SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1724

Duboki konvolucijski modeli za lokalizaciju objekata

Damjan Miko

Zagreb, srpanj 2018.

Zahvaljujem se mentoru, izv. prof. dr. sc. Siniši Šegviću na ukazanom trudu i znanju kojim mi je pomogao uspješno riješiti zadatak diplomskog rada. Također se zahvaljujem obitelji i prijateljima na podršci tijekom studija.

SADRŽAJ

1. Uvod **1**

2. Duboke neuronske mreže **2**

2.1. Neuronske mreže 2

2.1.1. Umjetni neuron 2

2.1.2. Arhitektura neuronske mreže 4

2.2. Konvolucijske mreže 5

2.2.1. Konvolucijski sloj 6

2.2.2. Sloj sažimanja 8

2.2.3. Optimizacijski algoritmi 9

3. Detekcija objekata dubokim modelima **11**

3.1. SSD - Single shot detector 11

3.1.1. Osnovni model 12

3.1.2. DSSD - Deconvolutional single shot detector 13

3.2. YOLO - You only look once 15

3.2.1. Osnovni model 15

3.2.2. YOLO 9000: Better, Faster, Stronger 19

3.2.3. YOLOv3: An Incremental Improvement 22

3.3. Usporedba brzih detektora 25

4. Korišteni algoritmi **28**

4.1. Generiranje referentnih okvira 28

4.2. Predikcija sa samo jednim unaprijednim prolazom 29

5. Detekcija objekata na sportskom susretu **31**

5.1. Video za detekciju 31

5.2. Izrada anotacija 32

6. Programska implementacija i vanjske biblioteke **35**
6.1. Programska implementacija 35
6.1.1. Struktura programske implementacije 35
6.1.2. Implementacija duboke neuronske mreže 37
6.2. Programska podrška 39
6.2.1. Python 39
6.2.2. TensorFlow 39
6.2.3. OpenCV 39

7. Eksperimentalni rezultati **41**
7.1. Evaluacijske mjere 41
7.2. Rezultati implementacija 44
7.2.1. Osnovni YOLOv2 model 44
7.2.2. Prilagođeni YOLOv2 model 45
7.2.3. Prilagođeni YOLOv2 model s podjelom slika 47
7.2.4. YOLOv3 model 49

8. Zaključak **53**

Literatura **54**

1. Uvod

Ljudski mozak brzo prepoznaje i lokalizira objekte na slici s velikom preciznošću. Iako to ljudima ne izaziva problem, računalu nije nimalo jednostavno razumjeti sliku. Iz ovog razloga započeo je razvoj računalnog vida. To je područje umjetne inteligencije koje se bavi obradom, analiziranjem i razumijevanjem slika. Iz slika se prikupljaju podaci sa svrhom izvlačenja korisnih numeričkih ili simboličkih značajka pomoću modela razvijenih na temelju strojnog učenja. Područja računalnog vida su klasifikacija (razvrstavanje slika u razrede), semantička segmentacija (podjela slike u dijelove i dodjeljivanje klasa svakom dijelu), detekcija, praćenje objekta i još mnoga druga. Ovaj rad se bavi detekcijom objekata u realnom vremenu.

Cilj detekcije je prepoznati objekte na slici i odrediti njihovu točnu lokaciju. Objekti mogu pripadati različitim kategorijama te ih treba smjesti u njihovu klasu. Problem detekcije nastaje zato što objekti mogu biti okrenuti na različite strane, mogu biti različitih veličina ili prekriveni nekim drugim objektima sa slike.

Detekcija objekata se nekada rješavala klasičnim metodama računalnog vida i strojnog učenja. Međutim, kada su se pojavili duboki konvolucijski modeli, oni su srušili sve rekorde u preciznosti i kvaliteti klasifikacije i detekcije. Razlozi za taj nagli rast u performansama neuronskih mreža su razvitak novih algoritama učenja, velik porast u performansama procesora, pogotovo grafičkih kartica te ogroman broj kvalitetno označenih skupova za učenje. Većina istraživanja okrenula su se prema dubokim modelima te je razvijeno mnogo novih metoda.

Prvi modeli koji su dobivali dobre rezultate detekcije podijeljeni su u dvije faze. Prva faza predlaže okvire unutar kojih su objekti, a druga ih klasificira. Ti modeli su precizni, ali imaju problem što im treba nekoliko sekunda za obradu slike. Zadatak ovog rada je detekcija objekata u realnom vremenu za što su korišteni modeli razvijeni u novije vrijeme. Oni spajaju faze predlaganja okvira i klasifikacije istih te jednim prolazom kroz duboku mrežu detektiraju objekte. Uvođenjem novih promjena drastično ubrzavaju vrijeme potrebno za detekciju objekata. U radu su objašnjeni brzi modeli detekcije te su korišteni za lokalizaciju igrača na nogometnom susretu.

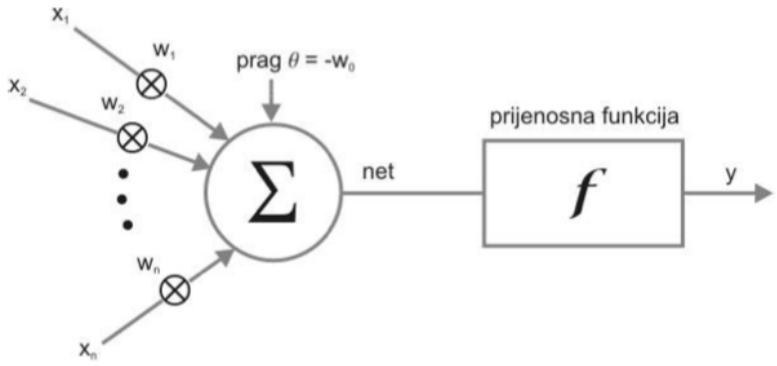
2. Duboke neuronske mreže

2.1. Neuronske mreže

Problemi detekcije i klasifikacije slike za računalo predstavljaju zahtjevan problem i ne mogu se riješiti jednostavnim algoritmima. Pokazalo se kako računalo ima velikih problema kada je u pitanju generalizacija. Za razliku od računala, čovjek takve probleme rješava brzo i efikasno. Upravo je ljudski mozak inspiracija za razvoj neuronskih mreža. One su skup umjetnih neurona koji su povezani u nizu slojeva kako bi rješili kompleksne probleme.

2.1.1. Umjetni neuron

Neuron je funkcija koja na ulazu prima vektor $\vec{x} = [x_1, x_2, ..., x_n]$ te matematičkim operacijama dolazi do skalarnog izlaza y. Primjer neurona prikazan je McCulloch-Pitts modelom (slika 2.1). Na slici vidimo da neuron osim ulaznog vektora $\vec{x}$ sadrži i težinski vektor $\vec{w}$, prag θ i aktivacijsku funkciju f.



Slika 2.1: Umjetni neuron prikazan McCulloch-Pitts modelom.

Svaki ulazni parametar x_j množi se s pripadajućom težinom w_j nakon čega se svi rezultati zbroje i dodaje im se prag θ. Česta je praksa da se prag θ prikazuje kao vrijednost težine $w_0 = -\theta$ te se radi kompaktnosti ulaznom vektoru doda vrijednost

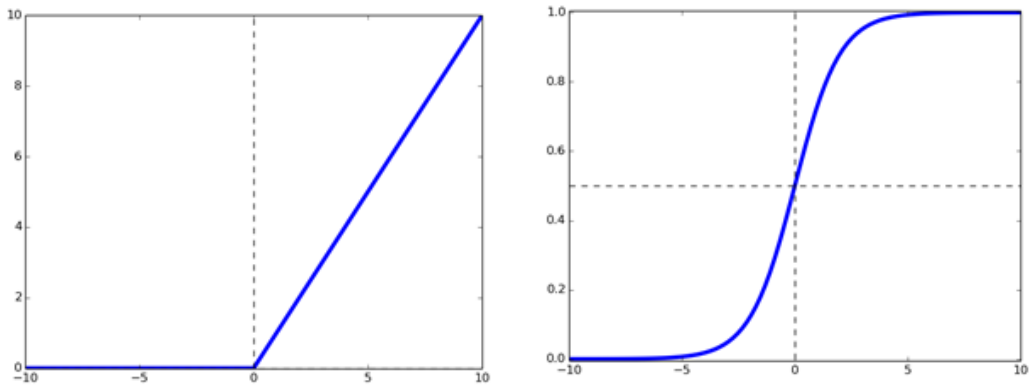
$x_0 = 1$. Na kraju dobiveni zbroj provučemo kroz aktivacijsku funkciju kako bi dobili izlaz y. Matematički izraz za izlaz neurona prikazan je u nastavku.

$$y = f(\vec{x} \cdot \vec{w}) = f(\sum_{j=1}^{n} x_j \cdot w_j) \tag{2.1}$$

Neuroni se razlikuju na temelju aktivacijske funkcije. Ona može biti proizvoljna te se odabire ovisno o problemu koji rješava. Aktivacijske funkcije služe da bi postigli nelinearnost između slojeva. Bez nelinearnih aktivacijskih funkcija, neuronske mreže ne bi se mogle prilagoditi svim podacima i naučiti nelinearnost u njima. Neke od najpoznatijih aktivacijskih funkcija su ReLU (2.2) i sigmoidalna funkcija (2.3). Njihovi grafovi su prikazani na slici 2.2.

$$f(x) = ReLU(x) = max(0, x) \tag{2.2}$$

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

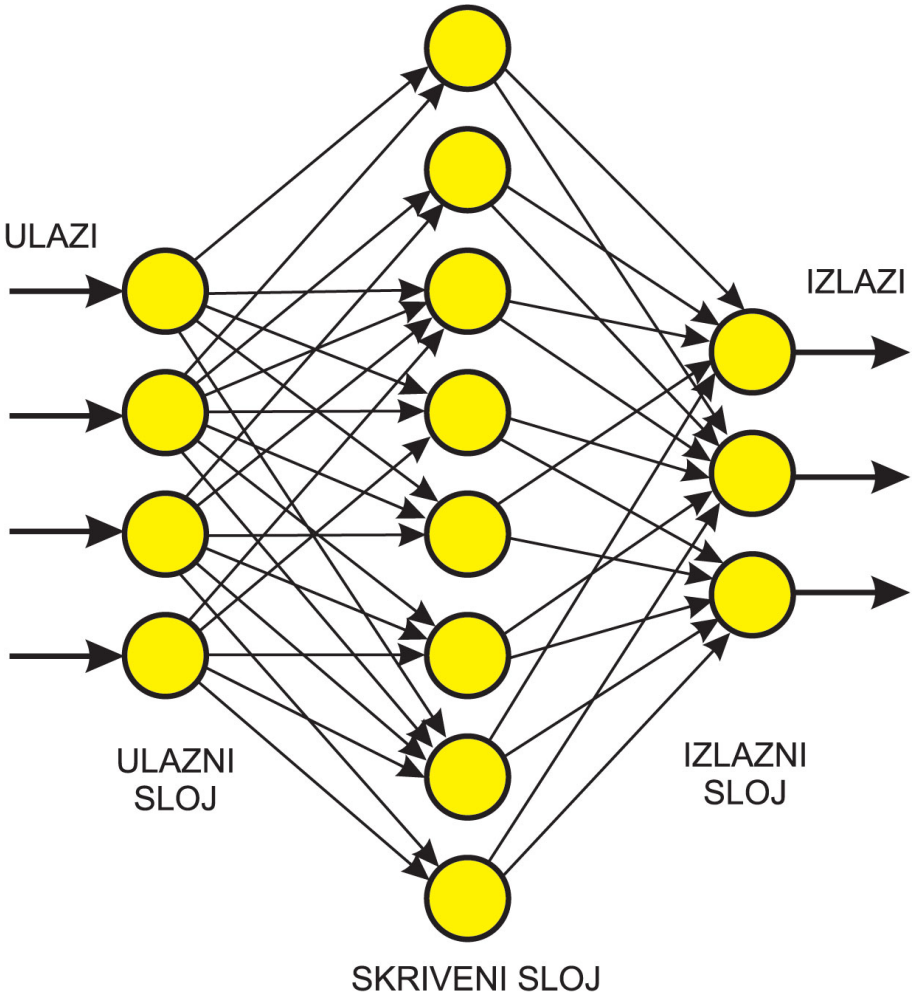


(a) **(b)**

Slika 2.2: Grafovi aktivacijskih funkcija: (a) ReLU, (b) sigmoidalna funkcija.

2.1.2. Arhitektura neuronske mreže

Neuronska mreža nastaje povezivanjem neurona. Osnovna arhitektura prikazana je na slici 2.3. Na njoj vidimo da su osnovni slojevi neuronske mreže: ulazni, skriveni i izlazni sloj. Ulazni sloj predstavlja ulazne podatke u mrežu, koji zatim idu preko skrivenog sloja do izlaznog te nam njegovi izlazi predstavljaju dobivene rezultate. Skriveni sloj se sastoji od jednog ili više podslojeva neurona. Svaki sloj u neuronskoj mreži ima veze sa svojim susjedima i to takve da izlazi prethodnog sloja predstavljaju ulaze za sljedeći sloj.



Slika 2.3: Arhitektura neuronske mreže. Osnovni slojevi su ulazni skriveni i izlazni sloj.

2.2. Konvolucijske mreže

Konvolucijske neuronske mreže mogu se opisati kao specijalizacija višeslojnih neuronskih mreža. Konvolucijska, kao i obična neuronska mreža sastoji se od jednog ulaznog, jednog izlaznog i jednog ili više skrivenih slojeva. Dobila je ime po matematičkoj operaciji konvoluciji, koja je definirana nad dvije funkcije realne ili kompleksne varijable koja kao izlaz daje treću funkciju koja je definirana kao količina preklapanja između prve funkcije te okrenute i pomaknute druge funkcije. Konvolucija realne neprekinute funkcije je definirana izrazom u nastavku.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.4}$$

Konvolucija ima široku primjenu, pogotovo u obradi slike i signala. U ovim područjima riječ je o konvolucijskim filterima koji se koristite za izoštravanje i zamućenje slika, kao i za detekciju rubova.

Konvolucijske mreže imaju veliki značaj u dubokom učenju zbog mogućnosti detekcije rubova, pomoću kojih se otkrivaju objekti na slici. Tamo gdje obične neuronske mreže pokazuju svoje nedostatke, tu konvolucijske pokazuju svoje prednosti. Na primjer, kada se kao ulazni podaci koriste slike, slika dimenzija $200 * 200$ piksela pretvara se u 40 000 neurona u ulaznom sloju, što je ogroman broj i praktično je nemoguće treniranje potpuno povezanih neuronskih mreža. Za sliku u boji treba uzeti u obzir i 3 kanala kojima su predstavljene osnovne boje: crvena, zelena i plava, što daje 120 000 neurona u ulaznom sloju. Ideja konvolucijske neuronske mreže je da se postavi veći broj slojeva za otkrivanje bitnih osobina ulaznih podataka. U skladu s tom idejom se konvolucijski filteri primjenjuju na sliku kako bi se izvukle korisne karakteristike i kreirale njihove mape značajki. Primjenom filtera na ulaznu sliku, pored otkrivanja značajnih karakteristika, vrši se i smanjenje rezolucije.

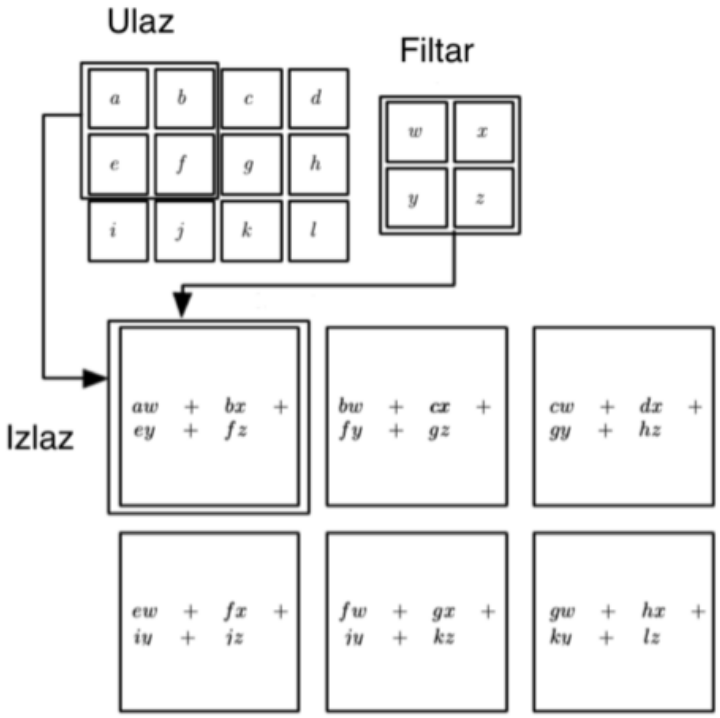
Duboki modeli imaju potrebu za značajnim hardverskim resursima, pa se za njihovo izvođenje koriste jake i skupe grafičke kartice. Također su dosadašnji modeli učeni nad velikim brojem slika, ali noviji rezultati govore da se visoke performanse detekcije mogu postići i s relativnom malim brojem uzoraka (Marko Dabović, 2017).

2.2.1. Konvolucijski sloj

Razlika konvolucijskih mreža u odnosu na druge tipove arhitektura neuronskih mreža je u sloju gdje se vrši konvolucija, po kome je i dobila ime. U računalnom vidu kod konvolucije u neuronskim mrežama (izraz 2.5) funkcija f se naziva ulazom, funkcija g jezgrom ili filtrom, a izlazi operacije konvolucije su mape značajki. Pri obradi slika ulazni podaci u konvoluciju su dvodimenzionalni, tada je i jezgra dvodimenzionalna, pa za slike vrijedi sljedeća jednadžba.

$$(f * g)(i, j) = \sum_{m=m_{min}}^{m_{max}} \sum_{n=n_{min}}^{n_{max}} f(i + m, j + n)k(m, n) \tag{2.5}$$

Primjer dvodimenzionalne konvolucije definirane u prethodnoj jednadžbi s kvadratnom jezgrom $2 * 2$ prikazan je na slici 2.4.

