema pojavio se manjak entuzijazma i interesa za daljnjim razvojem ove tehnologije te su 1969. M. Minsky i S. Papert objavili knjigu "Perceptrons: An Introduction to Computational Geometry" s ciljem da umanje značaj dosadašnjih postignuća vezanih za istraživanje neuronskih mreža. U knjizi ukazuju na prisutnost fundamentalnih problema te nedostatak procesorske snage za treniranje korisnih neuronskih mreža.

Sve do 1980-ih nije bilo previše interesa za daljnjim istraživanjem dok nije došlo do nekih novih postignuća. Jedno od takvih postignuća je primjena algoritma unazadne propagacije (*engl. backpropagation*) za minimiziranje greške. Iako je otkriven već 1974., tek je 1986. dobio na značaju.

Pojavom snažnijih procesora, složenijih struktura i algoritama daljnji razvoj počinje eksponencijalno rasti i traje sve do danas.

2.1. Neurobiološka analogija

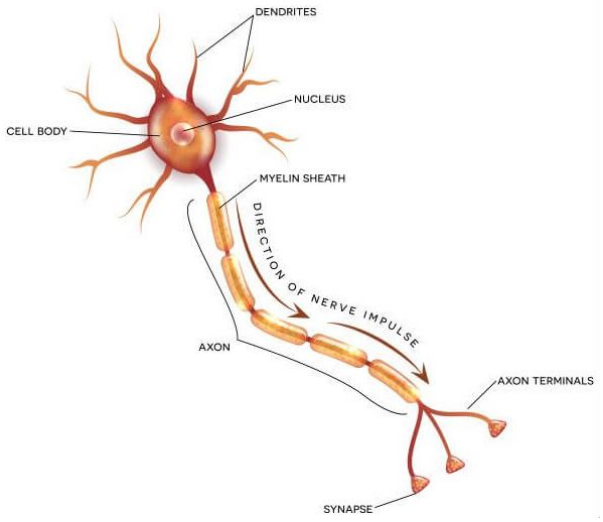
Radi lakšeg razumjevanja modela umjetnog neurona i umjetne neuronske mreže uobičajeno je najprije krenuti s objašnjenjem biološkog neurona. Tekst koji slijedi pisan je prema [5], [6].

Neuron ili živčana stanica osnovna je jedinica živčanog sustava. Uloga neurona je primanje

podržaja iz okoline, slanje električnih signala za pokretanje mišića te međusobna komunikacija s ostalim neuronima. Osnovni dijelovi neurona su dendriti, tijelo stanice (soma), akson i sinapsa.

Dendriti služe za primanje informacija s drugih neurona. Preko njih se informacija šalje u tijelo stanice. Akson je produžetak neurona te služi za prijenos živčanih impulsa s tijela stanice na druge neurone ili mišićna vlakna. Sinapsa služi za komunikaciju između neurona.

Na slici 2.1. prikazana je građa biološkog neurona.



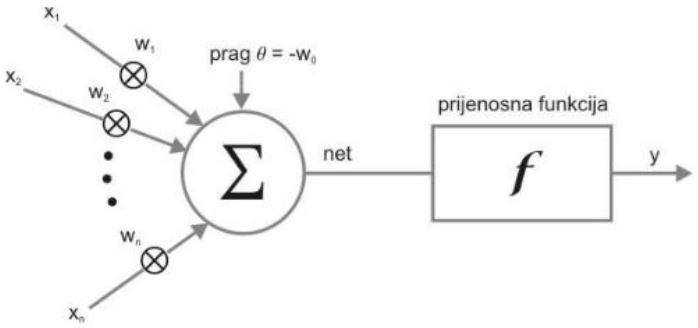
Slika 2.1. Građa neurona. Izvor:[6]

2.2. Umjetna neuronska mreža

Umjetna neuronska mreža (*engl. artificial neural network*) jedan je od najčešće korištenih algoritama strojnog učenja, a predstavlja računalno implementirani ekvivalent biološke neuronske mreže. Ono po čemu se ova vrsta algoritama razlikuje od "staromodnih" algoritama sposobnost je učenja direktno iz podataka. Drugim riječima, nije potrebno izričito napisati programski kod kojim će se rješiti određeni problem, već se umjetna neuronska mreža sama prilagođava problemu.

Umjetna neuronska mreža sastoji se od umjetnih neurona koji formiraju slojeve. Svaki neuron prima ulaze, primjenjuje aktivacijsku funkciju te generira izlaz. Model umjetnog neurona prikazan je na slici 2.2.

Ulazi, označeni s $x_1, x_2, ..., x_n$, predstavljaju numeričke vrijednosti koje neuron prima. Ti ulazi se zatim množe s težinskim faktorima (*engl. weights*), označnenim s $w_1, w_2, ..., w_n$, a predstavljaju ekvivalent jakosti sinapse kod biološkog neurona. Takvi skalirani ulazi se zbrajaju te im se dodaje prag θ (*engl. bias*) koji predstavlja granicu (*engl. threshold*) za aktivaciju neurona. Na kraju se na rezultat primjenjuje aktivacijska (prijenosna) funkcija (*engl. activation function*) čiji izlaz ujedno predstavlja i izlaz iz neurona.



Slika 2.2. Model umjetnog neurona. Izvor:[7]

Matematički se izlaz iz neurona može zapisati izrazom 2.1:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + \theta\right). \tag{2.1}$$

Da bi izraz 2.1 bilo lakše razumjeti, potrebno je detaljnije objasniti ulogu aktivacijske funkcije.

Problemi koji se pojavljuju u praksi uobičajeno su nelinearne prirode što znači da linearni modeli nisu prihvatljivi u većini primjena. Ako se promotri izraz 2.1, može se uočiti da, ukoliko ne bi postojala aktivacijska funkcija, izlaz iz neurona predstavljao bi samo rezultat skaliranih i zbrojenih ulaznih vrijednosti. Drugim riječima, takav model umjetnog neurona bio bi linearan, što bi značajno ograničilo njegovu primjenu, tj. primjenu umjetnih neuronskih mreža. Zbog navedenog razloga prirodno se nameće potreba za nelinearnim modelom, a to se postiže uvođenjem nelinearnih aktivacijskih funkcija. Na taj način neuronska mreža postaje puno fleksibilnija i primjenjivija na stvarne probleme.

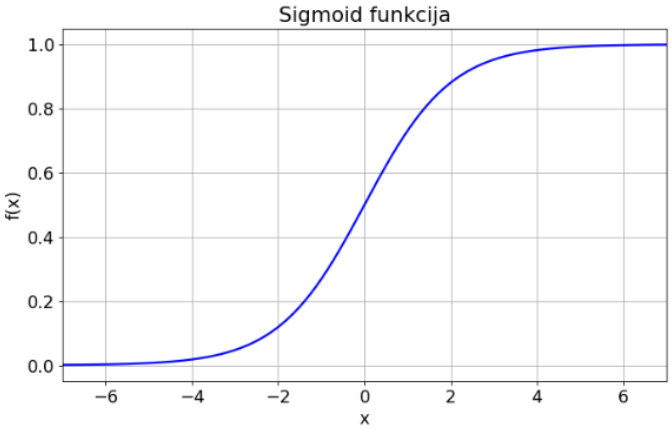
U nastavku se navodi nekoliko najčešćih aktivacijskih funkcija.

1. Sigmoidalna funkcija

   Sigmoidalna (logistička) funkcija definirana je izrazom 2.2, a prikazana na slici 2.3.:

   $$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.2}$$

   Sa slike je vidljivo da ova funkcija ograničava izlaznu vrijednost neurona na vrijednost iz intervala $[0, 1]$ što ju čini pogodnom za primjenu kod binarne klasifikacije [9]. Također, derivabilna je na cijeloj domeni, što kod nekih funkcija nije slučaj. Problem koji se javlja kod ove funkcije je to što kada je izlazna vrijednost neurona blizu 0 ili 1, vrijednost derivacije je približna nuli, što značajno usporava proces učenja mreže [1]. U engleskoj literaturi problem je poznat pod nazivom *vanishing gradient problem*.



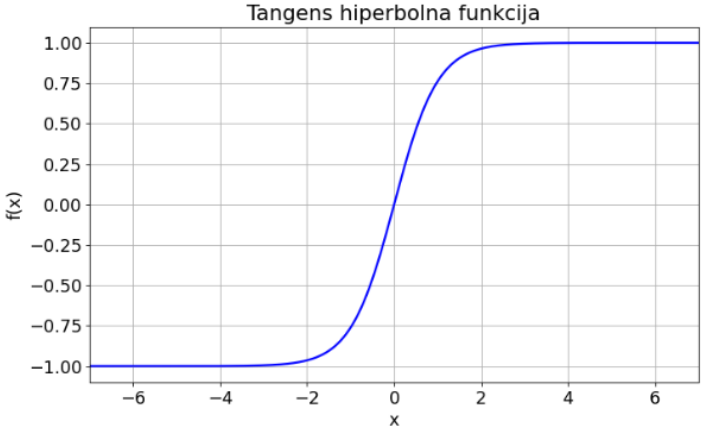
Slika 2.3. Sigmoidalna funkcija. Izvor:[8]

2. Tangens hiperbolna funkcija

   Tangens hiperbolna funkcija definirana je izrazom 2.3, a prikazana na slici 2.4.:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{2.3}$$

   Ova funkcija vrlo je slična sigmoidalnoj funkciji. Razlikuje se po tome što kod ove funkcije izlazna vrijednost neurona je iz intervala $[-1, 1]$, a ne $[0, 1]$ kao kod sigmoidalne funkcije. Također ova funkcija je brže rastuća od sigmoidalne, što znači da je i vrijednost derivacije veća, što utječe na brzinu učenja. Ova funkcija također ima problem u derivaciji približno jednakoj nuli kada je izlazna vrijednost blizu -1 ili 1 [10].



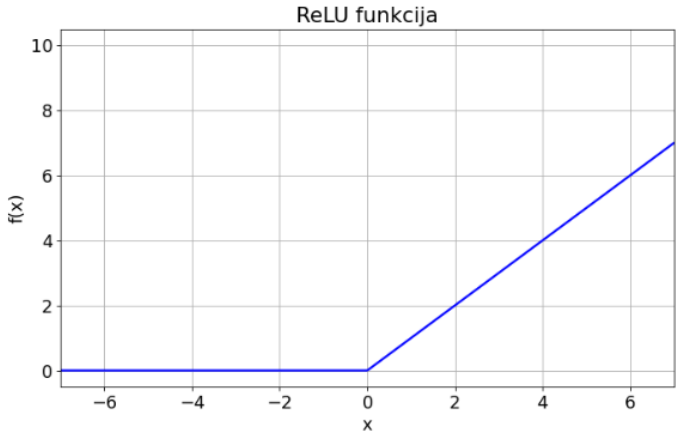
Slika 2.4. Tangens hiperbolna funkcija. Izvor:[8]

3. ReLU funkcija

   ReLU (*engl. Rectified Linear Unit*) funkcija definirana je izrazom 2.4, a prikazana na slici 2.5.:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{inače} \end{cases}. \tag{2.4}$$

ReLu je vrlo jednostavna funkcija koja na izlazu daje ulazne vrijednosti ukoliko su one pozitivne, a nulu ukoliko su ulazne vrijednosti negativne. Rezultati dobiveni korištenjem ove funkcije pokazali su se boljima od rezultata dobivenih korištenjem sigmoidalne i tangens hiperbolne funkcije što ovu funkciju čini vrlo popularnom, pogotovo kod dubokih neuronskih mreža. Kod ove funkcije ne postoji problem "nestajućeg gradijenta", već je derivacija konstantna. Međutim, problem koji se ovdje javlja je tzv. *dying ReLU problem*. Ukoliko se dogodi da su sve ulazne vrijednosti u neuron negativne, zbog svojstva ReLU funkcije, izlaz iz neurona će biti 0. To znači da će i derivacija biti jednaka nuli što će prilikom procesa učenja rezultirati da taj neuron ostaje neaktivan, a to može značajno utjecati na performanse neuronske mreže [11].



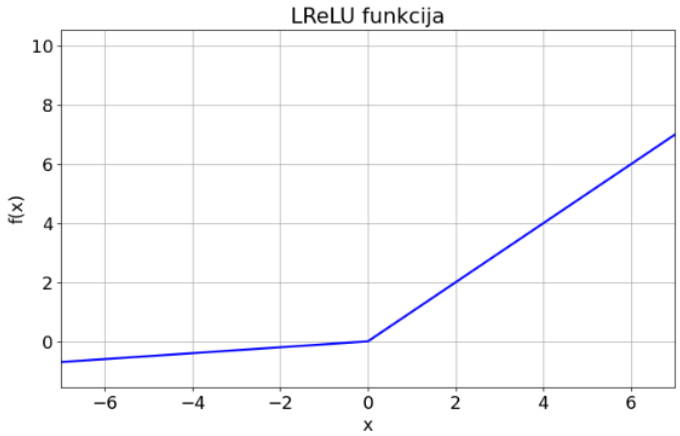
Slika 2.5. ReLU funkcija. Izvor:[8]

4. LReLU funkcija

LReLU (*engl. Leaky Rectified Linear Unit*) funkcija definirana je izrazom 2.5, a prikazana na slici 2.6.:

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}. \tag{2.5}$$

Ova funkcija je modifikacija ReLU funkcije. Spomenuti *dying ReLU problem* može se riješiti uvođenjem nagiba (parametar α) za negativne ulazne vrijednosti. Uobičajeno se radi o malom nagibu, npr. 0.01, no osigurava se postojanje ne-nulte derivacije na cijeloj domeni [11].



