SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2290

FINO UGAĐANJE KONVOLUCIJSKIH MODELA ZA LOKALIZACIJU OBJEKATA

Vjeran Hanžek

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2290

FINO UGAĐANJE KONVOLUCIJSKIH MODELA ZA LOKALIZACIJU OBJEKATA

Vjeran Hanžek

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2290

Pristupnik:	**Vjeran Hanžek (0036482927)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Fino ugađanje konvolucijskih modela za lokalizaciju objekata**

Opis zadatka:

Lokaliziranje objekata u slikama predstavlja važan problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme najbolji rezultati u tom području postižu se dubokim konvolucijskim modelima. Tema ovog rada je istražiti mogućnost prilagođavanja takvih modela konkretnim primjenama. U okviru rada, potrebno je proučiti model Mask R-CNN izveden nad piramidalnim ekstraktorom značajki FPN. Vrednovati lokalizacijsku točnost na označenoj snimci nogometnog susreta i analizirati rezultate. Po potrebi prilagoditi arhitekturu te provesti fino ugađanje na podskupu za treniranje. Validirati hiperparametre, ponovo provesti vrednovanje te prikazati i ocijeniti ostvarene rezultate. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

Zahvaljujem se svojoj obitelji i prijateljima za podršku tijekom studiranja.

SADRŽAJ

1. Uvod

Američki znanstvenik Arthur Lee Samuel bio je prvi koji je definirao pojam strojnog učenja, još sredinom prošlog stoljeća. Prema Samuelu, strojno učenje je područje proučavanja koje daje računalima mogućnost učenja bez njihovog eksplicitnog programiranja. Koliko je Samuel bio ispred svog vremena potvrđuje činjenica da se njegova definicija i danas koristi pri opisivanju inteligentnih sustava. Strojno učenje jedno je od danas najaktivnijih i najuzbudljivijih područja računarske znanosti, ponajviše zbog brojnih mogućnosti primjene koje se protežu od raspoznavanja uzoraka i dubinske analize podataka do robotike, računalnog vida, bioinformatike i računalne lingvistike. U ovom radu dotaknut ću se teme računalnog vida (engl. *computer vision*) koji se definira kao područje umjetne inteligencije čiji je cilj dati računalima vizualno shvaćanje svijeta, stjecanjem znanja iz digitalnih slika ili videa. Izazov računalnog vida je približiti se ljudskom vidu, što je i danas daleko od riješenog.

Tijekom proteklih godina došlo je do brze i uspješne ekspanzije istraživanja u području računalnog vida. Jedno od područja računalnog vida koje je postiglo veliki napredak je lokalizacija i prepoznavanje objekata. Cilj lokalizacije je odrediti prisutnost određenog objekta, kao i njegovu točnu lokaciju. U idealnom slučaju želimo detektirati svaki objekt na danoj slici, no budući da je broj različitih objekata vrlo velik, lokalizacija postaje problematična, pogotovo kada su ti objekti prekriveni drugim objektima ili pak odsječeni granicom slike.

Ovaj rad proučava problem detekcije objekata na slikama i videu i daje uvid kako se taj problem riješio korištenjem konvolucijskih neuronskih mreža zasnovanih na predlaganju područja (R-CNN). Brzina i točnost igraju važnu ulogu u prosuđivanju sustava detekcije objekata u stvarnom vremenu. Tako na primjer postoje jednofazni (engl. *one-stage*) modeli kao što su YOLO i SSD koji su prilično brzi, ali ne nude dobru točnost. Za razliku od njih, starije metode poput R-CNN-a nisu dovoljno brze kako bi se mogle koristiti u sustavu u stvarnom vremenu. R-CNN je prvi model koji je došao do lokalizacije koristeći predlaganje područja. Nakon toga, objavljivane su redom njegove poboljšane inačice Fast R-CNN, Faster R-CNN i Mask R-CNN, od kojih je svaka nad-

mašila svoju prijašnju verziju u smislu brzine i točnosti. Prva inačica koja je mogla konkurirati jednofaznim metodama bila je Faster R-CNN. Unatoč napretku, modeli zasnovani na predlaganju područja i dalje zaostaju za jednofaznim u pogledu brzine, zbog čega u ovom radu neće biti promatrana detekcija objekata u stvarnom vremenu. Zadatak ovog rada je detekcija igrača (objekata) na snimci nogometne utakmice. Učenje modela provedeno je nad prednaučenim implementacijama Faster i Mask R-CNN modela. Postupak gdje se težine već prednaučene neuronske mreže koriste kao inicijalizacija za novi model koji će se učiti nad podacima iz iste domene kao i prednaučeni model (najčešće slike) naziva se fino ugađanje (engl. *fine-tuning*). Koristi se za ubrzavanje postupka učenje i premošćivanje problema malog skupa podataka. Budući da je naš skup podataka relativno malen i ne razlikuje drastično od skupa podataka nad kojim je bazni model treniran, fino ugađanje je bilo idealan izbor. Provedeni eksperimenti i njihovi rezultati opisani su u posljednjem poglavlju.

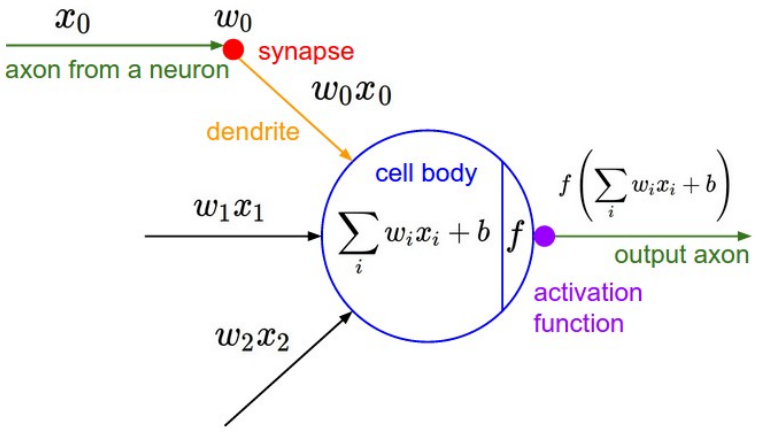
2. Duboke neuronske mreže

2.1. Neuronske mreže

Razvoj umjetnih neuronskih mreža započeo je kao pokušaj da se objasni inteligentno ponašanje ljudi i životinja. Istraživanja su pokazala da inteligentno ponašanje izranja iz velikog broja jednostavnih procesnih elemenata (neurona) koji podatke obrađuju masovno paralelno i uz veliku redundanciju - što je temelj pristupa danas poznatog pod nazivom konektivizam.

2.1.1. Model neurona

Biološki neuron ili živčana stanica se smatra osnovnom jedinicom živčanog sustava i najsloženija je u ljudskom organizmu. Pojednostavljeno, takav neuron se sastoji od dendrita, tijela neurona i aksona. Dendriti predstavljaju ulaze preko kojih neuron prikuplja informacije, tijelo stanice ih obrađuje i na kraju generira rezultat koji se prenosi kroz akson - izlaz neurona. Inspirirani tim činjenicama, Warren McCulloch i Walter Pitts su 1943. godine definirali model umjetnog neurona (tzv. TLU perceptron). Pojednostavljeni model prikazan je na slici 2.1.



Slika 2.1: Model neurona [9]

Model se sastoji od ulaza označenih sa x_i ("dendriti") pri čemu je $i \in (0, n)$, težina, označenih sa w_i koje određuju u kojoj mjeri svaki od ulaza pobuđuje neuron, praga paljenja neurona b (engl. *bias*), tijela neurona koje računa ukupnu pobudu te prijenosne funkcije f ("akson") koja pobudu obrađuje i prosljeđuje na izlaz neurona.

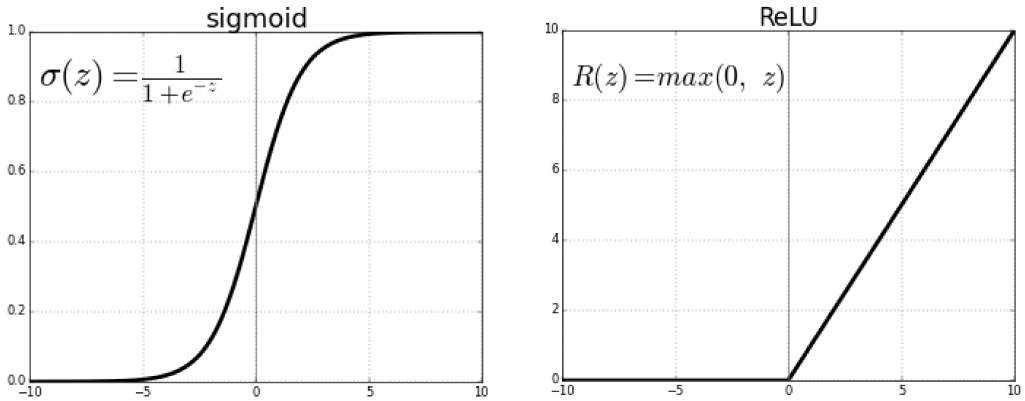
Ukupna pobuda y računa se prema izrazu:

$$y = \sum_{i=0}^{n} w_i x_i + b$$

Često se u literaturi može vidjeti da se prag paljenja neurona umjesto sa b označava sa w_0, pa ulazi i težine kreću od indeksa $i = 1$. U tom slučaju je potrebno dodati još jedan ulaz x_0 koji se naziva *dummy* značajka čija je vrijednost 1, pa gornji izraz možemo pisati u vektorskom obliku kao:

$$y = \mathbf{w}^T \mathbf{x}$$

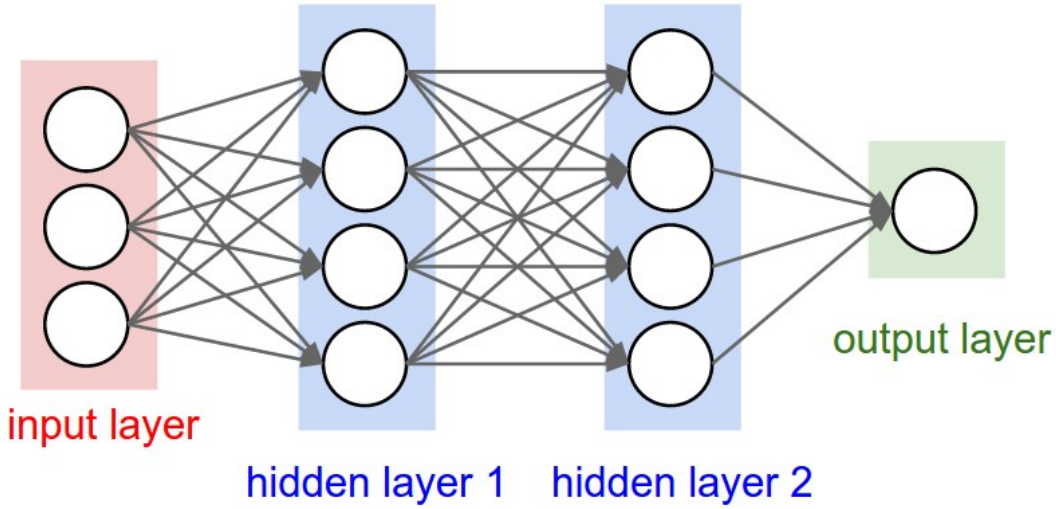
Prijenosna funkcija (engl. *activation function*) modelira ponašanje aksona i u praksi se koristi nekoliko tipičnih prijenosnih funkcija. Prijenosna funkcija može biti linearna i nelinearna. Linearna prijenosna funkcija ne radi nikakav koristan posao, već jednostavno preslikava ulaz u izlaz. Funkcije koje su nam od većeg značaja u izgradnji dubokih modela su nelinearne funkcije. Takve funkcije, kao što i samo ime govori, uvode nelinearnost u neuronsku mrežu. Bez njih mreža, bez obzira na broj slojeva, ne bi obavljala nikakvu korisnu funkciju. U dubokim neuronskim mrežama najčešće se koriste sigmoidalna (logistička) funkcija i ReLU (engl. *Rectified Linear Unit*). Konvolucijske neuronske mreže razmatrane u ovom radu najčešće koriste ReLU aktivacijsku funkciju. Obje aktivacijske funkcije prikazane su na slici 2.2.