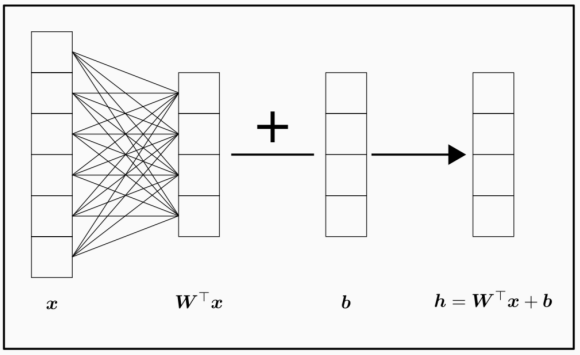


Slika 2.4: Prikaz rezultata konvolucije s filtrom dimenzije $2 * 2$. Filtar se pomiće po ulazu i formira izlaz prema jednadžbi 2.5 (Karapac, 2017).

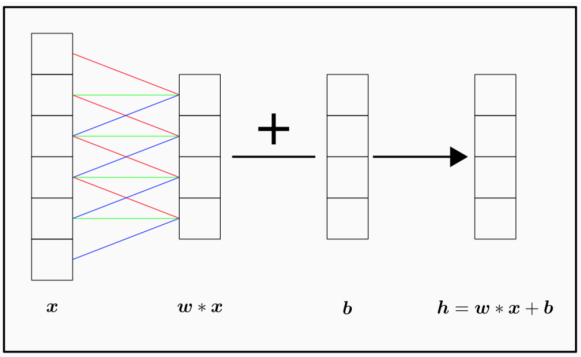
Na njoj se jasno može vidjeti kako funkcionira konvolucija. Vrijednosti ulaznog dijela unutar prozora se množe s filtrom i sumiraju te daju izlaze za pojedine dijelove izlazne mape značajki. Filtar se primjenjuje tako da se pomiče kroz cijeli ulaz. U novije vrijeme najčešće se koriste kvadratne jezgre dimenzije $k = 3, 5$ ili 7, k je ne-

paran broj kako bi određeni piksel predstavljao sredinu filtra. Rezultat konvolucije je dvodimenzionalna aktivacijska mapa koja predstavlja odziv filtra na svakoj prostornoj poziciji. Zapravo mreža će naučiti težine unutar filtra kako bi se filtar aktivirao na mjestima gdje prepoznaje određena slikovna svojstva, kao na primjer određene vrste rubova ili slično.

Konvolucijske mreže su pogodne za rad s dvodimenzionalnim podacima kao što su slike zato što modeliraju samo lokalne interakcije i podržavaju dijeljenje parametara. Bitna prednost konvolucijskih mreža je puno manji broj težina u svakom sloju u odnosu na potpuno povezani sloj. To je predočeno slikom 2.5. Kod potpuno povezanog sloja svaki izlaz je povezan sa svim ulazima, a kod konvolucije samo s malim dijelom ulaza ovisno o filterima koji su malih dimenzija. Broj parametra u potpunom povezanom sloju je $m \cdot n$, dok je u konvolucijskom $k \cdot n$, gdje je m dimenzija ulaznog vektora, k širina jezgre, a n veličina trenutnog sloja te vrijedi $k << m$ (Krapac, 2016). Zbog toga moramo pohraniti manje parametara, što smanjuje memorijske zahtjeve modela i znatno ubrzava evaluaciju.



(a)



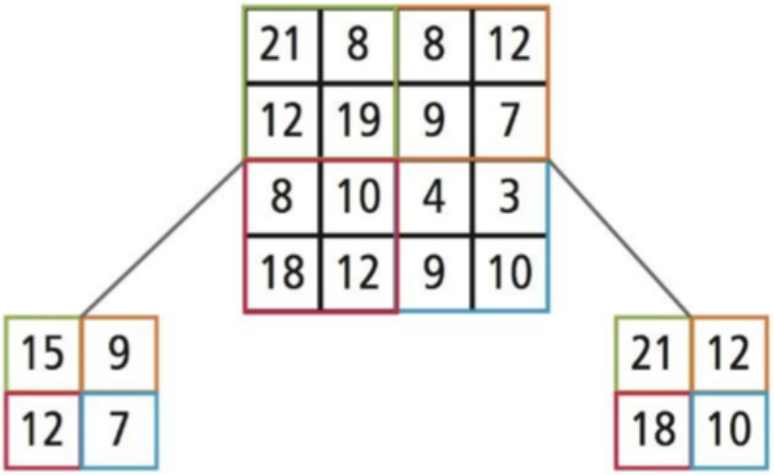
(b)

Slika 2.5: Usporedba (a) potpuno povezanog sloja i (b) konvolucijskog sloja. Potpuno povezani sloj koristi mnogo više parametara od konvolucijskog (Karapac, 2017).

2.2.2. Sloj sažimanja

Nakon konvolucijskih slojeva, obično dolaze slojevi sažimanja koji mapiraju skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu. Mapiranje se vrši tako da se računa statistički pokazatelj ulaznih značajki koji može biti maksimalna vrijednost, srednja vrijednost, L2 norma itd. Primjer mapiranja je prikazan na slici 2.6. U početku se radilo tako da se koristila aritmetička sredina, no kasnije se pokazalo da se malo bolji rezultati dobivaju korištenjem maksimalne vrijednosti.

Funkcija sažimanja prima dva hiperparametra. Prvi je širina filtra, a drugi je korak. Širina filtra nam govori koliko će susjednih elemenata obuhvatiti sažimanje, dok korak određuje koliko se filtar pomiče prilikom računanja izraza. Na slici 2.6 se koristi sažimanje sa širinom filtra 2 i korakom 2.



Slika 2.6: Rezultat sažimanja sa širinom filtra 2 i korakom 2. Na lijevoj strani je primjer sažimanja srednjom vrijednošću, dok je na desnoj maksimalnom vrijednošću.

Sažimanje povećava invarijantnost na blage rotacije i translacije. Invarijantnost je u matematičkom obliku definirana tako da je funkcija $f(x)$ invarijantna s obzirom na g ako vrijedi sljedeći izraz:

$$f(g(x)) = f(x) \tag{2.6}$$

Pomoću invarijantnosti mreža će uspjeti dobro prepoznati objekt iako je on blago pomaknut jer se operacijom sažimanja dobila jednaka vrijednost u oba slučaja. Osim toga, sažimanjem se smanjuju memorijski zahtjevi implementacije neuronske mreže. Ipak, treba biti oprezan, budući da se dio informacija može izgubiti ako je područje sažimanja preveliko.

2.2.3. Optimizacijski algoritmi

Cilj strojnog učenja je minimizirati pogrešku generalizacije. Za taj izračun koristimo optimizacijske algoritme. Klasični optimizacijski postupci koriste malene korake u suprotnom smjeru od gradijenta kako bi konvergirali i našli lokalni optimum. Oni se dijele na nekoliko vrsta ovisno o broju primjera koje koriste u jednom koraku algoritma, a to su: stohastički, grupni i algoritmi temeljeni na mini-grupama. Stohastički algoritmi rade izračun gradijenta funkcije gubitaka koristeći jedan po jedan primjer, dok grupni koriste čitav skup primjera za učenje. Kod grupnog načina procjena gradijenta je točnija u odnosu na stohastički jer u obzir uzima sve primjere. Međutim, u slučaju velikog broja primjera za grupni način potrebno je puno vremena i memorije te može biti preskup. Zbog toga algoritmi temeljeni na mini-grupama su se našli između ova dva načina. Oni računaju gradijent na dijelu podataka. Što je veća veličina mini-grupe to je gradijent precizniji, ali skuplji za izvršavanje. Moderni algoritmi su temeljeni na mini-grupama jer pokazuju najbolje rezultate.

Za optimizaciju se dugo vremena koristio gradijentni spust, međutim s dolaskom dubokih neuronskih mreža ispostavilo se da on nije najbolji odabir. Počeli su se razvijati novi moderni algoritmi, a neki najpoznatiji su: AdaGrad, RMSProp i Adam. U projektu su korišteni RMSProp i Adam algoritmi, pa će biti ukratko objašnjeni.

RMSProp algoritam

RMSProp je algoritam temeljen na gradijentnom spustu koji može ubrzati spust koristeći adaptivnu stopu učenja. On ima jedu stopu učenja za svaki od parametara i adaptira ih skalirajući inverzno proporcionalno kvadratnom korijenu sume propadajućih kvadriranih vrijednosti pripadne parcijalne derivacije funkcije gubitka kroz povijest. Parametri koji imaju velike parcijalne derivacije brzo će početi koristiti male stope učenja, dok će oni sa malom parcijalnom derivacijom dugo vremena koristiti velike stope učenja (Čupić, 2017). Pseudokod algoritma prikazan je na slici 2.7.

Ova metoda pokušava riješiti problem jako malih vrijednosti gradijenata tako da mijenja akumuliranje gradijenata dodajući eksponencijalno prigušivanje starih vrijednosti što znači da novije vrijednosti imaju veći utjecaj. Ovo je prikazano u pseudokodu 2.7 parametrom r, koji sadrži sume kvadratnih gradijenta i u svakom koraku se množi s parametrom $p < 1$ kako bi prigušio stare gradijente. To znači da ako komponenta gradijenta u nekom trenutku padne, relativno brzo će pasti i njeno prigušenje pa će se korekcije ponovno povećati (Čupić, 2017).

Algorithm 1 RMSProp algoritam

Require: stopa_ucenja η, parametar smanjenja p, inicijalni parametar θ

Require: konstanta δ (10^{-6}), koristi se za izbjegavanja dijeljenja s nulom

Inicijaliziraj varijablu akumulacije $r = 0$

while kriterij za zaustavljanje nije zadovoljen **do**

Kreiraj skup m primjera iz podataka za treniranje $\{x^{(i)}, ..., x^{(m)}\}$ zajedno s oznakama $y^{(i)}$

$g \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

$r \leftarrow pr + (1-p) g \odot g$

$\triangle\theta = -\frac{\eta}{\sqrt{\delta + r}} \odot g$($\frac{1}{\sqrt{\delta+r}}$ operacija po elementima)

$\theta \leftarrow \theta + \triangle\theta$

end while

Slika 2.7: Psudokod algoritma RMSProp (Čupić, 2017) .

Algoritam Adam

Adam je algoritam koji izračunava adaptivnu stopu učenja iz procjena prvog i drugog momenta gradijenata. On kombinira metode momenta i adaptivnog pomaka što pospješuje učenje dubokih modela. Prati uprosječeno kretanje gradijenta i kvadrata gradijenta uz eksponencijalno zaboravljanje. Pseudokod algoritma prikazan je na slici 2.8. Algoritam je dosta robusna implementacija koja radi sa širokim skupom parametara te se preporučuje kao prvi odabir za učenje dubokih neuronskih mreža.

Algorithm 2 Algoritam ADAM

Require: stopa_ucenja η, β_1 i β_2, δ, inicijalni parametar θ

Inicijalizacija prvog i drugog momenta $s = 0, r = 0$ i vremenskog koraka $t = 0$

while kriterij za zaustavljanje nije zadovoljen **do**

Kreiraj skup m primjera iz podataka za treniranje $\{x^{(i)}, ..., x^{(m)}\}$ zajedno s oznakama $y^{(i)}$

$g \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

$t \leftarrow t + 1$

$s \leftarrow \beta_1 s + (1 - \beta_1) g$

$r \leftarrow \beta_2 r + (1 - \beta_2) g \odot g$

$\hat{s} \leftarrow \frac{s}{1 - \beta_1^t}$

$\hat{r} \leftarrow \frac{r}{1 - \beta_2^t}$

$\triangle\theta = -\eta \frac{\hat{s}}{\sqrt{\hat{r}} + \delta} \odot g$($\frac{1}{\sqrt{\delta+r}}$ operacija po elementima)

$\theta \leftarrow \theta + \triangle\theta$

end while

Slika 2.8: Psudokod algoritma Adam (Čupić, 2017).

3. Detekcija objekata dubokim modelima

Detekcija objekata jedan je od problema kojeg rješava računalni vid. Cilj detekcije je prepoznati objekte na slici i odrediti njihovu točnu lokaciju. Objekti mogu pripadati različitim kategorijama te je cilj da se svaki objekt smjesti u svoju kategoriju. Problem detekcije nastaje zato što objekti mogu biti okrenuti na različite strane, mogu biti druge veličine ili prekriveni nekim drugim objektima sa slike. Iako je čovjeku lak zadatak odrediti točne lokacije objekata na slici neovisno o njegovom izgledu i položaju, računalu taj problem nije nimalo jednostavan.

Problemi detekcije su se nekada rješavali klasičnim metodama računalnog vida i strojnog učenja. Popularna rješenja bila su ručno oblikovane značajke SIFT (eng. *scale-invariant feature transform*) i HOG (eng. *histogram of oriented gradients*) metode. No, kada su se pojavili duboki konvolucijski modeli, oni su srušili sve rekorde u preciznosti i kvaliteti klasifikacije i detekcije. Većina istraživanja okrenula su se prema njima te je razvijeno mnogo novih metoda za duboke modele. Neki od prvih modela koji su dobivali dobre rezultate bili su Fast i Faster R-CNN, no oni su imali problem što im je trebalo po nekoliko sekunda za obradu slike. U nastavku ćemo opisati novije modele koji mogu detektirati objekte u realnom vremenu. To su YOLO (eng. *You only look once*) i SSD (eng. *Single-shot detector*) modeli te njihova nadogradnja.

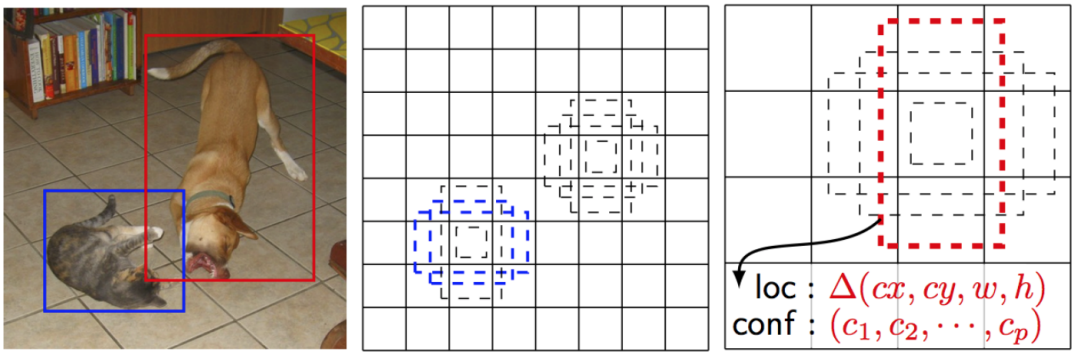
3.1. SSD - Single shot detector

Nekadašnji modeli detekcije rade na principu da odvoje faze za predlaganje okvira i klasifikaciju tih okvira. Na početku iz ulazne slike izdvoje regije unutar kojih će se tražiti objekti. Nakon toga slijedi druga faza u kojoj svaku regiju provuku kroz duboku mrežu te je klasificiraju. SSD spaja te dvije faze u jednu bez gubitka točnosti. Time se drastično ubrzava vrijeme detekcije objekata zbog čega je postao popularan. Kasnije je došlo do poboljšanja SSD-a i nastala je nova verzija DSSD (eng. *Deconvolutional*

Single Shot Detector). U nastavku će se opisati obje verzije.

3.1.1. Osnovni model

SSD model propusti sliku kroz konvolucijsku mrežu i time dobije nekoliko mapa značajki koje su različitih dimenzija ($10*10, 5*5, 3*3$ itd.). One se koriste kako bi mogle bolje detektirati objekte koji su različitih veličina. Na slici 3.1 prikazan je primjer u kojem jedan od okvira na mapi značajki dimenzija $4*4$ preklapa psa, dok u sloju $8*8$ to nije tako. Razlog tome je taj što u mapama značajki manjih dimenzija svaki okvir prekrije veći dio slike te su one pogodne za detekciju većih objekata, dok u mapama značajki većih dimenzija možemo detektirati manje objekte kao što je mačka prikazana na slici.



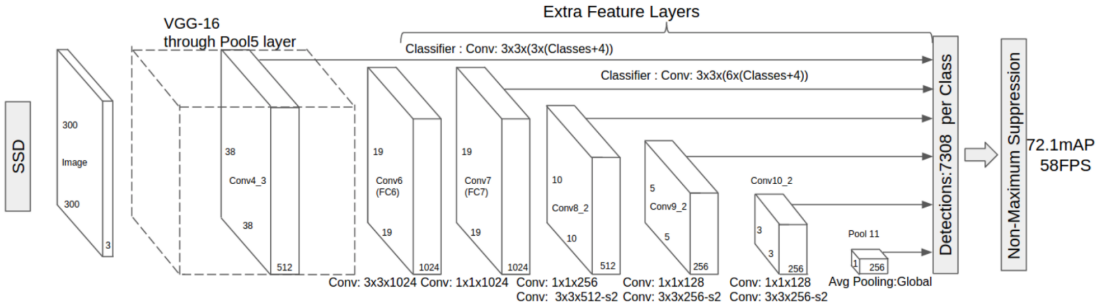
(a) SSD detekcija slike **(b)** $8*8$ mapa značajki **(c)** $4*4$ mapa značajki

Slika 3.1: Na slici (a) su označeni objekti dobiveni SSD detekcijom. Slika (b) prikazuje mapu značajki $8*8$ u kojoj je otkrivena mačka, a slika (c) mapu značajki $4*4$ u kojoj je otkriven pas (Wei Liu, 2015).

