Automatsko raspoznavanje akcija u rukometu

Konjuh, Niko

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:160404

Sveučilište u Rijeci – Odjel za informatiku

Poslovna informatika

Niko Konjuh

Automatsko raspoznavanje akcija u rukometu

Diplomski rad

Mentor: izv. prof. dr. sc. Marina Ivašić Kos prof. mat. i inf.

Rijeka, prosinac 2019.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 6.6.2019.

Zadatak za diplomski rad

Pristupnik: Niko Konjuh

Naziv diplomskog rada: Automatsko raspoznavanje akcija u rukometu

Naziv diplomskog rada na eng. jeziku: Automatic recognition of handball action

Sadržaj zadatka:

- teorijsko-istraživalački pregled dubokih neuronskih mreža
- usporedba i recenzija metoda detekcije objekata
- izradu kao i praktična primjena aplikacije koja bi se uz specijaliziranu doradu mogla upotrijebiti u svrhe poboljšanja i ispravaka sportskih poza kod sportaša svih uzrasti kao i u svrhu dobivanja statističke vrijednosti sportaša za trenere, menadžere, sportsku uprave te na kraju i za same roditelje sportaša mlađih uzrasta

Mentor: izv. prof. dr. sc. Marina Ivašić Kos prof. mat. i inf.

Voditeljica za diplomske radove: izv. prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 6.6.2019.

(potpis pristupnika)

Zahvala

Posebne zahvale mentorici izv. prof. dr. sc. Marini Ivašić Kos na zadavanju zanimljive teme za ovaj diplomski rad i na vrlo ugodnoj suradnji tijekom studija. Također zahvalio bi se na ukazanom povjerenju i pruženoj pomoći i podršci tijekom izrade samog diplomskog rada.

Hvala mome dragom rođaku Karlu Tomoviću koji je dobrovoljno ustupio svoje vrijeme i volju kako bi se dobili rezultati ovog istraživanja.

Na kraju, zahvale obitelji i prijateljima koji su mi bili podrška tijekom studija.

Sadržaj

Sažetak i ključne riječi

Ovaj rad prikazuje teorijsko-istraživalačku izradu kao i praktičnu primjenu aplikacije koja bi se uz specijaliziranu doradu mogla upotrijebiti u svrhe poboljšanja i ispravaka sportskih poza kod sportaša svih uzrasti kao i u svrhu dobivanja statističke vrijednosti sportaša za trenere, menadžere, sportsku uprave te na kraju i za same roditelje sportaša mlađih uzrasta.

U uvodu je opisan motiv izrade aplikacije za automatsko prepoznavanje pokreta sportaša kao i primjeri gdje bi se algoritam raspoznavanja mogao najučinkovitije primijeniti. Nadalje, sustavno je opisana podjela teoretskog i praktičnog dijela rada.

U drugom poglavlju objašnjeno što su to duboke konvolucijske mreže (engl. „deep convolutional networks“), gdje se primjenjuju i koji im je cilj.

U trećem poglavlju opisana je tehnika detekcije objekata kao i pripadajuće metoda za postizanje istoga. Također, na primjeru, prikazani su rad i rezultati svake od opisane metode, gdje se koristila jedno te ista biblioteka slika kako bi se dobio što bolji rezultat za usporedbu pojedinih metoda. Na kraju poglavlja, napravljena je usporedba svake korištene metode te su prikazane prednosti i slabosti istih.

U četvrtom poglavlju je opisana procedura izrade odnosno implementacije aplikacije u Android Studiju kao i popratni primjeri rada same aplikacije.

U zadnjem poglavlju se zaključuje cijelo istraživanje diplomskog rada, opisane metode te korišteni algoritam u mobilnoj aplikaciji.

Ključne riječi: duboke neuronske mreže, aplikacija, Android, detekcija pokreta, sport, TensorFlow, model

1. Uvod

Od začetka sportske rekreacije težilo se k tome da se sportaši razvijaju na osobnoj fizičkoj i motoričkoj razini kako bi se poboljšala ukupna kvaliteta samog sportaša u sportu kojem trenira te i u direktnom produžetku i kvaliteta same momčadi koja uvjetuje bolje plasmane na natjecanjima nacionalnih i internacionalnih razina. Napredak sportaša proizlazi iz dva faktora: upravljanjem cjelokupne kondicije sportaša i analiza dobivenih podataka o sportašu za vrijeme ili nakon treninga odnosno utakmice (natjecanja). Sportaši su u ovome radu prikazani kroz primjer rukometaša zbog autorovog iskustva u samome sportu. Kako bi se dobili precizni podaci, potrebno je pratiti pokrete odnosno akcije rukometaša u vremenu njihovog izvođenja, gdje se najučinkovitije pokazalo ono stvarno (realno) vrijeme kada se akcije izvode.

U rukometu, iznimno je interesantno pratiti sportaševe pokrete gornjeg dijela tijela sa naglaskom na položaj njihove pucačke ruke dok se igra napadački dio utakmice. Tu se može zamijetiti koliko je ispravna tehnika držanja ruke u trenutku pucanja na gol, točnije, drži li sportaš lakat pucačke ruke u razini, iznad razine ili ispod razine svoje glave. Za vrijeme izvođenja akcija rukometaša u napadačkom dijelu, interesantno je pratiti poziciju i položaj sportaša kroz cijeli vremenski ciklus izvođenja akcije koja obično traje od 3 do 10 sekundi te se na taj način može vidjeti koliko ispravno sportaš izvodi akcije svojih ekstremiteta i gornjeg dijela tijela u početku, u sred i na kraju izvođenja akcije. U obrambenom dijelu utakmice, sportaš poprima sasvim drugačiji stav tijela u odnosnu na onaj u napadačkom dijelu te je kod obrambenog dijela jako bitno za popratiti generalno držanje rukometaša, što proizlazi iz položaja njegove lijeve i desne ruke, visine laktova na obim rukama, položaj lijeve i desne noge, njihove raširenosti u odnosu jedne na drugu te zakrivljenost kralježnice. Kod rukometnih golmana prati se njihovo kretanje po golu za vrijeme pasivnog dijela utakmice (kada se lopta ne upućuje u neposrednom vremenu na gol) na način da se pregledava njihov položaj nogu i ruku kako prate kretanje lopte od jednog igrača do drugog na terenu. Također ono zanimljivije za pratiti je svakako akcije njihova tijela za vrijeme aktivnog dijela, kada je lopta upućena na gol ili će biti upućena u neposrednim sekundama. Tada se prati položaj golmanova tijela na temelju toga u koji dio gola lopta leti, odnosno hoće li golman promijeniti položaj svojih ruku, svojih nogu, glave i gornjeg dijela tijela.

Opisano praćenje bi se realiziralo kroz kameru pametnog telefona kako bi održala učinkovitost, a istovremeno maksimalno povećala trivijalnost i ekonomičnost dobivenih rezultata, bez potrebe korištenja velike, nezgrapne, profesionalne video opreme za koju su potrebna zamašna ulaganja. Mobilni uređaji imaju jednu bitnu manu u usporedbi sa digitalnim uređajima (kamere, fotoaparati). Uobičajene leće s fiksnim fokusom i manji senzori ograničavaju njihove performanse pri slabom osvjetljenju.

U profesionalnom sportu prati se gotovo sve, do najsitnijeg detalja, kako bi se dobili najprecizniji izlazni rezultati pomoću kojih sportaš te njihovi treneri i menadžeri mogu raditi na poboljšanju. Prate se način vođenja lopte, pucanja na gol, stava u pasivnoj obrani, stava u aktivnoj obrani, obrambena reakcija pucanja na gol, brzina ispucavanja lopte, broj timski i individualno zabijenih pogodaka, broj asistencija, broj timskih i individualnih promašaja,

posjed lopte, obrane golmana, pucanja u okvir gola, pucanja van okvira gola. Obično za to postoji educirana osoba kao timski statističar koji zapisuje detalje o svakom sportašu u svome timu. Rekreativni sportaši obično nemaju tu širinu svoje pomoćne klupe, te se ne može pratiti njihov napredak (ili nazadovanje) kao kod profesionalnih sportaša. U rekreaciji, naglasak je stavljen ili na tim ili na individuu, rijetko na oboje te to predstavlja otegotnu okolnost u razvoju samog sportaša. Valja napomenuti kako je u slučajevima profesionalnih sportaša prisutan faktor ljudske pogreške statističara, koji se ne može predvidjeti niti direktno uvjetovati. Implementacijom računalno stvorenog algoritma za praćenje akcija sportaša može se otkloniti taj problem kao i problem ljudske pogreške te na taj način pružiti svima, bez obzira na kvalitetu sportaša, uvid u načine poboljšanja tehnike i izvedbe akcija što dovodi sportsku izvedbu na novu razinu.

Diplomski rad je podijeljen u dva dijela, gdje je prvi istraživalačke i teoretske prirode, dok je drugi dio diplomskog rada praktična primjena i objašnjenje koda aplikacije za raspoznavanje pokreta sportaša.

U prvome dijelu bit će opisani principi rada dubokih neuronskih mreža, njihova primjena, povijest te ostale bitne značajke za razumijevanje poduzimanja daljnjih koraka u ovome radu, koji su krucijalni kako bi se na svrsishodan način koristili postupci detekcije samih sportaša te raspoznale njihove akcije.

Također, opisana je i detekcija objekata u realnom vremenu te različite vrste dubokih konvolucijskih mreža koje su bile istražene za potrebe ovoga rada. Uspoređene su njihove primjene na realnim uzorcima i to isključivo na najnovijim izdanjima spomenutih dubokih neuronskim mreža. Na primjerima su pokazane biblioteke TensorFlow i Keras koje se mogu koristiti za detekciju objekata.

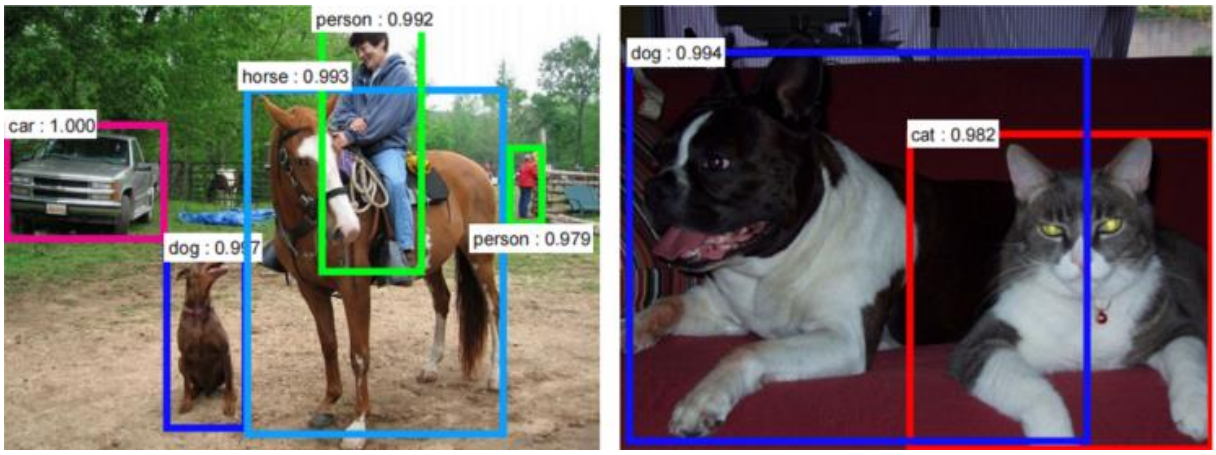
U drugome dijelu prikazana je izrada Android aplikacije za prepoznavanje pokreta sportaša. Aplikacija je izrađena u Android Studio programskom sučelju, a koristili su se Java, Kotlin i C++ programski jezici, te XML „markup“ jezik za izradu raznih sučelja aplikacije.

U prilogu diplomskog rada dani su različiti video primjeri korištenja aplikacije za prepoznavanje pokreta sportaša (rukometašu) tijekom razvoju.

2. Duboke neuronske mreže

Duboke konvolucijske neuronske mreže (engl. „deep convolutional neural networks“, skraćeno CNN) su vrsta dubokih neuronskih mreža kojima je najčešća primjena u klasifikaciji i analizi vizualnog sadržaja, najčešće slika ili videa. Najveću primjenu zasigurno imaju u prepoznavanju značajka u slikama i videozapisima, sustavima preporuke, klasifikaciji slika, detekciji objekata, medicinskoj analizi slika te obradi prirodnog jezika [1].

Slika 1: Pimjer skupa scena nad kojima CNN može vršiti detekciju: a) Nogometaš, b) Prometni znakovi c) Životinje



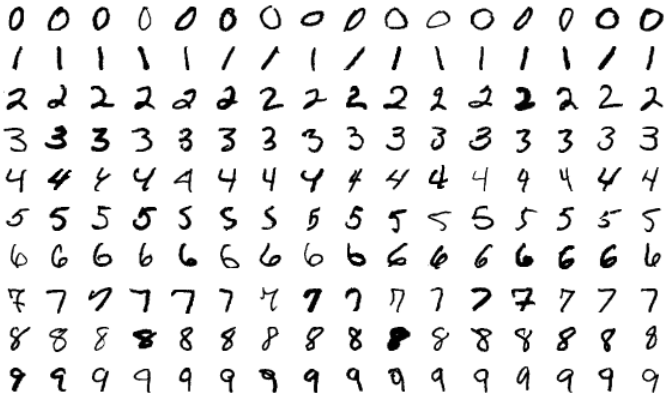
Slika 2: Primjer prikaza scena sa završenom detekcijom CNN-e te pripadajućim oznakama i numeričkim sigurnostima detekcije

Na Slici 1, CNN može prepoznati prizore i sustav je u mogućnosti predložiti odgovarajuće naslove (poput „nogometaš šutira nogometnu loptu) dok Slika 2 prikazuje rezultat detekcije konvolucijskih neuronskih mreža na slikama sa scenama svakodnevnih objekata, ljudi i životinja. U posljednje vrijeme, CNN se također pokazao dosta efikasan kod zadataka obrade prirodnog jezika kao što je na primjer klasifikacija rečenica [2].

2.1. Povijest konvolucijskih neuronskih mreža

Prvi rad sa modernim konvolucijskim neuronskim mrežama objavljen je devedesetih godina prošlog stoljeća, inspiriran „Neocognitron“-om. „Neocognitron“ se koristi za prepoznavanje rukopisa i drugih, jednostavnih zadataka prepoznavanja uzoraka [3]. Yann LeCun i suradnici, u svom radu „Gradient-Based Learning Applied to Document Recognition“ [4] (citirano sada preko 18000 puta), pokazali su da se CNN model koji objedinjuje jednostavnije značajke u progresivno složenije značajke može uspješno koristiti za prepoznavanje rukopisa.

Konkretno LeCun i suradnici naučili su CNN model koristeći MNIST bazu podataka [5] rukom pisanih znamenki, Slika 3. MNIST je skup podataka koji uključuje slike rukom napisanih znamenki uparenih s njihovom istinskom oznakom 0, 1, 2, 3, 4, 5, 6, 7, 8 ili 9. CNN model se uči na MNIST-u davanjem primjera slike, tražeći da klasificira koja se znamenka prikazuje na slici, a zatim ažurira postavke modela s obzirom da li je ispravno klasificirao znamenku ili nije. Najsuvremeniji CNN modeli danas mogu postići gotovo 100% točnost klasifikacije MNIST znamenki.



Slika 3. MNIST set podataka rukom pisanih znamenki

Postignuta točnost klasifikacije znamenaka omogućila je da se fizička pošta sortira sa strojevima koji koriste automatizirane tehnike prepoznavanja rukopisa za čitanje adresa.

Kroz 1990-e i rane 2000-e godine, istraživalo se CNN model, međutim značajno povećanje popularnosti CNN mreže dogodilo se 2012. godine nakon što je konvolucijska neuronska mreža zvana AlexNet postigla značajno bolje rezultate u označavanju slika od prethodno ostvarenih u ImageNet izazovu [6]. Alex Krizhevsky je sa suradnicima objavio rad „ImageNet Classification with Deep Convolutional Neural Networks“ [6] u kojem detaljno opisuje AlexNet model.

Slično MNIST-u koji se sastoji od sličica rukom pisanih znamenki 0 – 9 i odgovarajućih oznaka broja, ImageNet je također javni, slobodno dostupan skup podataka pripremljen za strojno učenje. ImageNet sadrži 14,197,122 "prirodnih slika" opisanih odgovarajućom klasom.

Sa laičkog stajališta, konvolucijske neuronske mreže koriste se za prepoznavanja obrazaca na određenom mediju, što je najčešće slika ili videozapis. To se radi tako što se na mediju traže odgovarajući obrasci, pomoću ručno unesenih parametara, ili pomoću gotovih, konačnih parametara. U prvih par slojeva, CNN-a mreža može prepoznati linije i kuteve, no ako te obrasce dalje propustimo kroz neuronsku mrežu prepoznaju se kompleksniji obrasci. To svojstvo čini CNN iznimno adekvatne za identificiranje objekata na slikama. Duboke neuronske mreže se sastoje od mnogo različitih slojeva od kojih su najbitnijih konvolucijski sloj, sloj sažimanja, aktivacijske funkcije te potpuno povezani sloj [7].

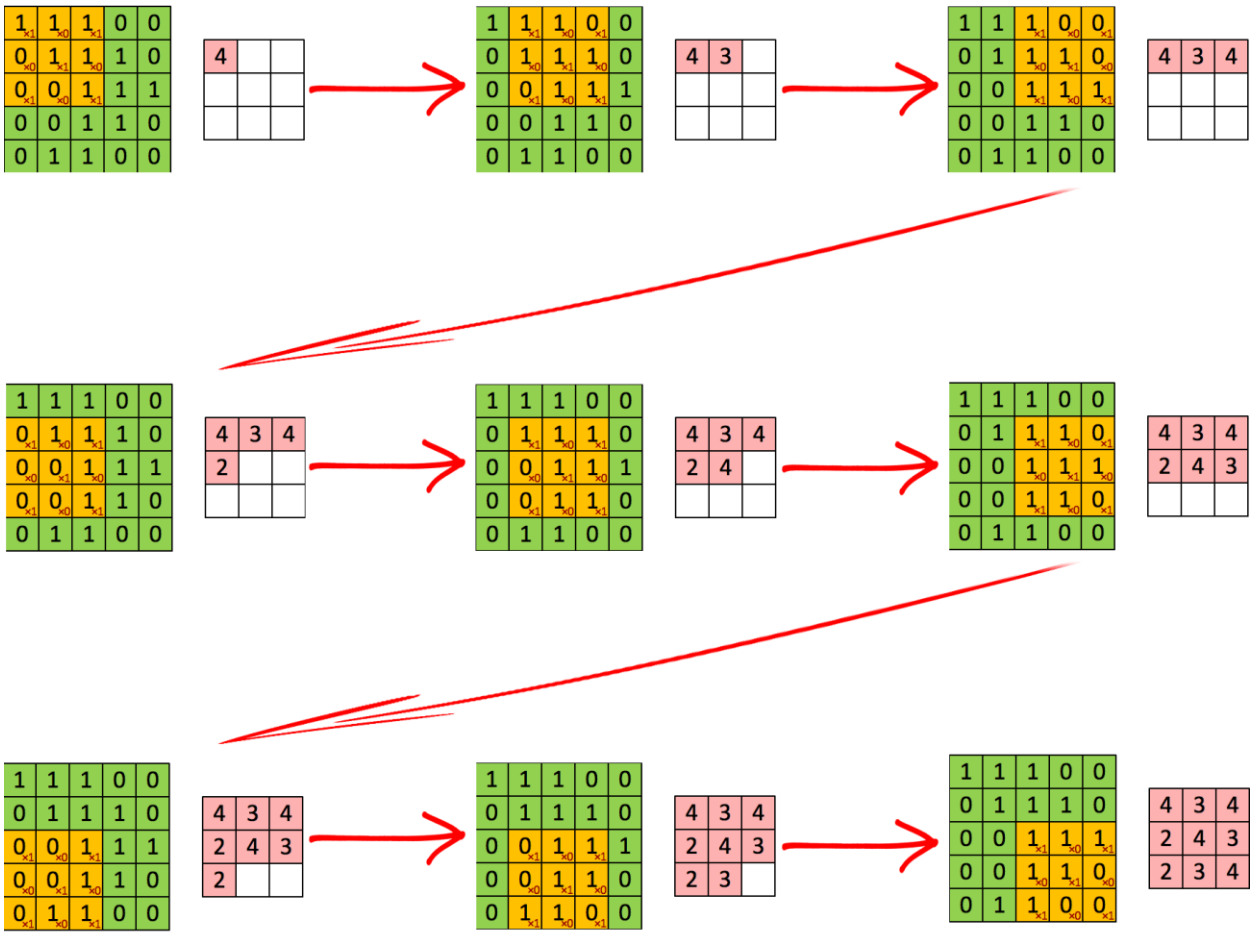
Da bi se dobio bolji uvid u rad dubokih konvolucijskih mreža, mora se najprije objasniti što su to konvolucije odnosno konvolucijski sloj, a onda i ostali slojevi.