Slika 2.6. LReLU funkcija. Izvor:[8]

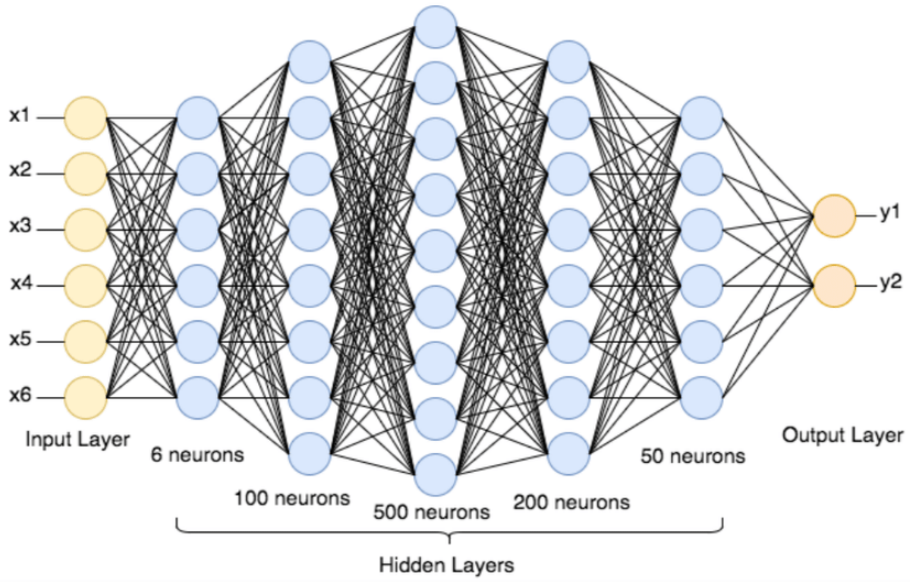
2.3. Struktura umjetne neuronske mreže

Nakon uvođenja osnovnih pojmova i opisa umjetnog neurona u ovom dijelu rada slijedi detaljniji uvid u strukturu umjetne neuronske mreže. U nastavku rada neće biti posebno naglašavano da se radi o umjetnoj vrsti neuronske mreže, već će se to podrazumijevati.

Kao što je već spomenuto na početku poglavlja, neuronska mreža sačinjena je od prethodno opisanih neurona. Ti neuroni međusobno su povezani te zajedno formiraju slojeve. Slojevi se mogu podijeliti na ulazni sloj (*engl. input layer*), skriveni sloj (*engl. hidden layer*) i izlazni sloj (*engl. output layer*). Za razliku od ulaznog i izlaznog sloja, čija uloga je sasvim jasna, značenje skrivenih slojeva teško je interpretirati te još uvijek nema konkretnog objašnjenja šta se točno događa i na koji način skriveni slojevi utječu na izlaz neuronske mreže.

Neuronska mreža može imati više skrivenih slojeva te se tada govori o dubokoj neuronskoj mreži (*engl. deep neural network*). Dodavanjem skrivenih slojeva povećava se složenost mreže te je samim time mreža primjenjivija za složenije probleme. Na slici 2.7. prikazana je duboka neuronska mreža s 5 skrivenih slojeva.

Projektiranje ulaznog i izlaznog sloja često je jednostavno, no projektiranje skrivenih slojeva puno je zahtjevniji postupak. Problem je u tome što ne postoje egzaktna pravila kojima je moguće voditi se prilikom projektiranja skrivenih slojeva, već su razvijene heurističke metode kojima se pokušava optimizirati neuronska mreža, npr. kako izabrati broj skrivenih slojeva u odnosu na vrijeme potrebno za uspješno treniranje mreže [1].



Slika 2.7. Duboka neuronska mreža. Izvor:[12]

2.4. Učenje neuronske mreže

Do sada je već nekoliko puta spomenut pojam treniranja, odnosno učenja, no nigdje nije dano objašnjenje istoga. Sposobnost učenja ključno je svojstvo zbog kojega se neuronske mreže, uz ostale vrste algoritama umjetne inteligencije, ističu u odnosu na "neinteligentne" algoritme te će iz tog razloga ovaj dio rada biti posvećen objašnjenju i razumijevanju matematičke pozadine učenja neuronskih mreža. Ovo podpoglavlje pisano je prema [1].

Cilj je učenja neuronske mreže pronaći optimalne parametre (težinske faktore i pragove) za obavljanje određenog zadatka. To se postiže tako da se na ulaz neuronske mreže dovede skup podataka za učenje (*engl. training set*) pomoću kojega će se neuronska mreža prilagođavati danom problemu. Drugim rječima, učenje je proces pronalaženja optimalnih parametara mreže na temelju pručavanja skupa podataka za učenje.

2.4.1. Funkcije gubitka

Da bi se moglo procijeniti koliko je mreža uspješna u obavljanju zadatka, potrebno je kvantificirati uspješnost mreže, a to se postiže uvođenjem funkcije gubitka (*engl. loss function*). Jedna takva funkcija je MSE (*engl. mean squared error*), a predstavlja srednju kvadratnu pogrešku i dana je izrazom 2.6.:

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N}\left[y(x_i) - a_i\right]^2, \tag{2.6}$$

gdje su w težinski faktori, b pragovi, N ukupan broj ulaznih podataka za treniranje, $y(x_i)$ željeni izlaz mreže za i-ti ulaz x_i, a a_i predstavlja trenutni izlaz za i-ti ulaz x_i. Valja napomenuti kako

su navedene varijable zapravo vektori, odnosno matrice, tako da w predstavlja matricu težinskih faktora, b predstavlja vektor pragova, x predstavlja vektor ulaza, a $y(x)$ i a predstavljaju vektore izlaza, no radi jednostavnosti zapisa izraza to se neće posebno naglašavati.

Iz izraza 2.6 vidi se da, ukoliko je trenutni izlaz iz neuronske mreže približno jednak željenom, odnosno ako vrijedi izraz 2.7:

$$y(x_i) - a_i \approx 0, \tag{2.7}$$

tada će i funkcija gubitka biti približno jednaka nuli. To znači da je potrebno pronaći one težinske faktore i pragove za koje će funckija gubitka biti minimalna. Navedeni problem se rješava minimizacijom funkcije gubitka o čemu će se detaljnije govoriti u nastavku ovog podpoglavlja.

Još je jedna često korištena funkcija gubitka unakrsna entropija (*engl. cross entropy*). Unakrsna je entropija mjera različitosti dviju distribucija [13]. Ukazuje koliko se predviđena vjerojatnost razlikuje u odnosu na stvarnu vjerojatnost te se uglavnom koristi za klasifikacijske probleme, a definirana je izrazom 2.8:

$$L(w, b) = -\sum_{j=1}^{K} y(x_j) \log(\hat{y}(x_j)), \tag{2.8}$$

gdje su w težinski faktori, b pragovi, K ukupan broj klasa, x_j ulaz j-te klase, $y(x_j)$ stvarna vjerojatnost j-te klase, a $\hat{y}(x_j)$ predstavlja predviđenu vjerojatnost j-te klase.

Ukoliko se želi usrednjiti rezultat unakrsne entropije tada se dobije izraz 2.9:

$$L(w, b) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y(x_{ij}) \log(\hat{y}(x_{ij})), \tag{2.9}$$

gdje je N ukupan broj ulaznih podataka za treniranje.

Da bi se dobila vjerojatnost predikcije pojedinih klasa koristi se *softmax* funkcija koja je definirana izrazom 2.10:

$$f(s)_j = \frac{e^{s_{y_j}}}{\sum_{i=1}^{K} e^{s_{y_i}}}, \tag{2.10}$$

gdje je s_{y_j} vrijednost pridružena j-toj predikciji, a s_{y_i} vrijednost pridružena i-toj predikciji.

Koristeći *softmax* funkciju, izraz 2.8 se može zapisati pomoću izraza 2.11:

$$L(w, b) = -\sum_{j=1}^{K} y(x_j) \log(f(s)_j), \tag{2.11}$$

gdje je $f(s)_j$ *softmax* funkcija.

2.4.2. Gradijentni spust

Nakon što je definirana funkcija gubitka, potrebno ju je minimizirati kako bi se optimizirala mreža. Metoda kojom se to postiže zove se gradijentni spust (*engl. gradient descent*).

Općenito je funkcija gubitka funkcija više varijabli koja ovisi o svim težinskim faktorima i pragovima u mreži. Radi jednostavnijeg objašnjenja, promatrati će se funkcija gubitka u ovisnosti samo o težinskim faktorima, a ovisnost o pragovima bit će zanemarena. Promjena funkcije gubitka $\Delta L(w)$ u ovisnosti o malim promjenama težinskih faktora Δw tada se može zapisati izrazom 2.12:

$$\Delta L(w) \approx \frac{\partial L(w)}{\partial w_1}\Delta w_1 + \frac{\partial L(w)}{\partial w_2}\Delta w_2 + ... + \frac{\partial L(w)}{\partial w_m}\Delta w_m. \tag{2.12}$$

Ideja je pronaći težinske faktore Δw za koje je promjena $\Delta L(w)$ negativna kako bi postigli minimum funkcije. Ako se definira vektor promjena težinskih faktora prema izrazu 2.13:

$$\Delta w = [\Delta w_1, \Delta w_2, ..., \Delta w_m]^T, \tag{2.13}$$

gdje T označava transponirani vektor i ako se definira gradijentni vektor prema izrazu 2.14:

$$\nabla L(w) = \left[\frac{\partial L(w)}{\partial w_1}, \frac{\partial L(w)}{\partial w_2}, ..., \frac{\partial L(w)}{\partial w_m}\right]^T, \tag{2.14}$$

tada se izraz 2.12 može zapisati kako je dano izrazom 2.15:

$$\Delta L(w) \approx \nabla L(w) \cdot \Delta w, \tag{2.15}$$

gdje je $\nabla L(w)$ gradijent funkcije gubitka, a Δw vektor promjena težinskih faktora. Ovakav zapis omogućuje takav odabir Δw da se postigne željena negativna promjena $\Delta L(w)$.

Ako se promjena težinskih faktora definira izrazom 2.16:

$$\Delta w = -\eta \nabla L(w), \tag{2.16}$$

gdje je η stopa učenja (*engl. learning rate*), izraz 2.15 se može zapisati pomoću izraza 2.17:

$$\Delta L(w) \approx \nabla L(w) \cdot (-\eta \nabla L(w)). \tag{2.17}$$

Stopa učenja η pozitivan je brojčani parametar koji upravlja brzinom učenja. Ukoliko se izabere premala vrijednost, model će presporo konvergirati. Nasuprot tome, ako se izabere prevelika vrijednost, učenje će se odvijati prebrzo što može rezultirati divergencijom modela.

Nakon sređivanja izraza 2.17 dobije se izraz 2.18:

$$\Delta L(w) \approx -\eta \|\nabla L(w)\|^2. \tag{2.18}$$

Budući da je $\|\nabla L(w)\|^2 \geq 0$, zadovoljeno je to da je $\Delta L(w) \leq 0$, što je i bio cilj.

Uvažavajući sve navedeno, koristeći izraz 2.16 moguće je zapisati iterativno pravilo za računanje težinskih faktora. Pravilo za računanje težinskih faktora dano je izrazom 2.19:

$$w_{k+1} = w_k - \eta \frac{\partial L(w, b)}{\partial w_k}, \tag{2.19}$$

gdje je s w_{k+1} označena nova vrijednost težinskog faktora, a s w_k označena stara vrijednost težinskog faktora.

Analogno tome, izrazom 2.20 dano je pravilo za računanje pragova:

$$b_{l+1} = b_l - \eta \frac{\partial L(w, b)}{\partial b_l}, \tag{2.20}$$

gdje je s b_{l+1} označena nova vrijednost praga, a s b_l označena stara vrijednost praga.

Ono što se u praksi češće koristi je modificirana verzija gradijentnog spusta koja se zove stohastički gradijentni spust (*engl. stochastic gradient descent*). Kod prethodno objašnjenog gradijentnog spusta javlja se problem sporog učenja, a razlog tomu je to što se gradijent funkcije gubitka mora računati za svaki pojedinačni primjer učenja. Problem je posebno izražen kod velikih skupova za učenje. Ideja stohastičkog gradijentnog spusta je da se skup podataka za učenje nasumično podijeli u manje skupove (*engl. mini-batch*) te se gradijent funkcije gubitka izračuna približno.

Aproksimacija gradijenta je tada dana izrazom 2.21:

$$\nabla L(w) \approx \frac{1}{M} \sum_{j=1}^{M} \nabla L(w)_{X_j}, \tag{2.21}$$

gdje je M broj manjih skupova za učenje, a $\nabla L(w)_{X_j}$ gradijent j-tog skupa za učenje.

Primjenom stohastičkog gradijentnog spusta težinski faktori i pragovi se računaju kako je dano izrazima 2.22 i 2.23:

$$w_{k+1} = w_k - \frac{\eta}{M} \sum_{j=1}^{M} \frac{\partial L(w, b)_{X_j}}{\partial w_k}, \tag{2.22}$$

$$b_{l+1} = b_l - \frac{\eta}{M} \sum_{j=1}^{M} \frac{\partial L(w, b)_{X_j}}{\partial b_l}. \tag{2.23}$$

2.4.3. Unazadna propagacija

Kod višeslojnih mreža dolazi do problema izračuna gradijenta iz razloga što je funkcija gubitka dobivena propagacijom ulaznih podataka kroz raličite slojeve u mreži s različitim parametrima [8]. Problem se rješava primjenom unazadne propagacije (*engl. backpropagation*).

Radi lakšeg objašnjenja prvo će biti prikazano kako se izlaz neurona mijenja u ovisnosti o promjeni težinskog faktora.