Na svakom izlaznom sloju koji služi za detekciju SSD postavlja referentne okvire (eng. *anchor boxes*). Oni su različitih veličina i imaju različite omjere stranica kako bi obuhvatili što više različitih dimenzija objekata (slika 3.1). Referentni okviri postavljaju se u svaku od lokacija na mapama značajki čime se dobiva skup pretpostavljenih okvira u kojima se mogu nalaziti objekti. Njima se pridružuje vektor koji sadrži vjerojatnosti da se unutar njega nalazi objekt pojedine kategorije i 4 relativna pomaka u odnosu na originalnu poziciju okvira. Dimenzionalnost vektora je $4 + k$, gdje je k broj kategorija u modelu, a 4 broj relativnih pomaka. U zadnjem koraku koristi se funkcija softmax kako bi klasificirala objekte (Wei Liu, 2015).

SSD koristi model koji je zasnovan na VGG-16 arhitekturi. Na bazne slojeve arhitekture dodaju se još novi slojevi te nastaje model prikazan na slici 3.2. Novi slojevi

se dodaju kako bi povećali mogućnost da uhvatimo još neke objekte.



Slika 3.2: Prikaz SSD arhitekture 300x300. Ona je zasnovana na baznoj arhitekturi VGG-16 (Wei Liu, 2015).

SSD model čini se jednostavnim, no njegovo treniranje je izazovan proces. Primjenom referentnih okvira na svim lokacijama mape značajki dobije se mnogo pretpostavljenih okvira koje mreža mora naučit klasificirati, a to nije lak zadatak. Modeli kao što su Fast i Faster R-CNN imaju fazu predlaganja regija koja kao rezultat daje manji broj okvira te svaki okvir ima neku minimalnu vjerojatnost da je uokvirio objekt. SSD preskače tu fazu predlaganja regije i umjesto nje postavlja okvire na svaku od lokacija u mapama značajki različitih dimenzija, koristeći različite veličine referentnih okvira. Kao rezultat ovog procesa SSD generira mnogo više okvira od kojih su skoro svi negativi, odnosno ne sadrže objekt (Xu, 2017).

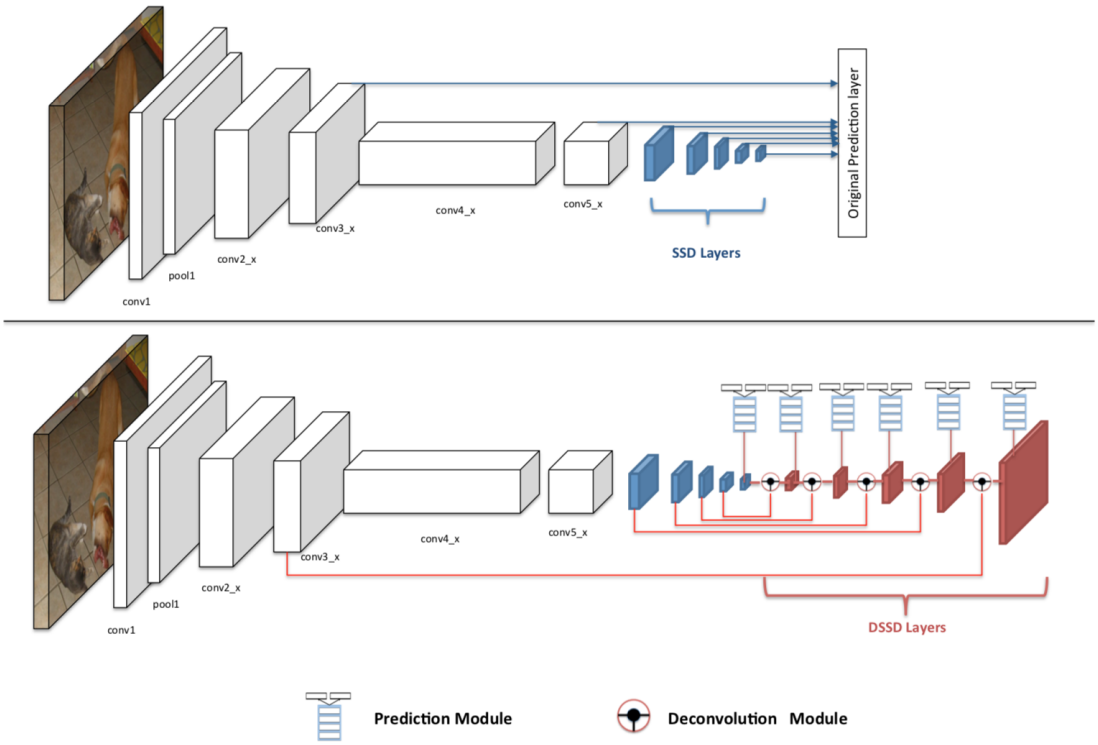
Kako bi se smanjila neuravnoteženost modela SSD koristi dva pristupa, algoritam eliminacije nemaksimalnih elemenata i traženje teških negativa. Eliminacija nemaksimalnih elemenata je postupak koji grupira okvire koji se preklapaju. Ako se na primjer pet okvira iste veličine nalaze na gotovo istom mjestu označavajući isti objekt, SSD će uzeti onaj okvir koji ima najveću vjerojatnost da uokviruje objekt i odbaciti ostale. Traženje teških negativa balansira primjere tijekom treniranja mreže kako ne bi bilo previše negativnih primjera u modelu što bi značajno otežalo treniranje. Taj postupak odbacuje dio negativa tako da se u svakoj mini-grupi broj negativa javlja u omjeru $3:1$ u odnosu na broj pozitiva.

3.1.2. DSSD - Deconvolutional single shot detector

Cilj modela DSSD je korištenje veće duboke mreže koja izvuče više bitnih značajka sa slike kako bi povećala točnost detekcije. Da bi to postigli autori modela prvo su odlučili spojiti najsuvremeniji klasifikator Residual-101 sa SSD-om. Potom su tako dobiveni model proširili sa dekonvolucijskim slojevima. To su slojevi koji se dodaju

na kraj duboke mreže kako bi povećali rezoluciju mapa značajki te se oni koriste za predviđanje lokacije objekata. Označeni su crvenom bojom na slici 3.3. Dekonvolucijom je model DSSD povećao preciznost detekcije, posebno za manje objekte jer njenim korištenjem se unosi više semantičkih informacija u mape značajki (Cheng-Yang Fu, 2017). Zbog dekonvolucijskih slojeva model je dobio ime dekonvolucijski SSD (DSSD).

Jedna od promjena u DSSD-u je korištenje bazne arhitekture ResNet-101 umjesto VGG-16 koja je korištena u originalnom SSD modelu. Cilj nove arhitekture bio je pronalazak više semantičkih značajaka kako bi se povećala točnost. Na slici 3.3 je prikazana arhitektura SSD i DSSD modela. Plavom bojom prikazani su dodatni slojevi koji su dodani u SSD-u, a crvenom novi slojevi dodani u DSSD-u. Na slici vidimo da DSSD samo proširuje mrežu SSD-a dekonvolucijskim slojevima.



Slika 3.3: Prikaz arhitekture mreže SSD-a (gore) i DSSD-a (dolje). DSSD dodaje dekonvolucijske slojeve označene crvenom bojom na kraj SSD mreže i koristi baznu arhitekturu ResNet-101, dok SSD koristi VGG-16 (Cheng-Yang Fu, 2017).

DSSD koristi referentne okvire, kao što i SSD, nad mapama značajki pojedinih slojeva mreže koji su usredotočeni na predviđanje objekata određenih veličina u tom sloju. Elementi mapa značajki u različitim slojevima imaju drugačija receptivna polja, veće objekte predviđaju oni slojevi koji imaju manju mapu značajki, odnosno oni

slojevi koji dolaze kasnije u arhitekturi mreže. Oni također imaju i veće receptivno polje, dok manje objekte predviđaju slojevi koji dolaze prije, a oni imaju manje receptivno polje i guste mape značajki. To dovodi do malih performansa u detekciji manjih objekata zato što plitki slojevi imaju malo semantičkih informacija o objektu.

Method	network	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SSD512*[18]	VGG	79.5	84.8	85.1	81.5	73.0	57.8	87.8	88.3	87.4	63.5	85.4	73.2	86.2	86.7	83.9	82.5	55.6	81.7	79.0	86.6	80.0
SSD 513	Residual-101	80.6	84.3	87.6	**82.6**	71.6	59.0	88.2	88.1	89.3	64.4	85.6	76.2	88.5	88.9	87.5	83.0	53.6	83.9	82.2	87.2	81.3
DSSD 513	Residual-101	**81.5**	**86.6**	86.2	**82.6**	**74.9**	62.5	**89.0**	**88.7**	88.8	65.2	87.0	**78.7**	88.2	89.0	**87.5**	**83.7**	51.1	**86.3**	81.6	85.7	**83.7**

Slika 3.4: Rezultati detekcije SSD i DSSD modela na skupu PASCAL VOC 2007. SSD512 koristi baznu mrežu VGG-16, dok SSD513 koristi ResNet-101 (Cheng-Yang Fu, 2017).

Na slici 3.4 nalazi se tablica s rezultatima detekcije objekata nad skupom PASCAL VOC 2007. Pomoću nje možemo usporediti točnost detekcije SSD modela sa VGG baznom mrežom, SSD modela s ResNet-101 baznom mrežom i DSSD modela. SSD512 su posljednji rezultati SSD-a i oni su bolji od većine modernih detektora. SSD s ResNet-101 mrežom daje bolje rezultate od originalnog SSD-a s time da i konvergira duplo brže. Najbolje rezultate ima DSSD te on postiže najvišu točnost od detektora s jednim unaprijednim prolazom nad skupom PASCAL VOC održavajući brzinu detekcije s najnovijim modelima.

3.2. YOLO - You only look once

YOLO je još jedan model za detekciju koji spaja faze za predlaganje okvira i njihovu klasifikaciju. Radi predikcije i vjerojatnosti pripadanja objekata određenoj kategoriji na temelju jednog prolaza kroz mrežu te zbog toga ima mogućnost detektiranja objekata u realnom vremenu. Razvijen je 2015. godina, neposredno prije SSD modela. Do sada su razvijene i dvije nadogradnje osnovnog YOLO modela. Za potrebe ovog diplomskog rada korištene su druga i treća verzija YOLO modela, odnosno YOLOv2 i YOLOv3. U nastavku će biti ukratko objašnjeni osnovni YOLO model i njegove nadogradnje.

3.2.1. Osnovni model

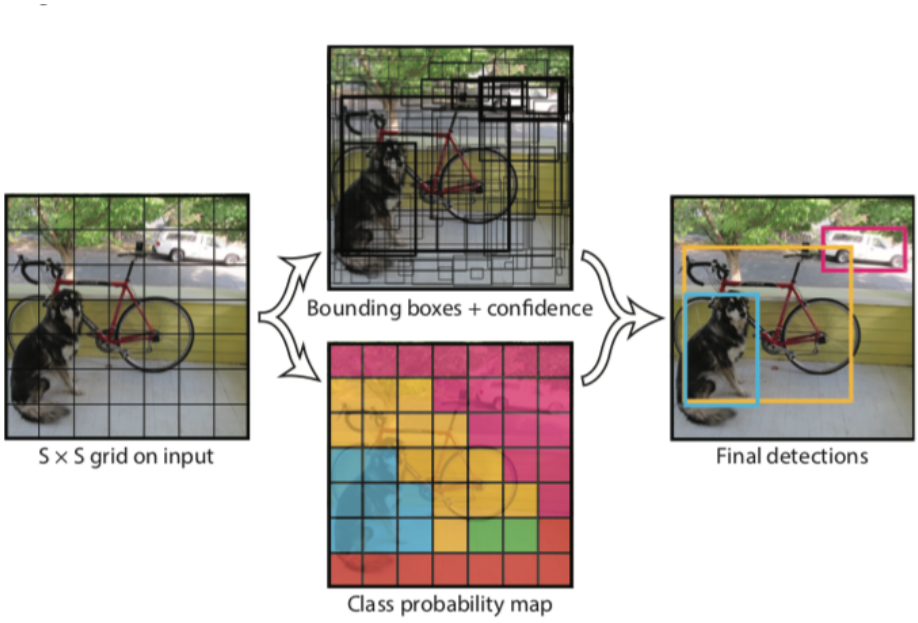
YOLO ujedinjuje odvojene komponente prijašnjih detektora u jednu neuronsku mrežu što ga čini mnogo bržim. Slika za detekciju skalira se tako da odgovara veličini ulaza i provuče se kroz duboku mrežu. Na temelju jednog prolaza model skupi značajke sa

slike koje su mu potrebne za predviđanje objekata te radi paralelne predikcije za sve klase u modelu.

Uspoređujući osnovni YOLO model s dotadašnjim detektorima on je bio mnogo brži od ostalih i imao mogućnost predviđanja lokalizacije objekata u realnom vremenu. Imao je daleko bolju točnost od modela koji su tada radili u realnom vremenu, međutim još uvijek slabiju od sporijih detektora.

Model

Rad osnovnog YOLO modela prikazan je slikom 3.5. On na početku podijeli sliku na $S * S$ jednakih ćelija. Ako centar objekta pada unutar ćelije, ona je odgovorna za njegovo detektiranje.



Slika 3.5: Princip rada modela YOLO. Na početku se slika podjeli na $S * S$ ćelija te se za svaku dodaju previđeni okviri i računaju pouzdanosti da se unutar njih nalazi objekt. Zatim odaberemo okvire s najvećom vjerojatnošću i dobijemo detektirane objekte (Joseph Redmon, 2015).

Svaka ćelija predviđa B okvira i rezultat pouzdanosti koji nam govori da li je unutar okvira objekt te koliko je model siguran u predikciju. Rezultat pouzdanosti računa se sljedećom jednadžbom:

$$Pr(Object) \cdot IOU_{truth} \tag{3.1}$$

Rezultat pouzdanosti jednak je umnošku vjerojatnosti da se u okviru nalazi objekt i $IOU - u$ (eng. *intersection over union*) između predviđenog okvira i stvarne lokacije objekta. IOU nam govori koliko se preklapaju predviđeni i stvarni okvir. Ako nema objekta unutar okvira rezultat pouzdanosti je 0.

Svaki predviđeni okvir sadrži 5 parametra koji ga opisuju: x, y, w, h i rezultat pouzdanosti. Točka (x, y) predstavlja centar okvira, dok parametri w i h relativnu širinu i visinu u odnosu na cijelu sliku. Svaka ćelija također predviđa C vjerojatnosti za svaku klasu u modelu. Ta vrijednost nam govori kolika je vjerojatnost da se unutar ćelije nalazi objekt određene klase. Svaka ćelija predviđa samo jedan set vjerojatnosti klase neovisno o broju okvira B. Zbog svih ovih parametra YOLO za svaku sliku stvara tenzor veličine $S * S * (B \cdot 5 + C)$. $S * S$ je broj ćelija, dok je $B \cdot 5$ broj predviđenih okvira po ćeliji i 5 parametra za svaki okvir, a C je broj klasa. Originalni YOLO u radu koristi vrijednosti $S = 7$ i $B = 2$.

Model nameće snažna ograničenja pošto svaka ćelija predviđa samo dva okvira koji mogu pripadati samo jednoj klasi. Zbog toga je dosta bitna veličina okvira jer se javlja poteškoća ako su objekti manji i nalaze se jedan pokraj drugoga. To dovodi do loših predikcija za manje objekte koji se nalaze u grupama.

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

Slika 3.6: Funkcija gubitka modela YOLO. Izražena je lokalizacijskim i klasifikacijskim kvadratnim pogreškama (Joseph Redmon, 2015).

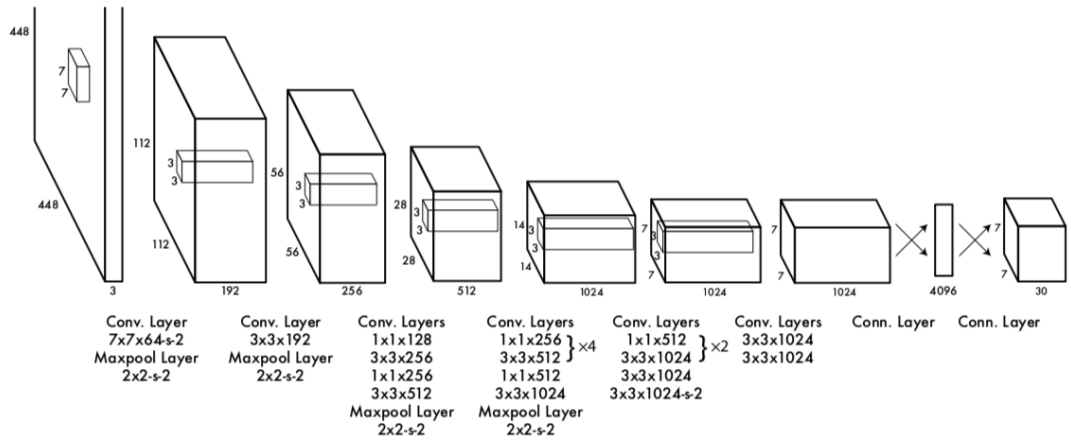
YOLO radi predikciju objekata u završnom sloju na temelju klasifikacijskih i lokalizacijskih pogrešaka okvira. Radi lakše optimizacije na izlaznom sloju funkcija gubitka (slika 3.6) izražena je kvadratnom pogreškom za razliku od SSD-a gdje je izražena običnim pogreškom. YOLO i SSD koriste iste pogreške za računanje funkcije gubitka, a to su pogreške između točne i predviđene pozicije centra okvira (x, y), re-

lativne širine w, relativne visine w, rezultata pouzdanosti $p(c)$ i klasifikacije objekata C. Međutim, ona nije savršeno usklađena sa ciljem da se poveća prosječna preciznost modela jer lokalizacijske pogreške izjednačuje s klasifikacijskim pogreškama što nije idealno. Također mnogo ćelija ne sadrži objekte zbog čega je rezultat klasifikacije negativan, odnosno jednak je nuli. Puno negativa potiskuje pozitivne rezultate što dovodi do neuravnoteženosti modela i treniranje divergira brzo.