Slika 2.2: Sigmoidalna i ReLU prijenosna funkcija

2.1.2. Unaprijedne neuronske mreže

Umjetna neuronska mreža (engl. *Artificial Neural Network - ANN*) je skup međusobno povezanih neurona, po uzoru na naš živčani sustav. Na slici 2.3 je prikazana unaprijedna (engl. *feedforward*) potpuno povezana slojevita neuronska mreža.



Slika 2.3: Unaprijedna neuronska mreža [9]

Unaprijedna mreža nema niti jedan ciklus, dakle nema povratnih veza među neuronima. Slojevita mreža znači da se u sloju k izlazi neurona računaju samo i isključivo temeljem izraza neurona sloja k - 1. Također, slojevita mreža nema lateralnih veza (veze između neurona u istom sloju). Kod potpuno povezane mreže svaki neuron sloja k je povezan sa svakim neuronom sloja k + 1.

Sve unaprijedne neuronske mreže se sastoje od jednog ulaznog, jednog izlaznog i više skrivenih slojeva. Mreža sa slike ima dva skrivena sloja.
Ulazni sloj se sastoji od neurona koji ne obavljaju nikakvu funkciju već naprosto preslikavaju dovedene ulazne podatke i čine ih dostupnima ostatku mreže.
Skriveni sloj se sastoji od neurona koji na svoje ulaze dobivaju isključivo vrijednosti prethodnog sloja. Svaki neuron u skrivenom sloju transformira dobivene ulaze korištenjem prijenosnih funkcija. Prijenosna funkcija mora biti nelinearna, u suprotnom bi se cijela mreža pretvorila u jednu veliku linearnost koja bi davala neispravne rezultate na svom izlazu. Povećanjem broja skrivenih slojeva se povećava ekspresivnost mreže, odnosno broj težina (parametara) mreže. Prevelik broj parametra može dovesti do prenaučenosti mreže, a premalen do podnaučenosti. Imajući to na umu, vrlo je

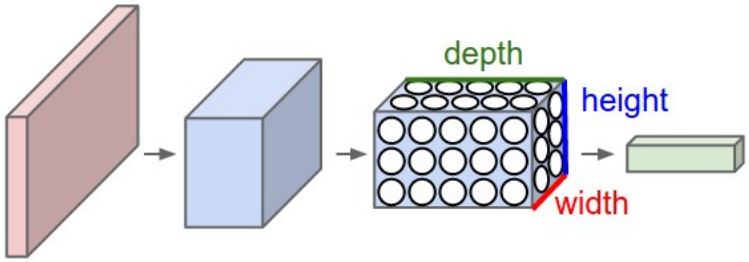
teško odabrati ispravan broj skrivenih slojeva mreže, onaj za koji mreža najbolje generalizira.

Izlazni sloj daje rezultat unaprijednog prolaza kroz mrežu. Najčešće izlazni neuroni nemaju prijenosnu funkciju tj. njihova prijenosna funkcija je linearna. Razlog tome je taj što se izlazni sloj uglavnom koristi za prikazivanje rezultata klasifikacije ili regresije, ovisno o namjeni mreže. Prikazana neuronska mreža ostvaruje preslikavanje podataka iz 6-dimenzijskog prostora $\vec{x}$ u 1-dimenzijski prostor $\vec{y}$.

Osim unaprijednih neuronskih mreža postoje i povratne neuronske mreže (engl. *Recurrent neural network - RNN*) kod kojih postoje ciklusi. Za razliku od unaprijednih mreža, povratne neuronske mreže mogu koristiti svoje unutarnje stanje (memoriju) za obradu nizova ulaza što ih čini primjenjivim na zadatke kao što su prepoznavanje rukopisa ili prepoznavanje govora.

2.2. Duboki konvolucijski modeli

Konvolucijske neuronske mreže (engl. *Convolutional Neural Network - CNN*) su vrlo slične običnim neuronskim mrežama opisanima u prošlom poglavlju što znači da su sastavljene od neurona koji imaju varijable težina (engl. *weights*) i pomaka (engl. *bias*) koje se mogu podešavati prilikom učenja mreža. Konvolucijska, kao i obična, neuronska mreža sastoji se od jednog ulaznog, jednog izlaznog i jednog ili više skrivenih slojeva. Razlika u odnosu na običnu neuronsku mrežu je ta što konvolucijske mreže eksplicitno pretpostavljaju da su ulazi slike, što nam omogućuje ugrađivanje određenih svojstava u arhitekturu mreže. Zbog toga, slojevi konvolucijske mreže imaju neurone raspoređene u tri dimenzije, širina, visina i dubina - kao što je to prikazano na slici 2.4. Ulaz u konvolucijsku mrežu je slika koja predstavlja podatke u trodimenzionalnom obliku WxHxD gdje su W i H širina, odnosno visina slike u pikselima, dok D predstavlja broj kanala slike. U slučaju slike u boji bismo imali 3 kanala (RGB).



Slika 2.4: Konvolucijski slojevi [9]

Kod konvolucijskih mreža koriste se tri vrste slojeva - konvolucijski sloj (engl. *Convolutional layer*), sloj sažimanja (engl. *Pooling layer*) i potpuno povezani sloj (engl. *Fully-connected layer*).

2.2.1. Konvolucija

Konvolucijski sloj je najvažniji građevni blok u konvolucijskim neuronskim mrežama, no prije nego što uđemo u priču o konvolucijskom sloju, bitno je shvatiti što je zapravo konvolucija.

Konvolucija je matematička operacija 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. U računalnom vidu koristi se diskretna varijanta konvolucije jer su podaci diskretizirani i ima ih konačan broj. U takvom slučaju integral zamijenimo sumom (uz pretpostavku da je domena konačna). Obje formula konvolucije, gdje je x prva funkcija, a w druga, prikazane su u nastavku:

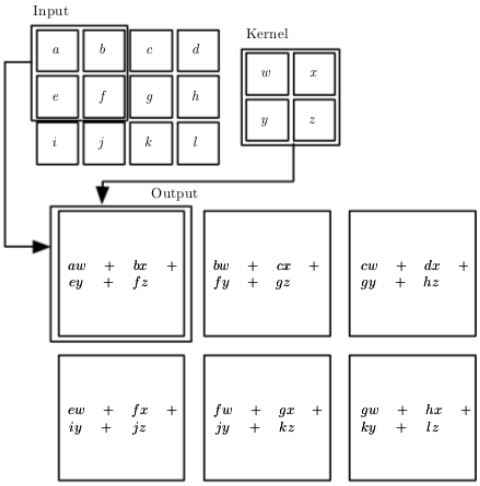
$$h(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau \overset{\text{diskr.}}{=} \sum_{\tau=\tau_{min}}^{\tau_{max}} x(\tau)w(t-\tau)$$

U kontekstu konvolucijskih modela funkcija x predstavlja ulaz, funkcija w jezgru (parametri) (engl. *kernel*), a izlaz konvolucije h se naziva mapa značajki (engl. *feature map*). Budući da su ulazni podaci dvodimenzionalni (jedan kanal slike), konvolucija se radi po obje dimenzije. Također, konvolucija je komutativna operacija pa se zbog jednostavnosti programske implementacije jezgra okreće s obzirom na ulaz. Kako je svejedno na kojoj će poziciji jezgre algoritam naučiti parametre, u primjeni se koristi kros-korelacija ili "neobrnuta konvolucija":

$$S(i,j) = (I * K)(i,j) = \sum_{m=m_{min}}^{m_{max}} \sum_{n=n_{min}}^{n_{max}} I(i+m, j+n)K(m,n)$$

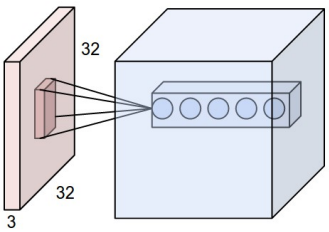
2.2.2. Konvolucijski sloj

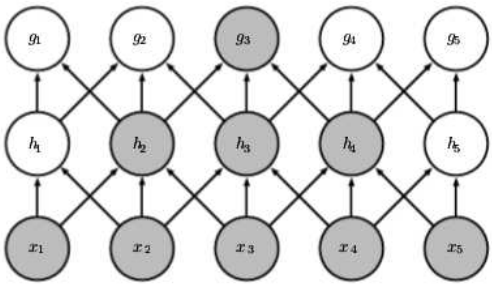
Konvolucijski sloj je sličan potpuno povezanom sloju, ali modelira samo lokalne interakcije. Drugim riječima, kod konvolucijskog sloja izlaz je povezan s manjim dijelom ulaza, za razliku od potpuno povezanog gdje je izlaz povezan sa svim ulazima. Ovdje je bitno naglasiti asimetriju u načinu na koji se tretiraju prostorne dimenzije (širina i visina) i dimenzija dubine - veze su lokalne u prostoru (duž širine i visine), ali uvijek



Slika 2.5: Dvodimenzionalna konvolucija [10]

potpune duž cijele dubine ulaznog volumena. Lokalni dio ulaza zove se receptivno polje neurona (ili značajke) te je ekvivalentan veličini jezgre. Drugim riječima, receptivno polje neurona je skup svih elemenata prethodnog sloja koje mogu utjecati na taj neuron. Primijetite da neuroni u dubokoj konvolucijskoj mreži mogu modelirati indirektno interakciju većeg dijela ulaznih značajki. Takvu situaciju prikazuje slika 2.7 gdje neuron g_3 preko srednjeg sloja utječe na svih pet značajki u ulaznom sloju.