Neka je a_j^l izlaz j-tog neurona u l-tom sloju i neka je w_{jk}^l težinski faktor koji povezuje k-ti neuron iz prethodnog sloja s j-tim neuronom u l-tom sloju. Promjena izlaza neurona je tada dana izrazom 2.24:

$$\Delta a_j^l \approx \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l. \tag{2.24}$$

Navedena će promjena utjecati na promjenu u sljedećem sloju. Radi jednostavnosti promatrati će se samo jedan put u mreži, tj. promjena jednog neurona u sljedećem sloju. Tada je promjena izlaza q-tog neurona u sljedećem sloju dana izrazom 2.25:

$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \Delta a_j^l. \tag{2.25}$$

Uvrštavanjem izraza 2.24 u izraz 2.25 dobiva se izraz 2.26 koji povezuje promjenu izlaza neurona u sloju $l+1$ u ovisnosti o težinskom faktoru iz l-tog sloja:

$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l. \tag{2.26}$$

Analogno prethodnim rezultatima, izrazom 2.27, može se definirati promjena izlaza p-tog neurona u sloju $l+2$:

$$\Delta a_p^{l+2} \approx \frac{\partial a_p^{l+2}}{\partial a_q^{l+1}} \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l. \tag{2.27}$$

Može se uočiti da se ponavljanjem ovog postupka rezultat ulančanim pravilom propagira unaprijed sve do izlaznog sloja gdje se na kraju dobije ovisnost promjene funkcije gubitka o promjeni težinskog faktora.

Kako je unaprijednom propagacijom dobiven izlazni rezultat, ideja je unazadne propagacije grešku propagirati u suprotnom smjeru te na taj način izračunati optimalne parametre mreže. Proces započinje izračunom greški neurona u izlaznom sloju te se pomoću toga računaju greške neurona u prethodnom sloju. Proces se ponavlja unazadno za svaki sloj. Na kraju se greške povezuju s parametrima mreže.

Greška j-tog neurona u l-tom sloju označit će se s δ_j^l. Također radi jednostavnijeg zapisa, uvodi se oznaka z_j^l kojom će se označit zbroj skaliranih ulaza u neuron, tj. rezultat nad kojim se primjenjuje prijenosna funkcija.

Unazadna propagacija definirana je pomoću 4 jednadžbe koje slijede u nastavku zajedno s objašnjenjima.

Jednadžba za izračun greške u izlaznom sloju, definirana izrazom 2.28:

$$\delta^L = \nabla_a L \odot f'(z^L), \tag{2.28}$$

gdje je δ^L vektor grešaka izlaznog sloja čije su komponente δ_j^L, $\nabla_a L$ gradijent funkcije gubitka u ovisnosti o izlaznom sloju, a $f'(z^L)$ je derivacija aktivacijske funkcije u ovisnosti o vektoru z^L čije su komponente z_j^L. Oznaka $\odot$ Hadamardov je produkt (*engl. Hadamard product*), a predstavlja umnožak dviju matrica istih dimenzija po elementima, gdje su elementi rezultirajuće matrice dobiveni množenjem odgovarajućih elemenata iz istih pozicija u matricama.

Jednadžba za izračun greške u l-tom sloju u ovisnosti o sljedećem sloju, definirana izrazom 2.29:

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot f'(z^l). \tag{2.29}$$

Ova jednadžba može se interpretirati kao propagacija greške unatrag. Množenjem greške s transponiranom matricom težinskih faktora $\left(w^{l+1}\right)^T$ dobiva se učinak unazadne propagacije. Zatim se primjenom hadamardovog produkta grešku "provlači" unatrag kroz aktivacijsku funkciju kako bi se dobila greška δ^l.

Jednadžba za mjeru promjene funkcije gubitka u ovisnosti o pragovima mreže, definirana izrazom 2.30:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \tag{2.30}$$

Jednadžba za mjeru promjene funkcije gubitka u ovisnosti o težinskim faktorima mreže, definirana izrazom 2.31:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l. \tag{2.31}$$

Zaključno, pomoću izraza 2.28 računaju se greške neurona u izlaznom sloju, a zatim se pomoću izraza 2.29 greška u izlaznom sloju povezuje s greškom u prethodnom sloju te se iterativno propagira unatrag. Pomoću izraza 2.30 i 2.31 računaju se promjene funkcije gubitaka u ovisnosti o pragovima i težinskim faktorima koristeći prethodno izračunate greške neurona.

3. Računalni vid i detekcija objekata

Računalni vid predstavlja područje umjetne inteligencije koje ima za cilj omogućiti računalima sposobnost vizualnog percipiranja i interpretacije okoline slično ljudskom vidu. Aspekti koje računalni vid uključuje su detekcija objekata, prepoznavanje objekata, detekcija događaja, praćenje pokreta i sl. [14]. Ovaj se rad konkretno bavi detekcijom i prepoznavanjem objekata te u nastavku ovog poglavlja slijedi više govora o navedenoj problematici.

3.1. Digitalna obrada slike i ekstrakcija značajki

Kako bi se razumjela računalna detekcija i prepoznavanje objekata, potrebno je razumjeti na koji način računalo "vidi" sliku.

Za razliku od čovjeka, računalo ne vidi sliku kao skup objekata različitih svjetlina i boja, već niz numeričkih vrijednosti koje definiraju sliku. Te numeričke vrijednosti zovu se pikseli (*engl. pixels*) i opisuju sliku pomoću matrice dimenzija $H \times W$, gdje H označava visinu slike, a W širinu slike. Uvažavajući rečeno slika A može se zapisati izrazom 3.1:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1W} \\ a_{21} & a_{22} & \cdots & a_{2W} \\ \vdots & \vdots & \ddots & \vdots \\ a_{H1} & a_{H2} & \cdots & a_{HW} \end{bmatrix}, \tag{3.1}$$

gdje elementi a_{ij} predstavljaju vrijednosti piksela.

Ova je definicija valjana za monokromatske slike tj., slike sačinjene od nijansi sive boje. Tada vrijednost svakog piksela označava intenzitet sive boje. Međutim, ukoliko se radi o slikama u boji (RGB) tada je slika sačinjena od nijansi crvene, zelene i plave boje te se tada za svaku boju posebno definira matrica, analogno izrazu 3.1, gdje elementi tih matrica predstavljaju intenzitet te boje na lokaciji ij. Tada se govori o višekanalnim slikama te se iste zapisuju pomoću tenzora dimenzija $H \times W \times D$, gdje H označava visinu slike, W širinu slike, a D broj kanala, tj. dubinu slike. RGB slika je trokanalna slika te se može zapisati tenzorom dimenzija $H \times W \times 3$ [8].

Ono što svaku sliku cini jedinstvenom u odnosu na neku drugu sliku njene su značajke (*engl. features*). Značajke mogu biti jednostavni uzorci poput rubova, krugova i sl., a mogu biti i neki složeniji uzorci. Ideja kojom se pristupa problematici prepoznavanja objekata sa slike temelji se na procesu ekstrakcije njezinih značajki (*engl. feature extraction*).

Ekstrakcija značajki vrši se pomoću filtara (*engl. kernel*) koji su posebno dizajnirani s ciljem prepoznavanja željenih uzoraka. U nastavku su dani primjeri nekih filtara koji se često koriste.

1. Filtar za zaglađivanje, definiran izrazom 3.2:

$$F = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{3.2}$$

2. Filtar za detekciju vertikalnih rubova (Sobel), definiran izrazom 3.3:

$$F = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

3. Filtar za detekciju horizontalnih rubova (Sobel), definiran izrazom 3.4:

$$F = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{3.4}$$

Matematička operacija koja omogućava ekstrakciju značajki zove se konvolucija (*engl. convolution*), a njenom primjenom dobiva se mapa značajki (*engl. feature map*). Konvolucija je definirana izrazom 3.5:

$$z_{pq} = \sum_{i=1}^{F_H} \sum_{j=1}^{F_W} w_{ij} a_{i+p,j+q}, \tag{3.5}$$

gdje je z_{pq} element mape značajki na poziciji pq, F_H visina filtra, F_W širina filtra, w_{ij} element filtra na poziciji ij, a a_{ij} element slike na poziciji ij.

Prethodno iznesena ideja služi kao temelj za igradnju alata za prepoznavanje objekata, a to su konvolucijske neuronske mreže (*engl. convolutional neural networks*) o kojima će više govora biti u sljedećem potpoglavlju.

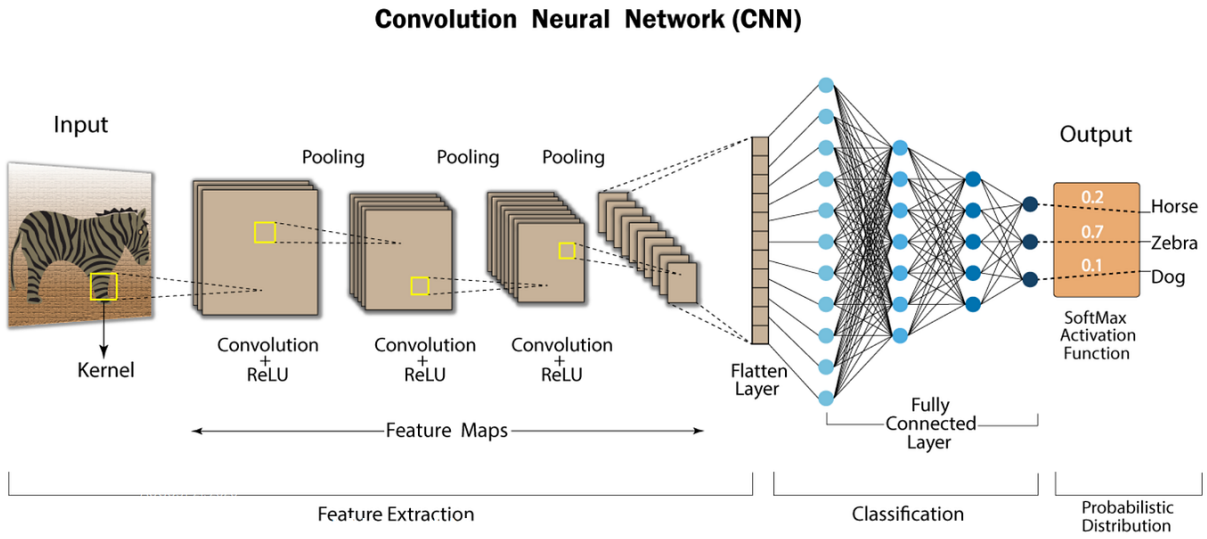
3.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (CNN) posebna su vrsta neuronskih mreža koje su prilagođene za obradu vizualnih informacija. Često se primjenjuju u zadacima kao što su prepoznavanje objekata, klasifikacija slika i segmentacija [16].

Razvoj konvolucijskih neuronskih mreža započinje u razdoblju od 1980. do 1990. godine, kada je Yann LeCun razvio LeNet, jednu od prvih konvolucijskih neuronskih mreža koja je služila za prepoznavanje rukom pisanih brojeva. Unatoč pionirskim radovima, interes za konvolucijskim neuronskim mrežama dosegao je vrhunac tek 2012. godine s pojavom AlexNet-a, duboke konvolucijske mreže kojom su natjecatelji Alex Krizhevsky, Geoffrey Hinton i Ilya Sutskever pobijedili

na ILSVRC-u (ImageNet Large Scale Visual Recognition Challenge) [8], [16]. Nakon uspjeha AlexNet-a, počeo je nagli porast interesa za dubokim konvolucijskim neuronskim mrežama. U to su vrijeme razvijene mreže poput VGGNet, GoogLeNet (Inception) i ResNet, koje su bile sve dublje i sposobne naučiti sve složenije značajke.

Struktura jednostavne konvolucijske neuronske mreže dana je slikom 3.1.



Slika 3.1. Struktura konvolucijske neuronske mreže. Izvor:[17]

Konvolucijska neuronska mreža sastoji se od ulaznog sloja, konvolucijskog sloja, aktivacijskog sloja, sloja sažimanja, potpuno povezanog sloja te izlaznog sloja.

Ulazni sloj (*engl. input layer*) prvi je sloj mreže koji prima ulazne podatke, a to su najčešće slike ili neki drugi tip strukturalnih podataka.

Konvolucijski sloj (*engl. convolutional layer*) osnovni je i najbitniji dio CNN-a. U tom sloju primjenjuje se konvolucija slike i filtra čime se dobiva prethodno spomenuta mapa značajki. Dio slike koji je obuhvaćen filtrom naziva se receptivno polje (*engl. receptive field*). Bitno je napomenuti da je u istom sloju moguće primjeniti više filtara s ciljem propoznavanja različitih uzoraka te se time dobiva i više mapi značajki. Drugim rječima, broj dobivenih mapi značajki odgovara broju primjenjenih filtara. Također, dubina filtra jednaka je dubini slike na koju se isti primjenjuje.

Aktivacijski sloj (*engl. activation layer*) slijedi nakon konvolucijskog sloja te se u njemu primjenjuje aktivacijska funkcija. Najčešće korištena aktivacijska funkcija je ReLu funkcija.

Sloj sažimanja (*engl. pooling layer*) sloj je koji služi smanjivanju dimenzija mapi značajki. Obično se primjenjuje sažimanje maksimumom (*engl. max pooling*) ili sažimanje uprosječivanjem (*engl. average pooling*) [8]. Ovisno o kojem načinu sažimanja se govori, proces se odvija tako da se "izvuče" ili najveći element receptivnog polja ili se izračuna prosjek vrijednosti unutar receptivnog polja.

Potpuno povezani sloj (*engl. fully connected layer*) predstavlja dio mreže gdje je svaki neuron

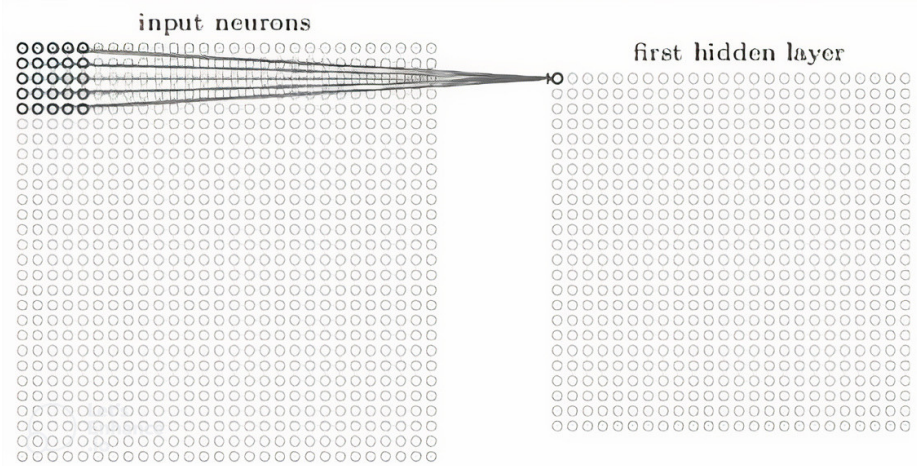
u sljedećem sloju povezan sa svakim neuronom iz prethodnog sloja te služi za klasifikaciju.