2.2 Konvolucijski sloj

Konvolucija uključuje, u najkraćem smislu, prolazak kroz sliku i primjenu filtera kako bi se pronašli obrasci.

U širem smislu, konvolucijski sloj (engl. „convolutional layer“) je ključna komponenta. Njegova uloga je, da koristeći određeni broj filtera, kojima se izlučuje neka korisna informacija koju mreža treba naučiti, provede konvoluciju nad ulaznim podacima i tako generira mapu značajki (engl. *feature map*s). Pritom su bitna tri svojstva konvolucije: raspršena povezanost (engl. *sparse connectivity*), dijeljenje težina i translatorna ekvivarijantnost (engl. *translation equivariance*). Pojedinom filteru je u svakom trenutku vidljiv samo manji dio ulaznih podataka – tzv. receptivno polje [7].

Razmotrimo sliku 4 predstavljenu matricom dimenzije 5x5. Ako se uzme 3x3 matrica iz originalne matrice te se miče po slici, množeći 3x3 matricu sa prozorom slike koja je bila korištena za stvaranje jedne vrijednosti. Sada je matrica pomiče udesno i prema dolje kako bi se dovršila čitava slika. Na kraju se dobiva konvoluirana značajka koja je prikazana na posljednjoj matrici. Korištena matrica 3x3 se naziva kernel.

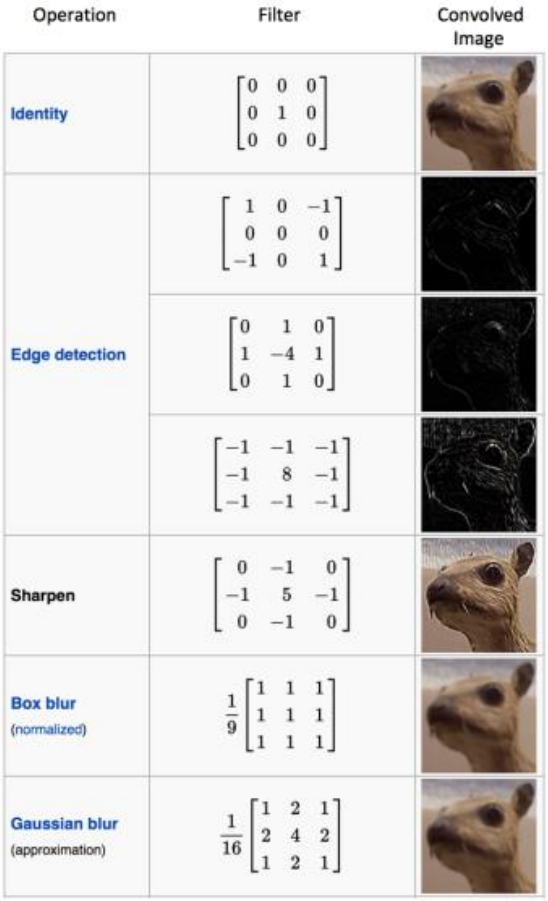


Slika 4: Koraci matrice konvolucijskog sloja

Cilj konvolucijske matrice je filtriranje. Dok se prolazi kroz sliku, efikasno se provjeravaju obrasci u tom dijelu/sekciji slike. To je učinkovito zbog filtera, snopa težinskih vrijednosti predstavljenih u vektoru, koji se množe sa vrijednostima koje ispisuje konvolucija. Kada se trenira slika, te težinske vrijednosti se mijenjaju, a onda kad je vrijeme za evaluaciju slike te težinske vrijednosti vraćaju visoke vrijednosti ako misle da su vidjeli neki obrazac prethodno. Kombinacije visokih težinskih vrijednosti iz raznih filtera omogućuju mreži da predvidi sadržaj slike. Na slici, lijeva matrica prikazuje sliku, dok desna matrica prikazuje konvoluirano svojstvo. Iz gornje slike evidentno je da će različite vrijednosti matrice filtra stvoriti različite mape značajki za istu ulaznu sliku. Kao primjer, ulazna slika je (Slika 5):

Slika 5: Ulazna slika

Na Slici 6. mogu se vidjeti učinci konvolucije na gornju sliku sa različitim filterima. Kao što je prikazano, mogu se izvoditi operacije otkrivanja ruba, izoštravanja, zamagljivanja sa jednostavnim mijenjanjem numeričkih vrijednosti filter matrice prije konvolucijske operacije - to znači da različiti filtri mogu otkriti različite značajke slike, na primjer, rubove, krivulje i tako dalje.



Slika 6: Različiti filtri ulazne slike

2.3 Sloj sažimanja

Sloj nakon konvolucijskog sloja najčešće je sloj sažimanja (engl. „pooling layer“) u CNN arhitekturi. U samome konceptu neuronskih mreža sažimanje ima nezanemarivu ulogu. Uobičajeno je da se u arhitekturi konvolucijskih neuronskih mreža periodično, između sukcesivnih konvolucijskih slojeva, umetne sloj sažimanja [8] s ciljem smanjivanja rezolucije mapi te povećanja prostorne invarijantnosti – nepromjenjivosti na manje pomake (rotacije, transformacije) značajki u nekome uzorku ili slici. Na Slici 7. prikazan je princip rada sloja sažimanja na matrici, gdje lijeva matrica prikazuje svojstvo konvolucije dok desna matrica prikazuje svojstvo sažimanja.

Slika 7: Koraci matrice sloja sažimanja

Dvije najkorištenije metode sažimanja su sažimanjem s izborom najvećeg elementa (engl. „max pooling“) i sažimanje sa prosječnom vrijednosti (engl. „average pooling“).

Sažimanje izborom najvećeg elementa proizvodi maksimalnu vrijednost neke podregije, dok prosječno sažimanje proizvodi prosječnu vrijednost neke podregije.

Generalno, sloj sažimanja koristi se za smanjenje prostornih dimenzija, ali ne i dubine, čije su glavne prednosti:

- Smanjivanje prostornih informacija (dimenzija) utječe na računske performanse
- Smanjivanje prostornih informacija znači da postoji manje parametara za treniranje modela te samim time se smanjuje šansa za pretreniranje (engl. overfitting)
- Dobiva se određeni translacija invarijance

2.4. Aktivacijske funkcije

Aktivacijske funkcije su čvorovi koji se stavljaju na kraj ili između neuronskih mreža. Oni pomažu u odlučivanju hoće li se neuron aktivirati ili ne. Aktivacijskih funkcija ima mnogo, no najpoznatije su:

- Linearne aktivacijske funkcije
- Funkcije skoka
- Sigmoidalne funkcije

Funkcije skoka i sigmoidalne funkcije puno su bolji odabir za neuronske mreže koje vrše klasifikaciju dok se linearne funkcije najčešće koriste u izlaznim slojevima u slučajevima kada je potreban neograničen izlaz.

Aktivacijska funkcija propagira ili zaustavlja ulaznu vrijednost u neuron ovisno o svom obliku. Jedna od najčešće korištenih aktivacijskih funkcija u konvolucijskim neuronskim mrežama je ReLU (engl. Rectified Linear Unit) aktivacijska funkcija pa se u literaturi može naći pojam «ReLU sloj», koji upućuje na sloj u kojemu se primjenjuje ReLU aktivacijska funkcija. ReLU funkcija je definirana na sljedeći način:

$$f(x) = \max(0, x)$$

gdje je *x* ulaz u neuron. Ova aktivacijska funkcija unosi nelinearnost u model te model najčešće znatno bolje opisuje stvarni objekt budući je svaki kompleksni sustav u manjoj ili većoj mjeri nelinearan.

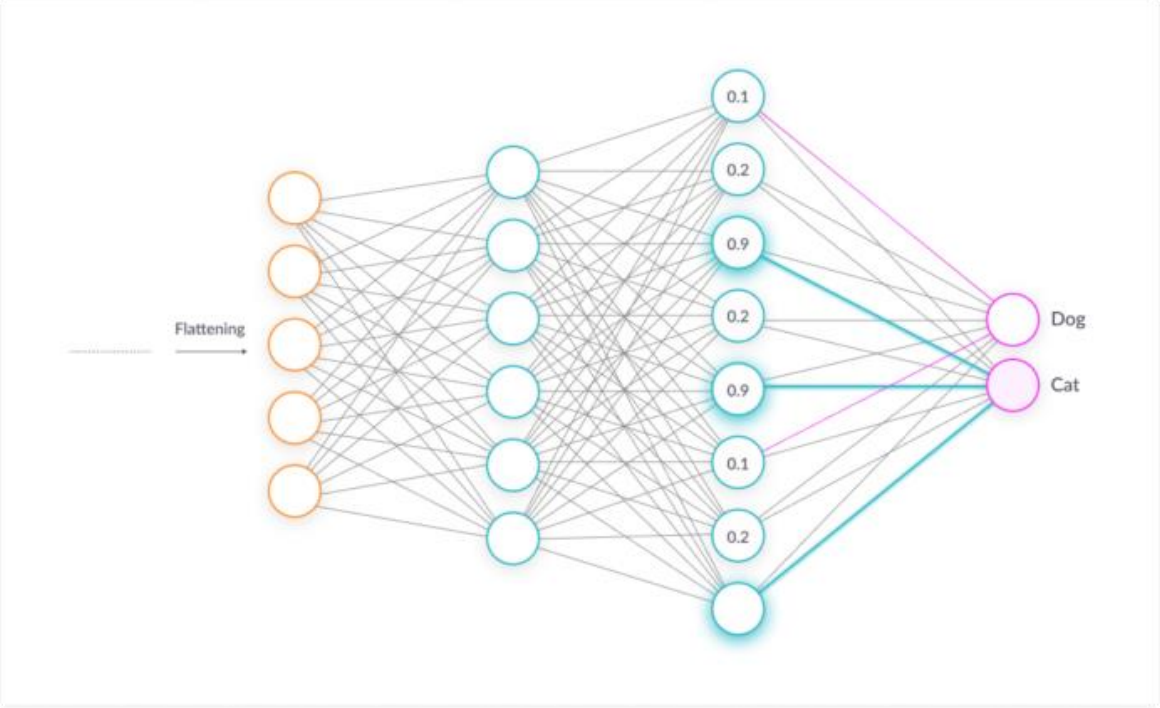
Pravi izbor aktivacijske funkcije može značajno poboljšati performanse konvolucijske neuronske mreže koja se primjenjuje na određene zadatke [9].

2.5. Potpuno povezan sloj

Cilj potpuno povezanog sloja je uzeti rezultate procesa konvolucije i parametare CNN mreže i koristiti ih za klasifikaciju slike ili objekata na slici u oznake. Izlaz iz CNN mreže „spljoštava“ se u jedan vektor vrijednosti, koji predstavljaju vjerojatnost da neko svojstvo pripada oznaci. Na primjer, ako je slika mačke, značajke koje predstavljaju stvari poput dlake, krzna ili njuške treba imati veliku vjerojatnost za oznaku "mačka".

Slika 8. ilustrira kako ulazne vrijednosti ulaze u prvi sloj neurona. Oni se množe s težinama i prolaze kroz aktivacijsku funkciju (tipično ReLu), baš kao u klasičnoj umjetnoj neuronskoj mreži. Zatim prelaze naprijed do izlaznog sloja u kojem svaki neuron predstavlja klasifikacijsku oznaku.

Potpuno povezani dio CNN mreže prolazi kroz vlastiti postupak povratnog širenja (engl. back propagation) kako bi se što je moguće točnije utvrdile težine. Svaki neuron dobiva težinu koja odgovara najprikladnijoj oznaci. Napokon, neuroni "glasaju" za svaku od oznaka, a oznaka s najviše glasova je pobjednik i predstavlja krajnja odluku klasifikacije.



Slika 8: Princip rada potpuno povezanog sloja

3. Detekcija objekata

Detekcija objekata je računalna tehnologija povezana sa računalnim vidom i obradom slika koja se bavi otkrivanjem primjera semantičkih objekata određene klase (ljudi, znakova, životinja, zgrada, automobila) u digitalnim slikama i videozapisima. Dobro istražene domene detekcije objekata uključuju detekciju lica i detekciju pješaka u ulici/cesti. Detekcija objekata ima aplikacije u mnogim područjima računalnog vida, uključujući pronalaženje slika u velikim skupovima slika i video nadzor.

Metode kojima se postiže detekcija objekata se generalno dijele na dva pristupa – pristup strojnog učenja (engl. „machine learning approach“) i pristup dubokog učenja (engl. „deep learning approach“). Za pristupe strojnog učenja, potrebno je najprije definirati i izlučiti značajke pomoću odgovarajućih metoda za izlučivanje značajki, a zatim pomoću tehnike kao što je vektorski strojevi za podršku odlučivanju (SVM) odrediti klasifikaciju objekata [10].

S druge strane, tehnike dubokog učenja sposobne su detektirati objekte bez posebnog definiranja značajki i one su upravo temeljene na konvolucijskim neuronskim mrežama.

U nastavku rada opisane su metode dubokog učenja koje se koriste u svrhu detekcije objekata te su objašnjeni principi koji prokazuju relevantne podatke.

3.1. Regionalni prijedlozi

3.1.1. Regionalna konvolucijska neuronska mreža

Umjesto da radi na jako velikom broju regija, algoritam regionalne konvolucijske neuronske mreže (skraćeno R-CNN) predlaže veliki broj „okvira“ na slici i provjerava sadrži li bilo koji od okvira neki određeni objekt. R-CNN koristi selektivno pretraživanje za izvlačenje tih okvira iz slike (okviri se u nastavku nazivaju regije) [11].

Kako bi se moglo dalje razumjeti regionalne konvolucijske neuronske mreže, potrebno je prvotno shvatiti što je selektivno pretraživanje i kako identificira različite regije. Postoje četiri regije koje stvaraju neki objekt: različite ljestvice (skale), boje, teksture i same granice slike

Selektivnim pretraživanjem identificiraju se obrasci na slici i na temelju toga predlažu različite regije. U nastavku je prikazano funkcioniranje selektivnog pretraživanja na slikovnim primjeru.

Slika 9: Proces rada regionalnih konvolucijskih neuronskih mreža [12]

Prvo se uzima slika kao ulazni podatak za obradu (slika 9. lijevo). Nakon toga, stvaraju se početne presegmentacije tako da iz ove slike dobijemo više regija (Slika 9. sredina). Tehnika zatim kombinira slična područja kako bi oblikovala veće područje na temelju sličnosti boja, sličnosti teksture, veličine i kompatibilnosti samog oblika (Slika 9. desno). Na kraju, te regije proizvode konačne lokacije objekta (regionalni prijedlog).

Ukratko rad R-CNN-a za otkrivanje objekata može se sažeti na nekoliko koraka:

1. Odabir unaprijed trenirane konvolucijske neuronske mreže
2. Model se ponovno trenira i to tako da se trenira zadnji sloj mreže na temelju broja klasa koje je potrebno detektirati
3. Cilj je dobiti regiju interesa za svaku sliku pa sve regije preoblikujemo tako da se mogu podudarati s veličinom ulaza CNN
4. Nakon dobivanja regija, treniramo SVM za klasificiranje objekata i pozadine. Za svaki razred osposobljavamo po jedan binarni SVM
5. Konačno, treniramo linearni regresijski model za generiranje čvršćih graničnih okvira za svaki identificirani objekt na slici. [13]

3.1.1.1. Nedostatci R-CNN mreže

Kao i svaka tehnika detekcije objekata, R-CNN dolazi sa svojim ograničenjima. Treniranje R-CNN modela je skupo i sporo zbog slijedećih koraka koji su potrebni [12]:

- Izdvajanje 2000 regija za svaku sliku na temelju selektivnog pretraživanja
- Izvlačenje značajki pomoću CNN-a za svaku regiju slike. Na primjer, ako imamo N slika, broj R-CNN značajki za tu sliku bi bio N*2000
- Proces detektiranja objekata sa R-CNN tehnikom sastoji se od tri modela (CNN za ekstrakciju značajki, linearni SVM klasifikator za identifikaciju objekata, regresijski model za generiranje čvršćih graničnih okvira).

Svi navedeni procesi u kombinaciji čine R-CNN vrlo sporim. Za izradu predviđanja svake nove slike potrebno je 40-50 sekundi, što model u stvari čini nezgrapnim i praktički nemogućim za izgradnju kada se suoči sa ogromnim skupom podataka.

Kako bi se otklonila gore navedena ograničenja, izumljena je nova tehnika detekcije objekata – Fast RCNN.

3.1.2. Brze regionalne konvolucijske neuronske mreže - Fast R-CNN

U usporedbi s R-CNN-om, brzi R-CNN ima veću prosječnu preciznost, jedno stepeni trening, trening koji ažurira sve mrežne slojeve, a pohrana na disku nije potrebna za pred memoriranje značajki. Umjesto da se CNN pokrene 2 000 puta po slici, može se pokrenuti samo jednom po slici i dobiti sva područja koja nas zanimaju (regije koje sadrže neki objekt).

Ross Girshick, autor R-CNN-a, smislio je pokretanje CNN-a samo jednom po slici i tada pronašao način da to računanje podijeli u 2 000 regija. U brzoj R-CNN ulaznu sliku dostavljamo CNN-i, što zauzvrat generira karte konvolucijskih značajki. Pomoću ovih karata izdvajaju se regije prijedloga. Potom se koristi regionalni interesni sloj za sažimanje kako bi se sve predložene regije preoblikovale u fiksnu veličinu kako bi se mogli uvesti u potpuno povezanu mrežu [12].

Raščlanimo ovaj koncept na nekoliko dijelova:

1. Kao i na prošlom primjeru, uzima se slika kao ulazni podatak
2. Ta se slika prosljeđuje u CNN koja zauzvrat generira regije interesa
3. Regionalni interesni sloj sažimanja primjenjuje se na ove regije kako bi ih preoblikovao prema ulazu konvolucijske mreže. Zatim se svaka regija prosljeđuje u potpuno povezanu mrežu.
4. Softmax sloj se koristi na vrhu potpuno povezane mreže za izlazne klase. Uz softmax sloj, paralelno se koristi i linearni regresijski sloj za izlazne koordinate okvira za predviđene klase

Dakle, umjesto da koriste tri različita modela (poput R-CNN-a), brzi R-CNN koristi jedan model koji izvlači značajke iz regija, dijeli ih na različite klase i vraća granične okvire za identificirane klase istovremeno [14].

3.1.2.1. Nedostatci Fast R-CNN mreže

Čak i brzi R-CNN ima određena problematična područja. Također koristi selektivno pretraživanje kao metodu prijedloga za pronalaženje regija interesa, što je spor i dugotrajan proces. Za otkrivanje objekata potrebno je oko 2 sekunde po slici, što je puno bolje u usporedbi

s RCNN-om. Ali kada razmotrimo velike skupove podataka iz stvarnog života, čak ni brzi RCNN više ne izgleda tako brzo.

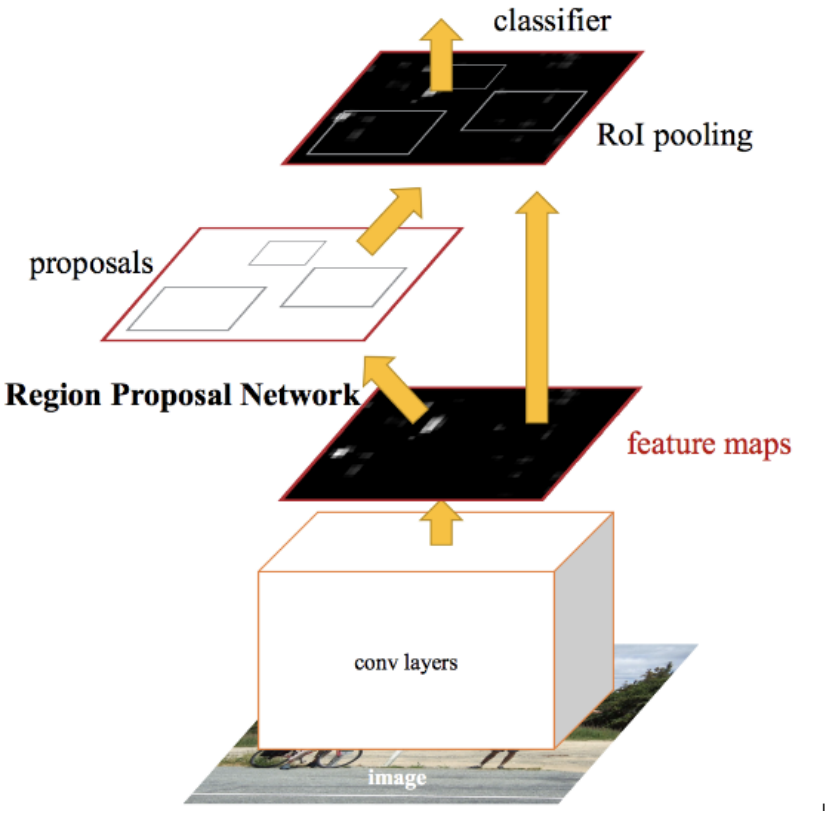
Iz toga razloga stvoren je još jedan algoritam za otkrivanje objekata sa ne tako originalnim imenom – brži R-CNN [12].