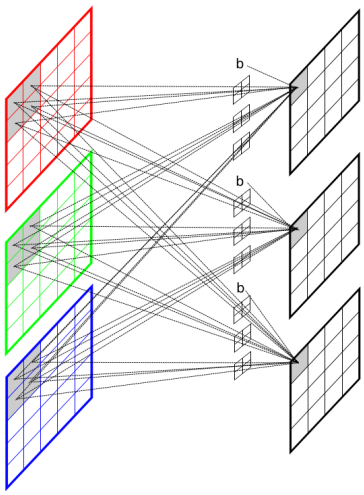


Slika 2.6: Stupac dubine [9] **Slika 2.7:** Indirektna interakcija neurona [10]

Manje veza u konvolucijskom sloju povlači i manji broj parametara, a time i smanjuje mogućnost da se model prenauči, što je jedan od glavnih razloga zbog čega se konvolucijski slojevi koriste kada na ulazu imamo visokodimenzionalne podatke kao što su slike. Manje parametara također povlači bržu evaluaciju $O(k \cdot n)$ u usporedbi s potpuno povezanim slojem $O(m \cdot n)$, gdje je k širina jezgre, a m dimenzija ulaznog vektora, pogotovo kada je $k << m$.

Što se tiče izlazne mape konvolucijskog sloja, postoje tri hiperparametra koji kon-

troliraju njegovu veličinu: broj kriški jezgri, pomak (engl. *stride*) i nadopuna nulom (engl. *zero-padding*). Broj kriški jezgre određuje kolika će biti dubina izlaznog volumena. Svaki skup parametara pojedine kriške će naučiti nešto drukčije u ulazu. Slika 2.8 prikazuje konvolucijski sloj s tri kriške jezgre. Skup neurona koji gledaju u istu regiju ulaza nazivamo stupac dubine (engl. *depth column*) što je prikazano na slici 2.6. Nadalje, moramo specificirati pomak kojim pomičemo jezgru prilikom konvolucije. Ako je pomak jednak 1, pomičemo jezgru jedan po jedan piksel duž širine i visine ulaza. Kako se konvolucijom prostorno (širina i dužina) smanjuje izlazna mapa, a najčešće želimo da su ulazne i izlazne mape prostorno jednake, potrebno je nadopuniti ulaz nulama sa svake strane. Broj redova nula oko ulaza je hiperparametar koji nam omogućava kontroliranje prostorne veličine izlazne mape. Nedostatak nadopune je taj što elementi na rubovima manje utječu na izlaz nego elementi na sredini. Moguće je izračunati prostornu veličinu izlazne mape koristeći formulu $(W - K + 2P)/S + 1$ gdje je W veličina ulaza, K širina jezgre, S pomak i P broj nula sa svake strane ulaza. Svaki izlaz iz konvolucijskog sloja se provlači kroz nelinearnu aktivacijsku funkciju. Nelinearnost je ključna za ispravan rad neuronske mreže, što je opisano na kraju potpoglavlja 2.1.1. U dubokim konvolucijskim mrežama najčešće se koristi ReLU aktivacijska funkcija.



Slika 2.8: Konvolucijski sloj [10]

Drastično smanjenje broja parametara može se postići uvođenjem pretpostavke da se isti parametri mogu koristiti za sve lokalne interakcije. Ako na ulazu imamo volumen 55x55x96, uvođenjem navedene pretpostavke dobili bismo samo 96 jedinstvenih skupova težina (jedan za svaku dubinsku kršku ulaza). Prema tome, jedan konvolucijski sloj ima $(K \cdot K \cdot D) \cdot C$ parametara koje koristi gdje je K širina jezgre, D

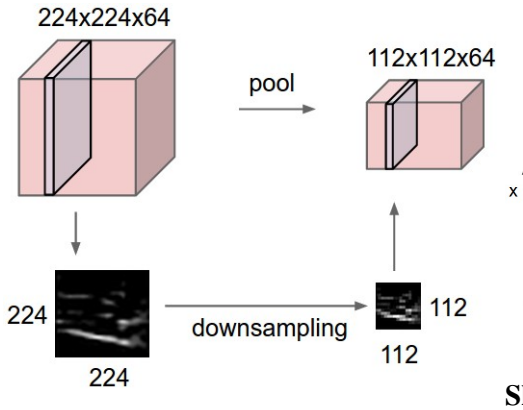
dubina ulaznog volumena, a C dubina izlazne mape (broj kriški jezgre). Također, valja spomenuti kako je konvolucija ekvivarijantna[1] s obzirom na pomak što znači da ako pomaknemo ulaz, pomaknut će se i mapa značajki. To je važno svojstvo jer u tom slučaju nije bitno na kojoj se lokaciji na slici nalazi objekt kojeg želimo klasificirati.

Ako označimo sa $\mathbf{W}$ 4-dimenzionalni tenzor jezgre, gdje element $w_{i,j,k,l}$ odgovora težini na lokaciji (k, l) jezgre koja povezuje mapu j na ulazu i mapu i na izlazu, te sa $\mathbf{x}$ označimo 3-dimenzionalni tenzor ulaza, gdje element $x_{j,k,l}$ odgovara vrijednosti ulaza na lokaciji (k, l) u mapi j, tada izlaz konvolucije možemo napisati kao:

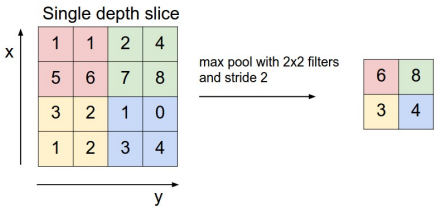
$$h_{i,m,n} = \sum_{j,k,l} x_{j,m+k,n+l} \cdot w_{i,j,k,l} + b_i$$

2.2.3. Sloj sažimanja

Uobičajeno je periodično stavljati slojeve sažimanja između uzastopnih konvolucijskih slojeva u arhitekturi mreže. Njegova funkcija je mapiranje skupa prostorno bliskih značajki na ulazu u jednu značajku na izlazu, smanjujući time broj parametara i računskih operacija. Sloj sažimanja djeluje nezavisno nad svakom dubinskom kriškom ulazne mape. Postoji nekoliko vrsta sažimanja od kojih se najčešće koristi sažimanje maksimalnom vrijednošću (engl. *max pooling*). Ostale vrste sažimanja su sažimanje srednjom vrijednošću (engl. *average pooling*), sažimanje L2 normom te sažimanje težinskim usrednjavanjem, gdje težine opadaju s udaljenosti od središnjeg neurona.



Slika 2.9: Sloj sažimanja [9]

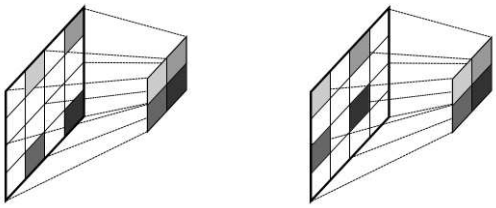


Slika 2.10: Sažimanje maksimalnom vrijednošću [9]

Sloj sažimanja ima dva hiperparametra: veličina regije i pomak. Sažimanje se tipično provodi bez preklapanja ulaza što znači da se ulazna mapa značajki podijeli na disjunktne regije te se svaka regija sažima u jednu značajku nepromijenjene semantičke

[1] $f(x)$ je ekvivarijantna s obzirom na g ako vrijedi: $f(g(x)) = g(f(x))$.

dimenzionalnosti. Drukčije rečeno, veličina regije i pomak su jednaki. Jedno bitno svojstvo sažimanja maksimalnom vrijednošću je invarijantnost[2] s obzirom na pomak što je vrlo korisno ako pretpostavimo da nam je za raspoznavanje bitnije detektirati prisutnost određene značajke, nego njezinu točno lokaciju. Veličina regije sažimanja regulira dozu invarijantnosti - što je regija veća, to je sažimanje invarijantno na veće pomake.



Slika 2.11: Invarijantnost na pomak [10]

Također, valja napomenuti da sloj sažimanja, za razliku od konvolucijskog i potpuno povezanog sloja, nema parametara koji se mogu podešavati prilikom učenja. Razlog tome je taj što je sažimanje fiksna operacija i uvijek se radi na isti način, bez potrebe za prilagođavanjem podacima.

2.2.4. Potpuno povezani sloj

Posljednji sloj koji se koristi u konvolucijskoj neuronskoj mreži je potpuno povezani sloj. Po načinu rada, on je identičan slojevima korištenim u unaprijednim neuronskim mrežama koje su opisane u potpoglavlju 2.1.2. Izlaz iz uzastopnih konvolucijskih slojeva i slojeva sažimanja se izravna (engl. *flatten*) kako bi dobili vektor značajki koji se predaje na ulaz potpuno povezanog sloja. Potpuno povezani sloj može obavljati dvije funkcije, regresiju ili klasifikaciju. Na primjer, možemo iskoristiti linearnu regresiju kako bi u lokalizaciji objekta dobili koordinate i dimenzije graničnog okvira koji nam govori gdje se na slici nalazi određeni objekt. S druge strane, klasifikacija nam govori koji je objekt nađen.

Postoje dva klasifikatora koji se koriste u potpuno povezanom sloju, SVM i Softmax klasifikator. SVM koristi višerazredni gubitak zglobnice (engl. *multiclass hinge loss*). Taj gubitak je postavljen tako da SVM "želi" da ispravna klasa za svaku sliku ima veću klasifikacijsku mjeru (engl. *classification score*) od pogrešne klase za neku marginu Δ. Klasifikacijske mjere predstavljaju izlaz potpuno povezanog sloja. Formula

[2] $f(x)$ je invarijantna s obzirom na g ako vrijedi: $f(g(x)) = f(x)$.

za višerazredni gubitak zglobnice je sljedeća:

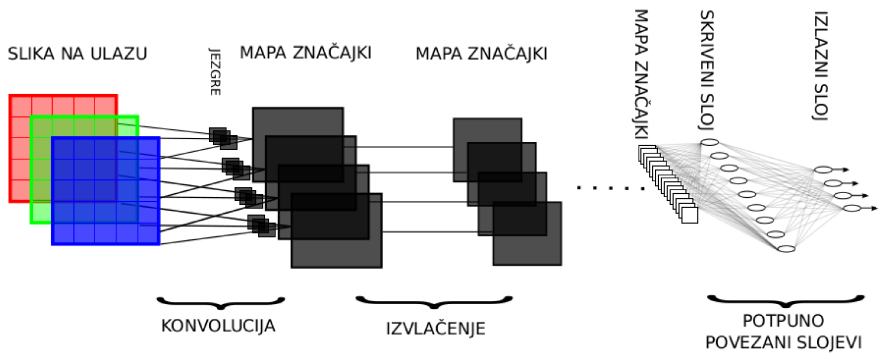
$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta)$$

gdje je s_j j-ti izlaz mreže tj. klasifikacijska mjera za klasu j, y_i stvarna oznaka klase, a s_{y_i} klasifikacijska mjera ispravne klase.