Izlazni sloj (*engl. output layer*) posljednji je sloj mreže koji daje rezultate. Ako se radi o klasifikaciji tada često predstavlja vjerojatnost predikcije pojedine klase.

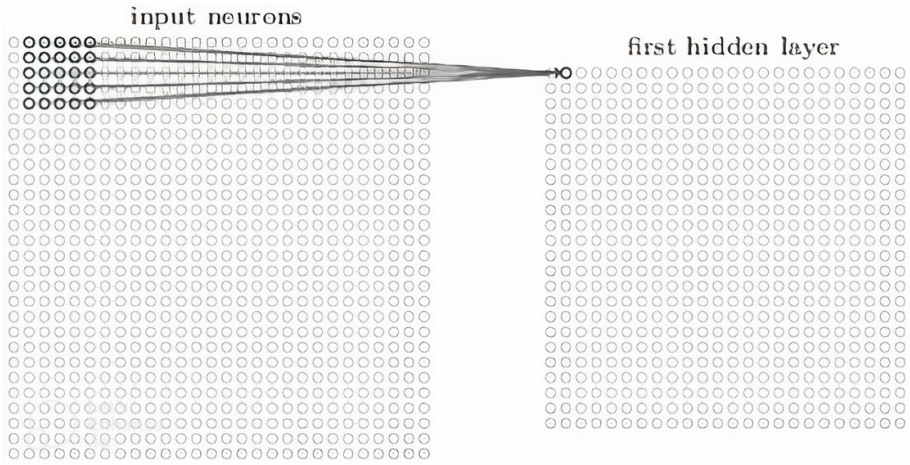
Nakon opisa strukture konvolucijske neuronske mreže, u nastavku slijedi detaljniji osvrt na princip rada.

Na ulaz konvolucijske neuronske mreže dovodi se slika koja je reprezentirana matricom, odnosno tenzorom, a svaki neuron u ulaznom sloju predstavlja vrijednost jednog piksela. Neuroni u skrivenom sloju povezani su na mali broj neurona iz ulaznog sloja, tj. "vide" samo mali dio ulazne slike (receptivno polje) te se kaže da imaju lokalnu povezanost (*engl. local connectivity*).

Na sliku se tada primjenjuje filtar koji "klizi" po slici s određenim korakom (*engl. stride*) te se nakon primjene aktivacijske funkcije generira mapa značajki koja služi kao ulaz u sljedeći sloj mreže. Iznesene tvrdnje prikazane su primjerom na slici 3.2. za filtar dimenzija 5×5 i korak $S = 1$.



(a) Dobivanje elementa mape značajki z_{11}



(b) Dobivanje elementa mape značajki z_{12}

Slika 3.2. Formiranje mape značajki. Izvor:[1]

Zatim slijedi sažimanje s ciljem smanjivanja dimenzija mape značajki.

Konvolucijski slojevi i slojevi sažimanja više se puta izmjenjuju u mreži kako bi se što više informacija izlučilo iz slike. Kako se ide dublje u mrežu, to se izlučuju složeniji uzorci.

Iako se postupkom sažimanja ciljano smanjuju dimenzije mape značajki, do smanjenja dimenzija također dolazi i nakon primjene konvolucije. Na izlazu iz svakog konvolucijskog sloja dolazi do moguće neželjenog smanjivanja dimenzija što može utjecati na gubitak bitnih informacija. Kako bi se to izbjeglo primjenjuje se postupak popunjavanja nulama (*engl. zero padding*). Tim postupkom dodaju se nule na rubove slike čime se smanjuje gubitak dimenzija slike te time i mogući gubitak korisnih informacija [8].

Uvažavajući sve spomenute parametre, dimenzija dobivene mape značajki određena je izrazom 3.6:

$$Z_H \times Z_W = \left(\frac{H - F + 2P}{S} + 1\right) \times \left(\frac{W - F + 2P}{S} + 1\right), \tag{3.6}$$

gdje je Z_H visina dobivene mape značajki, Z_W širina dobivene mape značajki, H visina ulazne slike, W širina ulazne slike, F veličina filtra, S korak konvolucije, a P predstavlja broj dodanih slojeva s nulama.

Ukoliko se želi postići da dimenzija mape značajki odgovara dimenzijama ulaznih podataka tada se parametar P odabire prema izrazu 3.7:

$$P = \frac{F - 1}{2}, \tag{3.7}$$

gdje je F veličina filtra.

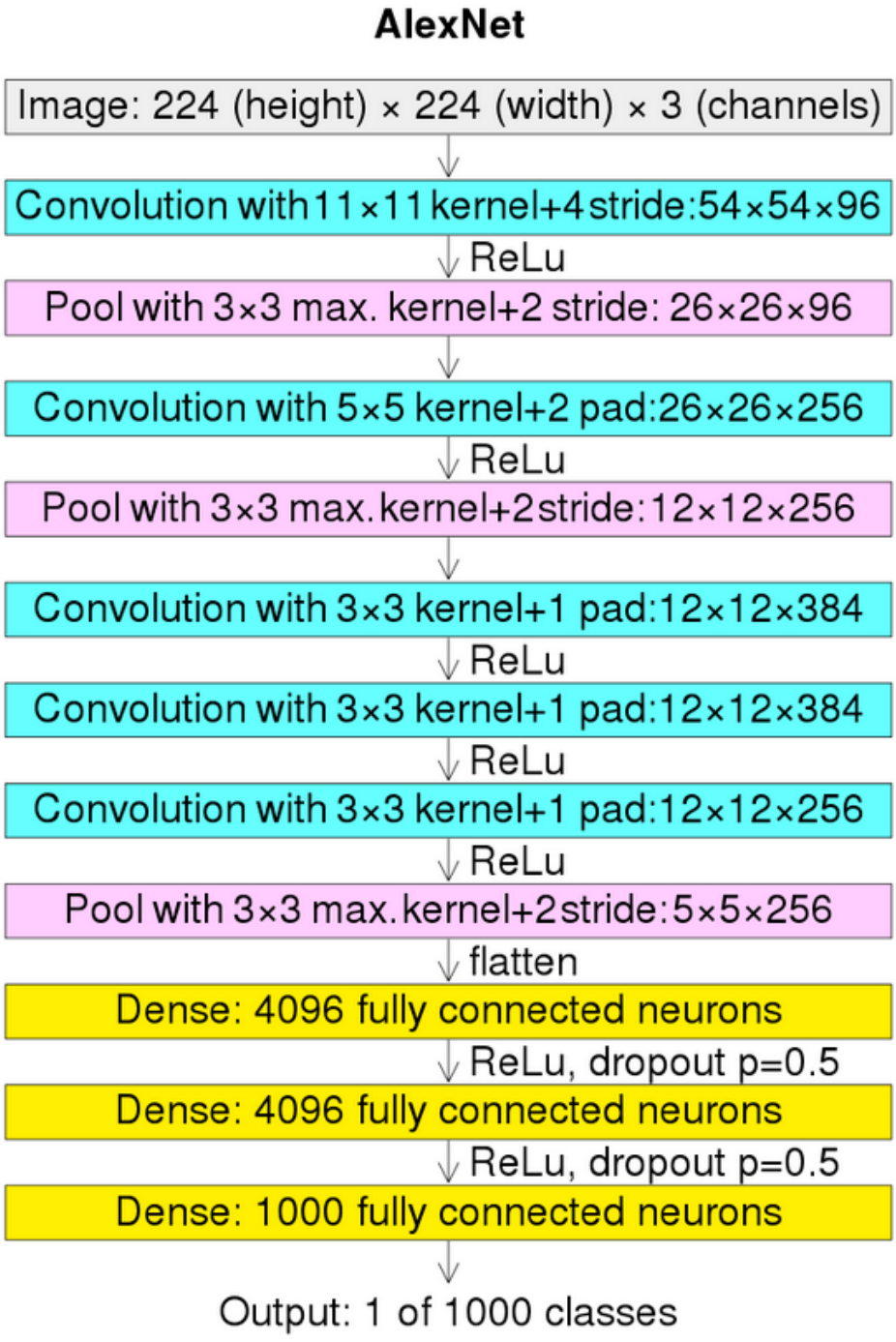
Nakon toga dobivene mape značajki pretvaraju se u vektor te se dovode na ulaz potpuno povezanog sloja gdje se potom vrši klasifikacija objekata. Nad dobivenim vrijednostima klasifikacije uobičajeno je da se primjenjuje *softmax* normalizacijska funkcija, kako bi se dobila distribucija vjerojatnosti predikcija pojedinih klasa, što ujedno predstavlja i izlaz iz konvolucijske neuronske mreže.

Proces treniranja konvolucijske neuronske mreže također se odvija primjenom unazadne propagacije na način kako je objašnjeno u prethodnom poglavlju, čime se nastoji optimizirati mreža pronalaskom odgovarajućih parametara, tj. težinskih faktora i pragova. Valja spomenuti kako se kod konvolucijskih neuronskih mreža težinski faktori pohranjuju u obliku parametara filtra te se učenje zapravo svodi na pronalaženje optimalnih parametara filtra (i pragova) kako bi se detektirale željene značajke sa slike.

3.2.1. Arhitektura AlexNet konvolucijske neuronske mreže

U ovom dijelu rada bit će prikazana arhitektura konvolucijske neuronske mreže na konkretnom primjeru mreže AlexNet. Tekst koji slijedi u nastavku pisan je prema [18].

AlexNet mreža trenirana je kako bi klasificirala 1.2 milijuna slika u 1000 različitih kategorija. Sastoji se od 60 milijuna parametara i 650 tisuća neurona. Sadrži pet konvolucijskih slojeva, tri slojeva sažimanja, tri potpuno povezana sloja te *softmax* sloj s 1000 izlaznih neurona. Arhitektura AlexNet mreže prikazana je na slici 3.3.



Slika 3.3. Arhitektura AlexNet konvolucijske neuronske mreže. Izvor:[19]

Ulazna slika je dimenzija $224 \times 224 \times 3$ te se u prvom konvolucijskom sloju konvoluira s 96 filtara dimenzija 11×11 uz korak $S = 4$. Zatim slijedi sloj sažimanja maksimumom, s filtrom dimenzija 3×3 i korakom $S = 2$. Izlazna mapa značajki dovodi se u sljedeći konvolucijski sloj gdje se primjenjuje 256 filtara dimenzija 5×5 uz popunjavanje nulama veličine 2. Ponovno slijedi sloj sažimanja maksimumom, s filtrom dimenzija 3×3 i korakom $S = 2$. Zatim slijede tri konvolucijska sloja, jedan za drugim, gdje se primjenjuju redom 384, 384 i 256 filtara dimenzija 3×3 te se u svakom konvolucijskom sloju primjenjuje popunjavanje nulama veličine 1. Nakon toga slijedi sažimanje maksimumom s filtrom dimenzija 3×3 i pomakom $S = 2$. Na kraju se rezultat vodi na tri potpuno povezana sloja od kojih svaki sadrži 4096 neurona. Izlaz iz potpuno povezanog sloja vodi se na *softmax* funkciju koja na izlazu daje distribuciju vjerojatnosti za 1000 klasa što je ujedno i izlazni sloj iz AlexNet mreže.

4. Evaluacijske metrike za određivanje uspješnosti modela

Kako bi se moglo procijeniti koliko je model bio uspješan u detekciji objekata potrebno je uvesti evaluacijske metrike, tj. parametre kojima će se vrednovati uspješnost modela.

Za početak potrebno je uvesti određene pojmove koji služe za definiranje evaluacijskih metrika, a to su:

1. TP (*engl. True Positive*)

   Objekt kojeg se detektira nalazi se na slici i postoji detekcija.

2. TN (*engl. True Negative*)

   Objekt kojeg se detektira ne nalazi se na slici i ne postoji detekcija.

3. FP (*engl. False Positive*)

   Objekt kojeg se detektira ne nalazi se na slici i postoji detekcija.

4. FN (*engl. False Negative*)

   Objekt kojeg se detektira nalazi se na slici i ne postoji detekcija.

Nakon prethodno uvedenih pojmova, u nastavku se navode često korišteni parametri za određivanje uspješnosti detekcije.

1. Presjek nad unijom IoU (*engl. Intersection over Union*)

   IoU predstavlja omjer površina $B \cap B_g$ i $B \cup B_g$, gdje B predstavlja predviđeni okvir detektiranog objekta, B_g stvarni okvir detektiranog objekta, $B \cap B_g$ je površina dobivena presjekom predviđenog okvira i stvarnog okvira, a $B \cup B_g$ je ukupna površina koju tvore predviđeni i stvarni okvir. IoU se može zapisati izrazom 4.1:

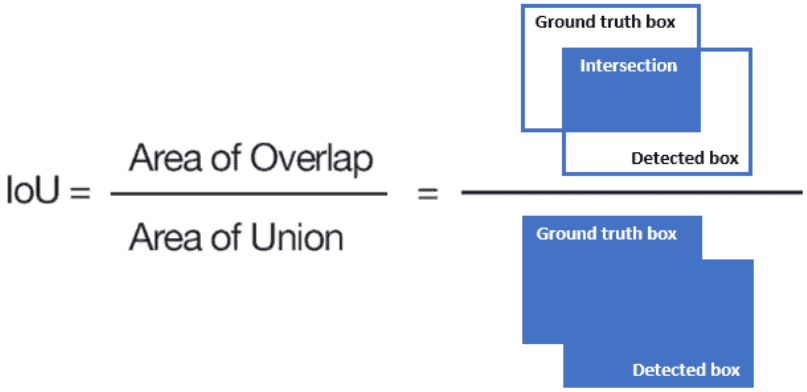
   $$IoU = \frac{B \cap B_g}{B \cup B_g}. \tag{4.1}$$

   Radi lakšeg razumjevanja, IoU je grafički prikazan na slici 4.1.