Kako bi spriječio ovaj problem YOLO model povećava vrijednost pogreške dobivene predikcijom lokalizacije okvira, a smanjuje pogrešku klasifikacije gdje je rezultat pouzdanosti jednak nuli, odnosno u kojoj okvir ne sadrži objekt. To radi tako da u funkciju gubitaka uvodi dva parametra λ_{coord} i λ_{noobj}. Lokalizacijsku pogrešku množi faktorom $\lambda_{coord} = 5$, a klasifikacijsku pogrešku koja predviđa da nema objekta u okviru s $\lambda_{noobj} = 0.5$

Dizajn neuronske mreže

Duboka neuronska mreža koju koristi originalni YOLO model prikazana je slikom 3.7. Početni slojevi mreže iz slike izdvajaju korisne značajke, dok potpuno povezani slojevi predviđaju vjerojatnosti i lokaciju predikcije. Arhitektura YOLO mreže temelji se na GoogLeNet arhitekturi. Sadrži 24 konvolucijska sloja, nakon kojih slijede dva potpuno povezana sloja. Dodani su i redukcijski slojevi (eng. *bottleneck*) s jezgrom $1 * 1$ kako bi smanjili dimenzionalnost značajki prethodnih slojeva, nakon čega dolaze konvolucijski slojevi s jezgrom $3 * 3$. Prilikom treniranja mreže nad novim skupom podataka koristi se inicijalizacija težina dobivena učenjem na ImageNetu. Na taj način se ubrza učnenje i izbjegne se prenaučenost modela ako je mali skup podataka.



Slika 3.7: Arhitektura mreže korištena u YOLO modelu temeljena na GoogLeNet arhitekturi. Na kraju sadrži dva potpuno povezana slojeva za lokalizaciju objekata (Joseph Redmon, 2015).

Završni sloj koristi linearnu aktivacijsku funkciju, dok svi ostali slojevi koriste propusnu (eng. *leaky*) aktivacijsku funkciju:

$$\phi(x) = \begin{cases} x, & \text{ako } x > 0 \\ 0.1x, & \text{inače} \end{cases} \tag{3.2}$$

Trenirana je još jedna verzija duboke mreže koja je dizajnirana s namjerom da pomiče granice brze detekcije. Ona je nazvana „Brzi YOLO“ (eng. *Fast YOLO*). Izbacuje nekoliko konvolucijskih slojeva i njihove filtere iz originalne mreže te ima 9 slojeva umjesto 24. Osim veličine mreže ostali parametri ostaju isti kao kod YOLO mreže. „Brzi YOLO“ ima mogućnost da radi predikcije nad 155 slika u sekundi, dok običan obrađuje 45 slika u sekundi. Zbog velike brzine vrijednost mAP mu je manja za otprilike 10% testirana nad skupom PASCAL VOC (Joseph Redmon, 2015).

3.2.2. YOLO 9000: Better, Faster, Stronger

YOLO 9000 je nadogradnja na YOLO model. Dobio je naziv po tome što može detektirati više od 9000 različitih kategorija objekata. Također se smatra njegovom drugom verzijom, pa ga neki zovu YOLOv2.

YOLOv1 je uspio riješiti problem brze detekcije, no imao je razne nedostatke naspram ostalih suvremenih detektora. U usporedbi s njima pokazalo se da čini značajan broj pogrešaka lokalizacije i da ima relativno mali odziv. Zbog toga javila se potreba za nadogradnjom kako bi se poboljšao odziv i lokalizacija objekata uz održavanje točnosti klasifikacije. Bolje implementacije u računalnom vidu idu prema većim i dubljim mrežama kako bi povećale točnost. Međutim, to bi usporilo detekciju, pa YOLOv2 umjesto povećanja mreže uvodi promjene koje pojednostavljuju mrežu i čine je jednostavnijom za učenje. Koristi nove koncepte preuzetih iz tadašnjih najnovijih istraživanja i uvodi ih u YOLOv1. Svi novi koncepti i rezultati detekcije njihovom primjenom na skupu PASCAL VOC 2007 prikazani su u tablici na slici 3.8, a bit će objašnjeni u nastavku.

YOLOv2 uvodi normalizaciju po grupi (eng. *batch normalization*) što poboljšava konvergenciju, istodobno eliminirajući potrebe za drugim oblicima regularizacije. Dodavanjem normalizacije po grupi na sve konvolucijske slojeve YOLO modela dobivamo povećanje mAP vrijednosti za 2 %.

Drugo poboljšanje koje se uvodi u ovaj model je povećanje rezolucije ulaza u mrežu (eng. *hi-res classifier*). YOLOv1 trenira mrežu na rezoluciji $224 * 224$ te ju

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Slika 3.8: Rezultati detekcije YOLO modela s novim konceptima dodanim u drugoj verziji nad skupom PASCAL VOC 2007. Jedini pad srednje prosječne preciznosti je kod dodaje referentnih okvira, ali su oni mnogo povećali odziv zbog kojeg su moguća bila daljnja poboljšanja (Joseph Redmon, 2016).

povećava na $448 * 448$ prilikom detekcije. YOLOv2 je povećao ulaznu rezoluciju prilikom treniranja na $448 * 448$ i time povećao mAP za 4%.

Sljedeća stvar koja se uvodi u YOLOv2 su referentni okviri (eng. *anchor boxes*). Izbacuju se potpuno povezani slojevi te umjesto njih referentni okviri rade predikciju lokalizacije objekata kao što to rade i kod modela SSD. Referentni okviri postavljaju se na svaku lokaciju u izlaznnoj mapi značajki i zaduženi su za detektiranje objekata koji imaju centar u toj lokaciji. Sa slike 3.8 vidimo da koristeći referente okvire točnost modela se malo smanjuje, ali se povećava odziv. Iako se točnost smanjuje povećanje odziva znači da model ima više prostora za napredak.

Iz mreže se izbacuje i jedan sloj sažimanja kako bi se dobio izlaz mreže veće rezolucije. Nadalje se smanjuje rezolucija ulaza u mrežu prilikom detekcije na $416 * 416$. To se radi iz razloga da se dobije neparna rezolucija na izlaznoj mapi značajki kako bi postojao jedinstven centar. Veliki objekti često zauzimaju središte slike pa je dobro imati jedinstveni centar koji ih predviđa. YOLOv2 arhitektura smanjuje sliku za faktor 32, tako da pomoću ulazne slike $416 * 416$ dobijemo na izlazu mapu značajki dimenzije $13 * 13$. Nova mreža (eng. *new network*) povećava srednju prosječnu preciznost za 0.4%.

Uvođenjem referentnih okvira u YOLO model suočavamo se s dva problema. Jedan je taj što su dimenzije tih okvira zadane ručno. Dimenzije referentnih okvira moraju se podesiti ovisno o arhitekturi mreže i objektima na skupu slika za učenje. Taj problem rješavamo algoritmom k najbližih susjeda koji izračunava pogodne dimenzije

referentnih okvira ovisno o modelu nakon čega se pokrene treniranje mreže . Drugi problem je nestabilnost modela koji se javlja zato što referentni okviri mogu završiti na bilo kojoj lokaciji na slici, neovisno o tome u kojoj ćeliji predviđaju objekt. To se protivi YOLO načinu rada te se uvode ograničenja predikcije (eng. *location prediction*) ovisno o ćeliji u kojoj referentni okvir mora imati svoj centar. Rješavanjem ovih problema dobijemo 5% povećanje mAP parametra u usporedbi s modelom s običnim referentnim okvirima. Pošto YOLOv2 mijenja originalnu implementaciju referentnih okvira i postupak njihovog korištenja naziva ih tzv. apriornim okvirima (eng. *bounding box priors, dimension priors*).

U YOLO modelu predikcije se temelje na izlaznoj mapi značajki. Kako bi prikupili više informacija o objektima u izlaznom sloju YOLOv2 dodaje prolazni (eng. *pass-through*) sloj u mrežu. On povezuje značajke iz sloja veće rezolucije sa slojem niže rezolucije tako da susjedne značajke slaže u različite kanale. To dovodi do povećanja performansa za 1%.

Kako bi podržali robusnost modela na slikama različitih veličina u YOLOv2 se uvodi promjena rezolucije ulaza u mrežu tijekom učenja (eng. *multi-scale*). Nakon svakih 10 grupa (eng. *batch*) mreža nasumično bira nove dimenzije ulaza. Pošto se YOLO mreža smanjuje za faktor 32, kao ulaznu rezoluciju uzimamo njegove višekratnike: {320, 352, . . . , 608}. Najmanja rezolucija je $320 * 320$, dok je najveća $608 * 608$. Na ovaj način omogućuje se da jedna mreža može predviđati predikcije objekata na različitim dimenzijama slika.

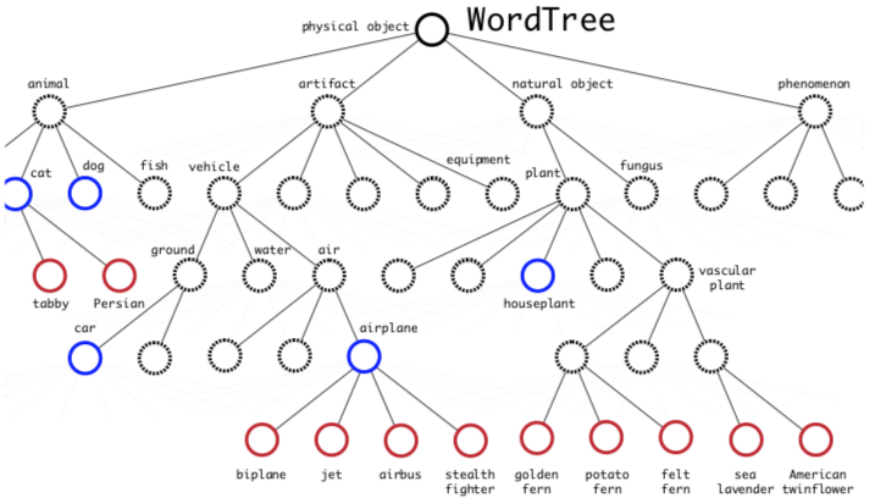
Detection Frameworks	Train	mAP	FPS
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

Slika 3.9: Rezultati detekcije istog YOLOv2 modela s različitim ulaznim rezolucijama nad skupom podataka PASCAL VOC (Joseph Redmon, 2016).

Arhitektura mreže koja je nastala uvođenjem svih novih promjena nazvana je Darknet-19. Ima 19 konvolucijskih slojeva i 5 slojeva sažimanja (Joseph Redmon, 2016). Prilikom učenja mreža se inicijalizira na težine dobivene učenjem na ImageNetu. S novom mrežom i ostalim promjenama YOLOv2 dolazi do značajnog povećanja točnosti modela. Osim toga možemo pokrenuti učenje na različitim rezolucijama ulaza što nam daje mogućnost podešavanje brzine i točnosti modela ovisno o tome što nam je bitnije.

Rezultati detekcije s različitim rezolucijama pokazane su na slici 3.9.

YOLOv2 uvodi još jednu zanimljivu promjenu koja iskorištava veliku količinu klasifikacijskih podataka koje model već ima i koristi ih kako bi proširio opseg postojećeg sustava za detekciju. Koristi hijerarhijsku strukturu objekata za klasifikaciju (slika 3.10) koji omogućuje jednostavno kombiniranje različitih skupova podataka. Na slici su prikazani objekti kobinirani iz COCO i ImageNet skupa podataka.



Slika 3.10: Primjer hijerarhijske strukture podataka i spajanje različitih skupova podataka. Plavom bojom su označeni objekti iz COCO, a crvenom iz ImageNet skupa (Joseph Redmon, 2016).

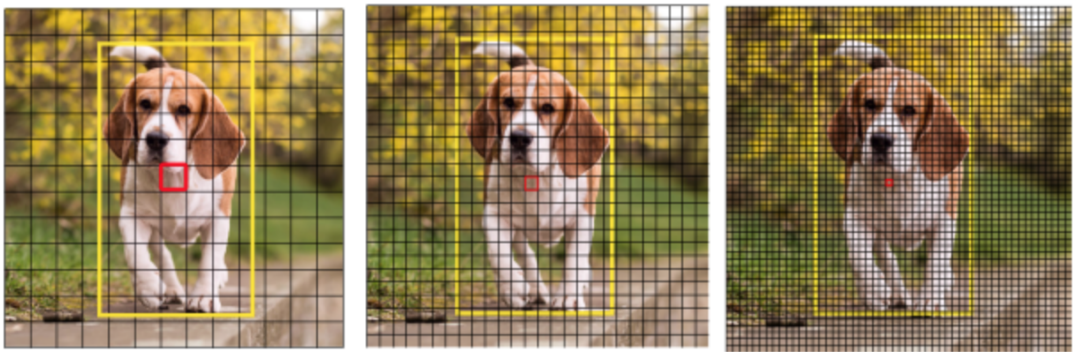
3.2.3. YOLOv3: An Incremental Improvement

YOLOv3 je moderan detektor razvijen ove godine. On uvodi mnogo manjih promjena u YOLOv2 model kako bi povećao točnost koristeći najnovije metode u dubokom učenju. Neke od najbitnijih bit će opisane u nastavku.

YOLOv3 mijenja način predviđanja vjerojatnosti da se objekt nalazi unutra okvira tako da koristi logističku regresiju. Vjerojatnost bi trebala biti 1 ako apriorni okvir preklapa originalan objekt više od drugih okvira. Ako apriorni okvir nije najbolji ali preklapa objekt više od nekog praga ignoriramo njegovu predikciju. U originalnom radu koristi se prag 0.5. Ovaj pristup je sličan eliminaciji nemaksimalnih elemenata u prijašnjim verzijama te njime sustav primjenjuje gubitak samo na apriornom okviru koji najbolje poklapa objekt, odnosno svaki objekat može imati samo jedan okvir. Korištenje logističke regresije je pogodno ako apriorni okvir nije dodijeljen objektu, zato što on ne uzrokuje nikakav gubitak za predviđanje lokalizacije i klasifikacije.

Sljedeće se uvodi višestruka klasifikacija u predviđanju objekata unutar okvira. Izbacuje se softmax funkcija za klasifikaciju pošto se pokazalo da nije nužna za dobre performanse modela te se umjesto višerazredne klasifikacije (eng. *multi-class*) koristi klasifikacija oznaka (eng. *multi-label*). Tijekom treniranja koristi se gubitak entropije za predviđanje klasa. Ove promijene pomažu kod klasifikacije kompleksnijih skupova podataka kao što je skup podataka Open Images. U njemu ima mnogo preklapajućih klasa, na primjer osoba i žena. Softmax funkcija pretpostavlja da svaki okvir može imati samo jednu klasu što često nije slučaj. Pristup s više klasa bolje predstavlja podatke.

YOLOv3 radi predikcije na temelju 3 skale, odnosno koriste se 3 mape značajki različitih dimenzija za predviđanje objekata. Predikcije se rade na mapama čija je dimenzija umanjena rezolucijom ulaza za faktore 32, 16 i 8. Ako je ulaz u mrežu $416 * 416$ predikcije se rade na slojevima dimenzija $13 * 13$, $26 * 26$ i $52 * 52$ (slika 3.11). Metoda koja se ovdje spominje je dekonvolucija, a nju koristi već spomenuti DSSD model. YOLOv3 na izlazni sloj YOLOv2 modela dodaje novi sloj koji je povećan faktorom 2 i spojen s prijašnjom mapom značajki jednakih dimenzija. Nakon njega dodani su još nekoliko konvolucijskih slojeva za obradu kombinirane mape značajki nakon kojih se vrši detekcija. Ovaj postupak se ponavlja još jednom samo za veću skalu. YOLOv3 koristi 9 apriornih okvira i to 3 za svaku od skala. Korištenjem dekonvolucije dobivamo više korisnih semantičkih informacija te nam ona omogućuje bolje detektiranje manjih objekata.



(a) 13 * 13 **(b)** 26 * 26 **(c)** 52 * 52

Slika 3.11: Prikaz detekcije objekta na različitim dimenzijama mapa značajki korištenjem dekonvolucije u YOLOv3 modelu. Crvenom bojom označena je ćelija zadužena za detekciju objekta u pojedinoj mapi značajki. Širinu i visinu žutog okvira predviđamo kao odstupanje od središta ćelije, dok centar okvira pomoću sigmoidalne funkcije.

Nova mreža koju koristi YOLOv3 je hibrid između prijašnje mreže u YOLOv2

modelu Darknet-19 i ResNet mreže. Ona ima uzastopne $3 * 3$ i $1 * 1$ konvolucijske slojeve, sadrži rezidualne veze i znatno je veća. Mreža ima 53 konvolucijska sloja, pa je zbog toga nazvana Darknet-53. Njezinu strukturu možemo vidjeti na slici 3.12. Ova nova mreža je mnogo moćnija od Darknet-19, ali nema toliko slojeva kao ResNet mreža zbog čega je i dalje dosta brza.

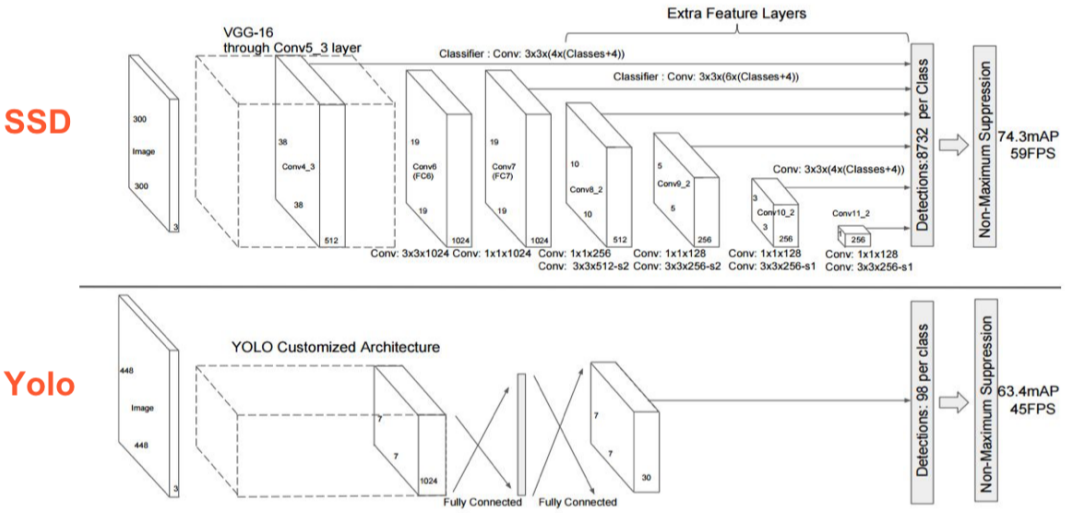
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Slika 3.12: Arhitektura mreže Darknet-53 (Joseph Redmon, 2018).