Drugi klasifikator koji se koristi je Softmax klasifikator. Za razliku od SVM-a koji tretira izlaz kao klasifikacijske mjere za svaku klasu (teško interpretabilne), softmax daje intuitivniji izlaz koji ima vjerojatnosnu interpretaciju. Softmax funkcija dana je formulom: $f_j(s) = \frac{e^{s_j}}{\sum_k e^{s_k}}$. Kao ulaz prima klasifikacijske mjere, a na izlazu daje normalizirane vjerojatnosti klasa. Drugim riječima, prima vektor proizvoljnih decimalnih brojeva i stišće ga u vektor vjerojatnosti čije su vrijednosti između 0 i 1 i čija je suma jednaka 1. S obzirom na to da izlaz Softmax-a ima vjerojatnosnu interpretaciju, minimizacijom negativne log izglednosti ispravne klase minimiziramo i ukupni gubitak, što se može tumačiti kao maksimalna procjena izglednosti (engl. *Maximum likelihood estimation (MLE)*). Kao funkciju gubitka koristimo gubitak unakrsne entropije (engl. *cross-entropy loss*) koji računa udaljenost između točne distribucije i distribucije koju predviđa model, a definiran je kao:

$$L_i = -\sum_{j=1}^{C} y_j log(s_j(\mathbf{x}))$$

gdje je C broj klasa, $\mathbf{x}$ ulaz Softmax funkcije koji predstavlja klasifikacijsku mjeru ili logit, $\mathbf{y}$ točna distribucija preko svih klasa za dani primjer (najčešće one-hot vektor), a $s_j(\mathbf{x})$ izlaz Softmax funkcije za klasu j. Radi jednostavnosti prikazali smo funkciju gubitka za samo jedan primjer dok u praksi definiramo gubitak nad skupom primjera pa će ukupan gubitak obično biti jednak prosječnom gubitku preko svih primjera.

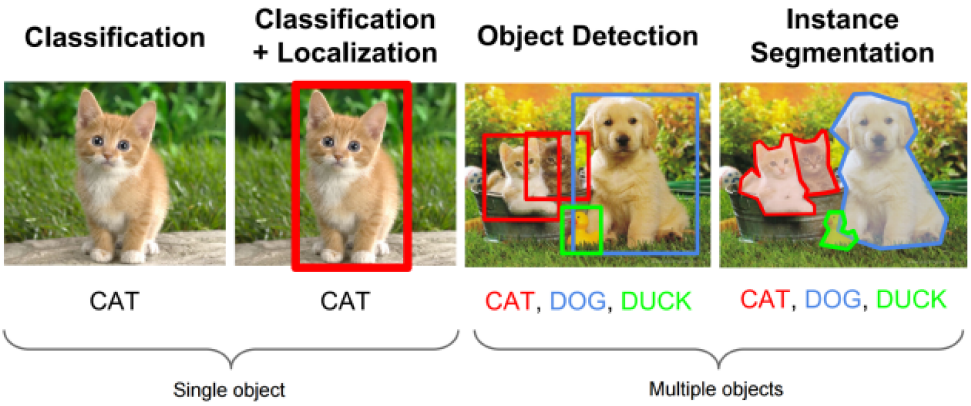


Slika 2.12: Konvolucijska mreža [10]

3. Detekcija objekata

3.1. Problem detekcije objekata

Problem lokalizacije i detekcije objekata na slikama ili videu se može podijeliti na četiri kategorije: klasifikacija, klasifikacija + lokalizacija, detekcija objekata i segmentacija. Primjeri tih kategorija, od najjednostavnije do najteže, prikazani su na slici 3.1.



Slika 3.1: Kategorije detekcije objekata [2]

Klasifikacija je najjednostavniji problem računalnog vida. Ulaz je slika određenih dimenzija koja se zatim provlači kroz konvolucijsku mrežu i na izlazu daje oznaku predviđene klase. Klasifikacija u kombinaciji s lokalizacijom objekta je malo kompliciraniji problem jer, osim što predviđa o kojem se objektu radi, nam govori i gdje se taj objekt na slici nalazi. To se najčešće radi uvođenjem regresije u mrežu koja će na izlazu dati koordinate i veličinu graničnog okvira (engl. *bounding box*) u kojem se nalazi objekt. Takva mreža će istovremeno obavljati dva zadatka, klasifikaciju i regresiju. Razlika između lokalizacije i detekcije objekata je u zadaći koju rade. Lokalizacija nalazi samo jedan objekt na slici, za razliku od detekcije objekata koja pronalazi sve različite objekte na danoj slici i klasificira ih. Prema tome, broj izlaza mreže koja rješava problem detekcije objekata nije fiksan, što uvodi dodatnu komplikaciju. Jedan od primjera takvih mreža je familija R-CNN o kojima će biti riječi u sljedećim potpo-

glavljima. Iako, treba napomenuti kako se u literaturi lokalizacija često poistovjećuje s detekcijom objekata. Posljednja kategorija je segmentacija primjerka koja je, u usporedbi s ostalim zadacima računalnog vida, jedna od najtežih. Treba napomenuti kako postoji razlika između semantičke segmentacije i segmentacije primjerka. Semantička segmentacija je proces pridjeljivanja semantičkih oznaka svim dijelovima slike, dok segmentacija primjerka detektira objekt, klasificira ga u jednu od klasa i zatim identificira obrise pronađenog objekta na razini piksela. Drugim riječima, segmentacija primjerka je kombinacija detekcije objekata i semantičke segmentacije. Primjer takve mreže je Mask R-CNN o kojem će biti govora u potpoglavlju 3.5.

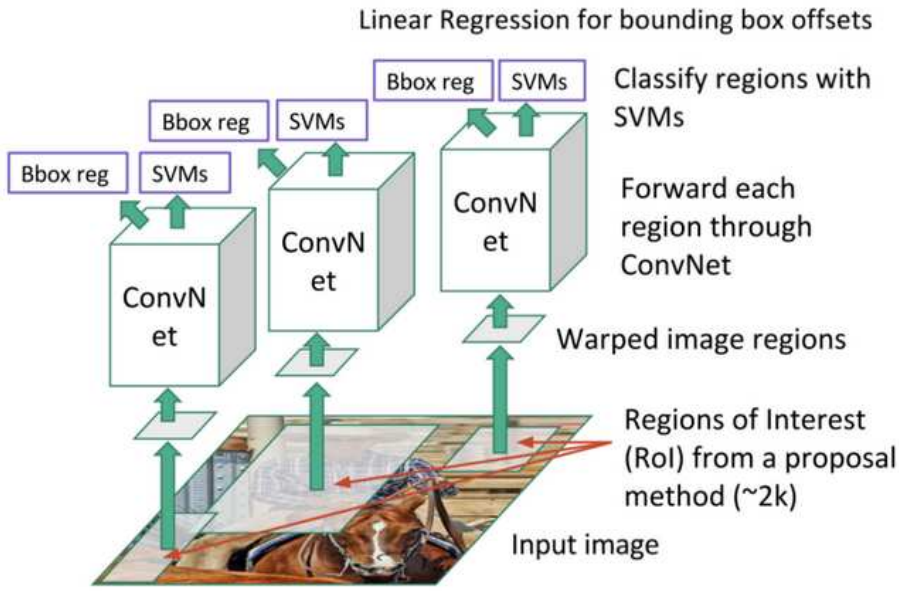
3.2. R-CNN

Prvi model koji je korištenjem konvolucijske mreže riješio problem detekcije objekata oslanjajući se pritom na vanjski sustav prijedloga područja (engl. *region proposal*) je R-CNN (engl. *Region-based Convolutional Neural Network*). Ujedno je to i prvi konvolucijski model koji je doveo do dramatično boljih rezultata u rješavanju zadatka detekcije objekata u odnosu na prijašnje sustave zasnovane na jednostavnijem HOG-u (engl. *histogram of gradients*). Cilj R-CNN-a je, kao i svim ostalim modelima za detekciju objekata, točno identificirati gdje se nalaze svi relevantni objekti na slici koristeći granični okvir.

Algoritam prijedloga područja koji se koristi u R-CNN-u je selektivno pretraživanje (engl. *selective search*). Algoritam selektivnog pretraživanja kombinira prednosti iscrpnog pretraživanja slike i segmentacije. Kao i iscrpno pretraživanje slike, nastoji pronaći sve moguće lokacije objekata, dok se za proces uzorkovanja koristi struktura slike što je karakteristično za segmentaciju. Ukratko, ideja je sljedeća: na početku se inicijaliziraju manja područja na slici koja se zatim spajaju hijerarhijskom grupacijom. Time će konačna grupa biti okvir koji sadrži cijelu sliku. Otkrivena područja se spajaju prema različitim teksturama, bojama i ostalim mjerilima sličnosti. Konačno, izlaz algoritma je manji broj predloženih područja od interesa (engl. *Regions of Interest (RoI)*) koje bi mogle sadržavati objekt. Primijetimo kako algoritam selektivnog pretraživanja ne daje kao izlaz područja jednake veličine, no to nas ne mora brinuti. Naprotiv, to i želimo jer na pojedinim slikama pojedini objekti ne moraju biti iste veličine, niti na istim pozicijama.

R-CNN kombinira algoritam selektivnog pretraživanja kako bi detektirao podru-

čja od interesa i duboko učenje kako bi klasificirao objekte u predloženim regijama. Svako predloženo područje, kojih ima 2000, je prilagođeno veličini (engl. *resize*) ulaza u konvolucijsku neuronsku mrežu, čija je arhitektura slična arhitekturi AlexNet-a, uz male preinake. Nadalje, provlačenjem kroz konvolucijsku mrežu se ekstrahira 4096-dimenzionalni vektor značajki koji se zatim predaje SVM klasifikatoru, opisanom u potpoglavlju 2.2.4, koji određuje jesu li i koji su objekti prisutni u predloženim područjima. Vektor značajki se dovodi i na ulaz linearnog regresora koji prilagođava granični okvir objekta i time smanjuje grešku lokalizacije. U zadnjem se koraku preklapajući okviri kombiniraju u jedan granični okvir koristeći potiskivanje višestrukih odziva (engl. *Non-maximal suppression*). Generalna ideja potiskivanje višestrukih odziva je smanjiti broj detekcija na slici na stvarni broj prisutnih objekata. Sažeto, mogli bismo postupak opisati na sljedeći način: generiranje područja od interesa, ekstrakcija značajki, klasifikacija regija temeljena na ekstrahiranim značajkama koristeći SVM klasifikator i na kraju potiskivanje višestrukih odziva. Gore opisani postupak ilustriran je na slici 3.2.