2. Točnost (*engl. Accuracy*)

   Točnost mjeri udio točnih predikcija u odnosu na ukupan broj predikcija, a definirana je izrazom 4.2:

   $$A = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}. \tag{4.2}$$



Slika 4.1. Grafički prikaz IoU metrike. Izvor:[22]

3. Preciznost (*engl. Precision*)

   Preciznost mjeri udio ispravno detektiranih objekata u odnosu na ukupan broj detektiranih objekata, a definirana je izrazom 4.3:

$$P = \frac{T_P}{T_P + F_P}. \tag{4.3}$$

4. Odziv (*engl. Recall*)

   Odziv mjeri udio ispravno detektiranih objekata u odnosu na ukupan broj prisutnih objekata, a definiran je izrazom 4.4:

$$P = \frac{T_P}{T_P + F_N}. \tag{4.4}$$

5. Konfuzijska matrica (*engl. Confusion matrix*)

   Konfuzijska matrica predstavlja matrični prikaz rezultata predikcije za sve klase koje se nastoje detektirati. Intuitivno prikazuje konfuziju tj., pogrešku modela u slučaju miješanja jedne klase s drugom prilikom klasifikacije.

Još jedan vrlo bitan parametar, koji se često koristi za evaluaciju modela za detekciju objekata, srednja je prosječna preciznost, mAP (*engl. mean Average Precision*). Ovaj parametar se koristi kada postoji više klasa objekata, a u nastavku slijedi objašnjenje kako se izračunava.

Da bi dobili srednju prosječnu preciznost modela potrebno je najprije dobiti informaciju o prosječnoj preciznosti AP (*engl. Average Precision*) posebno za svaku klasu koja se detektira. Prosječnu preciznost dobivamo kao površinu ispod PR krivulje.

PR krivulja prikazuje odnos između preciznosti i odziva. Sustav s visokim odzivom, ali niskom preciznošću daje velik broj detekcija, ali većina je pogrešni. Nasuprot tome, niski odziv uz visoku preciznost daje mali broj detekcija, ali većinom su točne. U praksi se nastoji pronaći optimalni odnos između te dvije mjere ovisno o primjeni modela za detekciju objekata [23].

Kada su dobivene informacije o prosječnim preciznostima za svaku klasu, potrebno je izračunati srednju vrijednost svih prosječnih preciznosti te se time dobiva srednja prosječna preciznost modela preko svih klasa objekata, a definirana je izrazom 4.5:

$$mAP = \frac{1}{C} \sum_{i=1}^{N} AP_i. \tag{4.5}$$

5. Prepoznavanje gesta za upravljanje dronom korištenjem YOLO algoritma

U ovom poglavlju bit će prikazana praktična primjena konvolucijskih neuronskih mreža za detekciju objekata. Cilj je ove primjene prepoznavanje gesta tijela kako bi se omogućilo upravljanje dronom bez upravljača. Algoritam kojim će se vršiti detekcija objekata zove se YOLO algoritam te u nastavku slijedi više govora o istome.

5.1. YOLO algoritam

YOLO, što stoji za "You Only Look Once", algoritam trenutno je jedan od najpopularnijih pristupa detekciji objekata. Glavnina razloga leži u tome što je ovaj algoritam brz i to ga čini pogodnim za primjenu u stvarnom vremenu. Detekcija objekata u stvarnom vremenu omogućena je svojstvom da se objekti detektiraju u samo jednom prolazu podataka, slika ili videozapisa kroz neuronsku mrežu za razliku od nekih drugih metoda koje zahtijevaju višestruke prolaze.

YOLO algoritam razvijen je 2015. godine na sveučilištu u Washingtonu od strane Josepha Redmona, Santosha Divvale, Rossa Girshicka i Alija Farhadija [8]. Algoritam je kroz niz godina razvijen u nekoliko verzija i njihovih nadogradnji od kojih je najnovija YOLOv9. Svaka sljedeća verzija predstavlja nadogradnju na onu prethodnu uz poboljšana svojstva kao što su povećana točnost, veća brzina obrade podataka, bolja detekcija malih objekata [20].

Kod detekcije objekata razlikuju se dvije problematike, a to su klasifikacija i lokalizacija. Klasifikacija objekata proces je prepoznavanja i dodjeljivanja klase objektima na slici ili videozapisu, a lokalizacija je proces dodjeljivanja koordinata objektima, u ovom slučaju ocrtavanje okvira objektu kojeg se detektira.

Rezultat detekcije, odnosno izlaz iz neuronske mreže y je tada definiran vektorom kako je dano izrazom 5.1:

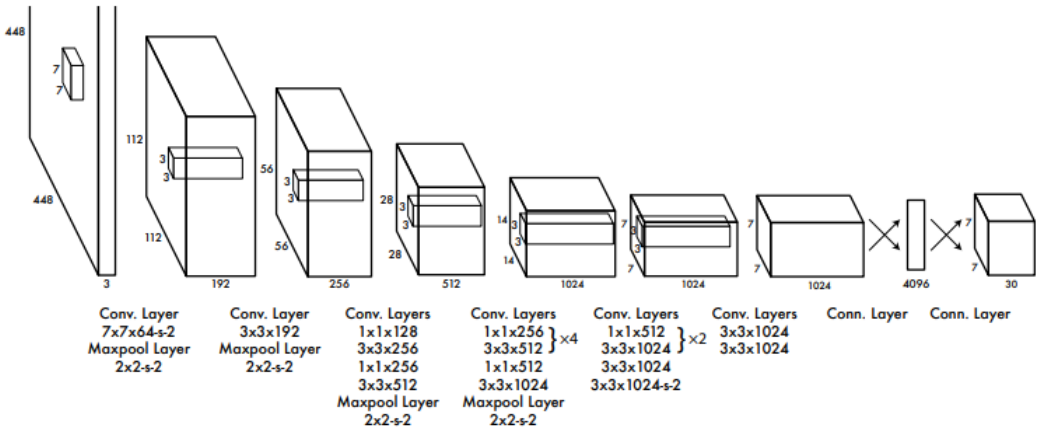
$$y = [P_c, B_x, B_y, B_w, B_h, C_1, C_2, ..., C_n]^T, \tag{5.1}$$

gdje je P_c vrijednost 0 ili 1 što ukazuje na to je li objekt prisutan na slici ili nije, B_x i B_y su koordinate središta objekta, B_w je širina okvira, B_h je visina okvira, a $C_1, C_2, ..., C_n$ su vrijednosti 0 ili 1 koje ukazuju na prisutnost pojedine klase na slici.

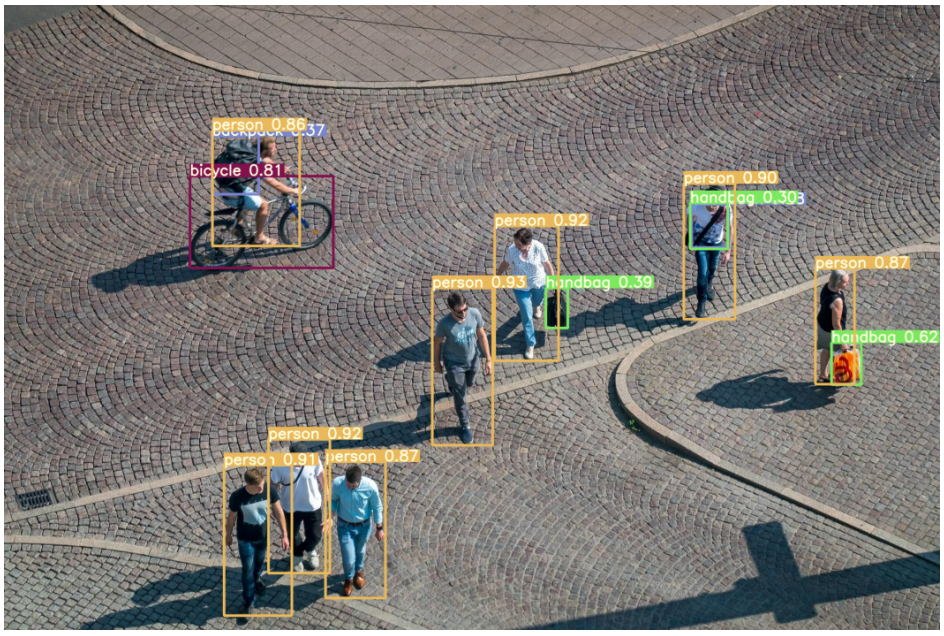
Treba napomenuti da prethodno iznesena tvrdnja vrijedi ako se na slici nalazi samo jedan objekt te samim time postoji samo jedan okvir. Međutim, ukoliko se na slici nalazi više objekata tada su potrebne informacije o prisutnosti i lokaciji svakog objekta kojeg se detektira posebno te izraz 5.1 više nije pogodan. YOLO algoritam taj problem rješava tako da ulaznu sliku podijeli na mrežu dimenzija $S \times S$. Tada je slika sačinjena od više ćelija te ukoliko se središte objekta nalazi unutar

neke ćelije, tada je ta ćelija odgovorna za njegovu detekciju [21]. U tom slučaju za svaku se ćeliju formira vektor sukladno izrazu 5.1 te je konačni rezultat tenzor dimenzija $S \times S \times (B \cdot 5 + C)$, gdje je B broj okvira koje pojedina ćelija može generirati, C broj klasa koje je moguće detektirati, a broj 5 stoji za broj parametara koji su potrebni za lokalizaciju objekta (B_x, B_y, B_w, B_h te mjera pouzdanosti detekcije) [8], [21]. Parametri B_x i B_y izraženi su u odnosu na rubove ćelije, B_w i B_h izraženi su u odnosu na cijelu sliku, a mjera pouzdanosti (*engl. confidence score*) predstavlja mjeru koja govori koliko je model siguran u svoju predikciju.

Na slici 5.1. prikazana je arhitektura YOLO algoritma, a na slici 5.2. je prikazan primjer kako detekcija YOLO algoritmom izgleda u praksi.



Slika 5.1. Arhitektura YOLO algoritma. Izvor:[21]



Slika 5.2. Primjer detekcije objakta primjenom YOLO algoritma. Izvor:[24]

5.1.1. Funkcija gubitka YOLO algoritma

U ovom dijelu bit će opisana funkcija gubitka YOLO algoritma prema [8], [21], [25].

Funkcija gubitka YOLO algoritma kvadratna je funkcija gubitka i sastoji se od tri dijela koji predstavljaju grešku klasifikacije (*engl. calssification loss*), grešku lokalizacije (*engl. localization loss*) i grešku pouzdanosti (*engl. confidence loss*).

1. Greška klasifikacije L_{class}

   Ako je objekt detektiran, greška klasifikacije u svakoj ćeliji računa se prema izrazu 5.2:

   $$L_{class} = \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2, \tag{5.2}$$

   gdje 1_i^{obj} iznosi 1 ukoliko se objekt pojavi u i-toj ćeliji, u suprotnom iznosi 0.

2. Greška lokalizacije L_{loc}

   Greška lokalizacije mjeri grešku u lokaciji i dimenzijama predviđenog okvira u odnosu na stvarni okvir i računa se prema izrazu 5.3:

   $$\begin{aligned} L_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right], \end{aligned} \tag{5.3}$$

   gdje 1_{ij}^{obj} iznosi 1 ukoliko je j-ti okvir u i-toj ćeliji odgovoran za detekciju objekta, u suprotnom iznosi 0, a λ_{coord} predstavlja parametar kojim se regulira značajnost greške za okvir koji sadrži detektirani objekt.

3. Greška pouzdanosti L_{conf}

   Greška pouzdanosti računa se prema izrazu 5.4:

   $$L_{conf} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2, \tag{5.4}$$

   gdje $\hat{C}_i$ predstavlja mjeru pouzdanosti j-tog okvira u i-toj ćeliji, λ_{noobj} predstavlja parametar kojim se regulira značajnost greške za okvir koji ne sadrži detektirani objekt, a 1_{ij}^{noobj} predstavlja komplement od 1_{ij}^{obj}.

   Drugim rječima, prvi dio izraza 5.4 predstavlja grešku pouzdanosti ako je objekt detektiran unutar okvira, a drugi dio predstavlja grešku ako objekt nije detektiran unutar okvira.

Ukupna funkcija gubitka dobiva se zbrajanjem prethodno navedenih dijelova kako je dano izrazom 5.5:

$$L = L_{class} + L_{loc} + L_{conf}. \tag{5.5}$$

Ukoliko se izrazi 5.2, 5.3 i 5.4 uvrste u izraz 5.5 dobiva se konačni izraz koji predstavlja ukupnu funkciju gubitka YOLO algoritma te je ista dana izrazom 5.6:

$$\begin{aligned} L = & \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \\ & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2. \end{aligned} \tag{5.6}$$

5.2. Skup podataka

Skup podataka za učenje prepoznavanja gesta tijela preuzet je s internetske stranice Roboflow [24]. Skup se sastoji od slika koje prikazuju različite naredbe koje uvijek isti subjekt daje gestikulacijom ruku i tijela i njima popratnih anotacija. Gestama pripadajuće naredbe su: poleti (*engl. take-off*), sleti (*engl. land*) i slijedi (*engl. follow*). Svaku od njih čovjek može dati položajem svoga tijela i/ili rukom što ukupno čini 6 razreda naredbi (*take-off, land, follow, take-off-hand, land-hand, follow-hand*), a prikazane su na slikama 5.3., 5.4., 5.5., 5.6., 5.7. i 5.8. redom.

Po preuzimanju, slike su podijeljene u skup za učenje (*engl. train set*), skup za validaciju (*engl. validation set*) i skup za ispitivanje (*engl. test set*) u omjeru 70:20:10. Skup za učenje sadrži 17830 slika, skup za validaciju sadrži 5094 slika, a skup za ispitivanje sadrži 2549 slika.