Još su se mnogo novih koncepata pokušali implementirati u YOLOv3, ali njihovom upotrebom nije došlo do poboljšanja modela. YOLOv3 je brz i točan detektor. Daje bolju točnost detekcije u odnos na prijašnju verziju. Za $IOU => 0.5$ daje super rezultate, no ima problem s novijom metrikom kada je IOU između 0.5 i 0.95.

3.3. Usporedba brzih detektora

Pojavom dubokih modela i velikog porasta u performansama grafičkih kartica, razvitak novih detektora krenuo je naglom brzinom. Godine 2015. pojavio se novi pristup u detekciji koji je omogućio predikciju objekata u realnom vremenu, a da je pokazivao dobre rezultate. Taj pristup je radio predikcije na temelju jednog unaprijednog prolaza kroz mrežu, a model koji ga je prvi predstavio bio je YOLO. Ovo je bio veliki uspjeh detektora te se pokrenula nova grana istraživanja. Nedugo zatim pojavio se SSD model koji je ubrzao vrijeme obrade i povećao točnost detekcije.



Slika 3.13: Usporedba arhitektura mreže SSD i YOLO modela. SSD koristi baznu arhitekturu VGG-16, dok YOLO koristi GoogLeNet arhitekturu. (Wei Liu, 2015).

Glavna razlika između YOLO i SSD modela je taj što YOLO model radi predikciju objekata pomoću dva potpuno povezana sloja na kraju mreže, dok SSD uvodi referentne okvire za predikciju. Potpuno povezani slojevi mogu biti dobri za predikciju objekata na skupu podataka kao što je Pascal, ali za slike s mnogo malih objekata dobije se bolji rezultat korištenjem referentinih okvira. SSD postavlja referentne okvire na svakoj lokaciji unutar nekoliko mapa značajki različitih dimenzija. Pomoću njih detektiraju se objekti na slici. YOLO i SSD koriste drugačiju arhitekturu mreže (slika 3.13). Mreža u YOLO modelu inspirirana je GoogLeNet arhitekturom, dok je kod SSD-a VGG-16 arhitekturom. Originalni SSD ima manju rezoluciju na ulazu u mrežu te nema potpuno povezane slojeve na kraju mreže zbog čega mu je brži prolazak, odnosno treba mu manje vremena za predikciju objekata. Na slici 3.14 se nalaze i rezultati testiranja YOLO i SSD modela nad skupom slika PASCAL VOC 2007. SSD ima vrijednost parametra mAP veću za otprilike 11% te obradi 59 slika u sekundi, dok

YOLO 45.

Sljedeće godine YOLO je izdao svoju drugu verziju. U njoj je uzeo neke korisne metode iz SSD pristupa. Izbacio je potpuno povezane slojeve i uveo referentne okvire za predikciju objekata. Smanjio je broj slojeva duboke mreže osnovnog YOLO modela i uveo moderne metode koje su se u međuvremenu razvile. Nova mreža nazvana je Darknet-19. Zbog svih promjena postao je brži i točniji od SSD modela.

Početkom 2017. nastala je nova metoda detekcije koja spaja SSD pristup i ResNet-101 arhitekturu. Tako dobiveni model je proširen dekonvolucijskim slojevima koji pomažu prilikom detekcije manjih objekata. Ova metoda nazvana je DSSD i mnogo je pretekla točnost prijašnjih modela.

Ove godine pojavio se još bolji detektor. To je treća verzija YOLO modela. Ona uzima neka poboljšanja iz DSSD modela kao što je dekonvolucija i duboku mrežu prilagođava ResNet arhitekturi. Koristi logističku regresiju za klasifikaciju okvira i uvodi još neke sitne promjene. U tom modelu razvijena je mreža Darknet-53.

Model	Train	mAP	FPS
SSD300	COCO trainval	41.2	46
SSD500	COCO trainval	46.5	19
YOLOv2 608x608	COCO trainval	48.1	40
Tiny YOLO	COCO trainval	23.7	244
SSD321	COCO trainval	45.4	16
DSSD321	COCO trainval	46.1	12
R-FCN	COCO trainval	51.9	12
SSD513	COCO trainval	50.4	8
DSSD513	COCO trainval	53.3	6
FPN FRCN	COCO trainval	59.1	6
Retinanet-50-500	COCO trainval	50.9	14
Retinanet-101-500	COCO trainval	53.1	11
Retinanet-101-800	COCO trainval	57.5	5
YOLOv3-320	COCO trainval	51.5	45
YOLOv3-416	COCO trainval	55.3	35
YOLOv3-608	COCO trainval	57.9	20
YOLOv3-tiny	COCO trainval	33.1	220

Slika 3.14: Rezultati najbržih detektora na skupu COCO. YOLOv3 daje najbolje rezultate u ovisnosti o brzini izvođenja (Redmon, 2018).

Osim YOLO i SSD pristupa razvijeni su još neki brzi modeli u moderno vrijeme. Neke koje treba spomenuti su Retinanet i FPN FRCN (eng. *Feature Pyramid Networks for Fast R-CNN*), koji svojom arhitekturom postižu jako dobre rezultate. Rezultati

najbržih detektora današnjice prikazani su na slice 3.14. Korištene su varijacije nekih modela te su prikazane srednje prosječne preciznosti mAP i brzine modela FPS. Na slici vidimo kako nove verzije modela doprinose točnosti. Jedni od najtočnijih modela su FPN FRCN, Retinanet-101 i YOLOv3-608 ($608 * 608$ rezolucija na ulazu u mrežu). Iako su im točnosti slične YOLOv3-608 uspije obraditi 20 slika u sekundi dok ostali oko 6 slika. Ako želimo detekciju u realnom vremenu treba obraditi videe koji znaju imati između 25-50 slika u sekundi. Za tu prigodu najbolji je YOLOv3 model pošto može brzo detektirati objekte s velikom točnošću.

4. Korišteni algoritmi

4.1. Generiranje referentnih okvira

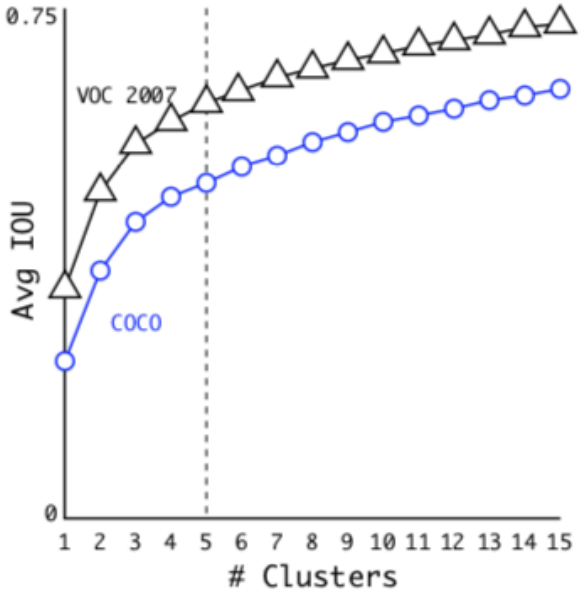
Kako bi se dobila učinkovita predikcija objekata na vlastitom skupu za učenje treba izabrati dimenzije referentnih okvira korištenih u YOLO modelu. Njihove početne vrijednosti su zadane ručno te ne moraju davati najbolja rješenja na našem skupu. Pošto su objekti obrađeni u ovom radu mali trebamo namjestiti referente okvire kako bi dobili dobru detekciju objekata.

Dimenzije referentnih okvira dobijemo algoritmom grupiranja k najbližih susjeda nad okvirima objekata iz skupa za učenje. Ispostavilo se da je on učinkovit u odabiru njihovih dimenzija. Ako koristimo standardni algoritam k najbližih susjeda s Euklidskom udaljenošću veći okviri generiraju veće pogreške u odnosu na manje. Međutim, ono što mi želimo su dimenzije početnih okvira koje vode do dobrog IOU rezultata koji su neovisni o veličini okvira. Zbog toga za udaljenost koristimo sljedeću metriku:

$$d(okvir, centroid) = 1 - IOU(okvir, centroid) \tag{4.1}$$

Broj referentnih okvira k biramo sami, što ih je više točnost modela je veća ali se smanjuje brzina i koristi više memorije. U YOLOv2 radu napravljen je eksperiment u kojem su autori pokrenuli algoritam generiranja referentnih okvira za različiti broj k i iscrtali prosječan IOU s najbližim centroidom (slika 4.1). Izabran je $k = 5$ kao najbolja opcija jer daje dobre rezultate, a model nije prekompleksan i ima veliki odziv. Algoritam generira referentne okvire od kojih je nekoliko manjih i širokih, a nekoliko viših i tanjih (Joseph Redmon, 2016).

Na slici 4.2 su uspoređeni prosječni IOU rezultati modela YOLOv2 u kojem su referentni okviri odabrani ručno i u kojem su generirani algoritmom k najbližih susjeda. Vidimo da 5 okvira generirani algoritmom daju sličnu točnost kao 9 okvira odabranih ručno. Dok s 9 okvira generiranih algoritmom dobivamo mnogo veći IOU. Ovim rezultatima vidljivo je da generiranjem referentnih okvira metodom k najbližih susjeda



Slika 4.1: Prikaz prosječnog IOU rezultata ovisno o broju referentnih okvira testiranim na skupu PASCAL VOC 2007 i COCO (Joseph Redmon, 2016).

dobijemo bolju točnost modela i olakšavamo treniranje mreže. Kod eksperimentalnog dijela u ovom diplomskom radu korišten je $k = 5$ za broj referentnih okvira.

Box Generation	#	Avg IOU
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

Slika 4.2: Usporedba prosječnog IOU rezultata ovisno o modelima koji referentne okvire izabiru ručno i korištenjem metode k najbližih susjeda (Joseph Redmon, 2016).

4.2. Predikcija sa samo jednim unaprijednim prolazom

U ovom radu za detekciju igrača na nogometnom susretu koristi se YOLOv2 model. Kao što je poznato iz prijašnjeg poglavlja znamo da taj model radi predikcije objekata na temelju jednog unaprijednog prolaza kroz duboku mrežu. Uzeta je implementacija originalnog YOLOv2 modela pretvorenog u Python programski jezik i dodane su neke nove stvari kako bi se model prilagodio problemu ovog rada.

Nakon što se generiraju dimenzije referentnih okvira, one se postave u programu i može početi treniranje mreže. Zbog velikog broja referentnih okvira i pogađanja regija gdje bi mogao biti objekt postoji puno negativa u modelu. Kako bi se oni smanjili koristi se algoritam eliminacije nemaksimalnih odziva. Kada mreža završi treniranje i postavi svoje parametre spremna je za detekciju objekata na novom skupu slika. Prije početka detekcije model određenim algoritmima mijenja rezoluciju slike kako bi se prilagodila dimenzijama ulaza u mrežu. Slika mora biti iste dimenzije kao i ulaz kako bi mogla cijela proći kroz mrežu Ako imamo veću sliku za detekciju ovim postupkom ona gubi neke značajke, pa je korisno povećati dimenziju ulaza u mrežu. Obrnuto, ako imamo manju sliku ona se poveća prilikom ulaska u mrežu, a time dobijemo jače semantičke značajke i predikcija objekata bit će preciznija. To možemo vidjeti i na rezultatima detekcije sa slike 3.9 gdje se koristi isti model, a rezolucija ulaza se mijenja. Općenito je bolje povećati rezoluciju ulaza u mrežu ako želimo bolju preciznost modela, no to dovodi do usporavanja brzine detekcije.

Kada se slika provuče kroz mrežu na izlazu se dobije mapa značajka malih dimenzija pomoću koje se radi predikcija objekata. U svakoj lokaciji mape značajka postavljaju se referentni okviri koji predviđaju objekte. Referentni okviri neke lokacije su odgovorni naći objekt ako je njegov centar unutar te lokacije. Svaki okvir sadrži vjerojatnost da se u njemu nalazi pojedina klasa. Na kraju se za predikciju objekata uzimaju okviri koji imaju vjerojatnost neke klase veću od zadane vrijednosti. Pošto se dešava da više okvira predvidi isti objekt, a oni se većim dijelom preklapaju radi se eliminacija okvira nemaksimalnih vrijednosti za $IOU = 0.1$. Ako je IOU manji to znači da se okviri moraju manje preklapati da bi zadovoljili ovaj kriterij. IOU je dosta nizak zato što su objekti u ispitnom skupu mali, pa promjena okvira za mali pomak dosta smanjuje njihovo preklapanje. Preostali okviri predstavljaju predikcije objekata.

5. Detekcija objekata na sportskom susretu

5.1. Video za detekciju

U ovome diplomskom radu radi se lokalizacija objekata u realnom vremenu metodom predikcije samo jednim unaprijednim prolazom. U tu svrhu koristi se snimka prvih nekoliko minuta nogometnog susreta 1. HNL lige između Dinama i Hajduka. Snimka je statična i snimana iz ptičje perspektive. Njezina je razlučivost $1632 * 288$, traje 5 minuta i koristi 25 sličica u sekundi. Primjer isječka iz snimke prikazan je na slici 5.1.

Slika 5.1: Isječak iz videa koji se koristi za detekciju objekata.

Na snimci je cijelo vrijeme vidljivo 11 igrača iz svake momčadi i jedan sudac. Zbog toga su izdvojene tri klase koje će se detektirati, to su igrači tima A, igrači tima B i sudac. Uz video postoji i formirana datoteka koja sadrži položaj svakog igrača na pojedinoj slici u videu. Iz nje izvađeni su odgovarajući podaci potrebni za učenje i testiranje detektora. Snimka sadrži 7514 sličica od kojih prvih 3000 predstavlja skup za učenje, dok se ostale koriste kao skup za testiranje.

Kao što je vidljivo na slici objekti koji se detektiraju jako su mali što otežava detekciju. Teško je detektirati objekte koji sadrže samo par piksela te čine mali postotak slike. Također vidimo da na snimci ima dosta sadržaja sa strane na kojem se objekti koje detektiramo skoro nikada ne nalaze. Oni se mogu zanemariti prilikom učenja kako bi dobili točnije rezultate.

Predikcija igrača na snimci nogometne utakmice ima još nekoliko problema zbog

kojih se otežava detekcija. Te probleme možemo vidjeti na slici 5.2. Prvi problem je taj da veličina igrača nije konzistentna. Igrači bliže kameri su mnogo veći od onih dalje te to izaziva problem pri učenju mreže. Međutim, današnji detektori ovo uspješno rješavaju korištenjem konvolucijskih slojeva i slojeva sažimanja. Sljedeći problemi koji se javljaju su stapanje igrača s pozadinom i njihovo međusobno prekrivanje. U tim situacijama je teško prepoznati i razlikovati objekte. Ove probleme nije jednostavno riješiti te se zbog njih često javljaju pogreške prilikom detekcije objekata.

(a) Nekonzistentna vličina igrača

(b) Stapanje igrača s pozadinom

(c) Međusobno prekrivanje igrača

Slika 5.2: Problemi sa snimke zbog kojih se otežava detekcija.

5.2. Izrada anotacija

Kako bi pokrenuli YOLO model nad vlastitim skupom slika za učenje i testiranje treba mu predati putanju do datoteke sa slikama i anotacijama. Skup slika dobili smo tako da smo iz videa izvadili svaku sličicu i spremili je u datoteku. Svaka slika u skupu treba imati svoju anotaciju koja ju opisuje. To je zapravo XML datoteka pomoću koje YOLO model saznaje informacije o slici. Pomoću nje možemo očitati broj objekata koji se nalaze na slici, njihovu pripadnost određenoj klasi i točnu lokaciju. Primjer jednostavne anotacije prikazan je u nastavku.

```
<annotation>
  <filename>img_2.jpg</filename>
  <owner>
    <name>Damjan Miko</name>
  </owner>
  <size>
    <width>1632</width>
    <height>288</height>
    <depth>3</depth>
  </size>
  <object>
    <name>soccerPlayerA</name>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>424</xmin>
      <ymin>86</ymin>
      <xmax>429</xmax>
      <ymax>102</ymax>
    </bndbox>
  </object>
</annotation>
```

Primjer koda 5.1: Prikaz jednostavne anotacije

Na početku anotacije navedeni su osnovni podaci o slici koju predstavlja. Prvo je ime slike pomoću kojeg ju algoritam može naći u skupu svih slika. Slijedi ime vlasnika slike, zatim dimenzije slike kao što je rezolucija i broj kanala. Rezolucija je u ovom primjeru $1632 * 288$, a broj kanala 3, odnosno slika je u boji. Jednobojne slike imaju 1 kanal (siva boja), dok slike u boji 3 kanala (RGB slika). Mogu se među ove osnovne podatke dodati informacije o skupu slika i anotacija, ali oni nisu potrebni.

Nakon osnovnih informacija slijedi popis svih objekata koji se nalaze na slici. Svaki objekt sadrži ime klase kojoj pripada, faktor odrezanosti i teškoće detekcije te lokaciju okvira. Faktor odrezanosti govori koliki je dio objekta prekriven, a faktor teškoće koliko ga je teško detektirati. Oni mogu biti između 0 i 1, ako ne znamo koliki su postavljamo ih na 0. Lokacija okvira sadrži početne i krajnje pozicija točaka na x i y osi. Iz primjera anotacije se može iščitati da se na slici nalazi jedan igrač iz tima A u

okviru između točaka $(424, 86)$ i $(429, 102)$

Uz video nogometne utakmice postoji i datoteka koja sadrži položaj svakog igrača na pojedinoj slici. Format datoteke nije dobar za korištenje u YOLO modelu. Međutim, pomoću nje napravili smo anotacije koje su pogodne YOLO modelu. Implementirali smo funkciju koja parsira formiranu datoteku i iz nje vadi bitne elemente potrebne za izradu anotacija. Funkcija sprema anotacije u definiranom XML formatu za svaku sličicu. Kada se izdvoje slike i anotacije u svoje datoteke u odgovarajućem formatu spremni smo za učenje YOLO modela nad vlastitim skupom podataka. Vjerojatno rezultati detekcije neće biti odmah najbolji, ovisno o težini vlastitog skupa podataka, no promjenom nekih hiperparametara i malih promjena u mreži može se doći do odličnih rezultata.