Slika 3.2: Arhitektura R-CNN [2]

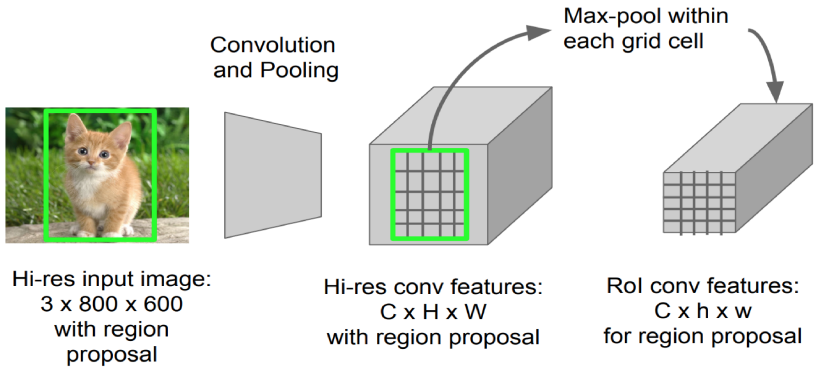
Već na prvi pogled možemo primijetiti da je navedeni model veoma spor. Za svaku sliku potrebno je generirati velik broj predloženih područja, što znači da algoritam mora za svaku sliku provesti selektivno pretraživanje kako bi pronašao moguća područja od interesa. Nadalje, za svako predloženo područje potrebno je ekstrahirati vektor značajki obavljanjem unaprijednog prolaza kroz konvolucijsku mrežu što veoma usporava rad algoritma. Također, cijeli proces uključuje treniranje tri različita modela

odvojeno kako bismo dobili konačne rezultate. Zbog svega navedenog, ovakav model ima veliku vremensku složenost što je definitivno velika mana budući da je krajnji cilj detektirati objekte u stvarnom vremenu (engl. *real-time*).

3.3. Fast R-CNN

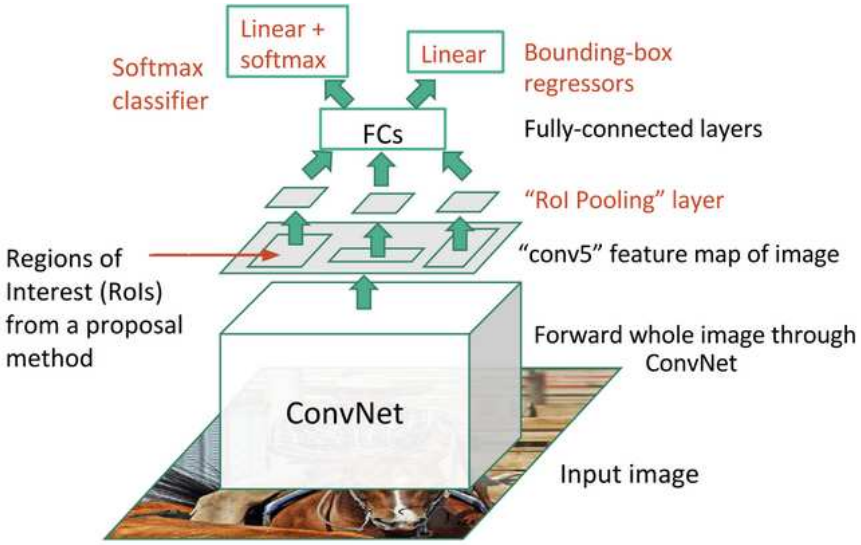
Neposredni potomak R-CNN-a u familiji mreža zasnovanim na sustavu predlaganja područja je njegova brža inačica pod nazivom Fast R-CNN. Svrha Fast R-CNN-a je smanjiti potrošnju vremena prouzročenu velikim brojem modela potrebnih za analizu svih područja od interesa. Brža inačica nalikuje izvornoj u mnogo segmenata, no ima dva bitna poboljšanja koja omogućuju efikasniju i bržu detekciju.

Prvo poboljšanje se odnosi na smanjenje broja unaprijednih prolaza kroz konvolucijsku mrežu što uvelike pridonosi ubrzanju algoritma. Sloj koji Fast R-CNN koristi kako bi ostvario navedeno poboljšanje naziva se sloj sažimanja područja od interesa (engl. *Region of Interest pooling layer (RoIPool)*). Mreža na ulazu prima cijelu sliku i skup predloženih područja. Cijela slika se provodi kroz konvolucijsku mrežu kako bismo dobili mapu značajki. Zatim, RoI sloj sažimanja iz dobivene mape značajki za svako predloženo područje od interesa ekstrahira vektor značajki unaprijed definirane veličine (npr. 7x7). Dio mape značajki koji predstavlja područje od interesa se podijeli na odjeljke jednake veličine kojih ima onoliko kolika je veličina izlaznog vektora značajki. Nakon toga se jednostavno provede sažimanje maksimalnom vrijednošću nad navedenim odjeljcima. Način rada RoI sažimanja ilustriran je slikom 3.3. Time je broj unaprijednih prolaza kroz mrežu smanjen s 2000 (jedan za svako područje od interesa) kod R-CNN-a na jedan u modelu Fast R-CNN.



Slika 3.3: RoI sloj sažimanja [21]

Drugo poboljšanje se odnosi na smanjenje broja različitih modela korištenih pri treniranju i evaluaciji. Kod R-CNN-a smo imali različite modele za ekstrakciju značajki slike (konvolucijska neuronska mreža), klasifikaciju (SVM) i stezanje graničnog okvira (linearni regresor), dok Fast R-CNN koristi jednu veliku mrežu koja računa sve troje. Svaki vektor značajki dobiven RoI sažimanjem se dovodi na ulaz slijeda potpuno povezanih slojeva koji se konačno granaju u dva izlazna sloja: jedan koji koristi Softmax klasifikator kako bi procijenio vjerojatnosti $C+1$ klasa objekata gdje C predstavlja broj različitih klasa objekata plus dodatna klasa "pozadina" (engl. *background*) koja predstavlja sve ostalo što nije jedan od mogućih objekata te drugi sloj koji na izlazu daje četiri decimalna broja (r, c, h, w) za svaki od C klasa objekata, gdje (r, c) predstavlja gornji lijevi kut, a (h, w) visinu i širinu graničnog okvira. Kako imamo dva izlaza, mreža se trenira koristeći višestruki gubitak (engl. *multitask loss*) koji je kombinacija gubitka unakrsne entropije (Softmax klasifikator) i L2 gubitka (linearni regresor). Arhitektura Fast R-CNN-a je prikazana na slici 3.4.



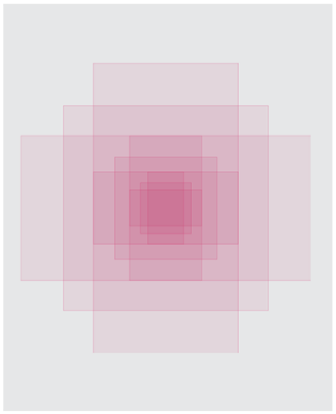
Slika 3.4: Arhitektura Fast R-CNN [15]

Iz svega možemo zaključiti kako je Fast R-CNN izveden puno bolje u smislu brzine, no još uvijek imamo usko grlo u našem algoritmu, a to je selektivno pretraživanje koje i dalje mora iscrpno pretražiti ulaz kako bi pronašao područja od interesa. Taj problem rješava poboljšana verzija našeg modela koja je opisana u sljedećem potpoglavlju, tzv. Faster R-CNN.

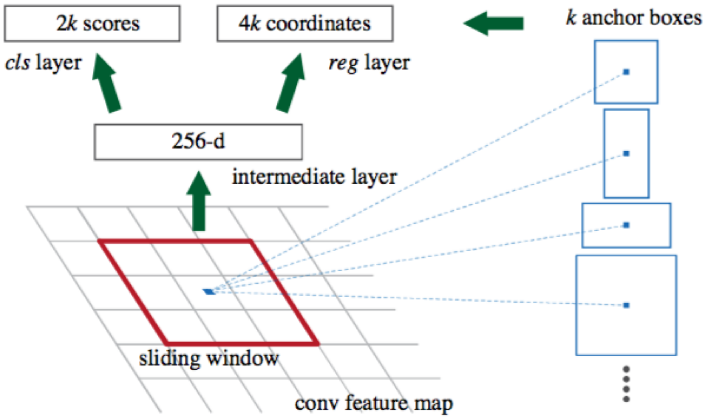
3.4. Faster R-CNN

U modelu Fast R-CNN se i dalje koristi selektivno pretraživanje za detekciju prijedloga područja, što je računalno skup proces. Faster R-CNN zamjenjuje algoritam selektivnog pretraživanja brzom neuronskom mrežom. Konkretno, uvodi se neuronska mreža za predlaganje područja (engl. *Region Proposal Network (RPN)*). Prema tome, Faster R-CNN se sastoji od dva modula. Prvi modul je duboka konvolucijska mreža za predlaganje područja, a drugi je Fast R-CNN detektor koji za detekciju koristi predložene regije. Cijeli sustav je jedna velika mreža koja efikasno rješava problem detekcije objekata.

Faster R-CNN na ulazu dobiva cijelu sliku te ju, kao i u prethodnom modelu, provlači kroz sljedove konvolucijskih slojeva i slojeva sažimanja kako bi na izlazu dobio mapu značajki. Klizni prozor veličine 3x3 se zatim pomiče preko dobivene mape značajki te se za svaki položaj kliznog prozora generira više mogućih područja koja se nazivaju sidreni okviri (engl. *anchor box*) koji imaju centar gdje i klizni prozor. Sidreni okviri su granični okviri s centrom gdje različitih veličina i omjera koji se koriste u RPN-u za predikciju lokacije mogućih objekata. Važno je shvatiti da, iako su sidreni okviri definirani na temelju mape značajki, konačni sidreni okviri referenciraju originalnu sliku. Slika 3.5 prikazuje sidrene okvire za jedan položaj kliznog prozora.