Kako bi se model dotrenirao, stvoren je dodatni, manji skup podataka koji se djelomično sastojao od nasumično odabranih slika iz prvobitnog skupa te od osobnih slika koje su prikupljene za ovu primjenu, a dobivene su ekstrakcijom slika iz snimljenog videozapisa.

Ekstrakcijom je dobiveno ukupno 625 osobnih slika koje su podijeljene u omjeru 85:15 čime je dobiveno 531 slika za treniranje i 94 slike za validaciju. Skup za ispitivanje nije posebno izrađen zato što se skup za validaciju kasnije koristio i za ispitivanje. Na ovaj je način osiguran veći broj slika za treniranje modela kao i prikladan broj slika za validaciju.

Nakon pridruživanja osobnih slika nasumično odabranim slikama iz prvobitnog skupa dobiven je skup od 1325 slika od kojih je 1126 korišteno za treniranje, a 199 za validaciju, odnosno ispitivanje.

Na slici 5.3. prikazana je naredba "poleti" dana položajem tijela. U toj naredbi subjekt, koji izdaje naredbu, podiže obje ruke u zrak pod kutom.

Slika 5.3. Naredba "poleti" dana položajem tijela

Na slici 5.4. prikazana je naredba "sleti" dana položajem tijela. U toj naredbi subjekt, koji izdaje naredbu, spušta ruke prema dolje pod kutom.

Slika 5.4. Naredba "sleti" dana položajem tijela

Na slici 5.5. prikazana je naredba "slijedi" dana položajem tijela. U toj naredbi subjekt, koji izdaje naredbu, spaja ruke s tijelom.

Slika 5.5. Naredba "slijedi" dana položajem tijela

Na slici 5.6. prikazana je naredba "poleti" dana dlanom. U toj naredbi subjekt, koji izdaje

naredbu, stišće šaku, no mali prst i kažiprst su usmjereni prema gore.

Slika 5.6. Naredba "poleti" dana dlanom

Na slici 5.7. prikazana je naredba "sleti" dana dlanom. U toj naredbi subjekt, koji izdaje naredbu, stišće šaku, no palac i kažiprst su odvojeni i podignuti u obliku slova "L".

Slika 5.7. Naredba "sleti" dana dlanom

Na slici 5.8. prikazana je naredba "poleti" dana dlanom. U toj naredbi subjekt, koji izdaje naredbu, otvara šaku te svi prsti su odvojeni.

Slika 5.8. Naredba "slijedi" dana dlanom

Ekstrakcija slika iz videozapisa izvršena je pomoću Python koda, koji je dan kodom 5..1. u nastavku, a pisan je prema [27].

Kod 5..1. Python kod za ekstrakciju slika iz videozapisa

```
import cv2
import os

cam = cv2.VideoCapture("./video.mp4")

try:
if not os.path.exists('data'):
os.makedirs('data')

except OSError:
print('Error: Creating directory of data')

intvl = 0.2 #interval in second(s)

fps= int(cam.get(cv2.CAP_PROP_FPS))
print("fps : " ,fps)

currentframe = 0
while (True):
ret, frame = cam.read()
if ret:
if(currentframe % (fps*intvl) == 0):
name = './data/screenshot' + str(currentframe) + '.jpg'
print('Creating...' + name)
cv2.imwrite(name, frame)
currentframe += 1
else:
break

cam.release()
cv2.destroyAllWindows()
```

Nasumičan odabir slika i njihovih popratnih anotacija iz prvobitnog skupa podataka izvršen je pomoću Python koda koji je dan kodom 5..2. u nastavku.

Kod 5..2. Python kod za nasumičan odabir slika i njihovih popratnih anotacija

```
import os
import random
import shutil

# Paths to the original dataset
images_path = './images'
annotations_path = './labels'

# Paths to the new subset dataset
subset_images_path = './images_new'
subset_annotations_path = './labels_new'

# Number of images to select
num_images_to_select = 700

# Get list of all image files
all_images = os.listdir(images_path)

# Randomly select a subset of images
selected_images = random.sample(all_images,
    num_images_to_select)

# Copy selected images and their annotations to the new
    subset directory
for image in selected_images:
# Copy the image file
shutil.copy(os.path.join(images_path, image), os.path.
    join(subset_images_path, image))

# Copy the corresponding annotation file
annotation_file = os.path.splitext(image)[0] + '.txt'
shutil.copy(os.path.join(annotations_path,
    annotation_file), os.path.join(subset_annotations_path
    , annotation_file))
```

5.3. Trening YOLOv9 modela

U ovom dijelu rada bit će opisan postupak treniranja YOLO algoritma s ciljem prepozavanja gesta tijela u svrhu upravljanja dronom. Za detekciju je izabran YOLOv9 algoritam.

Trening modela izvršen je pomoću razvojnog okruženja Google Colaboratory. Google Colaboratory je platorma koja korisnicima omogućava da pišu i izvršavaju Python kod direktno u pregledniku. Također, jedna je od njezinih vrlo korisnih mogućnosti besplatno korištenje računalnih resursa kao što je GPU (*engl. graphics processing unit*) koji je nužan za treniranje modela za detekciju objekata zbog svoje optimiziranosti za rad s grafičkim podacima kao što su slike i videozapisi. Isto tako, Google Colaboratory pruža mogućnost spajanja osobnog Google diska što omogućuje lako pohranjivanje i učitavanje datoteka.

Prije samog treniranja modela, izvršeno je spajanje osobnog Google diska, na kojem se nalaze potrebne datoteke, koje će se kasnije koristiti za treniranje modela, te je preuzet javno dostupni YOLOv9 kod s platforme GitHub [28]. Model, koji je korišten, prethodno je treniran na COCO (*engl. Common Objects in Context*) skupu podataka.

Proces treniranja započet je na prvobitno preuzetom skupu podataka s parametrima danima u tablici 5.1.

Tablica 5.1. Parametri za treniranje na prvobitnom skupu podataka

Batch size	Image size	Epochs	Weights
16	640	33	yolov9-e.pt

Nakon završenog prvog dijela, treniranje je nastavljeno na dodatno stvorenom skupu podataka s parametrima danima u tablici 5.2.

Tablica 5.2. Parametri za treniranje na dodatno stvorenom skupu podataka

Batch size	Image size	Epochs	Weights
16	640	30	best.pt

Težinski faktori *best.pt* predstavljaju najbolje težinske faktore iz prvog dijela treninga.

U sljedećem podpoglavlju biti će izneseni postignuti rezultati.

5.4. Rezultati

Nakon završenog treniranja postignuta je srednja prosječna preciznost mAP@0.5 od 98.1% što ukazuje na visoku uspješnost modela. Zatim je model ispitan na skupu za ispitivanje, a primjeri detekcije prikazani su na slikama 5.9., 5.10., 5.11., 5.12., 5.13. i 5.14. u nastavku.

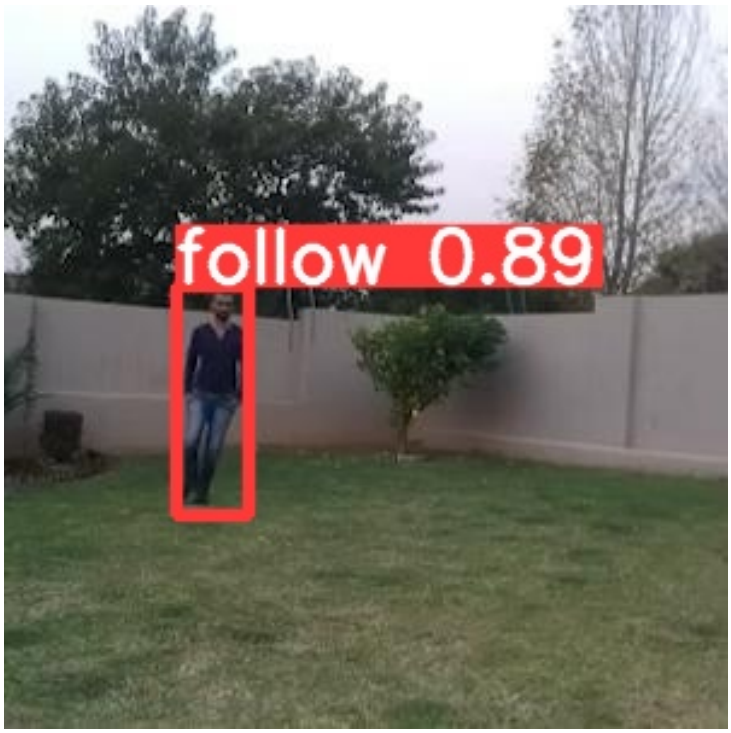
Na slici 5.9. prikazan je primjer uspješne detekcije naredbe "poleti" dane tijelom. Mjera pouzdanosti u ovom slučaju iznosi 0.91, odnosno 91%.

Slika 5.9. Detektirana naredba "poleti" dana položajem tijela

Na slici 5.10. prikazan je primjer uspješne detekcije naredbe "sleti" dane tijelom. Mjera pouzdanosti u ovom slučaju iznosi 0.93, odnosno 93%.

Slika 5.10. Detektirana naredba "sleti" dana položajem tijela

Na slici 5.11. prikazan je primjer uspješne detekcije naredbe "slijedi" dane tijelom. Mjera pouzdanosti u ovom slučaju iznosi 0.89, odnosno 89%.



Slika 5.11. Detektirana naredba "slijedi" dana položajem tijela

Na slici 5.12. prikazan je primjer uspješne detekcije naredbe "poleti" dane dlanom. Mjera pouzdanosti u ovom slučaju nije vidljiva zbog položaja detektirane geste na slici i ograničavajućih dimenzija iste, no vidljivo je da se radi o uspješnoj detekciji.

Slika 5.12. Detektirana naredba "poleti" dana dlanom

Na slici 5.13. prikazan je primjer uspješne detekcije naredbe "sleti" dane dlanom. Mjera

pouzdanosti u ovom slučaju iznosi 0.86, odnosno 86%.

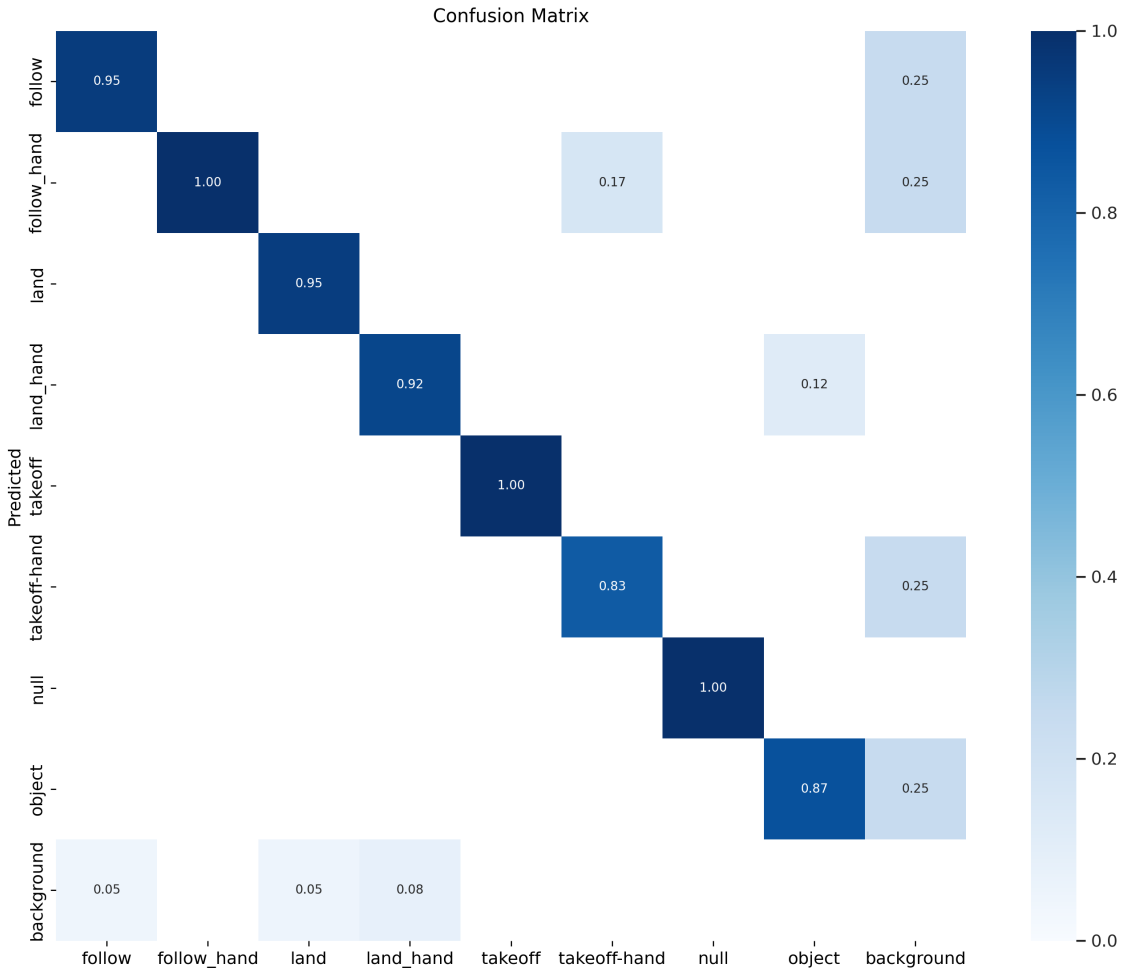
Slika 5.13. Detektirana naredba "sleti" dana dlanom

Na slici 5.14. prikazan je primjer uspješne detekcije naredbe "slijedi" dane dlanom. Mjera pouzdanosti u ovom slučaju nije vidljiva zbog položaja detektirane geste na slici i ograničavajućih dimenzija iste, no vidljivo je da se radi o uspješnoj detekciji.