3.1.3. Brže regionalne konvolucijske neuronske mreže - Faster R-CNN

Faster R-CNN je modificirana verzija brze R-CNN. Glavna razlika između njih je u tome što Brza R-CNN koristi selektivno pretraživanje za generiranje područja interesa, dok brža R-CNN koristi "Mrežu prijedloga regija", zvanu RPN. RPN uzima mape značajki slike kao ulaz i generira skup prijedloga objekata, od kojih svaka ima ocjenu predikcije nekog objekta kao izlaz.

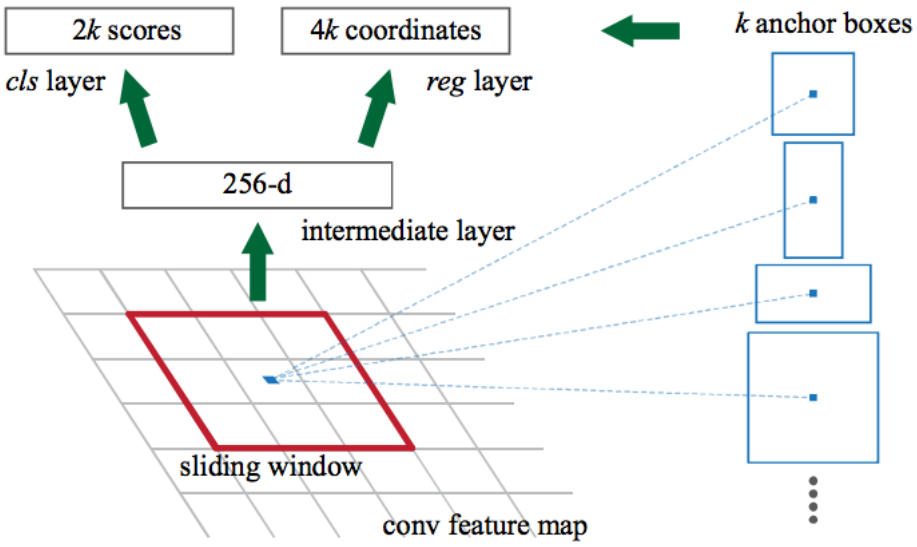
Koraci koji se koriste u bržim R-CNN-ma su prema [12] prikazani na Slici 10:

1. Uzimamo sliku kao ulaz i prosljeđujemo je u CNN-i koji vraća kartu značajki za tu sliku
2. Na ove se karte značajki primjenjuje mreža prijedloga regija. To vraća prijedloge objekata zajedno s ocjenom predikcije nekog objekta
3. Na ove prijedloge primjenjuje se regionalni interesni sloj sažimanja kako bi se svi prijedlozi primijenili na istu veličinu objekta
4. Konačno, prijedlozi se prosljeđuju u potpuno povezani sloj s softmax slojem i linearnim regresijskim slojem na vrhu kako bi se razvrstali i ispisali granični okviri za objekte



Slika 10: Princip rada bržih R-CNN-a [12]

Za početak, brži RCNN preuzima karte značajki s CNN-a i šalje ih u mrežu prijedloga regije, Slika 11. RPN koristi klizni prozor preko ovih karata značajki, a na svakom prozoru generira k kutije za sidrenje različitih oblika i veličina:



Slika 11: Princip rada RPN-a

Kutije za sidrenje su granične kutije fiksne veličine koje su smještene na cijeloj slici i imaju različite oblike i veličine. RPN za svako sidro predviđa dvije stvari:

- Prvo je vjerojatnost da je sidro objekt (ne uzima u obzir kojoj klasi objekt pripada)
- Drugo je regresor za ograničavajuću kutiju za podešavanje sidara kako bi bolje odgovarao objektu

S ovime su dobivene granične kutije različitih oblika i veličina koje se prenose u regionalno interesni sloj sažimanja. Sada je moguće da nakon koraka RPN-a postoje prijedlozi koji im nisu dodijeljeni. Možemo uzeti svaki prijedlog i obrezati ga tako da svaki prijedlog sadrži objekt. To radi regionalno interesni sloj sažimanja. Izdvaja karte značajki fiksnih veličina za svako sidro. Zatim se te značajke karte prenose u potpuno povezani sloj koji ima softmax i linearni regresijski sloj. Konačno klasificira objekt i predviđa granične okvire za identificirane objekte. [2]

3.1.3.1. Nedostatci Fast R-CNN mreže

Svi algoritmi za detekciju objekata spominjani dosad koriste regije za identifikaciju objekata. Mreža ne gleda cjelokupnu sliku u jednom potezu, već se dijelove slike fokusira uzastopno. To stvara dvije komplikacije [12]:

- Algoritam zahtijeva mnogo prolaza kroz jednu sliku kako bi izvukao sve predmete detekcije
- Kako postoje različiti sustavi koji rade jedan za drugim, uspješnost sustava koji napreduju ovisi o tome kako su prethodni sustavi funkcionirali

3.1.4. Maskirane regionalne konvolucijske neuronske mreže – Mask R-CNN

Mask R-CNN (engl. „Mask R-CNN“) u osnovi je produžetak Fast R-CNN-e. Fast R-CNN koristi se prvenstveno za zadatke detekcije objekata. Za datu sliku vraća oznaku klase i koordinate okvira za svaki objekt na slici.

Mask R-CNN izgrađen je na vrhu Faster R-CNN-a. Dakle, za datu sliku će Mask R-CNN, osim oznake klase i koordinata graničnog okvira za svaki objekt, vratiti i masku objekta.

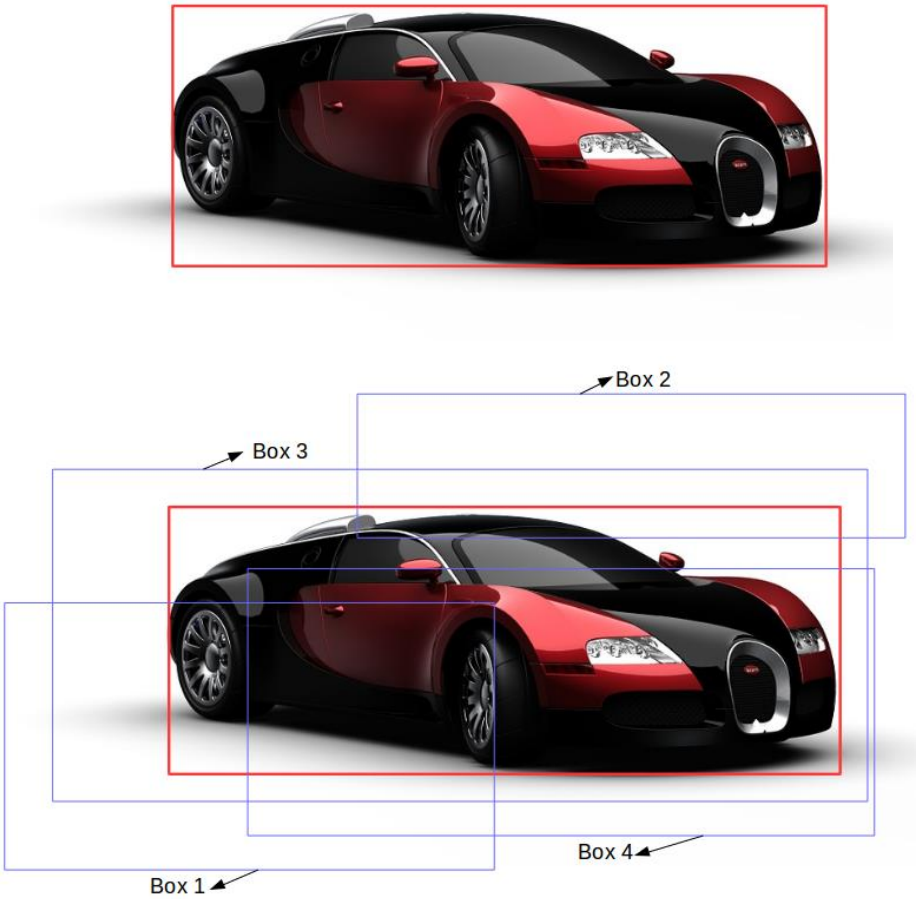
Slično kao u ConvNet-u koji se koristi u Faster R-CNN-u za vađenje značajki iz slike, Mask R-CNN koristi ResNet 101 arhitekturu za izdvajanje značajki iz slika. Ove značajke koriste se kao ulaz u sljedeći sloj mreže i primjenjujemo mrežu prijedloga regija (RPN) koja u osnovi predviđa je li objekt prisutan u toj regiji. U ovom koraku dobivamo one regije ili značajke za koje model predviđa da sadrže neki objekt.

Područja dobivena iz RPN-a mogu biti različitih oblika stoga se nanosi sloj za objedinjavanje i pretvara sve regije u isti oblik. Zatim ove regije prolaze kroz potpuno povezanu mrežu tako da se predviđaju oznake klase i granični okviri. Do ovog trenutka, koraci su gotovo slični načinu na koji djeluje Faster R-CNN. Sada dolazi razlika između ove dvije metode jer Mask R-CNN stvara masku segmentacije. Za masku segmentacije prvo se izračunava regija interesa da bi se smanjilo vrijeme računanja. Za sve predviđene regije, izračunava se odnos Presjeka i Unije (IoU) predviđenih regija s ispravno označenim okvirima, Slika 11. IoU možemo računati ovako:

$$\boldsymbol{IoU = Područje\ sjecišta\ /\ Područje\ unije}$$

Samo ako je IoU veći od ili jednak 0.5 smatramo predviđenu regiju regijom interesa. Inače se zanemaruje ta regija. To se radi za sve regije, a zatim se odabere samo skup regija za koje je IoU veća od 0.5. [15]. Kako bi se bolje razumjela opisana formula na slikama 12 i 13 prikazan je model sportskog automobila, preciznije na slici 12 automobil je prikazan unutar crvenog okvira koji predstavlja okvir istine (engl. „ground truth box“) dok je na slici 13 automobil, osim

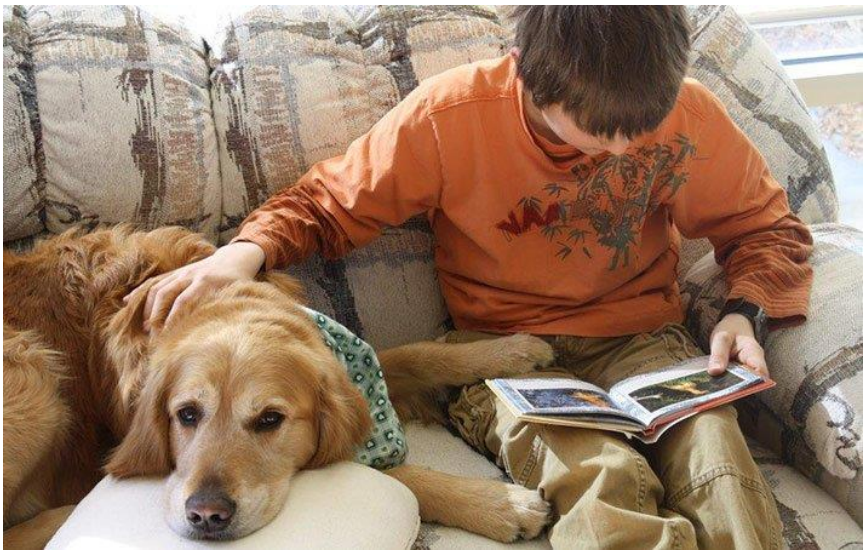
okvirom istine, prikazan i sa 4 interesne regije RPN-e koje su na slici 13 prikazane sa nazivima „Box 1“(hrv. Okvir 1), „Box 2“(hrv. Okvir 2), „Box 3“(hrv. Okvir 3), „Box 4“(hrv. Okvir 4),



Slike 12 i 13: Predviđanja IoU okvira 1,2,3,4 na slici sportskog automobila

IoU okvira 1 („Box 1“) i okvira 2 („Box 2“) moguće je da je niži od 0,5, dok je IoU okvira 3 („Box 3“) i okvira 4 („Box 4“) približno veći od 0,5. Stoga može se reći da su okviri 3 i okvir 4 područje interesa za primjer objekta na slici 12., dok će okviri 1 i okvir 2 biti zanemareni.

Tek nakon što se dobiju interesne regije utemeljene na IoU vrijednostima, može se dodati maska . Ovo vraća masku segmentacije za svaku regiju koja sadrži objekt. Vraća masku veličine 28 x 28 za svaku regiju koja se zatim povećava za zaključak. Ako se razmotri slika 14, vidi se originalni ulazni podatak na kojemu su prikazan dječak i njegov ljubimac. Segmentacijska maska tog ulaznog podatka prikazana je na slici 15.

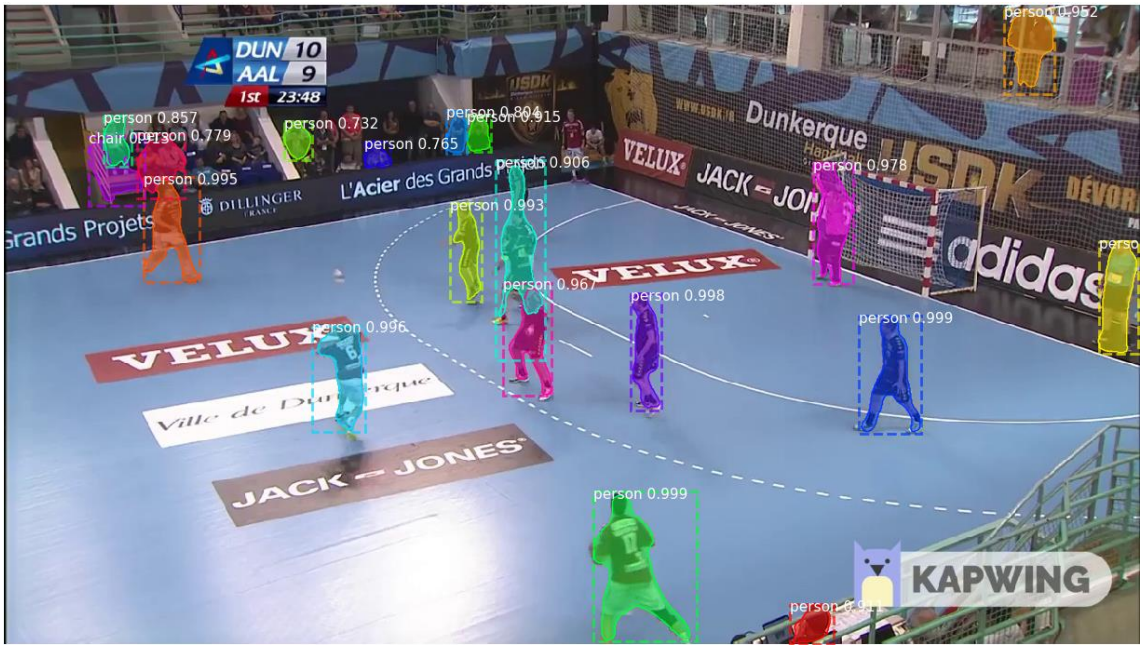


Slike 14 i 15: Mask R-CNN originalna slika i segmentacijska maska

Na slici 15, model je segmentirao sve objekte na slici. To je posljednji korak u Mask R-CNN gdje se predviđa maska za svaki objekt zasebno na ulaznom podatku odnosno slici.

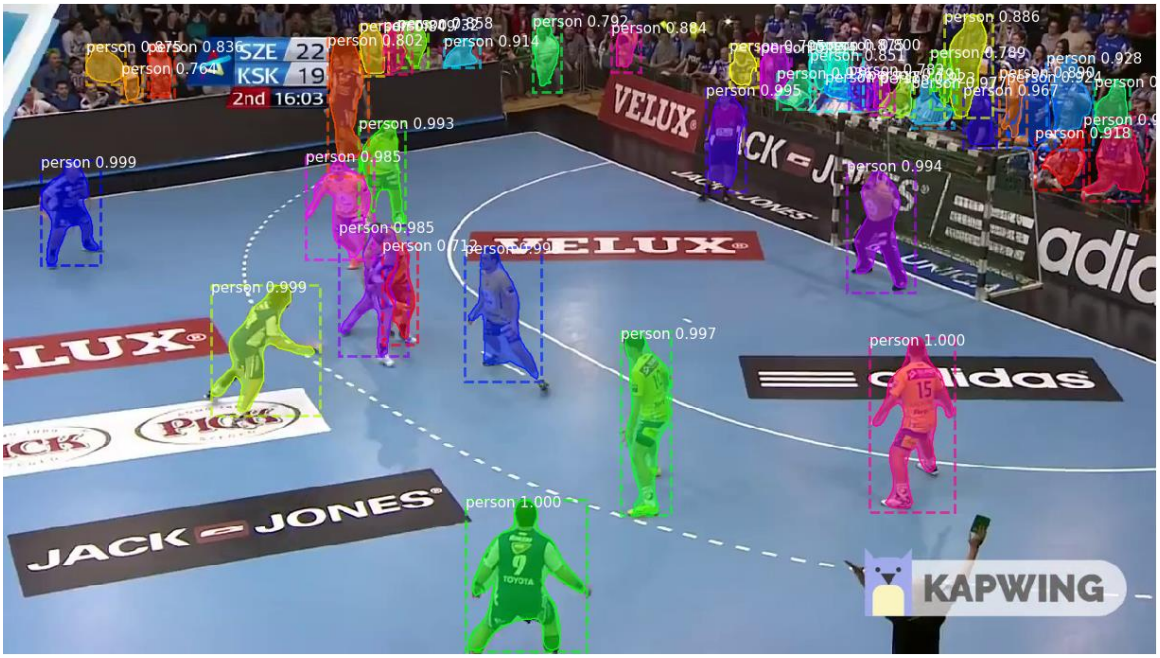
Mask R-CNN korištena je kao referentni primjer prikaza rada cijele kategorije regionalnih konvolucijskih neuronskih mreža kod zadatka detekcije objekata.

Slike prikazane na sljedećim primjerima izrezane su iz videa koji je priložen uz ovaj diplomski rad. Pri reprodukciji videa vidi se stvarna brzina i točnost Mask R-CNN-e, dok su slike prikazane u ovom diplomskom radu najbolji pokušaj prikaza rada algoritma na jednom frame-u.