Slika 3.5: Sidreni okviri

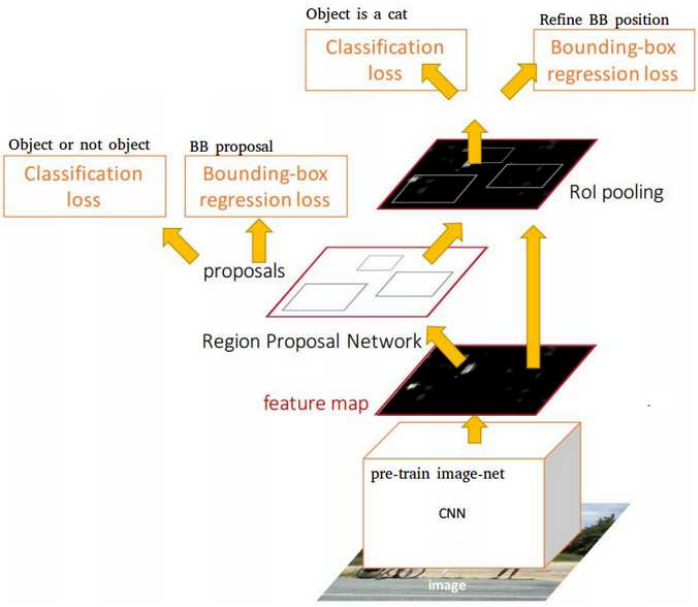


Slika 3.6: Detekcija sidrenih okvira [17]

Kao što je već spomenuto, RPN uzima sve okvire i na izlazu daje skup predloženih područja na kojima bi se mogli nalaziti objekti. Navedena mreža ima dva izlaza. Prvi izlaz je vjerojatnost da se u određenom okviru nalazi objekt. Primijetimo kako RPN ne mari za klasu objekta, već samo za to izgleda li nešto kao objekt, a ne kao pozadina.

Prema tome, za svaki položaj kliznog okvira dobit ćemo $2k$ vjerojatnosti na izlazu (2 različite klase - pozadina i "nepozadina" (engl. *foreground*)) gdje je k broj sidrenih okvira. To se može protumačiti kao "ocjena objektivnosti" za svako područje. Taj rezultat će biti iskorišten za filtriranje loših predikcija kao priprema za drugu fazu. Svi okviri ispod određenog praga "objektnosti" će biti odbačeni. Drugi izlaz je regresor graničnih okvira za podešavanje sidrenih okvira kako bi bolje odgovarali objektu koji se predviđa. Kako za svaki sidreni okvir imamo 4 decimalna broja, izlaz će dati $4k$ brojeva. Detektirani sidreni okvire se provode kroz algoritam potiskivanja višestrukih odziva kako bi se zadržali smo relevantni okviri. Opisani postupak za jedan klizni prozor ilustriran je na slici 3.6.

Jednom kada smo dobili prijedloge područja objekata, praktički se ponašamo kao kod Fast R-CNN modela. Dodajemo RoI sloj za sažimanje, potpuno povezane slojeve, te konačno Softmax klasifikacijski sloj i linearni regresor za granične okvire. Iz svega navedenog, možemo zaključiti kako je model Faster R-CNN, zbog uvođenja određenih optimizacijskih postupaka u svrhu ubrzanja procesa, kompliciraniji od svojih prethodnika, no njegov temeljni dizajn je praktički ostao identičan početnom R-CNN. Danas je ovo dominantan cjevovod za mnoge modele za detekciju objekata. Kompletna arhitektura modela prikazana je na slici 3.7.



Slika 3.7: Arhitektura Faster R-CNN [2]

3.5. Mask R-CNN

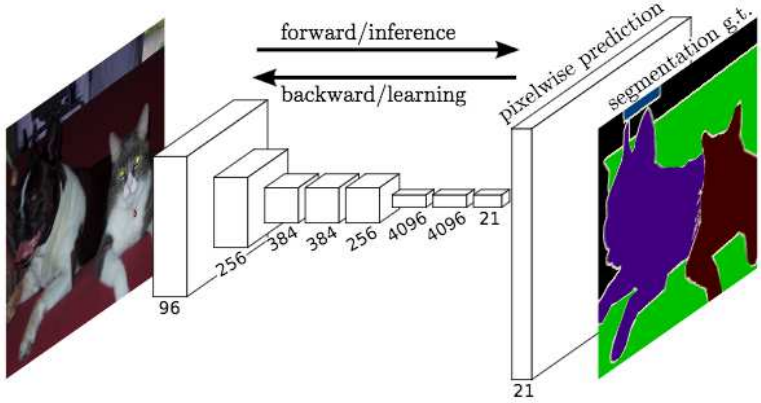
Posljednji model u R-CNN familiji je Mask R-CNN koji se koristi za segmentaciju primjerka. Segmentacija primjerka (engl. *instance segmentation*) kombinira elemente klasičnih zadataka računalnog vida - detekcije objekata, čiji je cilj klasificirati pojedine objekte i lokalizirati ih korištenjem graničnih okvira te semantičke segmentacije, čiji je cilj klasificirati svaki slikovni element (piksel) u odgovarajuću klasu bez mogućnosti razlikovanja primjeraka objekata. Mask R-CNN proširuje model Faster R-CNN dodajući mu granu za predikciju segmentacijskih maski za svako područje od interesa, paralelno s već postojećim granama za klasifikaciju objekta i regresiju graničnih okvira. Grana za predikciju maske je zapravo jednostavna potpuno konvolucijska mreža (engl. *Fully Convolutional Network (FCN)*) koja, primijenjena na svako područje od interesa, predviđa segmentacijske maske na razini piksela. Prema tome, Mask R-CNN možemo podijeliti na dva dijela - prvi dio je model Faster R-CNN, opisan u prošlom potpoglavlju, koji je zadužen za detekciju objekata i drugi dio, potpuno konvolucijska mreža (u daljnjem tekstu: FCN) zadužena za segmentaciju.

Glavni cilj FCN-a je provesti semantičku segmentaciju ulazne slike. To znači da ćemo na izlazu dobiti sliku istih prostornih dimenzija kao i ulazna slika, s različito obojanim pikselima. Kolika će biti dubina izlaza ovisi o broju kategorija koje segmentiramo. Semantička segmentacija dobivena potpuno konvolucijskom mrežom ilustrirana je slikom 3.8.

Slika 3.8: Semantička segmentacija

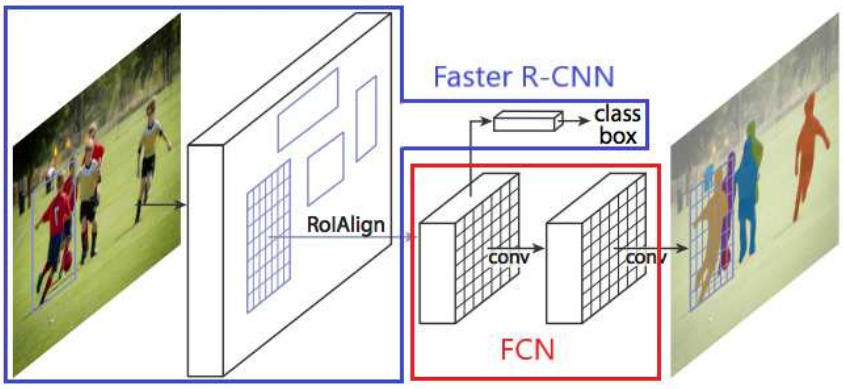
FCN je specifičan po tome što se sastoji samo od konvolucijskih slojeva i slojeva sažimanja. To znači da FCN nema potpuno povezani sloj na kraju, koji se tipično koristi za klasifikaciju. Umjesto toga koristi konvolucijske slojeve i Softmax funkciju kako bi pronašao najizgledniju klasu za svaki piksel slike. Primijetimo da mapa

značajki provlačenjem kroz sljedove konvolucijskih slojeva i slojeva sažimanja postaje prostorno sve manja i manja što predstavlja problem ako na izlazu želimo dobiti iste dimenzije kao i na ulazu. Kako bi riješio taj problem, FCN koristi "dekonvoluciju" ili unatražnu konvoluciju što je proces suprotan konvoluciji te naduzorkovanje (engl. *upsampling*) što je proces suprotan sažimanju tj. poduzorkovanju (engl. *downsampling*). Sažeto, FCN na ulazu dobiva sliku (odnosno mapu značajki u primjeru Mask R-CNN-a), a na izlazu daje matricu koja na pozicijama gdje se objekt nalazi ima 1, a svugdje drugdje 0, tzv. binarna maska. Arhitektura FCN-a je prikazana na slici 3.9.



Slika 3.9: Potpuno konvolucijska mreža [12]

Autori Mask R-CNN morali su napraviti jednu malu preinaku kako bi cjevovod radio ispravno. Problem u odnosu na Faster R-CNN bio je taj što se područja mape značajki dobivenih RoI sažimanjem nisu u potpunosti podudarala s područjima izvorne slike. Budući da segmentacija slike zahtjeva preciznost na razini piksela, za razliku od graničnih okvira, ovo je dovelo do netočnih rezultata. Problem je riješen uvođenjem metode pod nazivom usklađivanje područja od interesa (engl. *RoIAlign*). Način rada RoIAlign-a je sljedeći: zamislite da imamo sliku dimenzija 128x128 i mapu značajki dimenzija 25x25 te da želimo dobiti značajke koje odgovaraju veličini 15x15 piksela u lijevom gornjem kutu originalne slike. Jedan piksel u originalnoj slici odgovora $\approx 25/128$ piksela u mapi značajki što znači da 15 piksela odgovara $15 \cdot 25/128 \approx 2,93$ piksela u mapi značajki. U RoIPool-u bismo zaokružili ovaj broj na 2 što bi prouzročilo mali pomak. Međutim, u RoIAlign-u izbjegavamo takvo zaokruživanje. Umjesto toga koristimo bilinearno interpolaciju kako bismo dobili preciznu ideju kako izgleda piksel 2,93. Ovo nam na visokoj razini omogućuje izbjegavanje neusklađenosti koje uzrokuje RoIPool. Iako se čini kao mala promjena, RoIAlign daje puno bolje rezultate.



Slika 3.10: Arhitektura Mask R-CNN

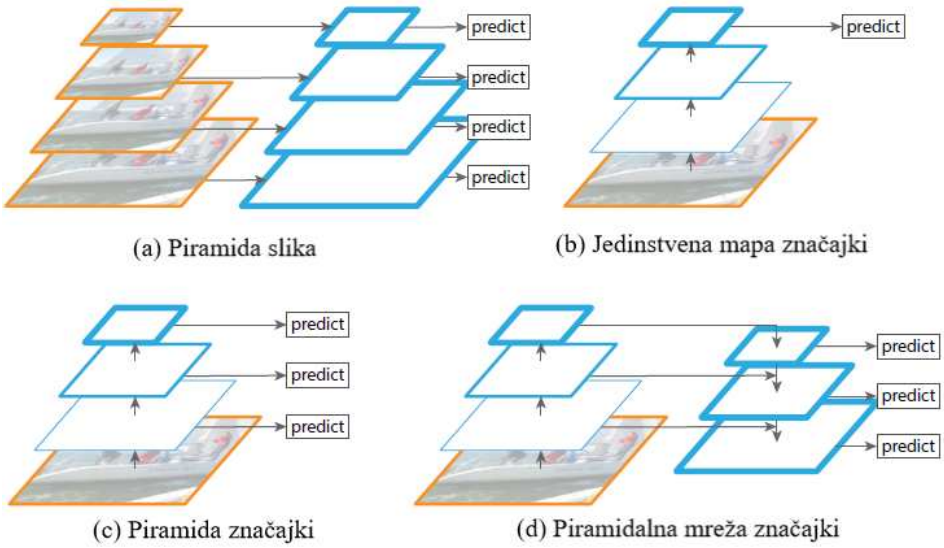
Jednom kad su maske generirane, Mask R-CNN ih kombinira s klasifikacijom i graničnim okvirima iz Faster R-CNN kako bi dobio konačne rezultate. Čitava arhitektura Mask R-CNN prikazana je na slici 3.10. Mask R-CNN se također koristi i za procjenu čovjekovog stava (engl. *Human pose estimation*), no to izlazi iz okvira ovog rada.