Slika 5.14. Detektirana naredba "slijedi" dana dlanom

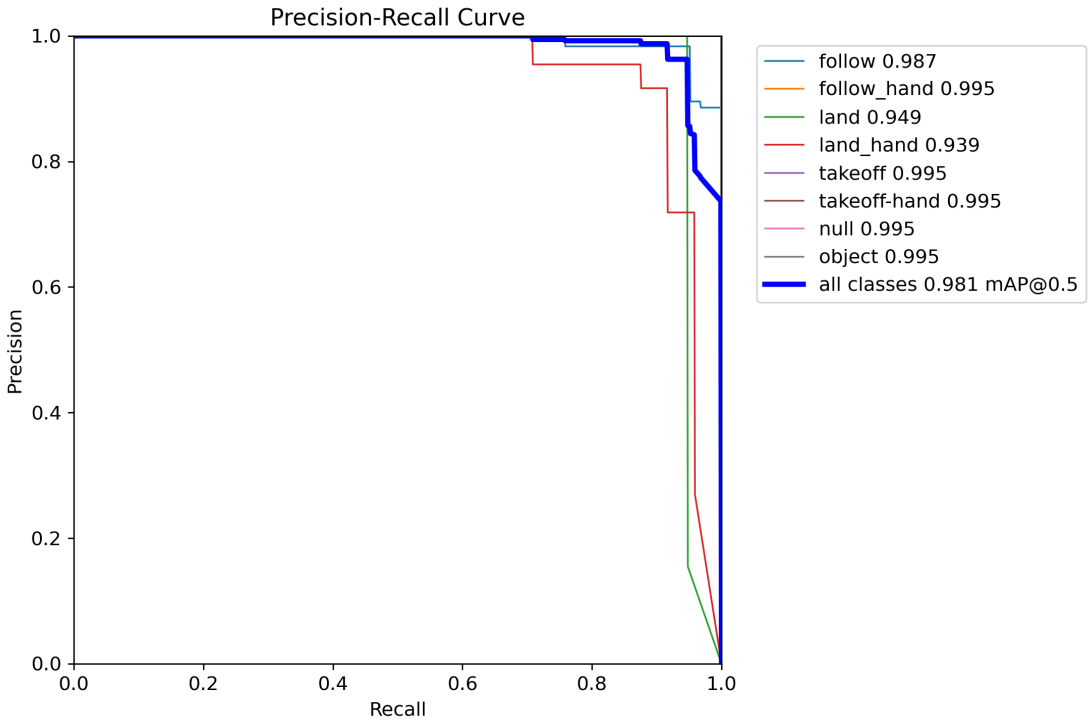
Uz dane primjere detekcije, u nastavku su dani i grafički prikazi uspješnosti modela. Za grafičke pokazatelje izabrani su sljedeći parametri: konfuzijska matrica, PR krivulja, greška klasifikacije i greška lokalizacije. Navedeni pokazatelji su dani na slikama 5.15., 5.16., 5.17. i 5.18. redom.

Iz konfuzijske matrice sa slike 5.15. vidljivo je da su rezultati na dijagonali dominantni, što ukazuje da ne dolazi do značajnog miješanja između klasa.



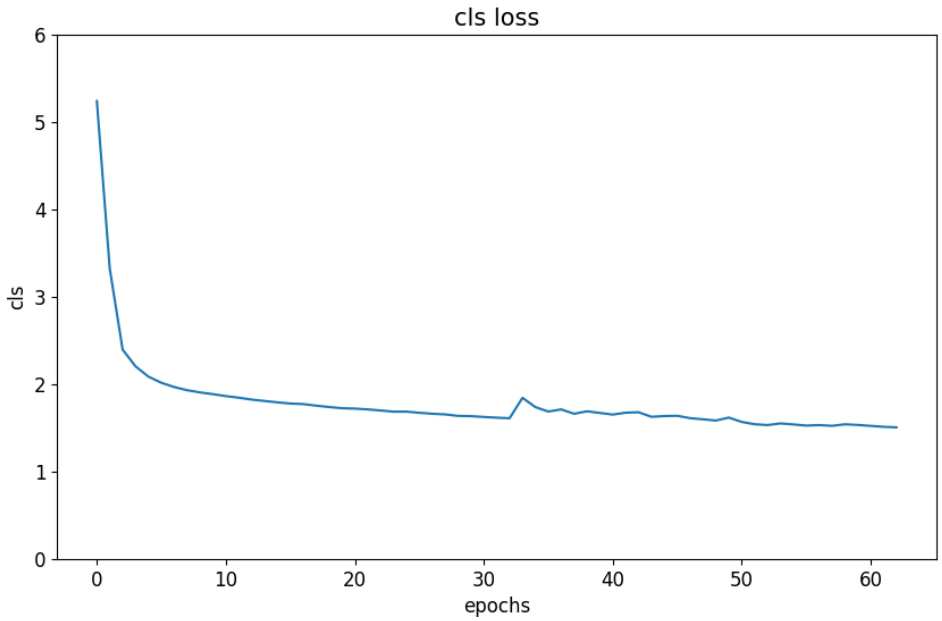
Slika 5.15. Konfuzijska matrica

PR krivulja, prikazana na slici 5.16., ukazuje na dobre performanse modela. Navedeno se može zaključiti po velikoj površini koja se nalazi ispod krivulje.



Slika 5.16. PR krivulja

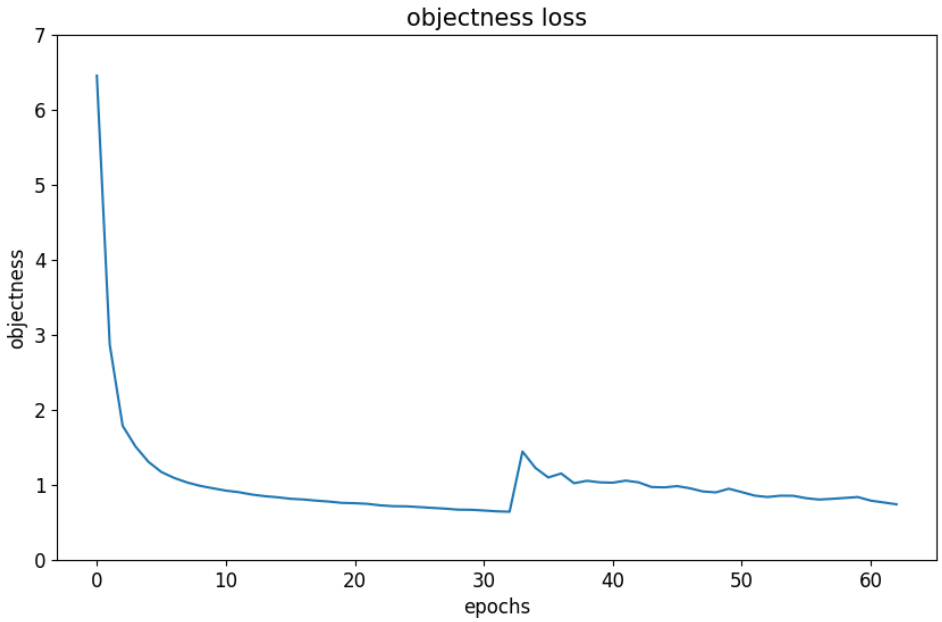
Slika 5.17. prikazuje grešku klasifikacije. S grafa je vidljivo da je došlo do znatnog smanjenja greške te se ista ustalila, što znači da daljnjim treniranjem, s istim skupom podataka, ne bi previše utjecalo na performanse modela.



Slika 5.17. Greška klasifikacije

Greška lokalizacije, prikazana na slici 5.18., ponaša se isto kao i greška klasifikacije što znači

da prethodno iznesena tvrdnja vrijedi i u ovom slučaju.



Slika 5.18. Greška lokalizacije

Sa slika 5.17. i 5.18., može se uočiti "ne-monotonost" krivulja, odnosno mali, ali nagli porast greške oko 30. epohe koji je prisutan zbog već spomenute promjene skupa podataka za treniranje.

Iako su rezultati zadovoljavajući, potencijalni nedostatak ovog sustava je u primjenjivost modela i na druge subjekte. Drugim rječima, model je naučen prepoznati naredbe dane od strane subjekta na kojemu je treniran, a ima poteškoća s prepoznavanjem naredbi koje daju drugi subjekti. Moguće rješenje ovog problema bilo bi u dotreniravanju modela pomoću slika drugih subjekata.

5.5. Detekcija u stvarnom vremenu

Na samom je kraju model za detekciju gesta za upravljanje dronom ispitan na detekciji u stvarnom vremenu (*engl. real time detection*) te će u ovom dijelu rada ona biti ukratko opisana zajedno s postignutim rezultatima.

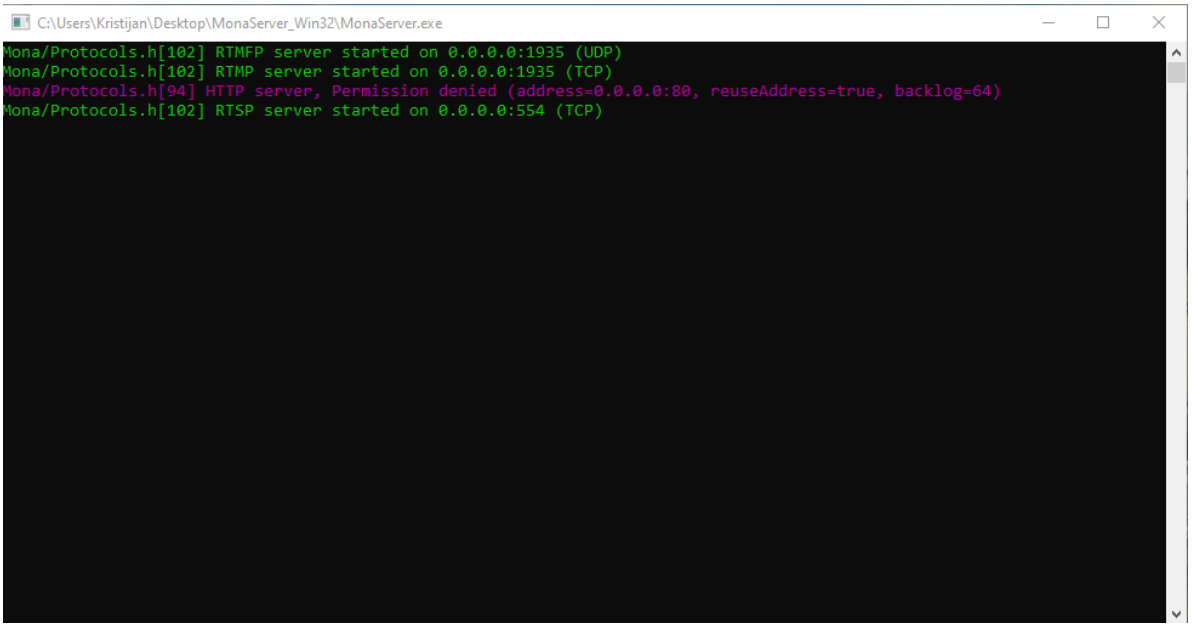
Za prijenos podataka u stvarnom vremenu korišten je RTMP (*engl. Real Time Messaging Protocol*). RTMP je komunikacijski protokol pomoću kojega je moguće uživo prenositi podatke putem interneta [29].

Za primjer RTMP servera uzet je MonaServer te u nastavku slijedi kratki opis procesa postavljanja RTMP servera prema [30], [31], [32].

Za uspješan prijenos potrebno je u direktoriju MonaServer-a stvoriti strukturu direktorija kako je dano u nastavku.

```
./MonaServer_Win32/www/live/test
```

Ukoliko gore navedena struktura postoji, potrebno je pokrenuti MonaServer te ga ostaviti upaljenog za vrijeme prijenosa. Na slici 5.19. prikazano je sučelje uspješno postavljenog MonaServera.



Slika 5.19. Sučelje MonaServer-a

Da bi se vršio prijenos uživo potrebno je koristitu neku od aplikacija koja podržava RTMP, u njoj odabrati RTMP opciju te proslijediti URL (*engl. Uniform Resource Locator*) koji je dan u nastavku.

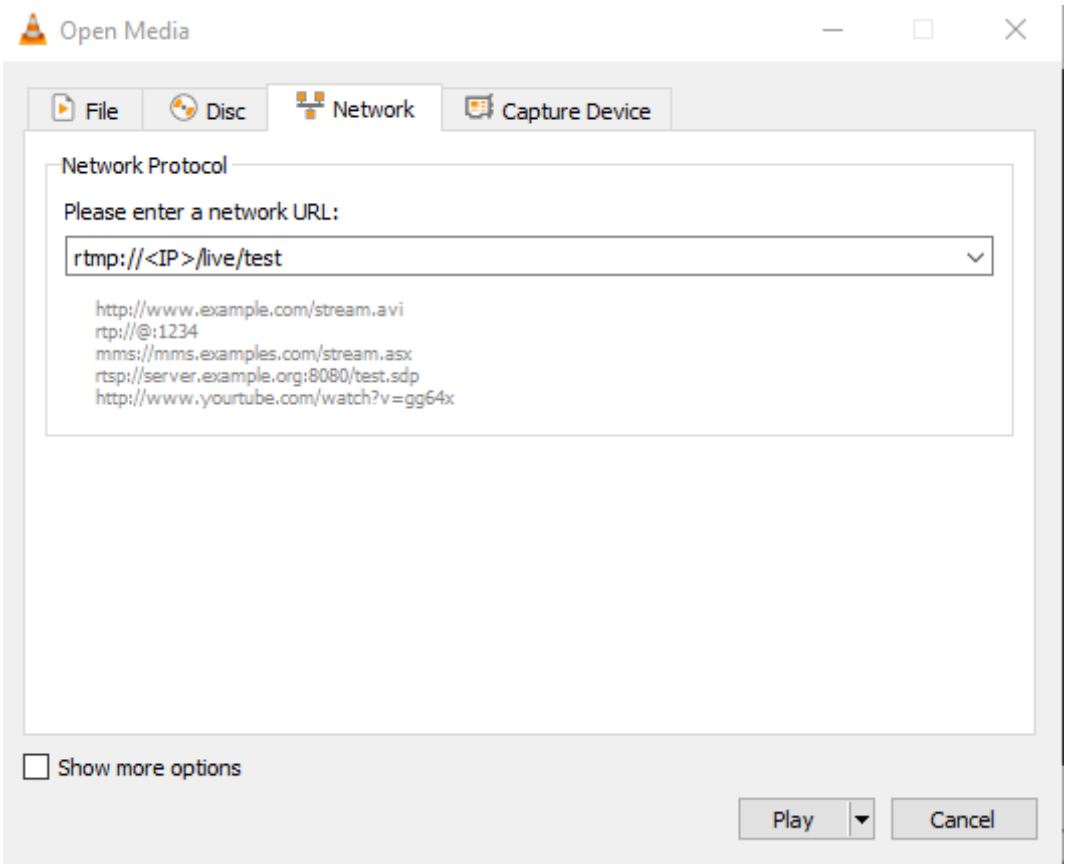
`rtmp://<IP>:1935/live/test`

IP u prethodno danom URL-u označava lokalnu IP adresu računala.