6. Programska implementacija i vanjske biblioteke

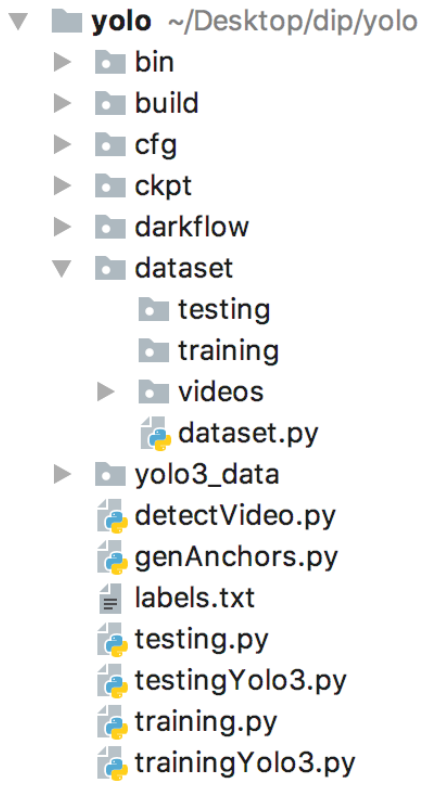
6.1. Programska implementacija

U okviru rada koristi se YOLO model za detekciju igrača na nogometnoj utakmici. Kao bazna implementacija YOLO i YOLOv2 modela klonirali smo Darkflow projekt sa Githuba (Trieu, 2017), dok je za YOLOv3 model kloniran keras-yolo3 projekt (Qqwweee, 2018). Originalna YOLO implementacija pisana je u C programskom jeziku zbog brzine izvođenja. Darkflow je zapravo pretvorba originalne implementacije u Python 3.6 koristeći razvojni okvir Tensorflow. U njemu su implementirane prve dvije verzije YOLO modela. Za potrebe ovog rada korištena je implementacija YOLOv2 modela s dodatnim promjenama. Povećana je rezolucija ulaza i parametar rastresenja podataka. Veličina grupe i rezolucija ulaza prilagodile su se memorijskom prostoru grafičke kartice. Izračunate su dimenzije referentnih okvira prilagođene za vlastiti skup podataka. Ovim promjenama mnogo se poboljšava detekcija objekata na vlastitom skupu podataka.

Eksperimentalne rezultate proveli smo na serverskom računalu zbog značajnog bržeg izvođenja. Serversko računalo je pokrenuto na Google Cloud platformi zato što ona nudi besplatnih 300 dolara za korištenje svojih usluga prilikom registracije. Taj iznos je bio dovoljan za potrebe ovog rada. Serversko računalo ima operacijski sustav Linux Ubuntu s Nvidia grafičkom karticom Tesla K80. Ono omogućuje treniranje modela u jednom danu.

6.1.1. Struktura programske implementacije

Program je pisan u programskom jeziku Python 3.6 u okruženju PyCharm. Njegova struktura prikazana je na slici 6.1. U nastavku ćemo detaljnije proći kroz nju i objasniti što se nalazi u pojedinim datotekama te kako koristiti program.



Slika 6.1: Struktura programske implementacije.

Na početku je kazalo `bin` koje sadrži težine prednaučene na skupu podataka ImageNet za modele YOLO, YOLOv2 i YOLOv3. One nam pomažu prilikom učenja vlastite mreže tako da postavimo početne težine i time ubrzamo konvergenciju kao i učenje. Ako ne bi postavili početne težine mreže učenje bi trajalo mnogo duže i rezultat bi bio znatno lošiji jer bi se model prenaučio zbog malog skupa podataka za učenje.

Sljedeća bitna komponenta je `cfg` datoteka u kojoj se nalaze implementirane duboke mreže koje su se koristile u eksperimentu. One su jednostavno prikazane u datotekama s `.cfg` ekstenzijom, a detaljnije ćemo ih objasniti u sljedećem poglavlju. Slijedi `ckpt` datoteka, a u nju se zapisuju podaci o stanju mreže tijekom učenja modela. Svakih određenih broj koraka zapisuje se stanje mreže što nam omogućuje prekidanje učenja u nekom trenutku bez gubitaka podataka. Također koristimo ove datoteke prilikom detekcije objekata gdje imamo mogućnost izabrati parametre mreže u nekom od koraka učenja.

U datoteci `darkflow` se nalaze glavni dijelovi programa, a to je implementacija svih YOLO modela. Oni su zapravo pretvorba originalnog YOLO modela pisanog u programskom jeziku C u Python i Tensorflow. U tom dijelu nalaze se svi algoritmi

dubokog učenja koji se koriste u ovim modelima te primjeri učenja i testiranja modela. U njoj se nalaze i funkcije koje parsiraju `cfg` datoteku neuronske mreže i uzimaju podatke iz nje kako bi ih koristili u programskom dijelu.

Kazalo `dataset` predstavlja skup podataka nogometne snimke. U njoj se nalaze skupovi slika za treniranje i testiranja te njihove anotacije. Ona još sadrži originalni video i rezultate detekcije objekata na slikama i videu. U modulu `dataset.py` se nalaze funkcije koje generiraju skupove slika pomoću videa i rade anotacije za njih.

U kazalu `yolo3_data` nalaze se datoteke za rad YOLOv3 modela. Nakon nje sljede klase potrebne za učenja mreže i predikcije objekata. Pokretanjem modula `genAnchors.py` dobijemo dimenzije referentnih okvira prilagođenih za pojedinu mrežu i skup podataka kojeg koristimo. Modul `training.py` i `trainingYolo3.py` pokreće treniranje mreže. Prije pokretanja treba izabrati mrežu i parametre za treniranje. U tekstualnoj datoteci `label.txt` nalazi se popis svih vrsta objekta koji se detektiraju. Ostaju još moduli `testing.py`, `testingYolo3.py` i `detectVideo.py`. Oni služe za predikciju objekata na slikama, odnosno videu i evaluiranju detektora s osnovnim mjerama.

6.1.2. Implementacija duboke neuronske mreže

Kao što je spomenuto u prijašnjem poglavlju konvolucijska mreža može se jednostavno dizajnirati pomoću cfg datoteke. Mogu se dizajnirati mreže iz početka ili možemo koristiti gotove mreže koje nam nudi YOLO model. One se mogu jednostavno prilagoditi problemu kojeg rješavaju. U nastavku su opisani dijelovi cfg datoteke i objašnjeno je kako dodati nove slojeve u nju.

```
[net]

batch=4
subdivisions=2
width=2304
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
learning_rate=0.001
```

(a)

```
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2
```

(b)

```
[region]
anchors =  0.16,0.47,
bias_match=1
classes=3
coords=4
num=5
softmax=1
jitter=.9
```

(c)

Slika 6.2: Implementacija duboke neuronske mreže pomoću cfg datoteke.

Na slici 6.2 su izvađeni dijelovi iz `yolov2.cfg` datoteke koja opisuje konvolucijsku mrežu korištenu u YOLOv2 modelu Darknet-19. Promijenjenili smo neke parametre i prilagodili ih za detekciju igrača na nogometnoj utakmici. Glavne komponente mreže u cfg datoteci pišu se unutar uglatih zagrada. Na početku je uvijek `[net]` komponenta (slika 6.2a) u kojoj se nalaze osnovni hiperparametri mreže. U njoj se zadaju visina i širina ulaza u duboku mrežu, veličina grupa za učenje, broj kanala, stopa učenja itd. Osim ovih osnovnih hiperparametara postoje i oni koji su manje poznati. Neki od njih su zasićenost, izloženost i vrijednost nijansi (eng. *saturation, exposure, hue*) koji predstavljaju raspone promjena boja slika tijekom učenja. Što su im vrijednosti veće to će se neuronska mreža više prilagoditi na promjene svjetlosti i boje objekata.

Nakon postavljanja osnovnih parametra mreže slijedi njezina izgradnja. Slojevi mreže dodaju se jedan ispod drugoga u slijedu izvođenja Na slici 6.2b prikazan je jedan konvolucijski sloj i jedan sloj sažimanja. Vidimo da je njihova implementacija dosta jednostavna. Komponenta `[convolutional]` dodaje konvolucijski sloj u mrežu, dok `[maxpool]` dodaje sloj sažimanja. Konvolucijski sloj sa slike ima 32 filtera dimenzija $3 * 3$ s pomakom 1. Podržava normalizaciju po grupama te koristi propusnu (eng. *leaky*) aktivacijsku funkciju. Sloj sažimanja je dimenzije $2 * 2$ s pomakom 2.

Na kraju cfg datoteka nalazi se sloj `[region]` koji sadrži dodatne hiperparametre modela potrebne za podjelu slike na ćelije. Neki parametri prikazani su na slici 6.2c. Na početku su navedene dimenzije referentnih okvira. One se odnose na rezoluciju izlaza duboke mreže i zapisane su slijedno širina i visina svakog okvira. Pošto ova mreža ima početnu rezoluciju $2304 * 416$ na izlazu će biti rezolucija umanjena za faktor 32, a to je $72 * 13$. Ta dimenzija je ujedno i maksimalna vrijednost referentnih okvira jer predstavlja cijelu sliku te objekt ne može biti veći od slike. U ovom primjeru prvi referentni okvir je dimenzija $0.16 * 0.47$ što znači da je jako mali u odnosu na sliku. Drugi parametar `bias_match` je korišten samo prilikom treniranja mreže. Ako je postavljen na 1 to znači da će mreža uzeti dimenzije referentnih okvira koje smo zadali, inače će ih mreža sama generirati. Pomoću parametra `classes` je zadano broj klasa u modelu, dok pomoću `num` broj referentnih okvira. Još jedan zanimljivi parametar je `jitter`, odnosno rastresanje podataka. On može biti između $[0, 1]$, a koristi se prilikom učenja mreže tako da ne uzima cijelu sliku nego samo njezin dio. Što je veća vrijednost parametra, uzet će se manji dio slike na ulaz u mrežu. Pomoću njega olakšava se predikcija manjih objekata.

6.2. Programska podrška

6.2.1. Python

Python je interaktivni, objektno orijentirani programski jezik. Python je programski jezik opće namjene, interpretiran i visoke razine koji je nastao 1990. godine. Po automatskoj memorijskoj alokaciji, Python je sličan programskim jezicima kao što su Perl, Ruby, Smalltalk itd. Python dopušta programerima korištenje nekoliko stilova programiranja. Mogućnost korištenja objektno orijentiranog, strukturnog i aspektnog programiranja čini Python sve popularnijim. Također postoje mnogo razvijenih biblioteka raznih namjena koje se jednostavno koriste pomoću Pythona. On ima sintaksu koja omogućuje laku čitljivost koda te ekstenzijske module koji mogu biti pisani u visoko efikasnim jezicima poput C-a ili C++-a. Python dolazi u verzijama Python 2 kojoj vrijeme podrške traje do 2020. godine te Python 3 koja je pod aktivnim razvojem. Python se najviše koristi na Linuxu, no postoje i inačice za druge operacijske sustave.

6.2.2. TensorFlow

Tensorflow je biblioteka otvorenog koda koja omogućava izražavanje numeričkih postupaka uz pomoć računskih grafova a posebno je pogodna za izradu modela strojnog učenja. Najviše je korišten pri izradi dubokih modela gdje podrška za grafičke procesore omogućava brže učenje modela. Tensorflow nudi podršku za automatsku diferencijaciju, što korisnicima omogućuje stvaranje novih modela bez potrebe za pisanjem izračuna gradijenata. Slojevi u Tensorflowu pisani su u jeziku C++ te je podržano izvođenje na arhitekturi CUDA. Tensorflow pruža sučelje prema programskom jeziku Python u kojem se model može definirati kao izračunski graf te se zbog prirode jezika Python tipično postiže jednostavnija implementacija (TensorFlow, 2015).

6.2.3. OpenCV

OpenCV (engl. Open Source Computer Vision) je skup biblioteka otvorenog koda pokrenut od strane američke tvrtke Intel 1999. godine. Biblioteke su višeplatformske (engl. *cross-platform*), mogu se koristiti u raznim aplikacijama, na raznim operacijskim sustavima. OpenCV služi raznim korisnicima da iskoriste i preoblikuju vlastiti kod. OpenCV se primjenjuje u području računalnog vida i služi za rad u realnom vremenu. Sadrži neke bitnije algoritme računalnog vida te metode za stjecanje, obrade,

analiziranja i razumijevanja slike. Biblioteka sadrži više od 2500 već gotovih optimiziranih algoritama. Algoritme možemo koristiti za pronalaženje, odnosno otkrivanje i prepoznavanje lica. Također se koristiti i za prepoznavanje objekata i praćenje objekata u pokretu.

Osnovna zadaća OpenCV-a pružanje je jednostavne platforme za rad s računalnim vidom i omogućavanje jednostavnog i brzog razvoja jednostavnih i složenih algoritama a s time i aplikacija koje koriste računalni vid. Zbog kompliciranih algoritama koji rade s velikim brojem operacija i podataka zahtjeva se optimiziran kod napisan na nižoj razini zbog veće brzine izvršavanja. Zbog toga su algoritmi OpenCV biblioteka originalno pisani u C programskom jeziku. Najčešći korisnici su tvrtke, istraživačke skupine, državna tijela i studenti (OpenCV, 2017).

7. Eksperimentalni rezultati

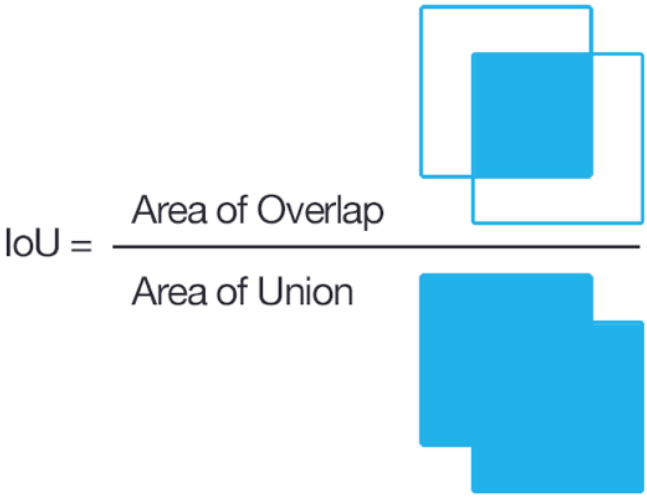
Eksperimente smo provodili nad skupom slika izvađenih iz video snimke nogometne utakmice koja je opisana u prijašnjim poglavljima. Video sadrži 7515 slika. Prvih 3000 slika koristile su se za treniranje modela, a ostale za testiranje. Detektor koji je korišten u projektu je YOLOv2 model. Pošto originalan model nije pokazivao dobre rezultate detekcije za zadani zadatak, trebalo ga je prilagoditi problemu. U tu svrhu promijenili smo neke hiperparametre i konvolucijsku mrežu modela te čak i skup slika za učenje. Tim promjenama uspješno je poboljšana točnost modela. U nastavku će se opisati glavne evaluacijske mjere korištene pri analizi modela i postupak dolaženja do modela sa sve boljim rezultatom detekcije.

7.1. Evaluacijske mjere

Glavne evaluacijske mjere koje se koriste za detekciju objekata su preciznost, odziv, krivulja preciznosti i odziva, prosječna preciznost i srednja prosječna preciznost. Kako bi dobili evaluacijske mjere bitno je objasniti kako smo odredili je li neka detekcija točna ili nije.

Svaki detektor lokalizira objekte tako da ih uokviruje pravokutnikom i klasificira u određeni razred. Izgled točnosti detekcije nad područjem interesa određuje se preko presjeka unije pravokutnika IOU (eng. *intersection over union*) . Za izračun presjeka unije pravokutnika koristi se funkcija prikazana na slici 7.1. IOU za neka dva okvira se računa tako da se podijeli presjek pravokutnika s njihovom unijom. Pravokutnik detekcije koji se savršeno poklapa s originalnim pravokutnikom imat će izlaz 1, dok će svaki pomaknuti imati manje ovisno o pomaku. Na temelju broja IOU određujemo točnost detekcije. Prvo se uzima pravokutnik koji predstavlja točan položaj objekta te svi detektirani pravokutnici koji se nalaze oko njega. Za njih se izračuna presjek unije pravokutnika i uzima se onaj koji ima najveći IOU. Prilikom evaluacije detektora najčešće se smatra da svaka detekcija koja ima $IOU => 0.5$ i točno je klasificirana je točna, dok su ostale netočne. Međutim, radi testiranja računaju se evaluacijske mjere

za različite vrijednosti parametra IOU.



Slika 7.1: Presjek unije pravokutnika koji je definiran kao omjer preklapanja pravokutnika i unije pravokutnika.

Evaluacijske mjere koje se koriste u detekciji definirane su preko broja točno pozitivnih TP (eng. *true positive*) primjera, broja pogrešno pozitivnih FP (eng. *false positive*) i broja pogrešno negativnih FN (eng. *false negative*). Točno pozitivni primjeri su oni kojima pravokutnici točno detektiraju objekt s obzirom na kriterij IOU, dok su pogrešno pozitivni oni koji nisu zadovoljili taj uvjet. Pogrešno negativnim se smatraju primjeri u kojima detektor nije prepoznao objekt. Pomoću ovih parametra mogu se lako izračunati preciznost P (eng. *precision*) i odziv R (eng. *recall*) modela jednadžbama u nastavku.