3.6. Feature Pyramid Network (FPN)

Obratimo sada pozornost na prvi dio cjevovoda: originalni dizajn bržih inačica familije R-CNN-a (uključujući Mask R-CNN) koristi mapu značajki u samo jednoj veličini (engl. *single-scale feature map*) koja se predaje na ulaz RPN-a (slika 3.11b). Detekcija objekata različitih veličina može biti vrlo izazovna, pogotovo kada se radi o manjim objektima. Iz tog razloga, model s ekstraktorom jedinstvene mape značajki neće davati maksimalnu performansu. Jedan način rješavanja ovog problema bio bi korištenje iste slike u različitim veličinama za detekciju objekata (slika 3.11a). Međutim, obrađivanje slika različitih veličina je dugotrajan proces i zahtjeva previše memorije za učenje takvog modela. Alternativno, ovaj problem se može riješiti stvaranjem piramide značajki (slika 3.11c). SSD (engl. *Single Shot Detector*) je jedan od modela koji koristi ovaj piramidalni koncept. Međutim, ni taj pristup ne daje najbolje rezultate. Mape značajki koje su bliže sloju slike su sastavljene od struktura niske razine i nisu učinkovite za preciznu detekciju objekata.

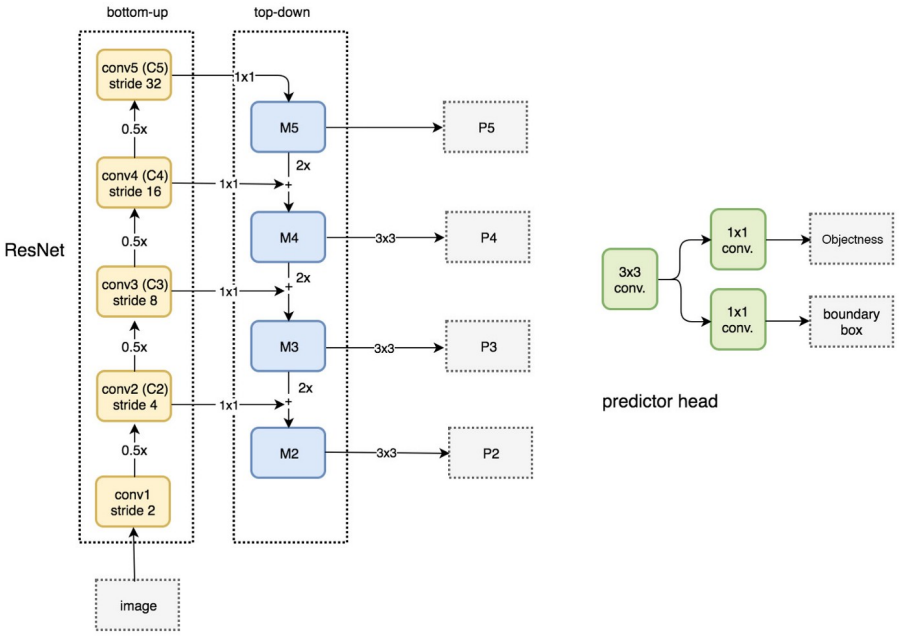
Ovdje na scenu dolazi piramidalna mreža značajki (engl. *Feature Pyramid Network (FPN)*). FPN je ekstraktor mape značajki dizajniran za piramidalni koncept koji daje veću preciznost, a pritom ne gubi na brzini. Važno je napomenuti da FPN nije detek-

tor objekata, već ekstraktor mapi značajki koje se zatim koriste u cjevovodu modela za detekciju objekata. Ovakav ekstraktor zamjenjuje postojeći u arhitekturi Mask R-CNN-a (ili Faster R-CNN-a) koji, za razliku od originalnog, na izlazu generira više slojeva mapa značajki (engl. *multi-scale feature map*) s boljom kvalitetom od običnih piramidi značajki.



Slika 3.11: Različite arhitekture ekstraktora značajki [11]

Tri su glavne komponente u radu FPN-a: put odozdo prema gore (engl. *bottom-up pathway*), put odozgo prema dolje (engl. *top-down pathway*) te lateralne veze (slika 3.11d). Put odozdo prema gore je obična konvolucijska mreža za ekstrakciju značajki. Kako idemo prema gore, prostorna razlučivost se smanjuje. Sa više detektiranih struktura visoke razine, semantička vrijednost svakog sloja se povećava. Prije spomenuti SSD za detekciju također koristi više mapi značajki. Međutim, donji slojevi nisu korišteni za detekciju objekata iako su u visokoj razlučivosti. Njihova semantička vrijednost nije dovoljno visoka da bi opravdala njihovu upotrebu. Zbog toga, SSD koristi samo gornje slojeve za detekciju i posljedično daje dosta lošiju performansu za manje objekte. FPN kombinira semantički jake značajke niske razlučivosti sa semantički slabim značajkama visoke razlučivosti pomoću lateralnih veza i , FPN koristi put prema dolje za konstruiranje slojeva veće razlučivosti iz semantički bogatog sloja (gornji sloj). Semantička vrijednost rekonstruiranih slojeva je visoka, no lokacije objekata nisu precizne zbog poduzorkovanja i naduzorkovanja. Iz tog razloga se dodaju lateralne veze između rekonstruiranih slojeva i odgovarajućih mapi značajki kako bi detektor mogao bolje lokalizirati objekte. Rad piramidalne mreže značajki ilustriran je na slici 3.12.



Slika 3.12: Rad piramidalne mreže značajki (FPN) [11]

Put prema gore koristi ResNet koji se sastoji od više konvolucijskih modula od kojih svaki ima više konvolucijskih slojeva. Kako se pomičemo prema gore prostorna dimenzija se smanjuje za 1/2 (dvostruko veći pomak). Izlaz svakog konvolucijskog modula je labeliran kao C_i, a kasnije korišten u putu prema dolje. Točnije, primjenjuje se konvolucija 1x1 na C_5 mapi značajki kako bi se smanjila dubina kanala i stvorila mapa značajki M_5 koja predstavlja prvi sloj mapi značajki koji će se koristiti u detekciji objekata. Kako idemo po putu odozgo prema dolje, naduzorkujemo prethodni sloj za 2 koristeći naduzorkovanje najbližeg susjeda. Nakon toga, ponovno primijenimo 1x1 konvoluciju na odgovarajućoj mapi značajki u putu prema gore, a zatim ih dodamo mapi značajki u putu prema dolje po elementu (engl. *element-wise*). Sve dobivene mape značajki provlačimo kroz konvolucijski sloj 3x3 kako bismo smanjili efekt preklapanja (engl. *aliasing*) prouzrokovan stapanjem sa naduzorkovanim slojem. Navedeni postupak se ponavlja sve do P_1 isključivo jer je prostorna dimenzija C_1 prevelika i previše bi usporila proces. Konačno, na svaku od dobivenih mapi značajki P se primjenjuje konvolucijski filter 3x3 nakon čega slijede zasebni filteri 1x1 za klasifikaciju da li se objekt nalazi u sidrenom okviru i za linearnu regresiju graničnih okvira. Ti 3x3 i 1x1 konvolucijski slojevi se nazivaju glavom RPN-a (engl. *RPN head*).

Nadalje, nakon što FPN generira piramidu mapi značajki, RPN se koristi za generiranje područja interesa. Na temelju veličine područja interesa, bira se sloj mapa

značajki u najprikladnijom veličini za ekstrakciju dijelova značajki. Formula za biranje mape značajki se temelji na širini i visini generiranog područja od interesa:

$$\lfloor k = k_0 + log_2(\sqrt{wh}/224) \rfloor$$

gdje je k_0 = 4 pošto originalni Faster R-CNN koristi C_4 kao mapu značajki jedinstvene veličine, dok k predstavlja redni broj P_k sloja koji će se koristiti. Nakon toga, provodi se RoI sažimanje i rezultat se daje šalje na ulaz Mask R-CNN za završnu predikciju. Dodavanjem piramidalne mreže značajki u RPN, poboljšao se prosječni odziv, kao i kvaliteta detekcije manjih objekata.

4. Mjera uspješnosti

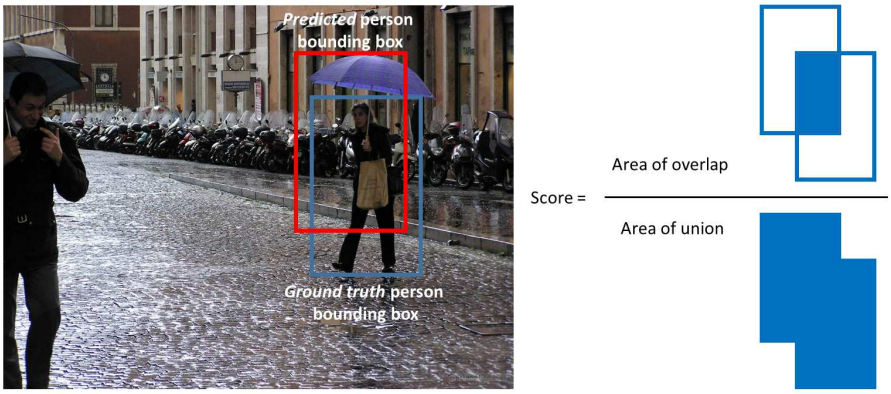
Sa svakim novim eksperimentom, jedno od najvažnijih pitanja je što zapravo želimo mjeriti. Kao što je prije spomenuto, izazov detekcije objekata je, u isto vrijeme, zadatak regresije i klasifikacije. Zbog složenosti problema, razvijene su određene mjere uspješnosti modela uzimajući u obzir prostornu lokaciju detektiranog objekta i točnost predviđenih klasa. Prije nego uđemo u dublje u evaluaciju modela za detekciju objekata, potrebno je razmotriti neke temeljne koncepte.

4.1. Osnovni koncepti

Mjera pouzdanosti (engl. *confidence score*) je rezultat predikcije klasifikatora koji predstavlja vjerojatnost da sidreni okvir sadrži objekt.