Za provjeru prijenosa najjednostavnije je koristiti VLC media player koji služi za reproduciranje medijskih datoteka.

U VLC-u je potrebno odabrati opciju mrežni prijenos (*engl. network stream*) te na odgovarajuće mjesto proslijediti identični URL kako je prethodno dan.

Na slici 5.20. dan je prikaz sučelja VLC-a za postavljenje RTMP opcije.



Slika 5.20. Sučelje VLC-a za postavljenje RTMP opcije

Nadalje, kako bi se uspostavila detekcija korišten je Windows Powershell.

Prvo što je potrebno je preuzeti YOLOv9 kod na svoje računalo. Zatim se u Windows Powershell-u pristupa YOLOv9 direktoriju te se instaliraju potrebne knjižnice pomoću naredbi koje su dane u nastavku.

```
cd ./yolov9/yolov9-main
```

```
python pip install -r requirements.txt
```

Kako bi se koristio prethodno istrenirani model, potrebno je da se datoteka koja sadrži težinske faktore nalazi u YOLOv9 direktoriju zajedno s detect.py datotekom.

Ukoliko je sve prethodno navedeno odrađeno, detekcija se može pokrenuti koristeći naredbu danu u nastavku.

```
python detect.py --weights ./best.pt --source rtmp://<IP>:1935/
live/test
```

Primjer detekcije u stvarnom vremenu dan je na slici 5.21. u nastavku.

Slika 5.21. Naredba "sleti" dana dlanom detektirana u stvarnom vremenu

Na slici 5.21. prikazana je uspješna detekcija naredbe "sleti" dana dlanom, u stvarnom vremenu, uz mjeru pouzdanosti 0.81, odnosno 81%.

6. Zaključak

Umjetna inteligencija, kao neizostavni segment svakodnevnog života modernog doba, predstavljena je u ovom diplomskom radu u obliku računalnog vida s ciljem detekcije objekata. Cilj rada bio je stvoriti sustav za prepoznavanje gesta tijela kako bi se omogućilo upravljanje dronom bez daljinskog upravljača.

Naredbe za upravljanje dronom koje su korištene su "poleti", "sleti" i "slijedi". Svaku od tih naredbi moguće je prikazati na dva načina, položajem ruku i tijela ili dlanom i prstima, što ukupno čini 6 različitih klasa koje je potrebno prepoznati.

Trening i ispitivanje modela implementirano je u programskom jeziku Python pomoću programskog okruženja Google Colaboratory, a za realizaciju navedenog cilja korišten je YOLO algoritam.

YOLO algoritam trenutno je jedan od najmodernijih i najbržih algoritama za detekciju objekata. Zbog svojih karakteristika pogodan je za detekciju u stvarnom vremenu što ga čini primjenjivim za ovu svrhu. U ovom je radu korišten YOLOv9 algoritam.

Kako bi se postigla veća robusnost sustava, model je treniran s dva skupa podataka gdje je jedan skup podataka preuzet s internetske stranice Roboflow, a drugi skup podataka je samostalno stvoren te je sadržavao manji dio prethodno preuzetog skupa i osobne fotografije s popratnim anotacijama.

Kroz razvoj i evaluaciju sustava za prepoznavanje gesta tijela, uspješno je pokazano da je moguće detektirati geste tijela s visokom točnošću u različitim uvjetima što dodatno potvrđuje praktičnost i robusnost ovog modela. Postignuta je srednja prosječna preciznost mAP od 98.1% što se smatra vrlo uspješnim.

Rezultati ovog rada pružaju podlogu za daljnji razvoj i istraživanje u ovom području, čime se doprinosi napretku tehnologije i njenoj integraciji u svakodnevni život omogućavajući primjenu inovativnih rješenja koja mogu unaprijediti mnoge aspekte ljudskih aktivnosti.

Osim za upravljanje dronom, geste tijela mogu služiti i za komunikaciju s osobom ili računalom, koje upravlja dronom, na način da svaka gesta tijela predstavljaja određenu informaciju. Situacija, u kojoj bi to bilo korisno je primjerice spasilačka misija za pronalazak osoba u nevolji. Još je jedan primjer davanje izvještaja o stanjima objekata kada se govori o inspekcijskim misijama, gdje je moguće gestama tijela putem drona poslati informaciju o postojećim oštećenjima ili potrebi za intervencijom.

Zaključno, razvoj umjetne inteligencije otvara mnoge nove mogućnosti u raznim sferama života pa tako i implementacija sustava za upravljanje dronom putem prepoznavanja gesta tijela predstavlja inovativan i praktičan pristup koji može značajno unaprijediti korisničko iskustvo i

otvoriti nove mogućnosti za primjenu dronova u različitim situacijama. Na ovaj način eliminira se potreba za tradicionalnim metodama upravljanja kao što su daljinski upravljači i sučelja na dodir što dodaje dodatan stupanj slobode i novu dimenziju sustavu upravljanja općenito.

Literatura

[1] Nielsen, M. A.: "Neural Networks and Deep Learning", Determination Press, 2015.

[2] "Artificial neural network", s Interneta, https://en.wikipedia.org/wiki/Artificial_neural_network, pristupljeno 13.12.2023.

[3] "History of artificial neural network", s Interneta, https://www.javatpoint.com/history-of-artificial-neural-network, pristupljeno 13.12.2023.

[4] "History of ann", s Interneta, https://libguides.aurora.edu/ChatGPT/History-of-AI-and-Neural-Networks, pristupljeno 13.12.2023.

[5] "Neuron", s Interneta, https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron, pristupljeno 14.12.2023.

[6] "Neuron anatomy", s Interneta, https://biologydictionary.net/nerve-cell/, pristupljeno 14.12.2023.

[7] Dalbelo Bašić, B.; Čupić, M.; Šnajder, J.: "Umjetne neuronske mreže", repozitorij Fakulteta elektrotehnike i računarstva, Zagreb, 2008.

[8] Cvija, T.: "Detekcija karcinoma mokraćnog mjehura korištenjem YOLO algoritma", repozitorij Tehničkog fakulteta, Rijeka, 2022.

[9] "Sigmoid function in neural network", s Interneta, https://www.analyticsvidhya.com/blog/2023/01/why-is-sigmoid-function-important-in-artificial-neural-networks/, pristupljeno 18.12.2023.

[10] "Sigmoid vs tanh activation function", s Interneta https://medium.com/@sthacruz/relationship-between-sigmoid-and-tanh-activation-function-53d289889d9a, pristupljeno 18.12.2023.

[11] "Activation functions in deep learning", s Interneta, https://www.theaidream.com/post/an-overview-of-activation-functions-in-deep-learning, pristupljeno 18.12.2023.

[12] "Deep neural network", s Interneta, https://www.researchgate.net/figure/Deep-Neural-Network-architecture_fig1_330120030, pristupljeno 21.12.2023.

[13] "Cross entropy cost function", s Interneta, https://www.aporia.com/learn/understanding-binary-cross-entropy-and-log-loss-for-effective-model-monitoring/, pristupljeno 27.12.2023.

[14] "Computer vision", s Interneta, https://en.wikipedia.org/wiki/Computer_vision, pristupljeno 16.01.2024.

[15] "Edge detection filter", s Interneta, https://sbme-tutorials.github.io/2018/cv/notes/4_week4.html, pristupljeno 17.01.2024.

[16] "Convolutional neural network", s Interneta, https://en.wikipedia.org/wiki/Convolutional_neural_networ pristupljeno 29.01.2024.

[17] "Structure of CNN", s Interneta, https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5, pristupljeno 31.01.2024.

[18] Krizhevsky, A., Sutskever, I., Hinton, G. E.: "Imagenet classification with deep convolutional neural networks.", University of Toronto, 2012.

[19] "AlexNet", s Interneta, https://en.wikipedia.org/wiki/AlexNet, pristupljeno 08.02.2024.

[20] "YOLO algorithm", s Interneta, https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z, pristupljeno 25.03.2024.

[21] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: "You Only Look Once: Unified, Real-Time Object Detection", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, 2016.

[22] "Intersection over union", s Interneta, https://www.baeldung.com/cs/object-detection-intersection-vs-union, pristupljeno 31.03.2024.

[23] "PR curve", s Interneta, https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_re pristupljeno 02.04.2024.

[24] "YOLO algorithm detection example", s Interneta, https://viso.ai/deep-learning/yolov7-guide/, pristupljeno 04.04.2024.

[25] "YOLO algorithm loss function", s Interneta, https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088, pristupljeno 04.04.2024.

[26] "Roboflow", s Interneta, https://public.roboflow.com/object-detection/drone-gesture-control/3, pristupljeno 05.05.2024.

[27] "Python take screenshot from video", s Interneta: https://stackoverflow.com/questions/57791203/python-take-screenshot-from-video, pristupljeno 14.06.2024.

[28] Wong, K.: "yolov9", s Interneta, https://github.com/WongKinYiu/yolov9, pristupljeno 23.03.2024.

[29] "RTMP", s Interneta, https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol, pristupljeno 21.06.2024.

[30] Baressi Šegota, S.: "Instructions for the preparation of YOLOv5 and RTMP-based meteorite drone detection environment", Sveučilište u Rijeci, Tehnički fakultet, 2022.

[31] "MonaServer", s Interneta, https://www.monaserver.ovh, pristupljeno 21.06.2024

[32] "RTMP streaming setup", s Interneta, https://restream.io/blog/rtmp-streaming, pristupljeno 21.06.2024.

Sažetak i ključne riječi

U ovom diplomskom radu prezentirana je detekcija objekata primjenom YOLO algoritma. Rad započinje s uvođenjem osnovnih pojmova vezanih za umjetne neuronske mreže te se problematika postepeno nadograđuje na konvolucijske neuronske mreže koje igraju ključnu ulogu kada se govori o detekciji objekata i računalnom vidu općenito. Također, uvode se evaluacijske metrike koje se koriste za procjenu uspješnosti modela. Nakon stvaranja teorijske podloge, rad nastavlja s praktičnom primjenom navedenih alata u obliku YOLOv9 algoritma koji je trenutno jedan od najmodernijih i najbržih pristupa detekciji objekata. YOLOv9 algoritam treniran je da prepozna naredbe dane gestikulacijom ruku i tijela s ciljem upravljanja dronom. Subjekt može izdati naredbu "poleti", "sleti" ili "slijedi" te svaku od njih može dati ili položajem svoga tijela i/ili rukom što ukupno čini 6 različitih klasa koje je potrebno detektirati. Na kraju rada model je ispitan na detekciji u stvarnom vremenu koristeći RTMP protokol.

Ključne riječi: umjetna inteligencija, umjetne neuronske mreže, računalni vid, detekcija objekata, konvolucijske neuronske mreže, YOLO algoritam, YOLOv9, RTMP

Summary and key words

In this master's thesis, object detection using the YOLO algorithm is presented. Thesis begins with the introduction of basic concepts related to artificial neural networks and is gradually upgraded to convolutional neural networks, which play a key role when talking about object detection and computer vision in general. Also, the evaluation metrics that are used to evaluate the performance of the model are introduced. After creating the theoretical basis, the thesis continues with the practical application of the mentioned tools in the form of the YOLOv9 algorithm, which is currently one of the most modern and fastest approaches to object detection. The YOLOv9 algorithm is trained to recognize commands given by hand and body gestures in order to control the drone. The subject can give the command "take off", "land", or "follow", and each of them can be given either by the position of their body and/or by their hand, which makes a total of 6 different classes that need to be detected. At the end of the thesis, the model was tested on real-time detection using the RTMP protocol.

Keywords: artificial intelligence, artificial neural networks, computer vision, object detection, convolutional neural networks, YOLO algorithm, YOLOv9, RTMP

Prilog

YOLOv9 kod za treniranje i detekciju objekata dan je kodom 1. u nastavku.

Kod 1. YOLOv9 kod za treniranje i detekciju objekata

```
!nvidia-smi

from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

%cd /mydrive/Diplomski

%cd yolov9

!wget -P /mydrive/Diplomski https://github.com/WongKinYiu/yolov9/
    releases/download/v0.1/yolov9-e.pt

!python train_dual.py --workers 8 --batch 16 --img 640 --epochs
    30 --data /mydrive/Diplomski/yolov9/data.yaml --weights /
    mydrive/Diplomski/custom_data/runs/train/exp2/weights/best.pt
    --device 0 --cfg /mydrive/Diplomski/yolov9/models/detect/
    yolov9_custom.yaml --hyp /mydrive/Diplomski/yolov9/data/hyps/
    hyp.scratch-high.yaml

!python detect.py --img 640 --conf 0.1 --device cpu --weights /
    content/gdrive/MyDrive/Diplomski/yolov9/runs/train/exp3/
    weights/best.pt --source /content/gdrive/MyDrive/Diplomski/
    drone_dataset/test/images
```