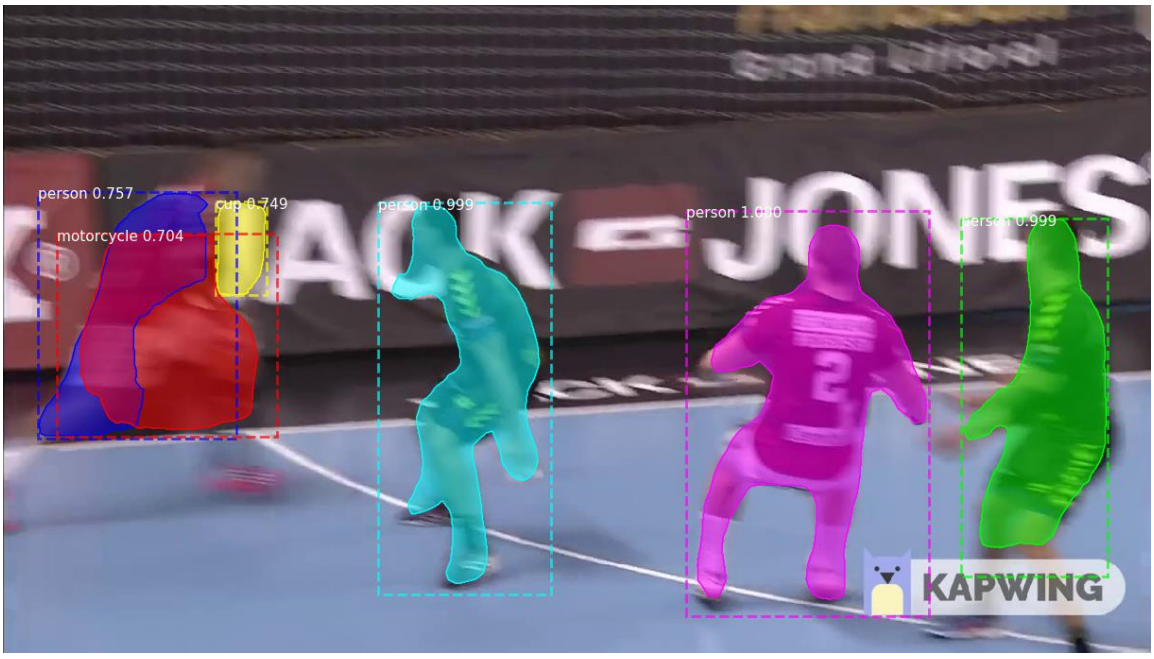


Slika 16: Mask R-CNN detekcija rukometnog razigravanja

Slika 17: Mask R-CNN detekcija rukometaša u profilu



Slika 18: Mask R-CNN detekcija rukometnog razigravanja 2



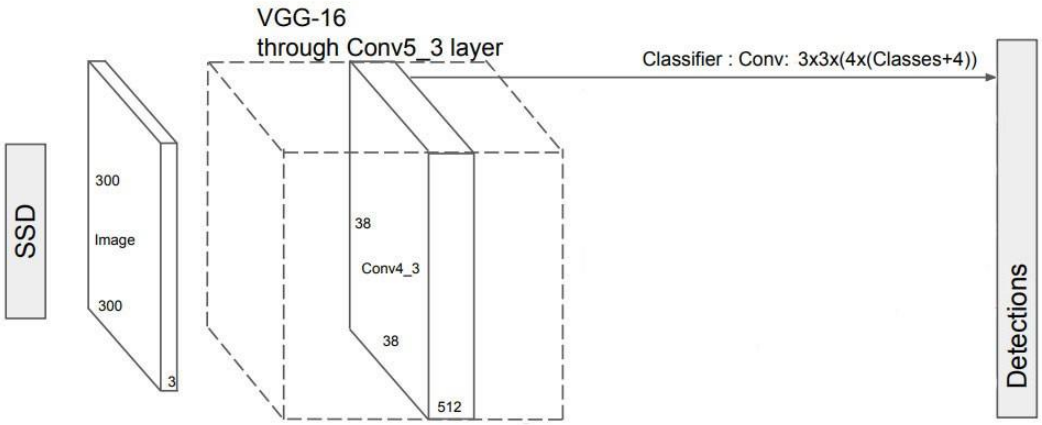
Slika 19: Mask R-CNN detekcija detekcija rukometaša u skoku prilikom zamućenog frame-a

3.2. Detektor s višestrukim snimkama

SSD *(Single Shot Multibox Detector, skraćeno SSD)* je dizajniran za detekciju objekata u stvarnom vremenu. Faster R-CNN koristi mrežu regionalnih prijedloga za stvaranje graničnih okvira i koristi ih za razvrstavanje objekata. Premda je točnost Faster RCNN-e iznimno dobra, ako ne i trenutno najbolja, cijeli proces se odvija u 7 sličica u sekundi. Daleko ispod onoga što je potrebna obrada u stvarnom vremenu. SSD ubrzava proces uklanjanjem potrebe mreže regionalnih prijedloga. Da bi se nadomjestio pad točnosti, SSD primjenjuje nekoliko poboljšanja, uključujući značajke u više skala i zadane okvire. Ova poboljšanja omogućuju SSD-u da odgovara bržim točnostima R-CNN koristeći slike niže rezolucije, što dodatno povećava brzinu.

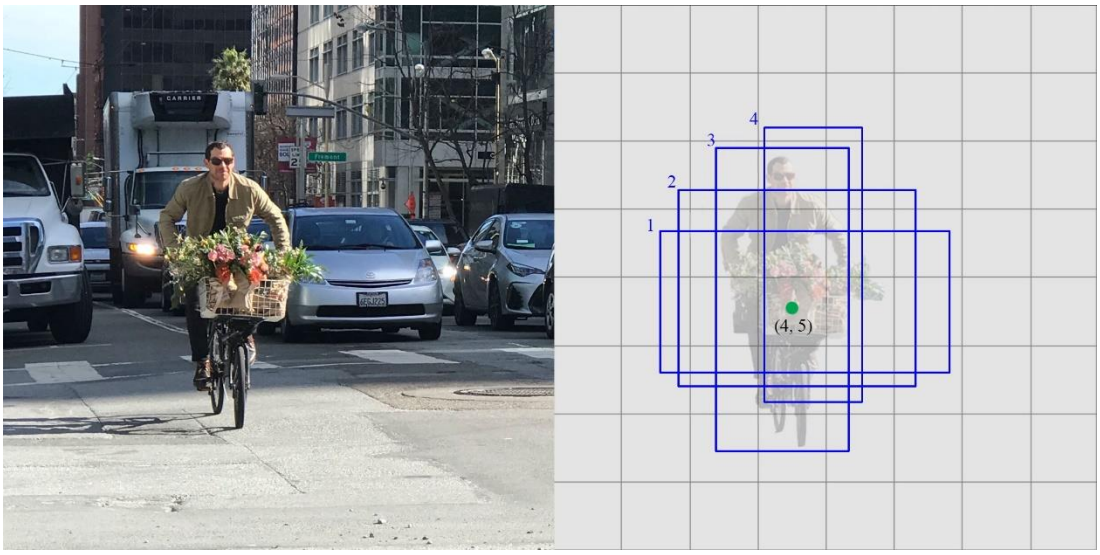
Detekcija SSD objekata sastoji se od 2 dijela:

- Izdvajanje značajki
- Primjenjivanje konvolucijskih filtera za detektiranje objekata



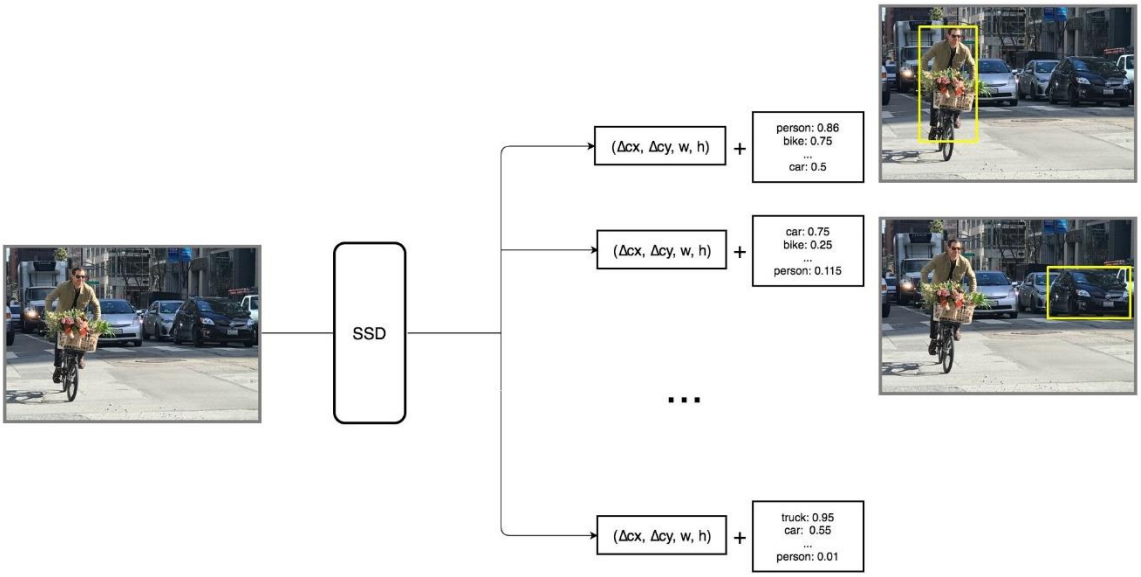
Slika 20: Grafički prikaz SSD detekcije

SSD koristi VGG16 za ekstrakciju značajki. Zatim otkriva objekte pomoću sloja Conv4_3. Za ilustraciju, crtamo da je Conv4_3 prostorno 8 × 8 (trebao bi biti 38 × 38). Grafički prikaz SSD-a prikazan je na slici 20. Za svaku ćeliju (koja se također naziva lokacija) čini 4 predviđanja objekta prikazanih na slici 21.



Slika 21 :SSD - Lijevo: izvorna slika. Desno: 4 predviđanja za svaku ćeliju

Svako predviđanje sastoji se od graničnog okvira i 21 rezultata za svaku klasu (jedna dodatna klasa bez objekta) prikazanih na slici 22, a kao rezultat za ograničeni objekt odabire se najveći rezultat. Conv4_3 daje ukupno 38 × 38 × 4 predviđanja: četiri predviđanja po ćeliji bez obzira na dubinu karata značajki. Kao što se i očekivalo, mnoga predviđanja ne sadrže objekt. SSD zadržava klasu "0" kako bi pokazao da nema objekata.



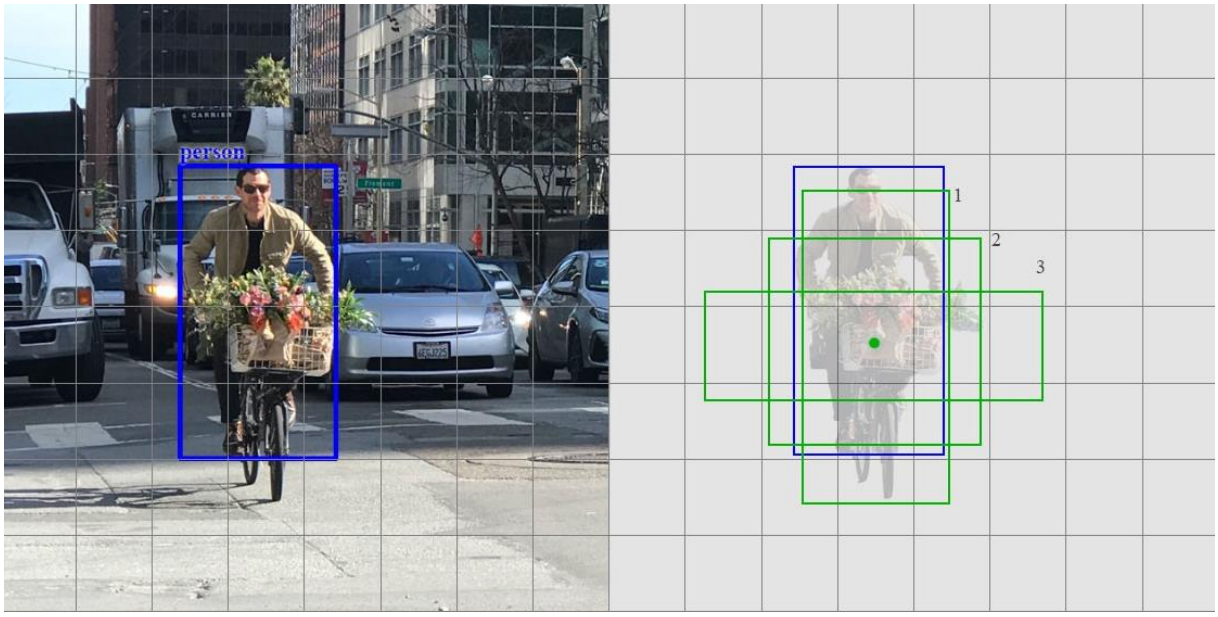
Slika 22: SSD - Svako predviđanje uključuje graničnu kutiju i 21 rezultat za 21 klasu (jedna klasa bez predmeta)

U početku SSD je bio definiran kao da otkriva objekte iz jednog sloja. Zapravo koristi više slojeva (karte značajki s višestrukim ljestvicama) za detektiranje svakog objekta zasebno. Kako CNN postupno smanjuje prostornu dimenziju, tako se smanjuje i razlučivost karata značajki.

SSD koristi slojeve niže razlučivosti za otkrivanje objekata većih razmjera. Na primjer, karte veličine 4 × 4 koriste se za veće objekte.

SSD dodaje još 6 pomoćnih slojeva konvolucije nakon VGG16. Pet od njih bit će dodano za otkrivanje objekata. U tri od tih slojeva, napravi se 6 predviđanja umjesto 4. Ukupno, SSD predviđa 8732 predviđanja koristeći 6 slojeva s čime karte značajki s višestrukim ljestvicama značajno poboljšavaju točnost detekcije.

SSD predviđanja klasificiraju se kao pozitivna ili kao negativna. SSD koristi samo pozitivne podudarnosti za izračunavanje troškova lokalizacije (neusklađenost graničnog okvira). Ako odgovarajući zadani granični okvir (nije isto što i predviđeni granični okvir) ima IoU veći od 0.5 sa ispravno označenim okvirima, podudaranje je pozitivno. Inače je negativan.



Slika 23: SSD - Osnovni objekt istine (plava) i tri zadana granična polja (zelena).

Na slici 23 prikazani su na lijevom dijelu slike osnovni objekt detekcije (čovjek, osoba) sa pripadnim okvirom istine te na desnom dijelu osnovni objekt istine sa okvirom istine plave boje i sa tri zadana granična okvira zelene boje. Samo zadani okviri 1 i 2 (ali ne 3) imaju IoU veći od 0.5 s ispravno označenim okvirima (plavi okvir). Dakle, samo polja 1 i 2 su pozitivne podudarnosti. Nakon što identificiramo pozitivne podudarnosti, za izračun troškova koristimo odgovarajuće predviđene rubričke okvire. Ova strategija podudaranja lijepo dijeli oblik temeljne istine za koju je predviđanje odgovorno. [16]

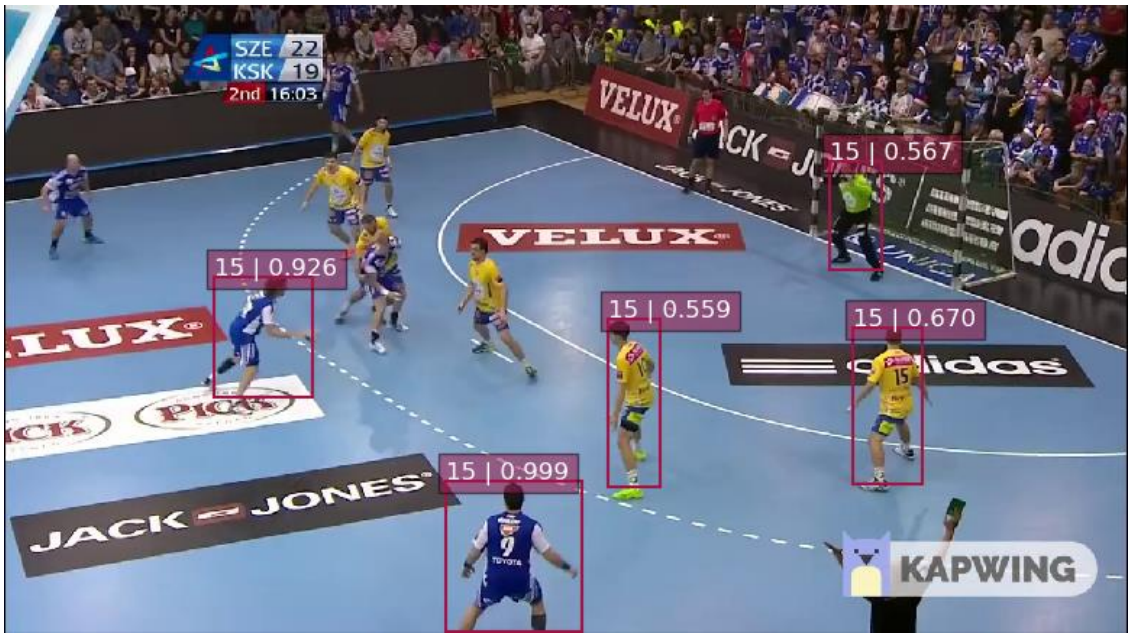
Svakako je bitno napomenuti kako SSD koristi ne-maksimalno suzbijanje za uklanjanje duplikata predviđanja koja upućuju na isti objekt. SSD sortira predviđanja prema rezultatima povjerenja. Polazeći od predviđanja s najvećom ocjenom pouzdanosti, SSD procjenjuje ima li prethodno predviđeni granični okvir IoU veći od 0.45 s trenutnim predviđanjima za istu klasu. Ako se pronađe, zanemarit će se trenutna predviđanja.

SSD metoda korištena je kao referentni primjer prikaza rada detekcije objekata.

Slike prikazane na sljedećim primjerima izrezane su iz videa koji je priložen uz ovaj diplomski rad. Pri reprodukciji videa vidi se stvarna brzina i točnost SSD-a, dok su slike prikazane u ovom diplomskom radu najbolji pokušaj prikaza rada algoritma na jednom frame-u.

Slika 24: SSD detekcija rukometnog razigravanja

Slika 25: SSD detekcija rukometaša u profilu



Slika 26: SSD detekcija rukometnog razigravanja 2

Slika 27: SSD detekcija detekcija rukometaša u skoku prilikom zamućenog frame-a

3.3. You Only Look Once (YOLO)

U usporedbi s drugim mrežama za razvrstavanje prijedloga regija (Fast RCNN) koje provode otkrivanje na različitim prijedlozima regije i na taj način završavaju višestruko predviđanje za različite regije na slici, YOLO arhitektura više liči na FCNN (potpunu konvolucijsku neuronsku mrežu) i provodi sliku jednom kroz FCNN te daje izlaz - predviđanje. Ta arhitektura dijeli ulaznu sliku na mreži i za svaku graničnu kutiju 2 granične okvire i vjerojatnosti klase za one

granične okvire. Treba se imati na umu da je ograničavajuća kutija vjerojatno veća od same rešetke.

Najveće prednosti YOLO metode (u usporedbi sa prijašnje opisanim):

- Brzina (45 slika u sekundi - bolja nego u stvarnom vremenu)
- Mreža razumije generaliziranu reprezentaciju objekata (ovo im je omogućilo da nauče mrežu za slike iz stvarnog svijeta, a predviđanja umjetničkih djela i dalje su prilično točna)

YOLO se smatra revolucionarnim algoritmom u polju detekcije objekata jer zauzima potpuno drugačiji pristup nego njegovi prethodnici. Ono nije tradicionalni klasifikator koji se pretvara u otkrivanje objekata. YOLO zapravo sliku prikazuje samo jednom (otuda i naziv: Samo jedno gledanje engl. „You Only Look Once“) i dijeli je na mrežu od 13 x 13 ćelija: Ćelije su prikazane na slici 28 na kojoj je ulazni podatak slika psa i bicikle sa pripadnom pozadinom.

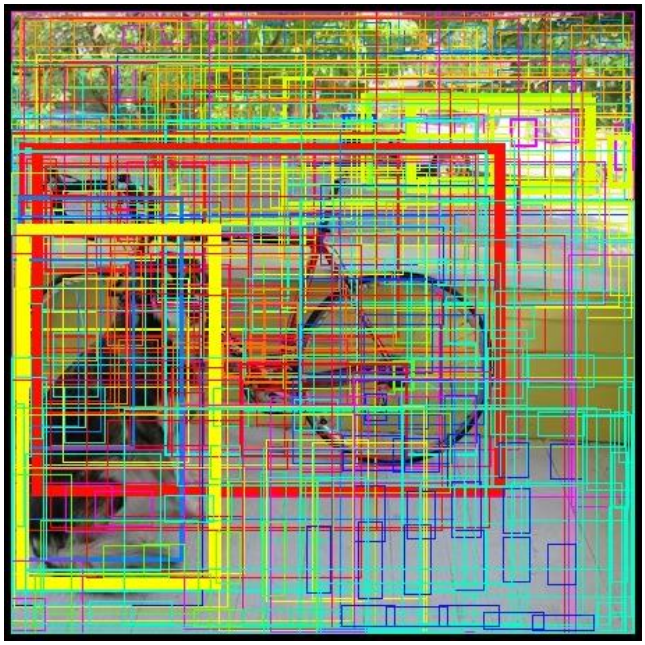
Slika 28: YOLO dijeljenje slike na ćelije

Svaka od ovih ćelija odgovorna je za predviđanje 5 graničnih okvira. Ograničavajući okvir opisuje pravokutnik koji zatvara objekt. YOLO također daje ocjenu pouzdanosti koja nam govori koliko je sigurno da predviđeni granični okvir zapravo zatvara neki objekt. Ovaj rezultat ne govori o tome kakav je predmet u kutiji, samo ako je oblik okvira dobar. Predviđene granične kutije mogu izgledati ovako: Što je veća ocjena pouzdanosti, to će se skupiti deblji okvir:

Slika 29: YOLO granične kutije

Za svaki granični okvir prikazanoj na slici 29, ćelija također predviđa klasu. To djeluje poput klasifikatora: daje distribuciju vjerojatnosti na sve moguće klase.