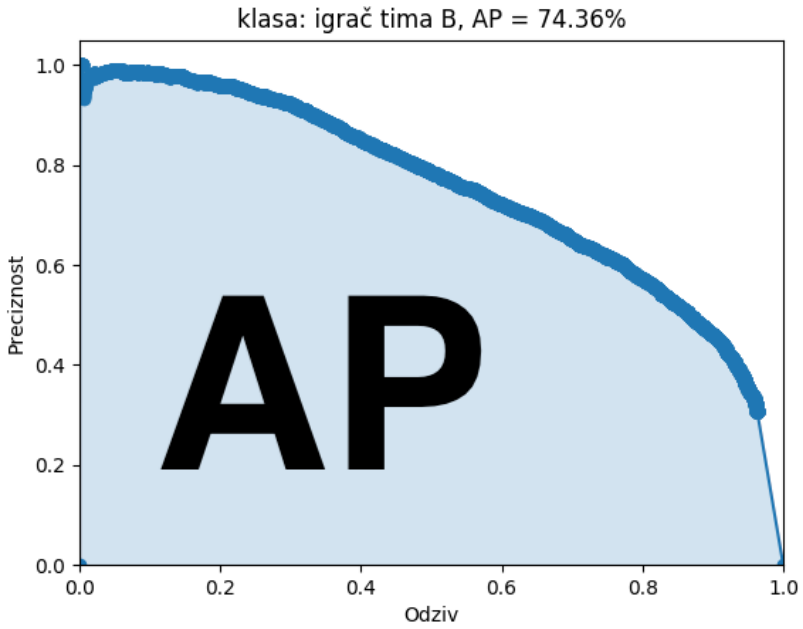
$$P = \frac{TP}{TP + FP} \tag{7.1}$$

$$R = \frac{TP}{TP + FN} \tag{7.2}$$

Preciznost nam govori koliki je udio točno klasificiranih primjera u skupu pozitivno klasificiranih primjera, a odziv koliki je udio točno klasificiranih primjera u skupu svih pozitivnih primjera. Mjera koja povezuje preciznost i odziv naziva se prosječna preciznost (eng. *average precision*).

Prosječna preciznost računa se tako da kontrolira vrijednost odziva nekim parametrom detekcije i za određeni odziv računa preciznost. Vrijednost odziva kontrolira se većinom pragom sigurnosti detektora (eng. *threshold*). Svaki detektor za pojedini

objekt daje svoju sigurnost u tu detekciju koja je između [0,1]. Na početku se za prag sigurnosti postavi visoka vrijednost tako da nema točno klasificiranih primjera pa je odziv 0. Kako smanjujemo taj prag odziv se povećava, dok preciznost pada. Tim postupkom dobije se krivulja prikazana na slici 7.2. Ona se naziva krivulja preciznosti i odziva (engl. *precision recall curve*). Površina ispod nje predstavlja prosječnu preciznost AP.



Slika 7.2: Primjer krivulje preciznosti i odziva za klasu igrač tima A. Površina ispod krivulje označena svijetlo plavom bojom je prosječna preciznost.

Prosječna preciznost računa se jednadžbom 7.3. Uzme se N jednako razmaknutih vrijednosti odziva između [0,1] te se izračuna preciznost za svaku od tih točaka, sumira i podijeli s brojem točaka.

$$AP = \frac{1}{N} \sum_{k=0}^{N-1} P[r = \frac{k}{N-1}] \tag{7.3}$$

Mjera koja najbolje opisuje kvalitetu nekog detektora je srednja prosječna preciznost mAP (eng. *mean average precision*). Ona jednim brojem opisuje rezultate detekcije nekog modela nad skupom slika i koristi se za usporedbu detektora. Izražena je sljedećom jednadžbom.

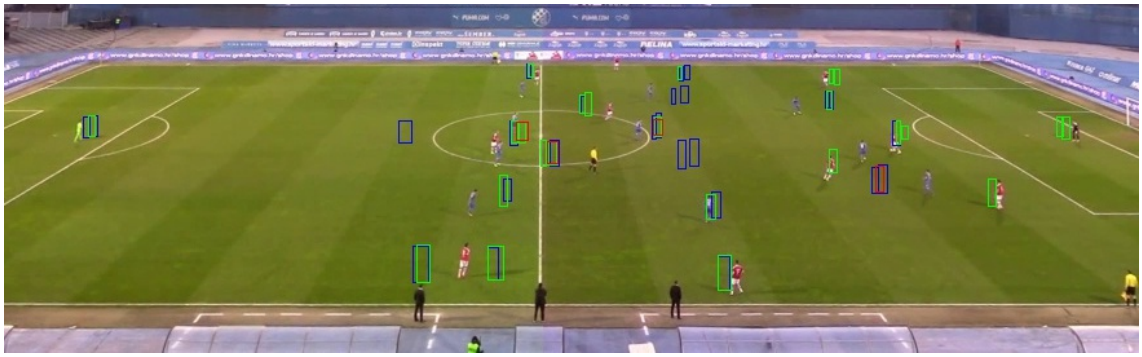
$$mAP = \frac{1}{K} \sum_{i=0}^{K-1} AP_i \tag{7.4}$$

7.2. Rezultati implementacija

7.2.1. Osnovni YOLOv2 model

Osnovni YOLOv2 model daje dobre rezultate nad PASCAL VOC i COCO skupu podataka te detektira objekte u realnom vremenu. Očekivano je bilo da će i na snimci nogometne utakmice točno prepoznati objekte. Preuzeli smo dobar kod sa Githuba koji implementira YOLOv2 model u Tensorflowu nazvan Darkflow te su preuzete težine dobivene na skupu slika PASCAL VOC za YOLOv2 duboku mrežu. Na početku smo pokrenuli model nad slikama iz nogometnog susreta tražeći klasu osoba. Međutim, originalan detektor nije dao dobre rezultate. On je detektirao točno nekoliko osoba koje su bile van terena, to su izbornici i pomoćni sudac, no nije detektirao niti jedan objekt na terenu. Vjerojatno su objekti na terenu premali i nemaju dosta značajka, dok su objekti van terena mnogo veći naspram njih.

Pošto se originalan model nije dobro prilagodio podacima, sljedeći korak je bio istrenirati mrežu nad skupom slika iz snimke. Izvadili smo slike za treniranje i njihove potrebne anotacije te je treniranje moglo početi. Nakon treniranja rezultati su bili poražavajući. Detektor je predviđao položaj igrača velikim okvirima diljem slike koji su bili poredani jedan do drugog i nisu imali nikakvih veza sa stvarnim igračima.



Slika 7.3: Primjer detekcije originalnog YOLOv2 modela sa promjenjenim referentnim okvirima.

Nakon neuspješnog pokušaja treniranja originalnog YOLOv2 modela proveli smo promjenu dimenzija referentnih okvira. Pošto su objekti na slici mnogo manji u odnosu na dimenziju slike, originalni referentni okviri su preveliki da bi ih uspjeli uokviriti. Pomoću algoritma k najbližih susjeda izračunali smo pogodne referentne okvire za duboku mrežu i zadani problem. Nakon učenja mreže s novim referentnim okvirima rezultat je bio opet neuspješan (slika 7.3), no ovaj put je čak detektor znao pogoditi

neku od lokacija objekata. Predviđeni pravokutnici bili su dobrih dimenzija, ali često su se pojavljivali oko stvarnog objekta ili negdje gdje nema igrača. mAP vrijednost ovog modela jednaka je 0.24% za $IOU => 0.5$ što nije dobro, ali pokazuje napredak. Sljedeći korak je bio promjena duboke mreže kako bi se bolje prilagodila zadanom problemu.

7.2.2. Prilagođeni YOLOv2 model

Pošto originalna YOLOv2 mreža nije prikazivala najbolje rezultate trebalo je promijeniti njezine parametre. Broj i raspored slojeva ostao je nepromijenjen jer su se oni pokazali dobrim u ostalim detekcijama. Problem zbog kojeg je model dao loše rezultate bio je taj što su objekti jako mali i mreža nije u stanju naučiti sve karakteristične značajke objekta.

Da bi mreža dobila više značajka od objekata treba joj proširiti dimenzije ulaza. Veća dimenzija daje na kraju i veću mapu značajki iz koje se mogu iščitati više podataka. U originalnoj YOLOv2 mreži rezolucija ulaza je $416*416$, dok je slika dimenzija $1624*288$. To znači da se ovako velika slika mora prilagoditi ulazu te se joj širina smanji za otprilike 4 puta. Pošto su igrači široki između 4 i 8 piksela njihova dimenzija se smanji na 1 do 2 piksela iz kojih je jako teško iščitati dobre značajke. Kako bi se spriječio ovaj problem odabrali smo veću rezoluciju ulaza proporcionalnu slici, a to je $2304 * 416$. Ova rezolucija još poveća ulaznu sliku čime se i objekti povećaju i njihove značajke ojačaju. Odabrana je ova rezolucija jer se mogla učiti na grafičkoj kartici, a da je imala dovoljno memorije. Veće rezolucije zahtijevaju više memorije pošto imaju više vektora na izlaznoj mapi značajka te bi popunile memoriju grafičke kartice. Već je za ovu rezoluciju trebalo smanjiti broj elemenata po grupi za učenje na 2 kako se ne bi popunila memorija.

Radi veće rezolucije, osim potrebe za više memorije, modelu treba duže za izvođenje prolaza kroz mrežu. Zbog toga ovim modelom se mogu detektirati oko 6 slika u sekundi što je znatno manje od originalnog modela, no povećala se mAP vrijednost. Iako je ona još uvijek mala, za $IOU => 0.5$ iznosi 2.29%, za manji kriterij $IOU => 0.1$ iznosi mnogo više $mAP = 52.24\%$. Pošto je za mali IOU srednja prosječna preciznost dosta visoka, model zna odrediti gdje je otprilike objekt, ali nije precizan u njegovoj lokaciji. Mala promjena pravokutnika detekcije za male objekte jako smanjuje vrijednost IOU-a.

Sljedeća stvar koja se promijenila, a da je uveliko poboljšala preciznost detektora je povećanje parametra za rastresanje podataka (eng. *jitter*). Prije ulaska slike u mrežu

YOLOv2 model ju raširi ili suzi za neku slučajno generiranu skalu. Na slici 7.4 je prikazan jedan takav primjer gdje je model povećao visinu, a smanjio širinu originalne slike. Ta promjena služi kako bi se model prilagodio različitim dimenzijama objekata i da može prepoznati šire i više objekte te da skupi više značajka o pojedinom objektu.

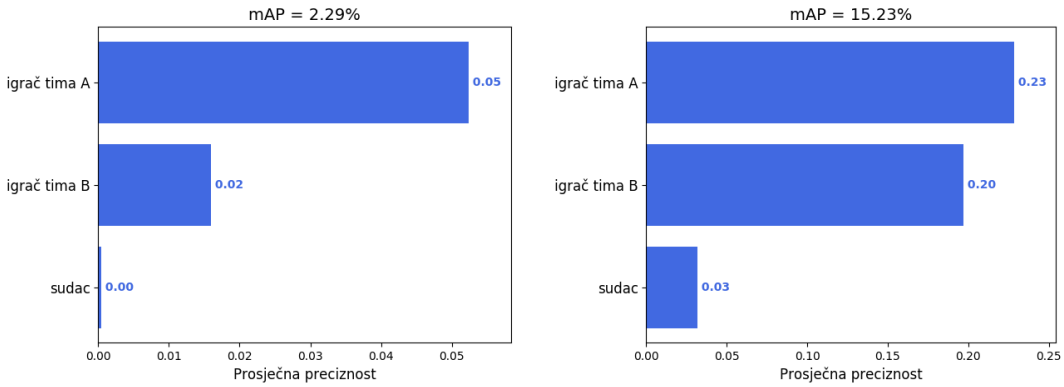
Slika 7.4: Slučajna promjena dimenzije originalne slike prije ulaska u mrežu.

Nakon promjene dimenzija originalne slike slijedi rastresanje podataka, odnosno uzimanje samo dijela slike na ulaz u mrežu. Parametar rastresenja podataka može biti između [0,1]. U originalnom YOLOv2 modelu on iznosi 0.3, a govori nam koliko posto slike najviše možemo maknuti. Ako je rastresanje 0.3, onda model može sliku skratiti za najviše 30% širine i visine. Taj postotak se dobije tako da se parametar rastresanja množi sa slučajno generiranim brojem između [0,1]. Primjer slike nakon rastresanja podataka prikazan je na slici 7.5. Kada se izvrši rastresanje podataka i uzme dio slike, ona se poveća na dimenziju ulaza u mrežu. Nad tako dobivenim slikama mreža se uči i iz nje vadi bitne značajke.

Slika 7.5: Izrezan dio slike od promijenjene slike i prilagođen rezoluciji ulaza u mrežu.

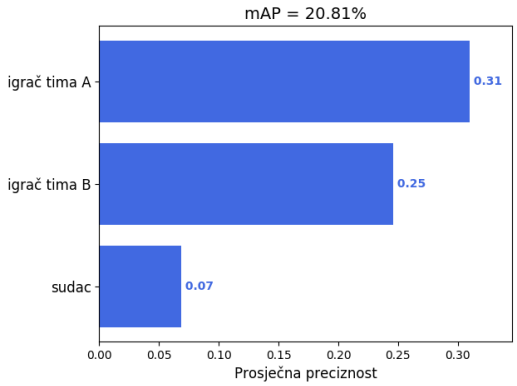
Rastresanje podataka koristi se kako bi model maknuo rubne dijelove slike i povećao objekte te izvadio više korisnih značajka. Na skupu slika sa velikim objektima nije dobro imati preveliki parametar rastresanja jer bi zbog njega model mogao izrezati dio objekta i izgubiti njegove značajke. Međutim, kod manjih objekata kojih je mnogo na slici parametar rastresanja je jako koristan. Ako je on veći, veća je vjerojatnost da se uzme manji dio slike te se objekti na tom dijelu uveliko povećaju i model tako „vidi “ više podataka za učenje. Na slici 7.6 prikazani su rezultati detekcije ovisno o rastresanju podataka. Vidi se kako povećanje vrijednosti rastresanja uveliko poveća srednju

prosječnu preciznost. Tako se za $jitter = 0.9$ dobije $mAP = 20.81\%$ što je solidan rezultat. Preciznost smanjuje klasa sudac za koju su predikcije dosta loše, a to je zato što imamo samo jednog suca po slici što je premali skup za učenje dobre lokalizacije u odnosu na igrače.



(a) rastresanje 0.5

(b) rastresanje 0.7



(c) rastresanje 0.9

Slika 7.6: Usporedba rezultata detekcije ovisno o rastresanju podataka. Povćanjem parametra rastresenja podataka na 0.9 rezultati se jako poboljšavaju. Kada bi ga još povećavali mAP vrijednost bi se smanjivala.

7.2.3. Prilagođeni YOLOv2 model s podjelom slika

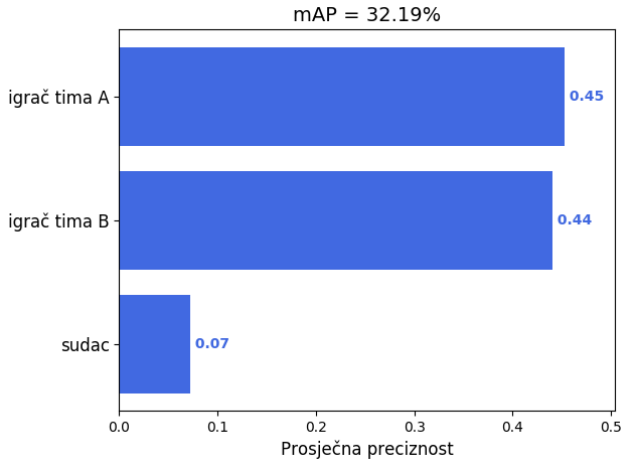
Sljedeća zamisao kako bi se poboljšala točnost modela je bila promjena skupa slika. Iz svake slike maknuli smo rubni dio gdje se nikada ne nalaze objekti te smo ostatak slike podijelili na 4 manje sličice. Sličice su dimenzija $288 * 216$ (slika 7.7), a anotacije su se preračunale za svaku od njih. Tako se iz 3000 slika za učenje dobije skup od 12000 sličica. U ovom modelu koristi se rezolucija ulaza u mrežu $960 * 960$ pošto su visina i širina sličica sličnih dimenzija. Parametar rastresanja podataka je smanjen te

je postavljen na 0.5 jer više nema potrebe za uzimanjem manjeg dijela sličice. Ovim postupkom se mnogo povećaju objekti i dobiju bolje značajke.

Slika 7.7: Podjela originalnih slika na manje sličice.

U ovom modelu glavni problem predstavljaju rubni objekti koji su prepolovljeni te se nalaze na dvije manje sličice. Kod skupa za treniranje taj problem se riješio produljenjem sličice za onoliko piksela koliko je potrebno da se vide cijeli objekti. Međutim, veći problem je bio kako riješiti ovaj problem prilikom detekcije objekata. Najboljim rješenjem uspostavili smo da se nakon podjele slike na 4 dijela svakoj manjoj sličici doda 8 piksela s desne strane. Zbog toga dvije susjedne sličice imaju jednakih 8 piksela slike. Odabrano je produljenje za baš 8 piksela zato što su objekti najviše široki 8 piksela te smo ovim produljenjem sigurni da se objekt koji je na početku prepolovljen sigurno nalazi na produljenoj sličici. Iako je i dalje moguće da postoje rubni objekti, oni su prepolovljeni samo na jednoj sličici jer se na susjednoj pojavljuju sigurno u cijelosti. Nakon što su napravljenje predikcije objekata za sve četiri sličice, rezultati detekcije se preračunavaju i postavljaju na originalnu sliku. Slijedi algoritam eliminacije nemaksimalnih elemenata koji traži okvire koji se preklapaju za $IOU => 0.3$ te izbacuje nemaksimalne okvire. Ovaj algoritam je implementiran već u YOLO modelu, no radi se opet zbog mogućnosti preklapanja rubnih elemenata.