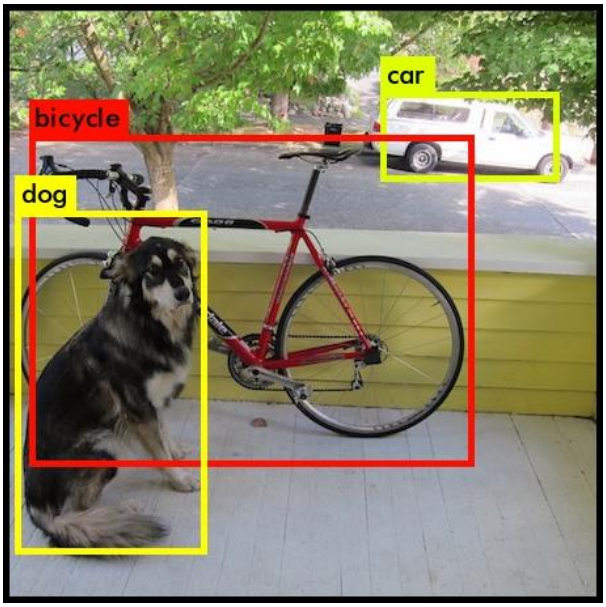
Ocjena pouzdanosti za granični okvir i predviđanje klase kombiniraju se u jedan konačni rezultat koji nam govori vjerojatnost da ovaj ograničavajući okvir sadrži određenu vrstu objekta. Na primjer, velika masno žuta granična kutija na lijevoj strani je 85% sigurna da sadrži objekt "pas":



Slika 30: YOLO Ocjena pouzdanosti za granični okvir i predviđanje klase

Budući da postoji 13 × 13 = 169 ćelija rešetke i svaka ćelija predviđa 5 ograničavajućih okvira, na kraju ćemo imati 845 ograničavajućih okvira kao što je prikazano na slici 30. Ispada da će većina ovih kutija imati vrlo niske ocjene pouzdanosti, tako da držimo samo kutije čiji je konačni rezultat 30% ili više (ovaj prag može se promijeniti ovisno o tome koliko se želi da detektor bude).

Konačno predviđanje tada je prikazano na slici 31:



Slika 31: YOLO Konačno predviđanje

Od više stotina ograničavajućih kutija zadržale su se samo tri sa posljednje slike jer su dale najbolje rezultate. Treba svakako naglasiti iako je bilo više stotina zasebnih predviđanja, sve su oni napravljeni u isto vrijeme – neuronska mreža se samo jednom pokrenula. To je ključan razlog zašto je YOLO smatran revolucionarnim algoritmom detekcije objekata. [17]

3.3.1. Ograničenja YOLO-a

YOLO nameće snažna prostorna ograničenja predviđanjima graničnog okvira jer svaka ćelija rešetke predviđa samo dva polja i može imati samo jednu klasu. Ovo prostorno ograničenje ograničava broj objekata u blizini koji model može predvidjeti. Model se „bori“ s malim predmetima koji se pojavljuju u skupinama, poput jata ptica. Budući da model nauči predvidjeti ograničavanje okvira iz podataka, on se bori s generalizacijom za objekte u novim ili neobičnim omjerima ili konfiguracijama. Model također koristi relativno grube značajke za predviđanje ograničavajućih okvira jer arhitektura ima više slojeva padanja uzoraka s ulazne slike. Konačno, dok treniramo na funkciji gubitaka koja se približava performansama otkrivanja, funkcija gubitka tretira iste pogreške u malim graničnim okvirima nasuprot velikim ograničenim okvirima. Mala pogreška u velikoj kutiji općenito je benigna, ali mala

pogreška u malom kutiji ima puno veći učinak na IOU (Intersection over Union[1]). Glavni izvor pogreške su pogrešne lokalizacije. [13]

3.3.2. YOLO v2

Kako bi se dodatno poboljšala pogreška lokalizacija, a zadržala točnost lokalizacije, razvijen je YOLO v2.

Na 67 FPS-a (engl. Frame per Second, sličica po sekundi) dobiva 76.8% srednje prosječne preciznosti detekcije, dok na 40 FPS-a dobiva 78.6% srednje prosječne preciznosti, što je više od Faster RCNN i SSD metode, čineći YOLO v2 najkorištenijim algoritmom posljednjih godina za prepoznavanje objekata na medijskim datotekama.

Značajke YOLO v2 u usporedbi sa originalnom YOLO metodom [18]:

1. Normalizacija serije
    a. Normalizacija serije (engl. Batch Normalization) koristi se na svim konvolucijskim slojevima
    b. 2% poboljšanja srednje prosječne preciznosti
2. Klasifikator visoke rezolucije
    a. Nakon što je istreniran sa 224x224 slike, YOLO v2 koristi i 448x448 slika za precizno prilagođavanje klasifikacijske mreže za 10 epoha ImageNet-a
    b. 4% porasta srednje prosječne preciznosti
3. Klasteri dimenzija
    a. Veličine i ljestvice kutija za sidrenje unaprijed su definirane bez prethodnih podataka, baš poput one u Bržem R-CNN-u
    b. Korištenje standardnih euklidskih k-sjedinjenja na osnovi udaljenosti nije dovoljno dobro, jer veće kutije stvaraju više pogreške nego manje kutije
4. Predviđanje izravne lokacije
    a. YOLOv1 nema ograničenja u predviđanju lokacije što model čini nestabilnim u ranim iteracijama. Predviđeni okvir za ograničavanje može biti daleko od izvornog mrežnog mjesta.
5. Finozrnate značajke
    a. Izlazni prikaz mape dimenzija 13 × 13 dovoljan je za otkrivanje velikih predmeta

[1] Intersection over Union je mjerna vrijednost koja se koristi za mjerenje točnosti detektora objekta na određenom skupu podataka. Svaki algoritam koji daje predviđene granične okvire kao izlaz može se procijeniti pomoću IoU.

b. Kako bi se kvalitetno otkrili mali objekti, karte značajki 26 × 26 × 512 iz ranijeg sloja preslikane su u mapu značajki 13 × 13 × 2048, a zatim povezane s izvornim mapama značajki 13 × 13 za otkrivanje
c. Postiže 1% povećanje srednje prosječne preciznosti

6. Treniranje s viša razina
    a. Za svakih 10 serija nasumično se biraju nove dimenzije slike
    b. Dimenzije slike su {320, 352,…, 608}
    c. Mreža mijenja veličinu i nastavlja treniranje

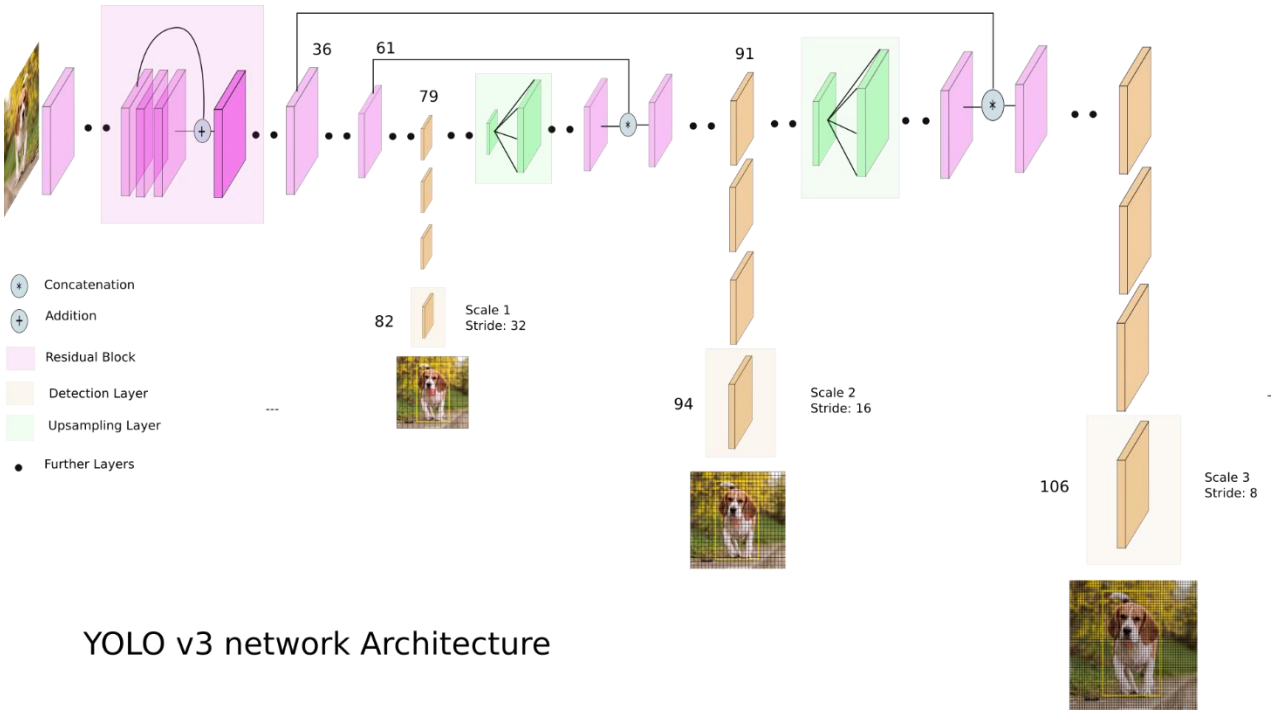
3.3.3. YOLO v3 ili YOLO9000

Trenutno najnovija edicija algoritma YOLO, službenog naziva „YOLO9000: Better, Faster, Stronger”. (hrv. YOLO9000: Bolji, Brži, Jači). U vrijeme začetka (2018. godina) YOLO v3 se uistinu pokazao kao najbolji izbor duboke konvolucijske mreže na tržištu. YOLO9000 bio najbrži i također jedan od najtočnijih algoritama. Međutim, nekoliko mjeseci unazad i više nije najpreciznji jer su razvijeni algoritmi poput RetinaNet-a i SSD-a.

Ali ta se brzina rasprodala zbog točnosti u YOLO v3. Dok je ranija varijanta na Titan X grafičkim karticama imala 45 FPS, trenutna verzija ima oko 30 FPS-a. To ima veze s povećanjem složenosti osnovne arhitekture koja se zove Darknet.

YOLO v2 koristio je prilagođenu dubinsku arhitekturu darknet-19, izvorno 19-slojnu mrežu dopunjenu s još 11 slojeva za otkrivanje objekata. S arhitekturom od 30 slojeva, YOLO v2 često se borio s otkrivanjem malih objekata. To se pripisuje gubitku sitnozrnih značajki jer su slojevi smanjili ulazne podatke. Da bi to ispravio, YOLO v2 koristio je mapiranje identiteta, objedinjujući značajke značajki iz prethodnog sloja za snimanje značajki niske razine.

U usporedbi s YOLO v2, YOLO v3 koristi varijantu Darknet-a, koji u početku ima 53 slojnu mrežu obučenu na Imagenetu. Za zadatak otkrivanja, na njega su složeni još 53 sloja, čime se dobije 106 slojeva potpuno konvolucijski temeljne arhitekture za YOLO v3. To je razlog sporosti YOLO v3 u usporedbi s YOLO v2. Arhitektura YOLO v3 izgleda kao na slici 32.



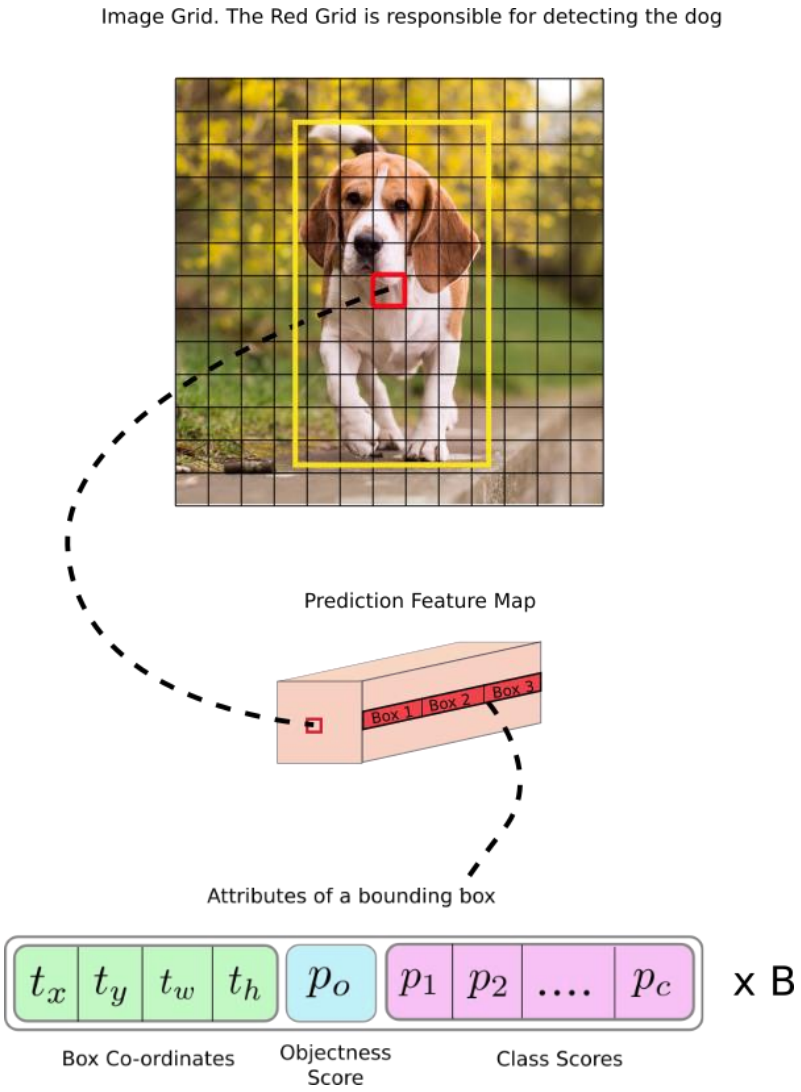
Slika 32: YOLO v3 mrežna arhitektura

Novija se arhitektura može pohvaliti mogućnošću da otkriva objekte u tri različite skale, prikazane vizualno na slici 33. YOLO je potpuno konvolucijska mreža i njezin se konačni izlaz generira primjenom kernela 1 x 1 na mapi značajki. U YOLO v3, otkrivanje se vrši primjenom 1 x 1 detekcijske jezgre na mapama značajki tri različite veličine na tri različita mjesta u mreži. Oblik jezgre za otkrivanje je 1 x 1 x (B x (5 + C)). Ovdje je B broj ograničavajućih okvira u kojima ćelija na karti značajke može predvidjeti, "5" predstavlja 4 atributa ograničavajućeg okvira i povjerenje u točnost detekcije objekta (engl. „one object confidence), a C je broj klasa. U YOLO v3 treniranom na COCO skupu, B = 3 i C = 80, tako da je veličina jezgre 1 x 1 x 255. Karta značajki koju je proizvela ovaj kernel ima identičnu visinu i širinu prethodne mape značajki, a atributi detekcije nalazi se duž dubina kao što je gore opisano [19].

Očekujemo da svaka ćelija karte značajki predvidi objekt kroz jedan od njegovih ograničavajućih okvira ako središte objekta padne u prijemno polje te ćelije.

Ovo ima veze s time kako je YOLO treniran, gdje je za otkrivanje bilo kojeg objekta odgovoran samo jedan granični okvir. Prvo mora se utvrditi kojoj od ćelija pripada ovaj granični okvir.

Da bi se to učinilo, podijeli se ulaznu sliku na mrežu dimenzija jednakih vrijednosti konačne mape obilježja. Donji primjer prikazan na slici 33, gdje je ulazna slika 416 x 416, a korak mreže 32. Korak mreže (engl. „stride of the network“) je sloj koji je definiran kao omjer u kojem se smanjuje ulazni podatak. Kao što je ranije istaknuto, dimenzije značajke bit će 13 x 13. Zatim se podijeli ulazna slika na 13 x 13 ćelije.



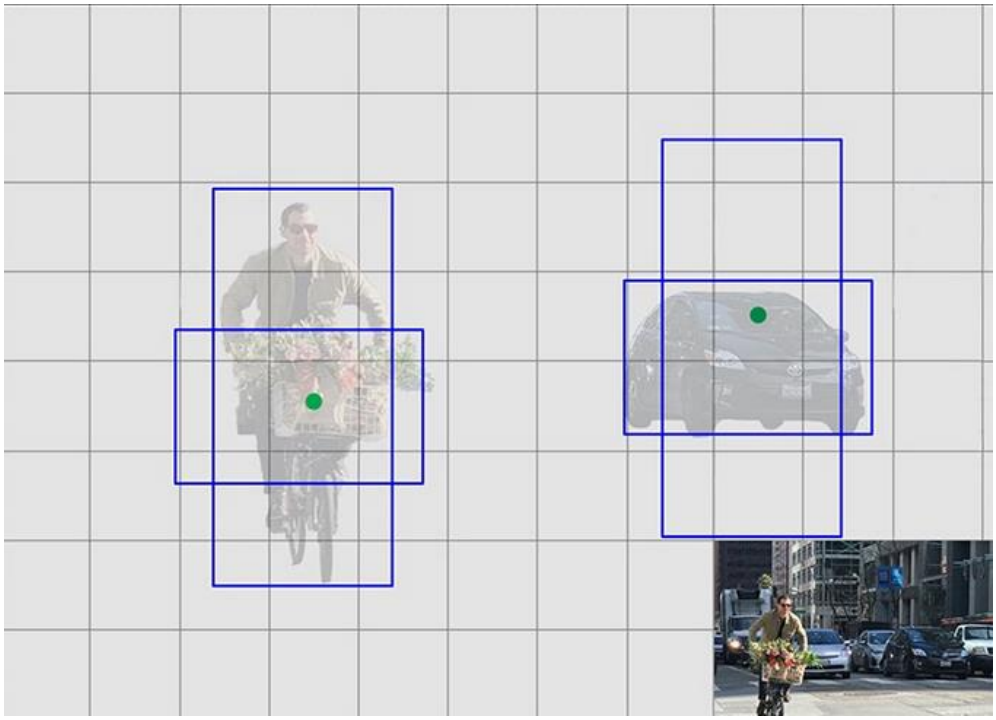
Slika 33[19]: YOLO v3 detekcija na tri skale

Zatim odabrana ćelija (na ulaznoj slici) koja sadrži središte polja prividne istine objekta je odgovorna za predviđanje objekta. Na slici ćelija koja je označena crvenom bojom, a sadrži središte polja istine (žuto).

Sada, crvena ćelija je 7. ćelija u 7. redu na mreži. Sada dodaje se 7. ćeliju u 7. redu na karti značajke (u odgovarajuću ćeliju na karti značajke) kao onu koja je odgovorna za otkrivanje psa.

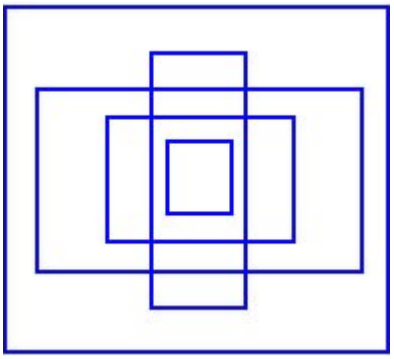
Sada ova ćelija može predvidjeti tri granična okvira. Koji će biti dodijeljen oznaci istine za psa sa slike? Da bi se to razumjelo, mora se shvatiti koncept sidra. Kao što je naznačeno u YOLO znanstvenom radu, rani trening podložan je nestabilnim gradijentima. U početku, YOLO izrađuje proizvoljne nagađanja na graničnim okvirima. Ova bi nagađanja možda dobro djelovala na nekim objektima, a loše za druge što rezultira u naglim promjenama nagiba. U ranom treningu, predviđanja se međusobno bore oko oblika u kojima se posebno treba specijalizirati [17]. U stvarnom životu, granični okviri nisu proizvoljni. Na primjeru prometa, automobili imaju vrlo slične oblike, a pješaci imaju približan omjer 0,41.

Budući da je potrebna samo jedna pretpostavka, početni će trening biti stabilniji ako se započne s raznim nagađanjima koja su uobičajena za predmete iz stvarnog života kao što je prikazano na slici 34.



Slika 34: Više različitih predviđanja

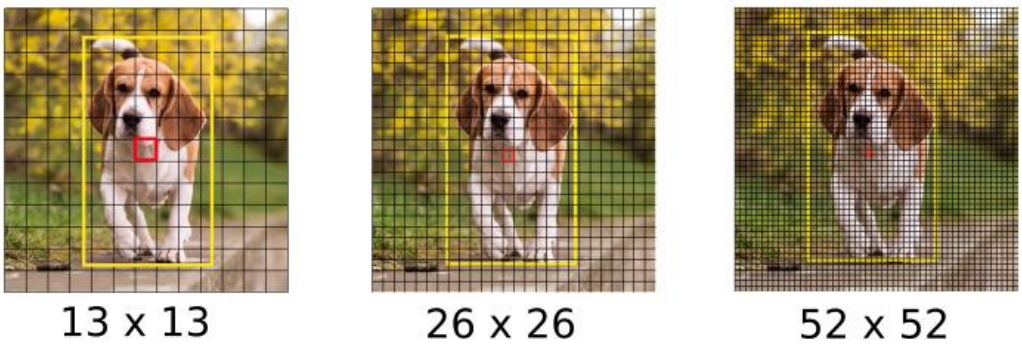
Na primjer, možemo stvoriti 5 okvira za sidro sa sljedećim oblicima kao na slici 35.



Slika 35: 5 okvira sidra za ulazni podatak sa slike 34

YOLO v3 predviđa tri različite skale. Detekcijski sloj koristi se za otkrivanje na mapama značajki tri različite veličine, koje imaju korake 32, 16, 8. To znači, s unosom od 416 x 416, vršimo otkrivanje na ljestvici 13 x 13, 26 x 26 i 52 x 52. Mreža smanjuje uzorke ulazne slike do prvog detekcijskog sloja, gdje se otkrivanje vrši pomoću karakterističnih mapa sloja s korakom 32. Nadalje, slojevi se povećavaju s faktorom 2 i spajaju se s mapama značajki prethodnih slojeva s identičnom mapom značajki veličina. Još jedno otkrivanje vrši se na sloju sa korakom 16. Isti se postupak preklapanja ponavlja i konačna detekcija vrši se na sloju koraka

8. Na svakoj skali svaka ćelija predviđa 3 granične kutije pomoću 3 sidara, što čini ukupni broj upotrjebljenih sidara 9. (Sidra su različita za različite ljestvice) [19] kao što se vidi ina primjeru slike 36.



Slika 36 [19]: YOLOv3 predviđanja na različitim skalama

Prvu detekciju vrši se 82. slojem. Za prvih 81 sloj slika se smanjuje na uzorku od mreže tako da 81. sloj ima korak 32. Ako imamo sliku 416 x 416, rezultirajuća mapa značajki bila bi veličine 13 x 13. Jedna detekcija je napravljena ovdje pomoću 1 x 1 detekcijske jezgre, što nam omogućava mapu značajke 13 x 13 x 255. Zatim se karta značajki iz sloja 79 podvrgne nekoliko kovolucijskih slojeva prije nego što se uzme uzorak za 2x do dimenzija 26 x 26. Ta se značajka zatim dubina povezuje s kartom značajki iz sloja 61. Zatim se kombinirane mape značajki ponovo podvrgnuo nekoliko slojeva 1 x 1 kako bi spojio značajke ranijeg sloja (61). Zatim, druga detekcija izrađuje 94. sloj, čime se dobiva karta značajki detekcije 26 x 26 x 255.

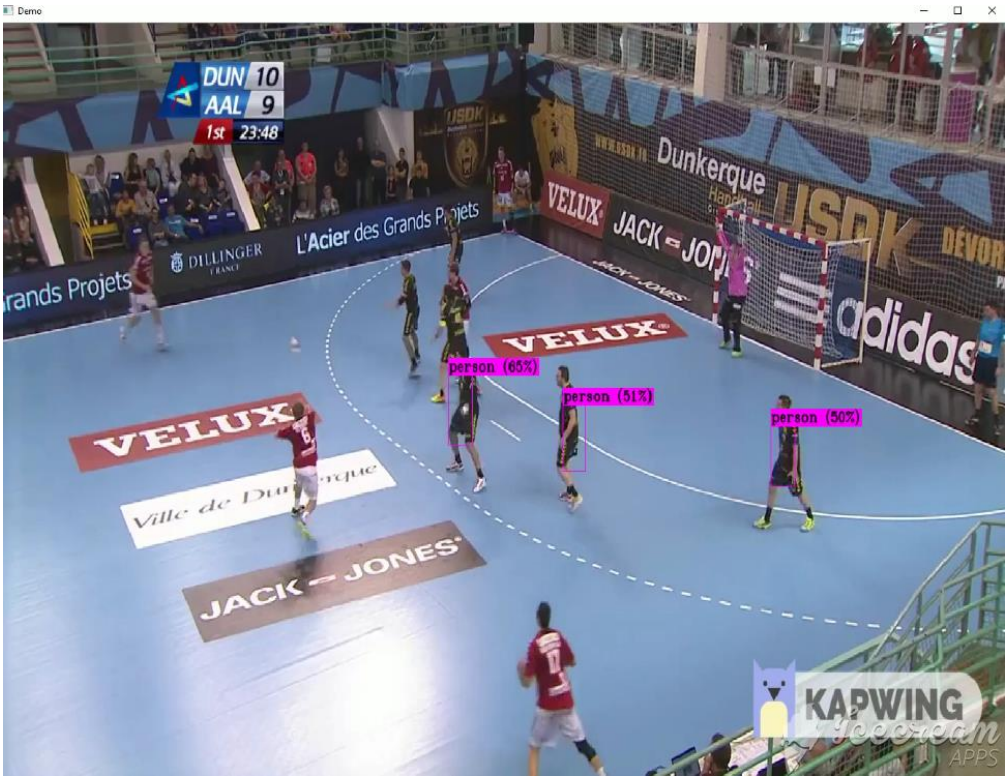
Ponovno slijedi sličan postupak, gdje je mapa značajki iz sloja 91 podvrgnuta nekoliko slojevitih slojeva prije nego što se dubina konkatenizira s kartom značajki iz sloja 36. Kao i prije, nekoliko slojeva 1 x 1 slijedi kako bi se osigurali podaci iz prethodnog sloj (36). Izvršimo završni sloj 3 na 106., čime se dobiva karakteristična mapa veličine 52 x 52 x 255.

Detekcija na različitim slojevima pomaže u rješavanju problema otkrivanja malih objekata, što je česta pritužba kod YOLO v2. Slojevi spojeni s prethodnim slojevima u slučaju YOLO v3 pomažu u očuvanju sitnozrnatih značajki koje pomažu u otkrivanju sitnih objekata.

Sloj 13 x 13 odgovoran je za otkrivanje velikih predmeta, dok sloj 52 x 52 otkriva manje objekte, dok sloj 26 x 26 otkriva srednje objekte. Ovdje je usporedna analiza različitih objekata koji su u istom objektu pokupili različiti slojevi.

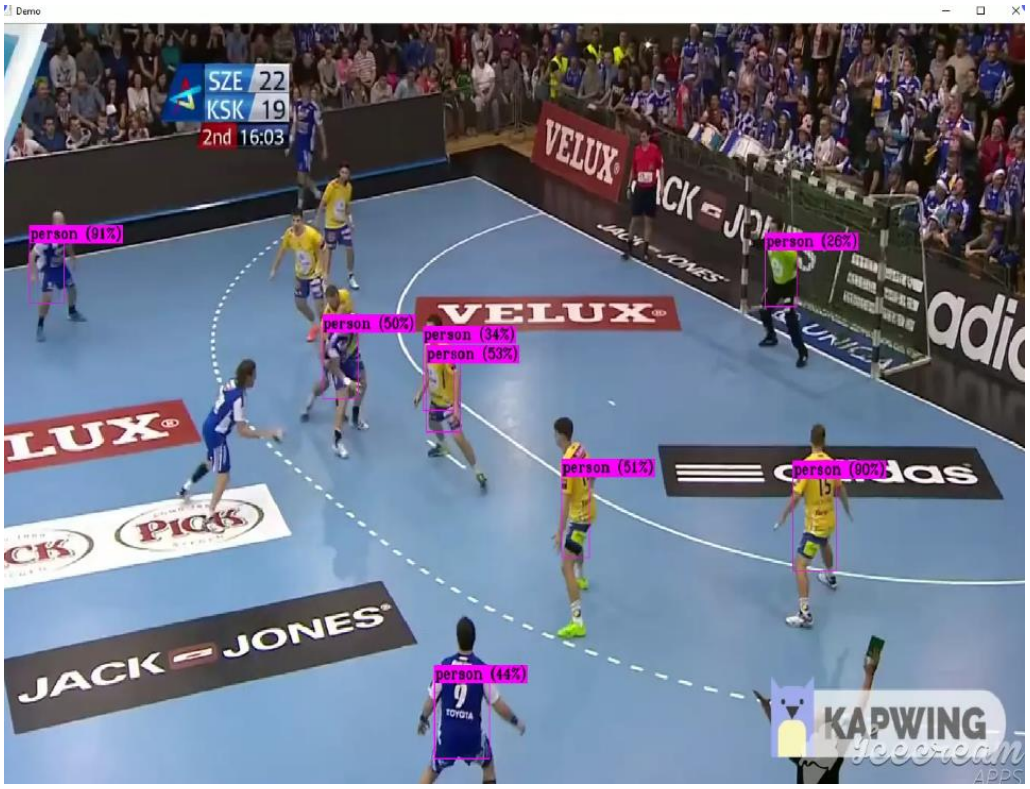
YOLO v3 korišten je kao referentni primjer prikaza rada za cijelu kategoriju YOLO metode detekcije objekata.

Slike prikazane na sljedećim primjerima izrezane su iz videa koji je priložen uz ovaj diplomski rad. Pri reprodukciji videa vidi se stvara brzina i točnost YOLO v3 algoritma, dok su slike prikazane u ovom diplomskom radu najbolji pokušaj prikaza rada algoritma na jednom frame-u.



Slika 37: YOLOv3 detekcija rukometnog razigravanja pomoću metode logističke regresije

Slika 38 : YOLOv3 detekcija rukometaša u profilu pomoću metode logističke regresije



Slika 39: YOLOv3 detekcija rukometnog razigravanja 2 pomoću metode logističke regresije

Slika 40: YOLOv3 detekcija rukometaša u skoku prilikom zamućenog frame-a pomoću metode logističke regresije

3.4. RetinaNet

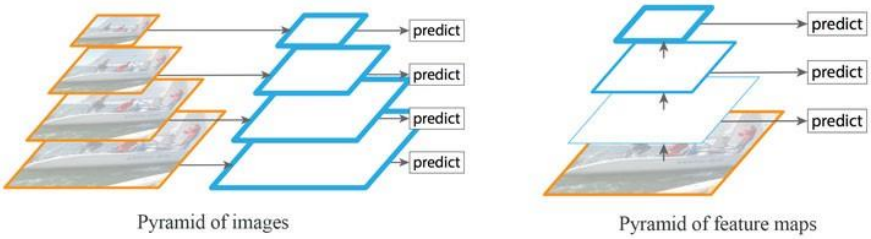
U konvencionalnim detektorima objekata, recimo, R-CNN, u početku se generira skup lokacija objekata, a zatim se ta mjesta klasificiraju pripadaju li prednjem ili pozadinskom razredu koristeći CNN. To radi dvofazni detektor. U slučaju jednofaznih detektora poput SSD-a, točnost je veća kada se primjenjuje na gustom uzorkovanju lokacija objekata, razmjera i omjera. Jednofazni detektori generiraju veliki skup lokacija objekata koji gusto prekrivaju nekoliko područja slike. To stvara neravnotežu klase jer se negativi povećavaju i klase objekata prisutne na tim lokacijama ostaju neotkrivene. Kako bi se riješio problem guste detekcije razvijena je RetinaNet od strane Facebook AI Research tima [20].

U RetinaNet-u, jednofaznom detektoru, korištenjem fokusnog gubitka, manji gubitak doprinose "laki" negativni uzorci tako da se gubitak usredotočuje na "tvrde" uzorke, što poboljšava preciznost predviđanja. ResNet + FPN čine okosnicu za vađenje značajki ove metode, te dvije pod-mreže specifične za zadatak za razvrstavanje i regresiju graničnog okvira, formiraju RetinaNet, mrežu koja postiže vrhunske performanse, nadmašuje bržu R-CNN, dobro poznati dvofazni detektor objekata.

Funkcija gubitaka koja se koristi u ovom pristupu je gubitak izlaza klasifikacijske pod-mreže. Taj se gubitak primjenjuje na sva sidra u svakoj uzorkovanoj slici. Ukupni fokusni gubitak slike je zbroj fokusnih gubitaka nad svim sidrima. Normalizacija se vrši na dodijeljenim sidrima, a ne na ukupnim sidrima kako bi se izbjegli negativi generirani ukupnim sidrima.
RetinaNet omogućen fokusnim gubitkom djeluje bolje od svih postojećih metoda, snižavajući trend niske točnosti [21].

3.4.1. Mreža piramidi značajki (engl. „Feature Pyramid Networks“, skraćeno FPN)

Detekcija objekata u različitim razmjerima je izazovno posebno kod malih predmeta. Možemo koristiti piramidu iste slike u različitim mjerilima za otkrivanje objekata (lijevi dio potonje slike). Međutim, obrada slika s više razmjera oduzima puno vremena i potražnja za memorijom je previsoka da bi se istovremeno trenirala s kraja na kraj. Stoga bi se mogla koristiti samo u zaključivanju kako bis se povećala točnost što je moguće više, kada brzina nije presudni faktor. Alternativno, stvaramo piramidu značajki (slika 41) i koristimo ih za otkrivanje objekata (desni dio potonje slike). No, prikazujte značajke bliže sloju slike sastavljenom od struktura niske razine koje nisu učinkovite za precizno otkrivanje objekta.



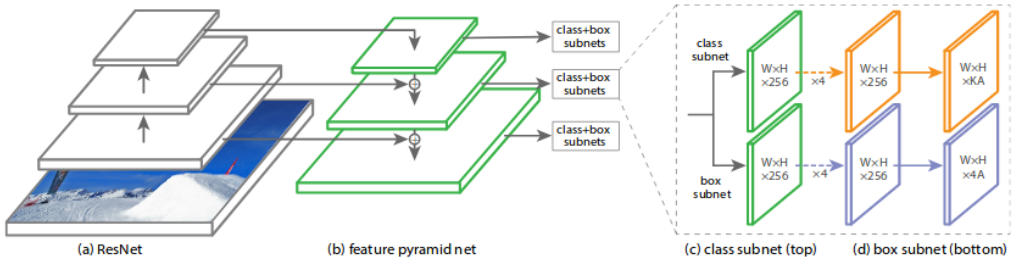
Slika 41: Mreža piramidi značajki

Mreža piramidi značajki (FPN) je ekstraktor značajki dizajniran za takav koncept piramida s točnošću i brzinom u vidu. Zamjenjuje ekstraktor značajki detektora poput onog kod brže R-CNN-e i generira više slojeva mapa značajki s kvalitetnijim informacijama od uobičajenih piramida značajki za otkrivanje objekata [22].

3.4.2. Rezidualna neuronska mreža

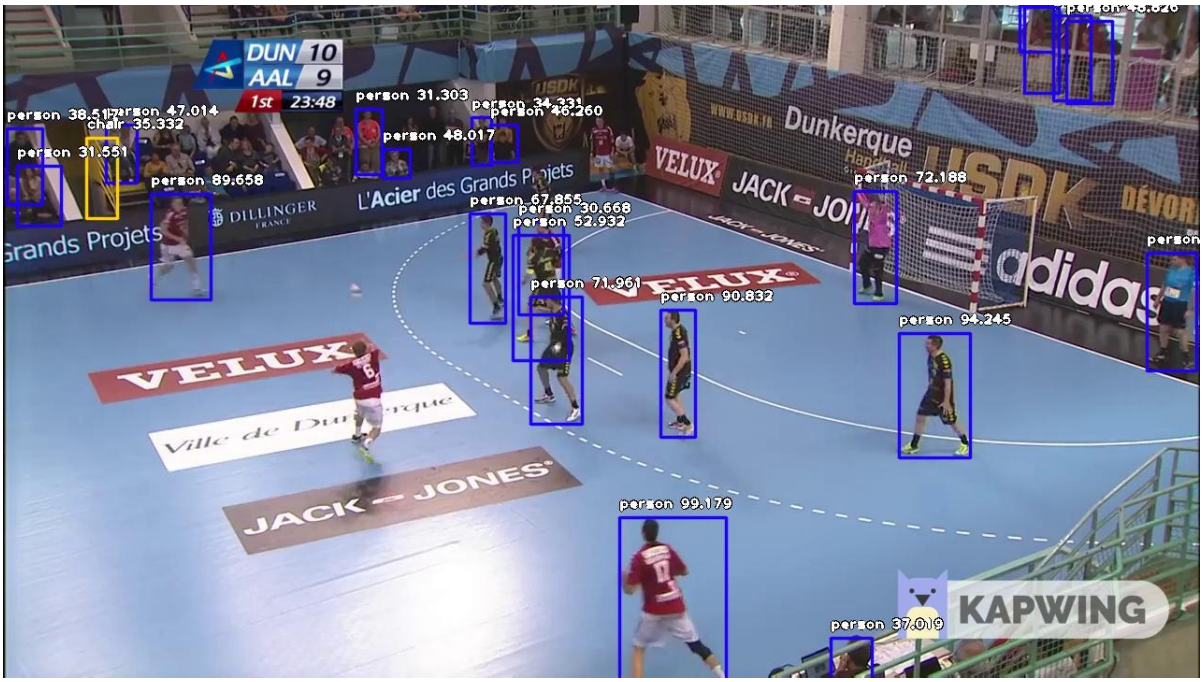
Rezidualna neuronska mreža ((engl. „Residual neural network“, skraćeno ResNet) je umjetna neuronska mreža (ANN) vrste koja se temelji na konstrukcijama poznatim iz piramidalnih stanica u moždanoj kore. Preostale neuronske mreže to čine pomoću preskočenih veza ili prečaca da biste preskočili neke slojeve. Tipični ResNet modeli implementirani su s dvoslojnim- ili troslojnim preskocima koji sadrže nelinearnosti (ReLU) i normalizaciju serije između njih. Grafički prikaz ResNet-a prikazan je na slici 42.
Jedna motivacija za preskakanje slojeva je izbjegavanje problema nestajanja gradijenata ponovnom uporabom aktivacija iz prethodnog sloja sve dok susjedni sloj ne nauči težinu. Tijekom treninga, utezi se prilagođavaju kako bi utišali uzlazni sloj i pojačali prethodno preskočeni sloj. U najjednostavnijem slučaju, prilagođavaju se samo utezi za vezu susjednog sloja, bez izričite težine za gornji sloj. Ovo najbolje uspijeva kada se prelazi jedan nelinearni sloj ili kada su međufazni slojevi linearni. Preskakanje učinkovito pojednostavljuje mrežu, koristeći manje slojeva u početnim fazama treninga. Ovo ubrzava učenje smanjujući utjecaj nestajućih gradijenata, jer ima manje slojeva za širenje kroz njih. Mreža zatim postupno obnavlja preskočene slojeve kako uči prostor s značajkama. Pred kraj treninga, kada se svi slojevi šire, ostaje bliže razdjelniku i na taj način brže uči. Neuronska mreža bez zaostalih dijelova istražuje više prostora s značajkama. To ju čini osjetljivijom na uznemirenosti zbog kojih napušta razdjelnik i zahtjeva dodatne trening podatke kako bi se oporavila. [23]

RetinaNet je jedinstvena, objedinjena mreža sastavljena od mrežne okosnice i dviju pod-mreža specifičnih za određeni zadatak. Okosnica je odgovorna za izračunavanje mape značajki konvolucije preko cijele ulazne slike i to ju čini samostalnom konvolucijskom mrežom. Prva pod-mreža vrši klasifikaciju na izlazu okosnice; druga pod-mreža vrši regresiju konvolucije granične kutije [24].



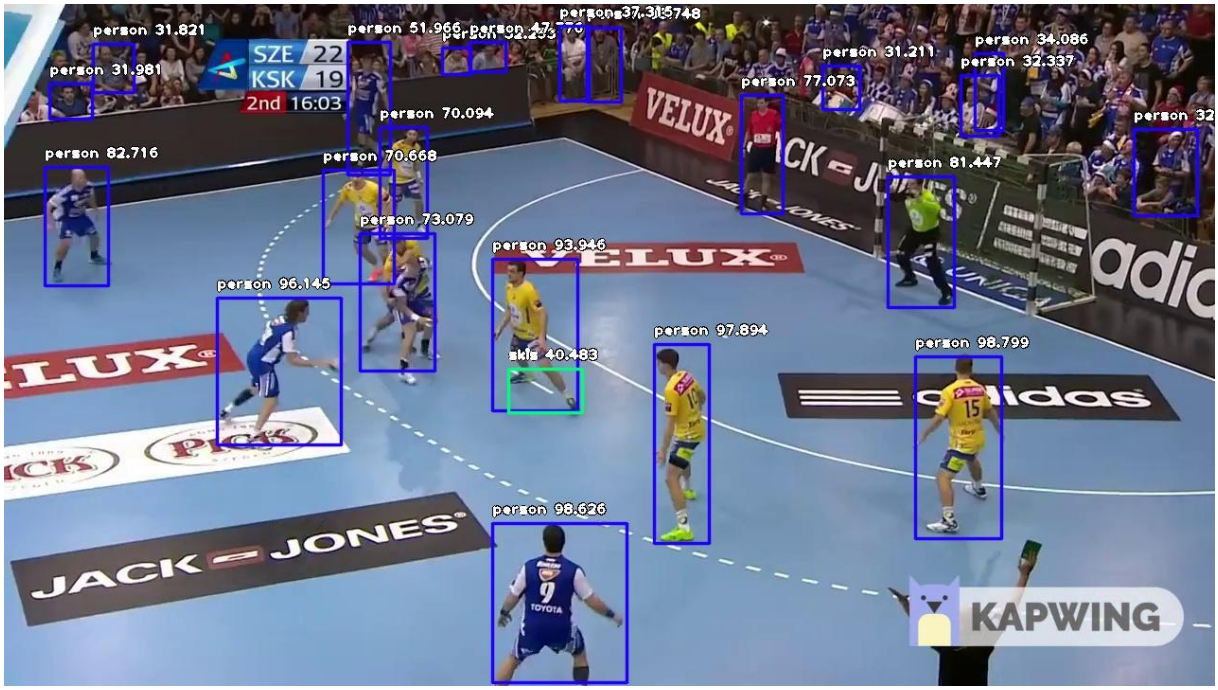
Slika 42: Arhitektura ResNet-e

Slike prikazane na sljedećim primjerima izrezane su iz videa koji je priložen uz ovaj diplomski rad. Pri reprodukciji videa vidi se stvara brzina i točnost RetinaNet algoritma, dok su slike prikazane u ovom diplomskom radu najbolji pokušaj prikaza rada algoritma na jednom frame-u.