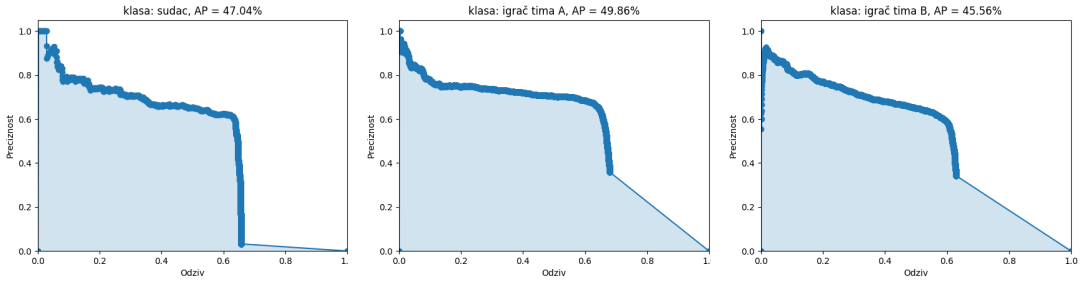
Mogućnost izrade ovakvog modela je zbog statične snimke i dobro definiranih pravila kojim se zna gdje se igrači nalaze. Njime su dobiveni mnogo bolji rezultati od prijašnjih modela te ima mAP vrijednost 32.19%. Rezultati detekcije prikazani su na slici 7.8. Kao što je vidljivo prosječna preciznost detekcije igrača se mnogo poboljšala te iznosi približno $mAP = 45\%$, dok je detekcija suca još uvijek slaba. Ovaj model je dosta precizan, no vrijeme detekcije se znatno produljilo. Uspije obraditi jednu originalnu sliku u sekundi na grafičkoj kartici Tesla K80.



Slika 7.8: Rezultati detekcije modela s podjelom originalne slike na dijelove. Igrači imaju mnogo bolje prosječne preciznosti od klase sudac jer imamo više njihovih primjera za učenje.

7.2.4. YOLOv3 model

Prijašnji postupak daje dobre rezultate, no on ipak dijeli slike iz skupa za učenje što nije dobra praksa i vrijeme detekcije traje dosta dugo. Kako bi dobili dobar detektor koji radi na originalnoj slici isprobali smo YOLOv3 model. Njegova glavna značajka je da dodaje dekonvolucijski sloj na kraj YOLOv2 modela. A taj sloj pospješuje prepoznavanje manjih objekta što je pogodno za ovaj problem. Rezolucija ulaza u novu mrežu je $1632 * 416$, a na izlazu su dobivene 3 mape značajka sljedećih dimenzija: $51 * 13$, $102 * 26$, $204 * 52$. U njima se rade predikcije objekata, a pošto su neke većih dimenzija imaju više značajka manjih objekata. Parametar rastresanja podataka postavljen je na 0.7. Duboka mreža koja se koristi je Darknet-53, s 53 konvolucijska sloja i dekonvolucijom na kraju mreže.



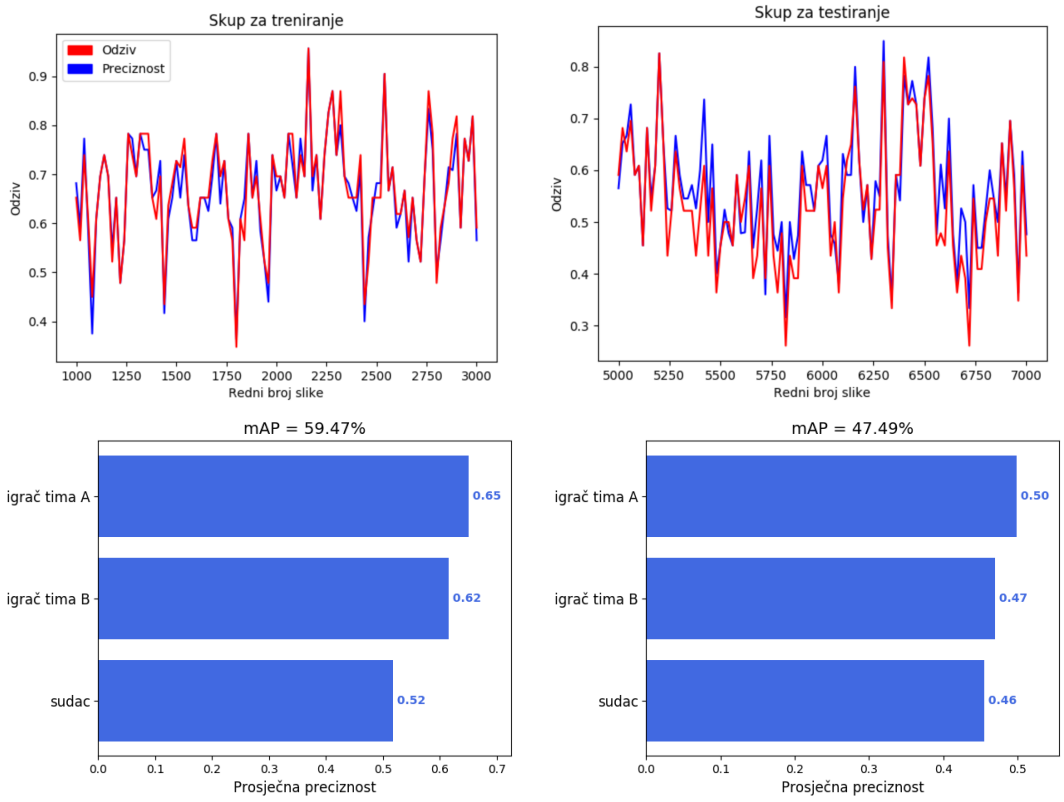
(a) sudac **(b)** igrač tima A **(c)** igrač tima B

Slika 7.9: Krivulje preciznosti i odziva za svaku klasu. YOLOv3 daje bolje rezultate od prijašnjih modela i uspijeva poboljšati prosječnu preciznost za klasu sudac.

Rezultati detekcije prikazani su krivuljom preciznosti i odziva na slici 7.9. Ovim

modelom poboljšana je klasa sudac i konačno pokazuje slične rezultate kao i ostale klase. Sa slike vidimo da je prosječna preciznost klase sudac $AP = 47.04\%$, klase igrač tima A $AP = 49.86\%$ te klase igrač tima B $AP = 45.56\%$. Srednja prosječna preciznost iznosi $mAP = 47.49\%$. Ovaj model daje daleko najbolje rezultate od prijašnjih modela i detektira slike relativno brzo. U sekundi uspije obraditi oko 5 slika na grafičkoj kartici Tesla K80.

Treniranje traje mnogo duže nego kod prijašnjih modela baziranim na YOLOv2 verziji. Mreža je dosta veća, ima 53 konvolucijska sloja, dok YOLOv2 samo 19 te je potrebno odrediti mnogo više parametara. YOLOv2 modelu dovoljno je oko 4000 koraka da se mreža nauči što traje 2 sata, dok je YOLOv3 modelu potrebno 10 puta više koraka te učenje traje oko dan i pol na grafičkoj kartici. Učenje modela YOLOv3 odvija se u dva koraka kako bi bilo što brže. Na početku prvi slojevi ostaju ne taknuti, a uče se samo izlazni slojevi kako bi se funkcija gubitka što prije smanjila nakon čega ide učenje nad cijelom mrežom.



(a) skup za treniranje **(b)** skup za testiranje

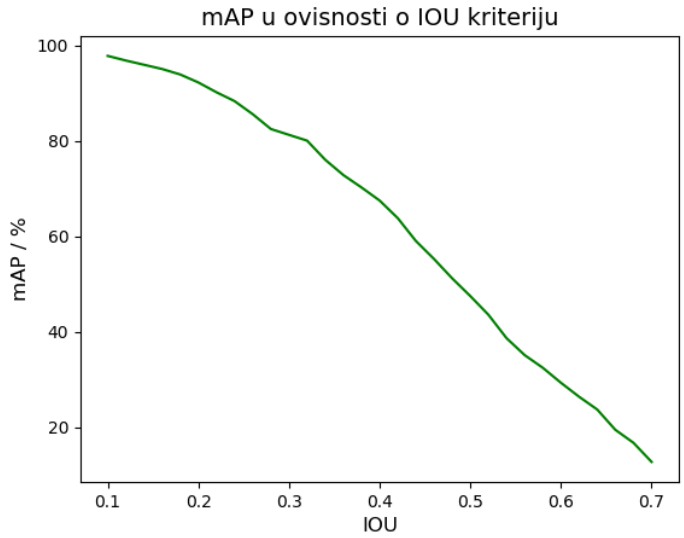
Slika 7.10: Usporedba rezultata dobivenih nad 2000 slika iz skupa za testiranje i ispitnog skupa. Detekcija na skupu za treniranje daje 12% bolju srednju prosječnu preciznost.

Na slici 7.10 uspoređeni su rezultati detekcije dobiveni nad 2000 slika iz skupa za

treniranje i 2000 slika iz skupa za testiranje. Kao što je očekivano rezultati na skupu za treniranje su bolji. Preciznost na skupu za treniranje je $P = 66.49\%$, dok je na ispitnom skupu $P = 56.42\%$. Odziv je dosta sličan preciznosti te iznosi na skupu za treniranje $R = 67.04\%$, a na ispitnom skupu $R = 53.70\%$. Srednja prosječna preciznost je za 12% veća na skupu za treniranje. Razlog tome je možda i taj što je rano prekinuto treniranje zbog ograničenja na virtualnom stroju koji je korišten. Naime iskorištene su bile sve besplatne minute, pa treniranje nije nastavljeno.

Slika 7.11: Rezultat detekcije YOLOv3 modela.

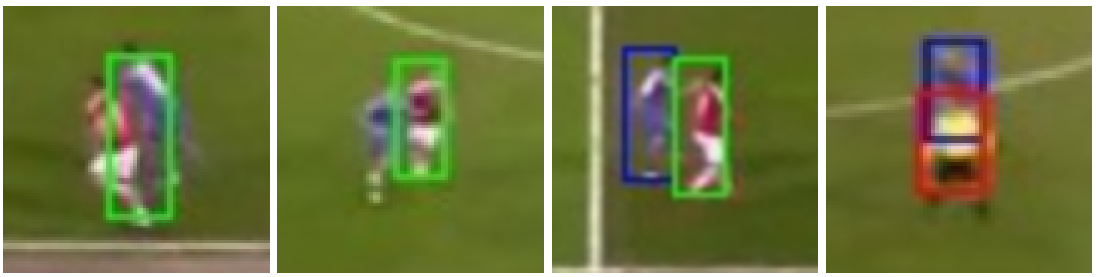
Ovaj model je uspješno riješio problem detekcije malih igrača na terenu što je prikazano na slici 7.11. On ih uspije skoro sve detektirati, no kod nekih pravokutnici ne uokviruju precizno objekt.



Slika 7.12: Rezultati srednje prosjčne preciznosti u ovisnostu o kriteriju IOU. Za kriterij $IUO => 0.1, 0.3, 0.5, 0.7$ vrijednost srednje prosječne preciznosti je redom $mAP = 96.83\%, 81.02\%, 47.49\%, 12.74\%$.

Rezultate za različiti kriterij vrijednosti IOU parametra prikazani su na slici 7.12. Srednja prosječna preciznost za manji IOU je mnogo veća. Tako na primjer za $IOU =>$ 0.3 model ima srednju prosječnu preciznost $mAP = 81.02\%$, dok za $IOU => 0.1$ ima $mAP = 96.83\%$ što je mnogo više od prijašnjeg rezultata za $IOU => 0.5$. YOLOv3 za veći IOU ne daje precizna rješenja, pa je tako za $IOU >= 0.7$ srednja prosječna preciznost jednaka $mAP = 12.74\%$.

Ovaj model uspješno detektira manje objekte u daljini i rješava problem nekonzistentnosti veličina igrača. Jedino se ponekad zna dogoditi da ne uspije dobro detektirati objekte ako su oni jedan pokraj drugoga ili ako se prekrivaju. Na slici 7.13. su prikazani pozitivni i negativni primjeri takvih situacija.



Slika 7.13: Prikaz pozitivnih i negativnih primjera u situacijama kada su objekti jedan blizu drugoga.

8. Zaključak

U okviru rada razmatrane su novije metode detekcije objekata koje se baziraju na jednom unaprijednom prolazu kroz duboku mrežu. Te metode su popularne jer uspijevaju detektirati objekte s dobrom preciznošću mnogo brže nego prijašnji detektori. Ako se izvode na grafičkoj kartici mogu raditi predikcije u realnom vremenu. Detektori koji su detaljnije proučeni u radu su YOLO i SSD. Kako se razvija područje dubokog učenja tako se razvijaju i detektori pa postoji više verzija YOLO i SSD modela koje uvode dodatna poboljšanja.

U radu su korištene najnovije verzije YOLO modela za lokalizaciju igrača na nogometnoj utakmici. Verzija YOLOv3 pokazala je dobre rezultate detekcije te detektirala male objekte na video snimci. Može se dogoditi da model ne pogodi precizno objekt već pomakne okvir u odnosu na originalnu poziciju. To se događa jer su objekti na videu niske razlučivosti u odnosu na sliku što otežava preciznost detekcije. YOLOv3 se pokazao kao dobar detektor koji se može prilagoditi skupu za učenje te je brz u predikciji.

Dobiveni model upotrebljen je za praćenje igrača na nogometnoj utakmici tako što je uzastopno ponovljen proces detekcije na svakoj slici. Tako dobiveno praćenje izgleda dobro samo što pravokutnici znaju brzo mijenjati dimenzije i vibrirati oko objekta. Problem je taj što se ne uzima kontekst prijašnjih slika nego je svaka slika detekcija za sebe. Da bi se ovo poboljšalo u budućem radu bi se mogla dodati metoda praćenja. Prvo bi se napravila detekcija igrača pomoću YOLO modela te nakon toga svakom igraču pridružila povratna neuronska mreža čiji ulaz će biti tekući izgled, a izlaz predikcija budućeg izgleda. Ta mreža bi predviđala konvolucijske značajke novog položaja. Predikciju bi na sljedećoj slici interpolirali s detekcijom trenutnog položaja i tako dobili novi. Ovaj princip se prikazao uspješnim u radu gdje je korišten (Pauline Luc, 2018). U njemu je za detekciju korišten Faster R-CNN, koji je sporiji i daje lošije rezultate nego YOLO model. Ako bi se umjesto njega koristio YOLOv3 model ovaj pristup bi se mogao primjeniti na našem problemu s nogometašima i dati zadovoljavajuću preciznost pri praćenju objekata.

LITERATURA

Ananth Ranga Ambrish Tyagi Alexander C. Berg Cheng-Yang Fu, Wei Liu. *DSSD : Deconvolutional Single Shot Detector*, 2017. URL https://arxiv.org/pdf/1701.06659.pdf.

Marko Čupić. *Optimizacija parametra modela*. FER, 2017. URL http://www.zemris.fer.hr/~ssegvic/du/du3optimization.pdf.

Ali Farhadi Joseph Redmon. *YOLO9000: Better, Faster, Stronger*, 2016. URL https://arxiv.org/pdf/1612.08242.pdf.

Ali Farhadi Joseph Redmon. *YOLOv3: An Incremental Improvement*, 2018. URL https://pjreddie.com/media/files/papers/YOLOv3.pdf.

Santosh Divvala Joseph Redmon. *You Only Look Once: Unified, Real-Time Object Detection*, 2015. URL https://arxiv.org/pdf/1506.02640.pdf.

Josip Karapac. *Konvolucijske neuronske mreže*. FER, 2017. URL http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf.

Josip Krapac. *Konvolucijski modeli*. FER, 2016. URL http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf.

Igor Tartalja Marko Dabović. *Duboke konvolucijske neuronske mreže – koncepti i aktuelna istraživanja*. ETRAN, 2017. URL https://www.etran.rs/common/pages/proceedings/ETRAN2017/VI/IcETRAN2017_paper_VI1_1.pdf.

OpenCV. *OpenCV*, 2017. URL https://opencv.org.

Camille Couprie Pauline Luc. *Predicting Future Instance Segmentations by Forecasting Convolutional Features*, 2018. URL https://arxiv.org/pdf/1803.11496.pdf.

Qqwweee. Keras-yol3, 2018. URL https://github.com/qqwweee/keras-yolo3.

Joseph Redmon. Yolo, 2018. URL https://pjreddie.com/darknet/yolo/.

TensorFlow. Tensorflow, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Trieu. Darkflow, 2017. URL https://github.com/thtrieu/darkflow.

Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Alexander C. Berg Wei Liu, Dragomir Anguelov. *SSD: Single Shot MultiBox Detector*, 2015. URL https://arxiv.org/pdf/1512.02325.pdf.

Joyce Xu. Deep learning for object detection: A comprehensive review, 2017. URL https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d

Duboki konvolucijski modeli za lokalizaciju objekata

Sažetak

Ovaj rad bavi se metodama detekcije objekata konvolucijskim modelima. Detaljnije su proučeni i uspoređeni modeli SSD i YOLO te njihove novije verzije. Oni se baziraju na jednom unaprijednom prolazu kroz mrežu što ih čini brzim, a također daju i dobre rezultate. Napravljen je eksperiment koji koristi novije YOLO modele za lokalizaciju objekata na vlastitom skupu podataka. U tu svrhu koristi se snimka nogometne utakmice gdje su objekti za detekciju igrači oba tima i sudac. Modeli su evaluirani osnovnim mjerama i prikazani su njihovi rezultati detekcije.

Ključne riječi: detekcija objekata, konvolucijske neuronske mreže, duboko učenje, SSD, DSSD, YOLO.

Deep convolution models for detecting objects

Abstract

This paper deals with the object detection method with convolutional models. There are studied and compared models like SSD, YOLO and their modern versions. They are based on only one forward pass through network what makes them fast and also give good results. An experiment is made which use newer YOLO models for localization objects on own dataset. For this purpose a tape of football game is use where object for detection are both teams and referee. Models are evaluated with basic measures and their detection results are shown.

Keywords: object detection, convolutional neural networks, deep learning, SSD, DSSD, YOLO.