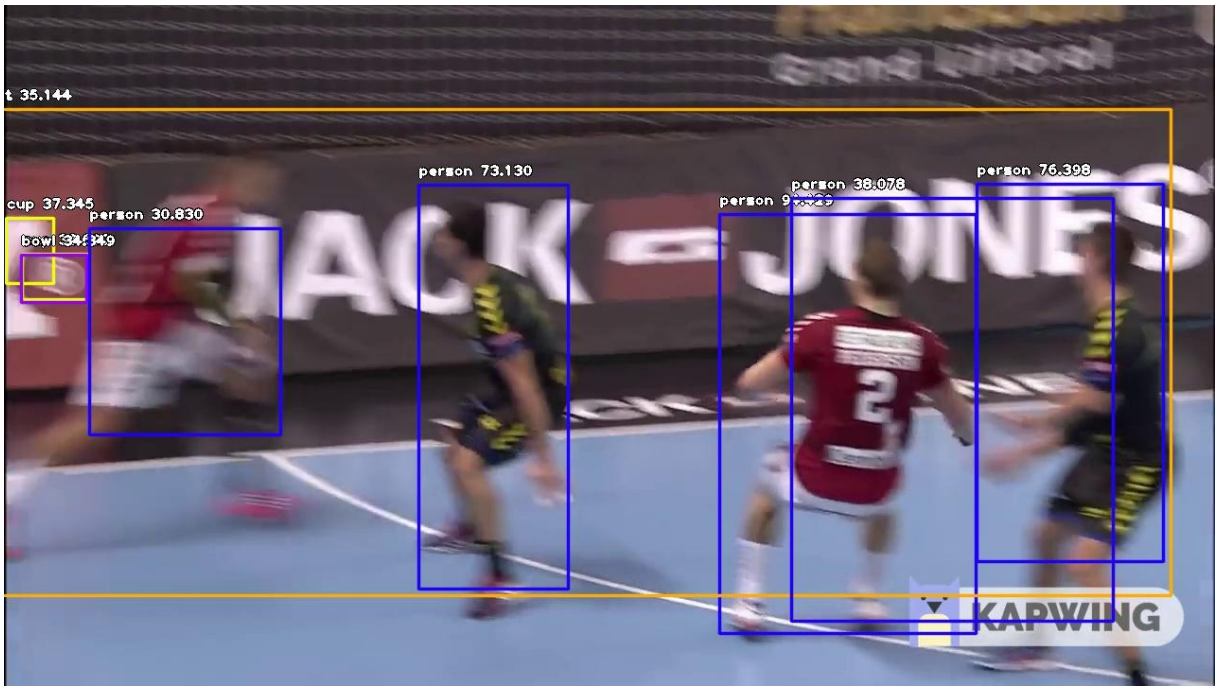


Slika 43: RetinaNet detekcija rukometnog razigravanja

Slika 44: RetinaNet detekcija rukometaša u profilu



Slika 45: RetinaNet detekcija rukometnog razigravanja 2



Slika 46: RetinaNet detekcija rukometaša u skoku prilikom zamućenog frame-a

3.5. Usporedba metoda (arhitektura) za detekciju objekata

Vrlo je teško uspostaviti usporedbu različitih detektora objekata. Ne postoji izravan odgovor koji je model najbolji. Za aplikacije u stvarnom životu donosimo odluke za uravnoteženje točnosti i brzine. Osim vrsta detektora, mora se biti svjesno kako postoje mnogi drugi atributi koji utječu na performanse kao što su: izlučivanje značajki, izlazni koraci za ekstraktor (engl. „output strides of the extractor“), rezolucija slike, odgovarajuća strategija i prag IoU, broj prijedloga ili predviđanja, povećavanje podataka treniranje seta podataka i tako dalje.

U nastavku slijedi sažeta tablica u kojoj su ocjenjene metode po sljedećim parametrima:

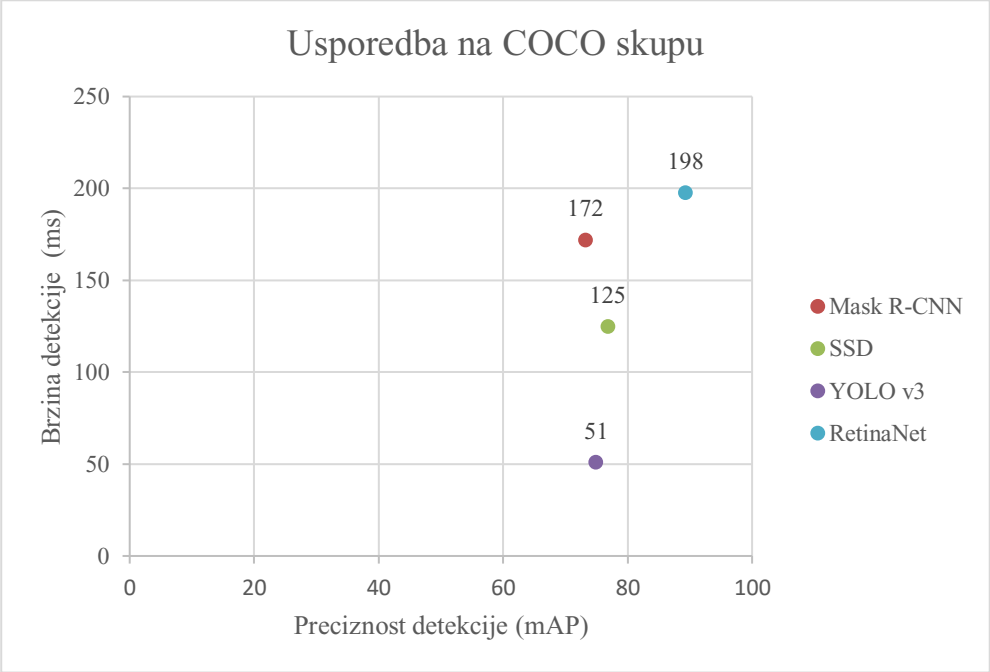
- Brzina detekecije – koliko metodi treba da raspozna objekte detekcije. Mjerena u milisekundama (ms)
- Preciznost detekcije – koliko metoda točno raspoznaje objekte detekcije, mjerena kao srednja prosječna preciznost detekcije (mAP – engl. Mean Average Precision)
- Kompleknost – koliko je metoda napredna za razumijevanje i implementaciju
- FPS(min) – koliko najmanje sličica u sekundi postiže detektor
- FPS(max) – koliko najviše sličica u sekundi postiže detektor

Korišteni set podataka (dataset) je COCO set podataka pošto je on najnovije ažuriran (2015. godina)

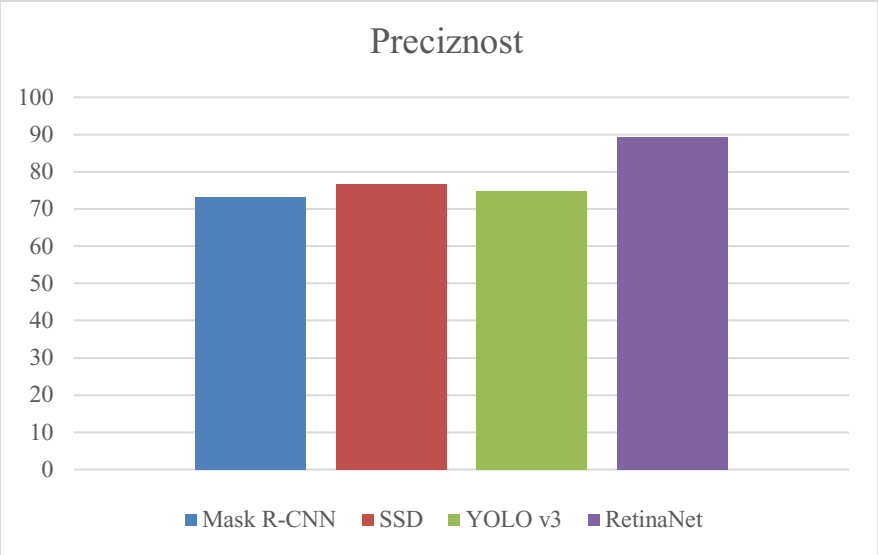
Slijedi tablica usporedbe na COCO skupu podataka:

Tablica 1: COCO dataset usporedba arhitektura

Arhitektura	Brzina detekcije	Preciznost detekcije	Kompleksnost	FPS(min)	FPS(max)
Mask R-CNN	172 ms	73.2 mAP	Visoka	5	17
SSD	125 ms	76.8 mAP	Niska	22	59
YOLO v3	51 ms	74.9 mAP	Umjerena	40	91
RetinaNet	198 ms	89.3 mAP	Visoka		



Slike 47: Usporedba preciznosti detekcije i brzine detekcije predstavljenih neuronskih mreža



Slike 48: Grafička usporedba preciznosti detekcije predstavljenih neuronskih mreža

Uspoređujući rezultate prikazane u različitim znanstvenim radovima (navedeni u poglavlju popis literature i izvora) dobio se uvid u točnost i brzinu metoda predstavljenih u ovom radu. Najpreciznijom metodom uvjerljivo pokazala se RetinaNet, dok najbržom YOLOv3. Najjednostavnijom za korištenje i implementaciju pokazala se metoda SSD. Mask R-CNN nije odnijela „pobjedu“ u nijednoj kategoriji, no bitno je za naglasiti kako je Mask R-CNN jedina metoda od navedenih koja koristi dvofazni detektor, dok ostale metode koriste jednofazni detektor.

Upravo jer je najjednostavnija za implementaciju, varijacija metode SSD zvana MobileNetV2 korištena je u sljedećem poglavlju u Android aplikaciji za prepoznavanje pokreta.

4. Android aplikacija za prepoznavanje poze/pokreta (sportaša)

U sklopu praktičnog dijela diplomskog rada napravljena je Android aplikacija za prepoznavanje pokreta – pokreti ljudskog tijela ovdje su ekstenzija objekta odnosno objekata te se različiti dijelovi tijela (vrat, lijeva noga, desna noga, lijeva ruka, desna ruka, lakat, zapešća, ramena) mogu klasificirati kao objekti neke veće klase, u ovom slučaju osobe.

Za potrebe ovog projekta bilo je potrebno instalirati Android Studio, službeno integrirano razvojno sučelje (IDE) za Google-ov operativni sustav Android, koji je izgrađen na JetBrains-ovom IntelliJ IDEA softveru i dizajniran isključivo za razvoj Android aplikacija. Također potrebno je bilo instalirati i dodatne ovisne nadogradnje kao što je ustav Gradle. Gradle je alat za automatizaciju koji se često koristi za JVM („Java Virtual Machine“ - virtualna mašina koja može izvršavati Java kompajlirani byte kod) jezike kao što su Java, Groovy ili Scala. Gradle se

može konfigurirati za pokretanje zadataka koji rade stvari poput sastavljanja JAR datoteka („Java Runtime Environment“), pokretanja testova, izrade dokumentacije i još mnogo toga

Također, kako bi se postojeći projekt detekcije pokreta mogao implementirati unutar novog projekta, bilo je potrebno preuzeti i instalirati biblioteke unutar Android Studio-a: NDK, SDK-buildTools, SDK Platform-Tools. Android NDK je skup alata koji omogućuje implementaciju dijelova aplikacije u izvornom kodu koristeći jezike poput C i C ++. Android SDK (skraćeno od engl. „Software Development Kit“ – komplet za razvoj softvera) skup je razvojnih alata koji se koriste za razvoj aplikacija za Android platformu.

Frontend kod pisan je u XML-u jer je to trenutno najpodržaniji mark-up jezik za Android.

Backend kod pisan je U Javi i u Kotlinu[2] po osobnoj preferenciji autora.

Set podataka za treniranje je inicijalno preuzet od AI Challenger [25] no za potrebe ovog rada transformiran je u COCO format te se sastoji od slijedećih primjera:

- 22446 trening primjeraka
- 1500 test primjeraka

Model korišten u aplikaciji znan je kao CPM koristeći biblioteku TensorFlow. CPM se sastoji od niza kovolucijskih mreža koje više puta proizvode 2D mape vjerovanja za položaj svakog dijela. U svakoj fazi CPM-a kao značajke ulaza koriste se značajke slike i karte vjerovanja proizvedena u prethodnoj fazi. Karte vjerovanja pružaju narednoj fazi ekspresivno neparametarsko kodiranje prostorne nesigurnosti lokacije za svaki dio, omogućujući CPM-u da nauči bogate prostorne modele odnosa između dijelova koji ovise o slici.

Umjesto klasične konvolucije, unutar modela se koristi invertirani rezidualni modul (također poznat kao Mobilenet V2 [26]) za zaključivanje u stvarnom vremenu.

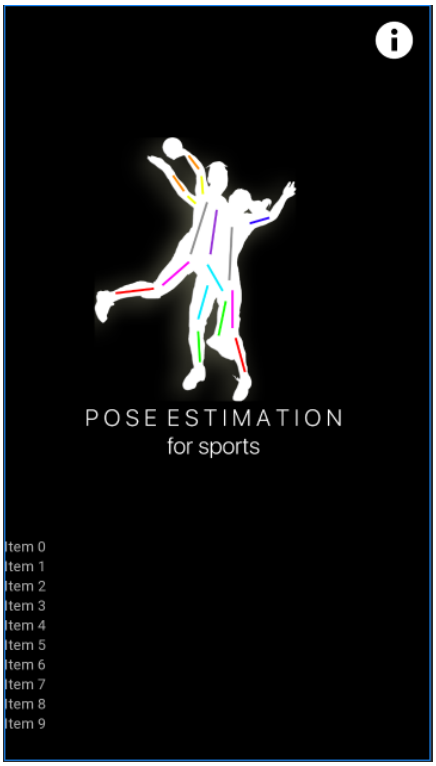
Kako je model bio unaprijed istreniran za prepoznavanje poza (model.tflite), model nije dodatno treniran pa se fokus prebacuje na izradu sučelja i implementaciju detektora u stvarnom vremenu.

Sam izgled aplikacije je napravljen sa naglaskom na lakoću i brzinu responzivnosti kako hardverske komponente smartphone-a ne bi utjecale na rad aplikacije.

[2] Kotlin je otvoreni programski jezik opće namjene, statički tipkani „pragmatični“ programski jezik za JVM i Android koji kombinira objektno orijentirane i funkcionalne značajke programiranja. Usredotočena je na interoperabilnost, sigurnost, jasnoću i podršku alata

4.1. Prvi modul – Korisničko sučelje

Prilikom ulaska u aplikaciju prva aktivnost predstavlja ujedno i glavnu aktivnost aplikacije na kojoj su prikazani ime aplikacije i njen logo.

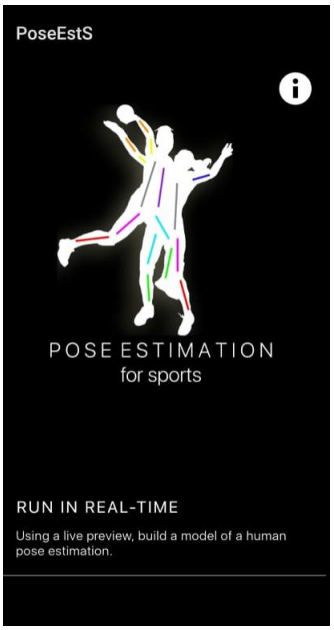


Slika 49: Izgled naslovne aktivnosti aplikacije u Android Studio-u (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="254dp"
        android:layout_height="403dp"
        android:layout_marginBottom="@android:dimen/thumbnail_height"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.496"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.625"
        app:srcCompat="@drawable/logo2" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/demo_list_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:scrollbars="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

    <Button
        android:id="@+id/button"
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/info"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:onClick="OpenInfo"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Slika 50: Detalji u XML formatu za naslovnu aktivnost (activity_main.xml)

Prazan prostor ispod imena i loga aplikacije pušten je iz razloga što na donjem dijelu ekrana se generira demo adapter prilikom pokretanja aplikacije. Demo adapter je postavljen iz razloga možebitne nadogradnje aplikacije, gdje će se pomoću adaptera po potrebi dodavati dodatne funkcionalnosti. Demo adapter na smartphone-u izgleda kao na slici:



Slika 51: Izgled naslovne aktivnosti aplikacije na smartphone-u (activity_main.xml)

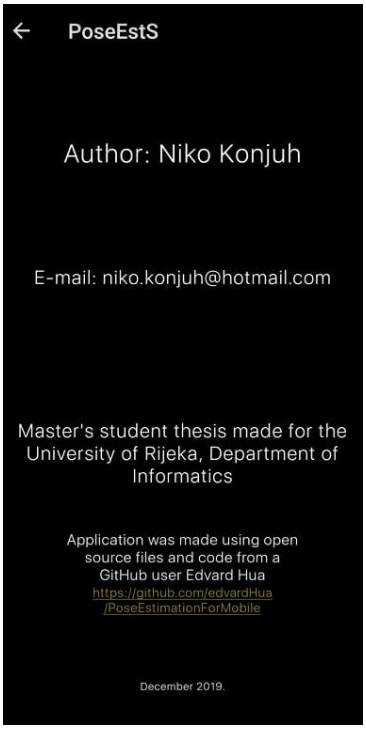
Sa backend strane, kod je prikazan na sljedećim slikama:

```
package com.example.poseests;

import android.content.ComponentName;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.Window;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;
import java.util.List;

import ui.DemoAdapter;
import ui.DemoItem;
import ui.SeparatorDecoration;

public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.demo_list_view);
        recyclerView.setHasFixedSize(true);
        LinearLayoutManager rvLinearLayoutMgr = new LinearLayoutManager( context: this);
        recyclerView.setLayoutManager(rvLinearLayoutMgr);

        // Add a divider
        SeparatorDecoration decoration = new SeparatorDecoration( context: this, Color.GRAY,  heightDp: 1);
        recyclerView.addItemDecoration(decoration);

        // Add the adapter
        DemoAdapter adapter = new DemoAdapter(getDemoItems());
        recyclerView.setAdapter(adapter);
        recyclerView.setClickable(true);
    }

    private List<DemoItem> getDemoItems() {
        List<DemoItem> demoItems = new ArrayList<>();

        demoItems.add(new DemoItem(
                getString(R.string.poseestimation_title),
                getString(R.string.poseestimation_description),
```

Slika 52 i 53: MainActivity.java

Nakon učitavanja biblioteka potrebnih za funkcionalnost i izgled ove aktivnosti unutar aktivnosti deklarira se varijabla *recyclerView* koja je odgovorna za prikaz na kojemu su izlistani sve stavke (u ovom slučaju trenutno samo detekcija pokreta). Unutar metode *onCreate,* koju je potrebno definirati u svakoj aktivnosti, postavlja se kontekst na *activity_main.xml* kako bi se povezale frontend i backend datoteke. Nakon toga povezuje se stvorena *recyclerView* varijabla sa pripadnom vrijednošću. Slijedi inicijalizacija samog djelitelja stavki (engl. „divider“) te postavljanje vrijednosti demo adaptera. U konačnici poziva se funkcija DemoAdapter.java (koja ispunjava estetsku funkcionalnost generiranom adapteru, njegovim stavkama i djelitelju stavki, zajedno sa klasama DemoItem.java i SeperatorDecoration.java) koja se ispunjava vrijednostima naslova same stavke i opisa stavke te ono najbitnije, same „namjere“ (engl. „Intent) te stavke. Intent stavke se prebacuje na modul detekcije pokreta u stvarnom vremenu razvijenim od strane Edvarda Hue. Modul detekcije ima jako puno zavisnih varijabli, biblioteka i datoteka te mu je potrebno oko 10 sekundi da se potpuno inicijalizira.

Pritiskom gumba „i“ u desnom gornjem okviru aplikacije učitava se nova scena koja prikazuje informacije o samoj aplikaciji, ime autora, e-mail autora, svrha izrade aplikacije te izvor i mjesec izrade.



Slika 54: Izgled scene o generalnim informacijama

4.2. Drugi modul – Prepoznavanje poze

Vraćajući se s gumbom nazad, dolazi se na naslovnu scenu gdje nam je preostao gumb (demo adapter) 'Run in real time' (hrv. Pokreni u stvarnom vremenu). Klikom na gumb učitava se drugi modul aplikacije – samo prepoznavanje pokreta koristeći kameru smartphone-a.

Ovaj modul sa backend strane sastavljen je od 8 Kotlin datoteka/aktivnosti: *AutoFitFrameLayout, AutoFitTextureView, Camera2BasicFragment, CameraActivity, DimensionExtensions, DrawView,ImageClassifier, ImageClassifierFloatInception*.

Prva aktivnost koja se inicijalizira pritiskom na gumb 'Run in real time' je *CameraActivity.java.* Veže se sama scena (*activity_camera.xml)* sa klasom te se poziva nova instanca *Camera2BasicFragmet* klase. Unutar te aktivnosti poziva se još jedna nezavisna komponenta *openCVlibrary* sa svojim pripadnim klasama i modulima. OpenCV je open source knjižnica programskih funkcija usmjerenih na računalni vid u stvarnome vremenu. Komponenta *openCVlibrary* omogućuje da se na vrh same kamere „zalijepe“ opcije detekcije pokreta – sami kostur dijelova tijela (objašnjen naknadno u radu), latentnost detekcije izraženo u milisekundama (ms) te legenda prikaza s kojom se može referencirati što točno kostur na zadanom videozapisu točno prepoznaje.

Frontend same aktivnosti kamere je vrlo trivijalan, napravljen je tako da je ubačen samo jedan kontejner unutar kojeg će se prikazivati rezultat kamere u stvarnom vremenu

```
@Suppress( ...names: "DEPRECATION")
class CameraActivity : Activity() {

  private val mLoaderCallback = object : BaseLoaderCallback( AppContext: this) {
    override fun onManagerConnected(status: Int) {
      when (status) {
        LoaderCallbackInterface.SUCCESS -> isOpenCVInit = true
        LoaderCallbackInterface.INCOMPATIBLE_MANAGER_VERSION -> {
        }
        LoaderCallbackInterface.INIT_FAILED -> {
        }
        LoaderCallbackInterface.INSTALL_CANCELED -> {
        }
        LoaderCallbackInterface.MARKET_ERROR -> {
        }
        else -> {
          super.onManagerConnected(status)
        }
      }
    }
  }

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_camera)
    if (null == savedInstanceState) {
      fragmentManager
          .beginTransaction()
          .replace(R.id.container, Camera2BasicFragment.newInstance())
          .commit()
    }

  }

  override fun onResume() {
    super.onResume()
    if (!OpenCVLoader.initDebug()) {
      OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION, AppContext: this, mLoaderCallback)
    } else {
      mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS)
    }
  }

  companion object {

    init {
      //        System.loadLibrary("opencv_java");
      System.loadLibrary( libname: "opencv_java3")
    }
  }
```

CameraActivity

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000"/>
```

Slike 55 i 56: Backend (CameraActivity.kt) i frontend (*activity_camera.xml)*aktivnosti kamere

Camera2BasicFragment.kt je aktivnost unutar koje se sve ostale aktivnosti povezuju i prožimaju kako bi se definirale i povezale sve potrebne varijable za detekciju. Također, bitno je naglasiti kako su ovdje definirane metode ponašanja stanja kamere, kad je otvorena, kad je zatvorena, kad je isključena kad je posrijedi neka greška. Kako bi kamera funkcionirala na smartphone-u potrebno je definirati sva moguća stanja koja se mogu dogoditi.

Nadalje, napravljene se manje optimizacije kako bi se detekcija mogla izvršavati sa manje napora na hardver pametnog telefona.

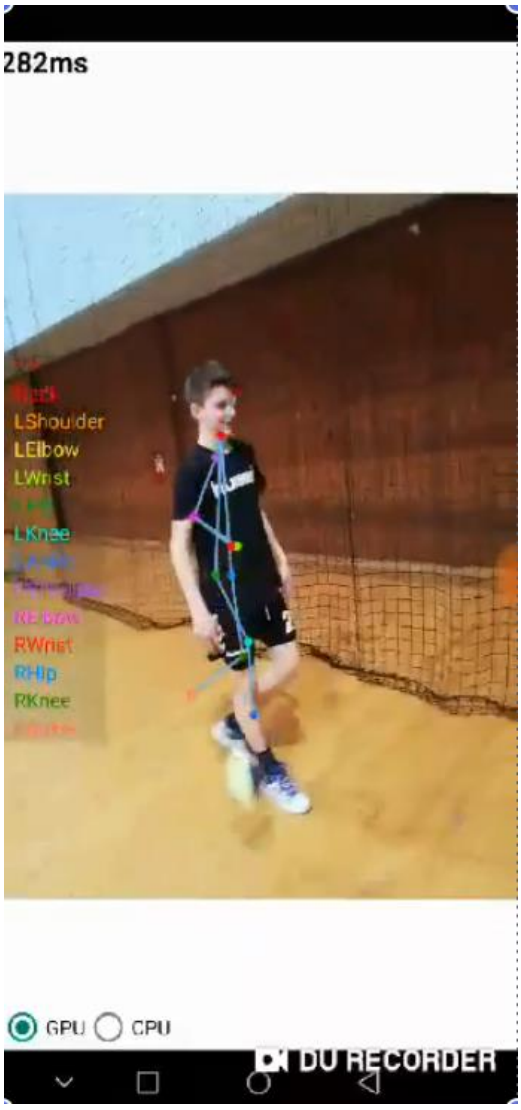
Nakon toga slijedi isječak koda koji se fokusira na povezivanje okvira, pogleda i ostalih komponenti koje se pozivaju za vrijeme detekcije.

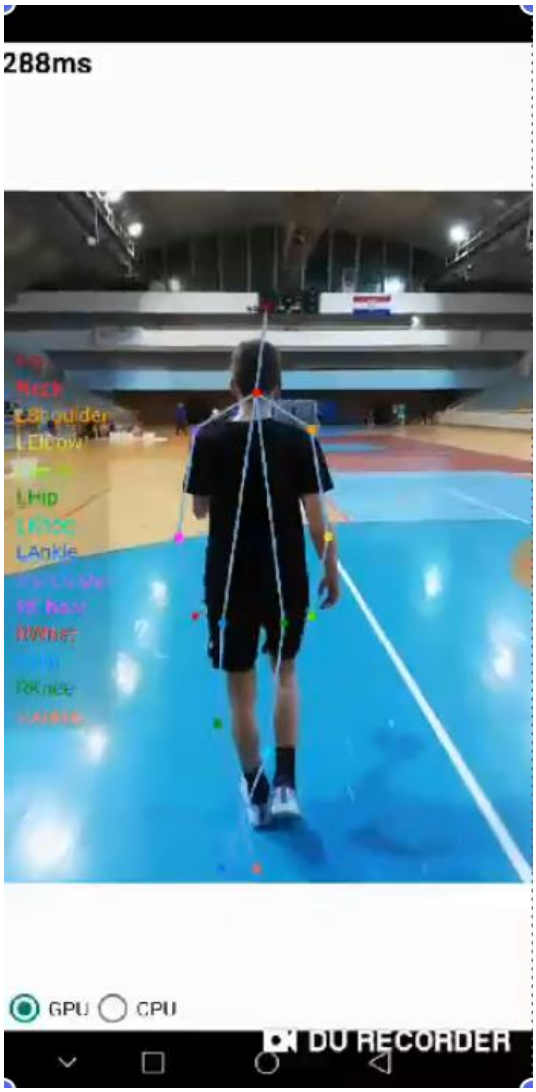
Slijedi najbitniji dio ove aktivnosti a to unutar funkcije *onActivityCreated()* gdje se učitavaju model i pripadne etikete. Također, funkcija *onResume()* radi tako da ukoliko se ugasi ekran pametnog telefona, prilikom ponovnog pokretanja model će biti već prethodno učitan i moći će se odmah raditi detekcija. Funkcija *onPause()* gasi trenutno otvorenu kameru i pauzira detekciju. Funkcija *onDestroy()* gasi detekciju i „uništava“ trenutno otvoren pregled kamere.

Nadalje u ovoj aktivnost definiraju se i povezuju svi potrebni parametri kako bi se detekcija napravila kao što se to očekuje, pa tako, na primjer, dodaju se karakteristike da se koristi samo

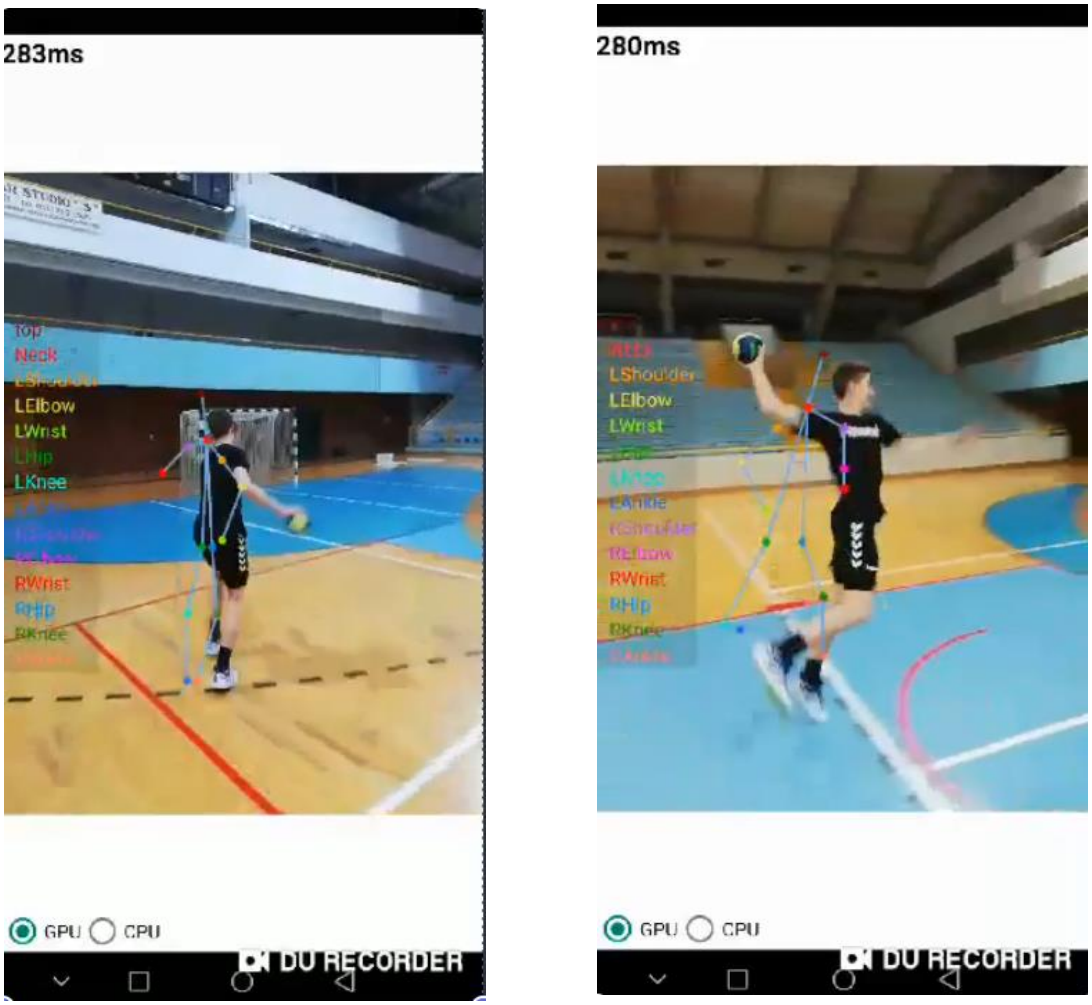
zadnja kamera smartphone-a, definiraju se dimenzije okvira, postavljaju se svi mogući slučajevi greške i slično.

U konačnici, detekcija na mladom sportašu (rukometašu) izgleda:




288ms
LHip
LAnkle
GPU
CPU
DU RECORDER


261ms
LShoulder
LHip
GPU
CPU
DU RECORDER



Slike 57,58,59,60,61,62: Prikaz rada detekcije u stvarnom vremenu

Slike prikazane u diplomskom radu su referentnog sadržaja te ukoliko se želi vidjeti stvaran rad aplikacije, treba se pokrenuti priloženi video *Videozapis rada aplikacije PoseEstS.mp4*

4.3. Završne riječi vezane uz aplikaciju

Detaljna analiza koda nije uključena u rad iz razloga što je sam model detekcije pokreta preuzet od vanjskog izvora. Naglasak je stavljen na procese koji se pozivaju za rad same detekcije.

Aplikacija je predstavlja inicijalni korak za izgradnju projekta koji bi u sadržavao opciju poboljšanja poza sportaša pri određenim akcijama i pokretima, posebice mladim sportašima i sportašima u razvoju.

Projekt je zamišljen po slijedećim koracima:

1. Prilikom pokretanja odabire hoće li se koristiti postojeći video zapis (iz lokalnog ili web direktorija)
2. Na odabrani video se aplicira detekcija pokreta, kostur detekcije se sprema eksterno

3. Kostur spremljene detekcije se uspoređuje po matematičkim parametrima sa kosturom koje detekcija ima u svojoj biblioteci (lokalno ili web), a to su kosturi profesionalnih sportaša određenog sporta
4. Aplikacija nakon usporedbe ispiše što sve na inicijalnom videu treba promijenit u pozi, na primjer: podignuti lijevi lakat za 10%, podignuti desno koljeno prilikom trčanja za 15%, ispraviti leđa prilikom skoka i slično

5. Zaključak

Premda se algoritmi za detekciju objekata ubrzano razvijaju i posljednjih godina su doživjeli značajna poboljšanja, problem očito još nije u potpunosti riješen. Razvijene su različite metode od kojih su ovdje predstavljene metode regionalnih prijedloga (regionalne konvolucijska neuronska mreža, brze regionalne konvolucijske mreže, brže regionalne konvolucijske mreže, maskirane regionalne konvolucijske mreže), Detektor s višestrukim snimkama (SSD) i njegova poboljšana arhitektura za mobilne modele zvana MobileNet V2, metoda YOLO (YOLO v1, YOLO v2 i YOLO v3 ili 9000), RetinaNet

Svi razvijeni modeli imaju svoje prednosti i nedostatke, tako da nema dominantnog modela koji univerzalno dobro radi. Modeli su naučeni na velikom skupu slika ali rezultati koje daju kada se testiraju na novim slikama ovise o tome koliko su te slike bliske onima na kojima je model učen. Srećom postoji mnogo biblioteka i dostupnim implementacija tako se može odabrati metoda koja najbolje odgovara danim zahtjevima, u ovom slučaju je to bila detekciji rukometaša te detekciji posture prilikom izvođenja rukometnih akcije. Izrađena aplikacija koristi dostupne modele za detekciju posture igrača bez dodatnog treniranja.

Vizualno najbolje i najatraktivnije rezultate na videu dao je model Mask R-CNN-najpreciznije prepoznajući objekte na svakom frame-u. Mask R-CNN posebit je u ondosu na sve ostale opisane i istražene modele upravo zbog svoje mogućnosti stvaranja segmentacijske maske, koja, iako ispunjava funkciju graničnog okvira, pruža znatno točniju detekciju i samim time veću numeričku sigurnost.

Za detekciju posture korišten je „Bottoms Up Approach“ algoritam. „Bottoms Up“ uključuju prvo otkrivanje dijelova ili zglobova za jednog ili više ljudi na slici, a zatim sastavljanje dijelova zajedno i povezivanje s određenim čovjekom. Jednostavnije rečeno, algoritam prvo predviđa sve dijelove tijela / zglobove koji se nalaze na slici. Obično slijedi formulacija kostura, temeljenog na modelu tijela, koji povezuje zglobove čovjeka.

Također detekcija objekata kakvu danas poznajemo se tek počela razvijati i algoritmi će s vremenom postajat sve precizniji i sve brži, do trenutka kada kašnjenje rezultata detekcije u usporedbi sa originalnim ulaznim podatkom ne bude 0 milisekundi, a sam detektor bude stopostotno siguran u detekcije koje prepoznaje.

Popis literature i izvora

[1] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network. [Accessed 14. 11. 2019.].

[2] K. H. R. G. J. S. Shaoqing Ren, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Beijing, 2015.

[3] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Neocognitron. [Accessed 14. 11. 2019.].

[4] L. B. Y. B. P. H. Yann LeCun, "Gradient-Based Learning Applied to Document Recognition," 1998.

[5] C. C. C. J. B. Yann LeCun, "THE MINST DATABASE of handwritten digits," [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed 15. 12. 2019.].

[6] l. S. G. E. H. Alex Krizhevsky, "ImageNet Classification with Deep Convolutional Neural Networks," Toronto , 2012.

[7] Q. C. S. Y. Min Lin, Network In Network, 2013.

[8] A. K. J.Johnson, "Convolutional Neural Networks," [Online]. Available: https://cs231n.github.io/convolutional-networks/.

[9] Z. W. J. K. L. M. A. S. B. S. T. L. X. W. G. W. J. C. T. C. Jiuxiang Gua, "Recent advances in convolutional neural networks," Singapore, 2016.

[10] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Object_detection. [Accessed 17. 11. 2019.].

[11] R. Girshick, "Information Engineering," [Online]. Available: http://www.robots.ox.ac.uk/~tvg/publications/talks/fast-rcnn-slides.pdf. [Accessed 15. 12. 2019.].

[12] P. Sharma, "Analytics Vidhya," [Online]. Available: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/. [Accessed 15. 12. 2019.].

[13] J. D. T. D. J. M. Ross Girshick, "Rich feature hierarchies for accurate object detection and semantic segmentation," Berkeley, 2013.

[14] R. Girshick, "Fast R-CNN," Berkeley, 2015.

[15] P. Sharma, "Analytics Vidhya," [Online]. Available: https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/. [Accessed 15. 12. 2019.].

[16] D. A. D. E. C. S. S. R. C.-Y. F. A. C. B. Wei Liu, "SSD: Single Shot MultiBox Detector," Chapel Hill, 2015.

[17] S. D. R. G. A. F. Joseph Redmon, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[18] "Towards Data Science," 4. 2018.. [Online]. Available: https://towardsdatascience.com/review-yolov2-yolo9000-you-only-look-once-object-detection-7883d2b02a65. [Accessed 2019. 11. 22.].

[19] P. Lessons, "Medium," [Online]. Available: https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193. [Accessed 15. 12. 2019.].

[20] P. G. R. G. K. H. P. D. Tsung-Yi Lin, "Focal Loss for Dense Object Detection," Long Beach, 2017.

[21] S.-H. Tsang, "Towards Data Science," [Online]. Available: https://towardsdatascience.com/review-retinanet-focal-loss-object-detection-38fba6afabe4. [Accessed 15. 12. 2019.].

[22] J. Hui, "Medium," [Online]. Available: https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c. [Accessed 15. 12. 2019.].

[23] X. Z. R. J. S. Kaiming He, "Deep Residual Learning for Image Recognition," Microsoft, 2015.

[24] C. Shorten, "Towards Data Science," [Online]. Available: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4. [Accessed 15. 12. 2019.].

[25] "AI Challenger," [Online]. Available: https://challenger.ai/. [Accessed 15. 12. 2019.].

[26] A. H. M. Z. A. Z. L.-C. C. Mark Sandler, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018.

[27] S. Mallat, Understanding deep convolutional networks, 2016.

[28] "GitHub," [Online]. Available: https://github.com/llSourcell/YOLO_Object_Detection. [Accessed 28. 11. 2019.].

[29] "GitHub," [Online]. Available: https://github.com/matterport/Mask_RCNN. [Accessed 25. 11. 2019.].

[30] "GitHub," [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed 25. 11. 2019.].

[31] "GitHub," [Online]. Available: https://github.com/balancap/SSD-Tensorflow. [Accessed 26. 11. 2019.].

[32] "Github," [Online]. Available: https://github.com/OlafenwaMoses/ImageAI/. [Accessed 26. 11. 22019.].

Popis priloga

Datoteke i mape priložene uz diplomski rad:

- Videozapis rukomet.mp4
- Videozapis rezultati rukomet mrccn.mp4
- Videozapis rezultati rukomet ssd.mp4
- Videozapis rezultati rukomet yolov3.webm
- Videozapis rezultati rukomet retinanet.avi
- cjelokupna izvorišna mapa 'Android aplikacija za prepoznavanje pokreta'
- generirana aplikacija za instaliranje na Android uređaje PosEst.apk
- python/CMD kod za izvršavanje detekcije objekata
    - mrcnn video kod.py
    - ssd video kod.py
    - yolov3.txt
    - retinanet video kod.py
- Videozapis rada aplikacije PoseEstS.mp4

Sadržaj