Da bismo procijenili preciznost lokalizacije objekata, potrebno je algoritmom potiskivanja višestrukih odziva odbaciti sve okvire s niskim povjerenjem. Za tu svrhu koristimo Jaccardov indeks, koji se u literaturi često naziva **IoU** (engl. *Intersection over Union*). IoU predstavlja površinu preklapanja predviđenog okvira i ispravnog (engl. *ground truth*) okvira s rasponom [0,1]. Što je veći IoU, to je predviđena lokacija danog objekta bolja. Obično ostavljamo granične okvire koji imaju IoU veći od nekog postavljenog praga. Iako u praksi radi dobro za relativno velike objekte, poznat je problem da za vrlo male objekte (npr. 10x10 piksela) IoU ne funkcionira najbolje. Čini se da je to zato što male varijacije u predviđenim koordinatama piksela mogu znatno pogoršati preklapanje na sitnim objektima. Intuitivna vizualizacija IoU-a vidljiva je na slici 4.1.

Obje navedene mjere se koriste kao kriterij za utvrđivanje je li model ispravno ili pogrešno detektirao objekt. Detekcija se smatra ispravnom (engl. *true positive*) (**TP**) ako zadovoljava sljedeća tri uvjeta: mjera pouzdanosti je veća od zadanog praga, predviđena klasa odgovara ispravnoj klasi, predviđeni granični okvir ima IoU s ispravnim



Slika 4.1: Vizualizacija IoU

okvirom veći od praga (npr. 0.5). Kršenje bilo kojeg od zadnja dva uvjeta daje pogrešno detektirani objekt (engl. *false positive*) (**FP**). U slučaju da više predviđenih graničnih okvira odgovara istom ispravnom okviru, samo okviri s najvišom mjerom pouzdanosti se smatraju ispravnima, dok se ostali smatraju pogrešno detektiranim. Nadalje, ako je mjera pouzdanosti okvira koji bi trebao detektirati objekt manja od određenog praga, model neće detektirati stvarni objekt (engl. *false negative*) (**FN**). S druge strane, ako je mjera pouzdanosti detekcije, koja nije trebala detektirati ništa, veća od praga, model ispravno neće detektirati nepostojeći objekt (engl. *true negative*) (**TN**), no u detekciji objekata ovakve detekcije uglavnom ne uzimamo u obzir.

U tipičnom skupu podataka bit će više klasa i njihova distribucija je neujednačena. Prema tome, jednostavna mjera uspješnosti temeljena na točnosti neće davati dobre rezultate. Također je važno procijeniti rizik pogrešnih klasifikacija. Dakle, potrebno je povezati mjeru uspješnosti modela sa svakim detektiranim graničnim okvirom. Za tu svrhu je uvedena **prosječna preciznost** (engl. *Average Precision (AP)*). Za razumijevanje prosječne preciznosti, potrebno je razumjeti što znači preciznost i odziv klasifikatora.

Preciznost (engl. *precision*) predstavlja omjer ispravno detektiranih objekata i ukupnog broja objekata koje je klasifikator predvidio.

$$precision = \frac{TP}{TP + FP}$$

Odziv (engl. *recall*) predstavlja omjer ispravno detektiranih objekata i ukupnog broja objekata u skupu podataka.

$$recall = \frac{TP}{TP + FN}$$

4.2. Prosječna preciznost

Vrlo je važno napomenuti da postoji inverzni odnos između preciznosti i odziva te da su navedene mjere ovisne o postavljenom pragu ocjene modela (IoU). Postavljanjem različitih pragova, dobivamo različite parove preciznosti i odziva. S odzivom na x-osi i preciznošću na y-osi, možemo nacrtati krivulju preciznosti i odziva. Sada možemo definirati prosječnu preciznost: AP predstavlja površinu ispod krivulje preciznost-odziv (engl. *Precision-Recall (PR) curve*).

$$AP = \int_0^1 p(r)dr$$

Kao i njegove komponente, i prosječna preciznost će se nalaziti u rasponu [0,1]. Najčešće korištena mjera u detekciji objekata je srednja prosječna preciznost (engl. *mean Average Precision (mAP)*). To je jednostavno uprosječena vrijednost prosječnih preciznosti izračunatih za sve klase.

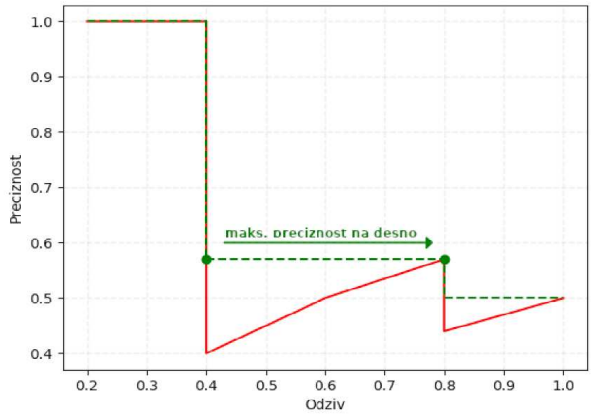
Kako bi čitatelju približio postupak crtanja PR krivulje, opisat ću pojednostavljeni primjer izračuna prosječne preciznosti. Za svrhu primjera, uzmimo da čitavi skup podataka sadrži samo pet objekata klase koju želimo detektirati. Prikupimo sve predikcije iz svih slika u skupu podataka za željenu klasu te ih silazno sortiramo po predviđenoj mjeri pouzdanosti. Sortirane vrijednosti su prikazane u tablici 4.1. Pouzdanosti su proizvoljno izabrane radi bolje ilustracije, a za IoU je postavljen na 0.5.

Rang	Ispravnost	Pouzdanost	Preciznost	Odziv
1	True	95%	1.0 ↑	0.2 ↑
2	True	95%	1.0 -	0.4 ↑
3	False	91%	0.67 ↓	0.4 -
4	False	87%	0.5 ↓	0.4 -
5	False	83%	0.4 ↓	0.4 -
6	True	81%	0.5 ↑	0.6 ↑
7	True	75%	0.57 ↑	0.8 ↑
8	False	65%	0.5 ↓	0.8 -
9	False	61%	0.44 ↓	0.8 -
10	True	50%	0.5 ↑	1.0 ↑

Tablica 4.1: Vrijednosti preciznosti i odziva

Uzmimo šesti red kako bismo pokazali kako se preciznost i odziv računaju. U oba slučaja brojnik u omjeru predstavlja broj ispravnih predikcija modela. Preciznost se može izračunati kao 3/6 = 0.5 gdje nazivnik predstavlja ukupni broj predikcija. Odziv se računa kao 3/5 = 0.6 gdje nazivnik predstavlja ukupni broj objekata u skupu podataka. Vrijednosti odziva se povećavaju kako idemo prema dolje u tablici, što bi značilo da se odziv povećava sa smanjenjem mjere pouzdanosti prilikom ispravne detekcije. Što se tiče preciznosti, ona raste s ispravnom detekcijom, a pada s pogrešnom.

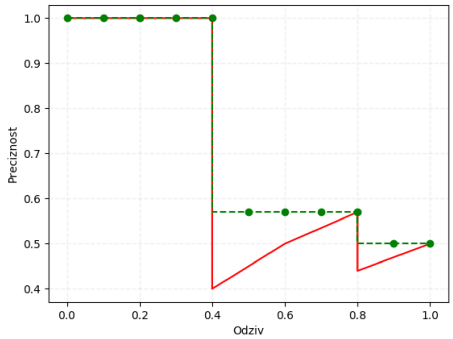
Budući da, zbog varijacija u rangiranju primjera po mjeri pouzdanosti, graf krivulje nalikuje na "pilu", prije računanja prosječne preciznosti, graf se najčešće izgladi (engl. *smooth*) kako bismo dobili stepenasti graf koji monotono pada i pogodniji je za izračun. Grafički, na svakoj vrijednosti odziva, vrijednost preciznosti je zamijenjena s maksimalnom preciznošću desno od trenutne vrijednosti odziva, što je prikazano na slici 4.2. Crveni graf predstavlja originalnu krivulju nacrtanu prema gornjoj tablici, dok zeleni graf označava izglađenu PR krivulju. Izračunata vrijednost bit će manje osjetljiva na male varijacije u rangiranju. Matematički, zamjenjuje se vrijednost preciznosti za odziv $\hat{r}$ s maksimalnom preciznošću za svaki odziv veći od $\hat{r}$: $p_{interp}(r) = \max_{\hat{r} \geqslant r} p(\hat{r})$



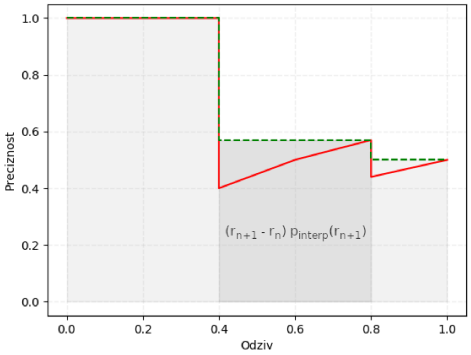
Slika 4.2: Uglađivanje PR krivulje

Sada kada imamo izglađeni graf, možemo izračunati vrijednost prosječne preciznosti. Postoje dva glavna načina kako aproksimirati površinu ispod krivulje - interpolacija krivulje i AUC (engl. *Arena Under Curve*) metoda. Tradicionalni način na koji se krivulja interpolira je koristeći 11 točaka (engl. *11-point interpolated average precision*). Za svaku od 11 različitih vrijednosti odziva u rasponu [0,1] s pomakom od 0.1 se iščita iz grafa vrijednost preciznosti, a zatim se te vrijednosti uprosječe kako bismo

dobili konačnu vrijednost prosječne preciznosti. Interpolirana krivulja prikazana je na slici 4.3. Ova metoda se koristi u PASCAL VOC2008 izazovu za detekciju objekata o čemu će biti više riječi u sljedećem potpoglavlju. Međutim, ovakva metoda interpolacije je aproksimacija koja ima dva glavna problema. Kao prvo, manje je precizna jer uzimamo u obzir samo 11 točaka na grafu, a kao drugo, izgubila je sposobnost mjerenja razlike između metoda s niskom prosječnom preciznošću. Stoga je nakon 2008. za PASCAL VOC usvojen drugačiji način izračuna prosječne preciznosti. Za kasnije PASCAL VOC izazove krivulja je uzorkovana na vrijednostima odziva, kad god padne maksimalna vrijednost preciznosti. S ovom promjenom, može se točno izračunati površina ispod izglađene PR krivulje. Ideja je prikazana na slici 4.4.



Slika 4.3: Interpolacija PR krivulje



Slika 4.4: AUC metoda

4.3. Izazovi detekcije objekata

Nekoliko skupova podataka je objavljeno za izazove detekcije objekata. Istraživači objavljuju rezultate svojih algoritama evaluiranih nad tim skupovima podataka. Za ovaj rad nam nisu toliko bitni skupovi podataka koliko definirane mjere za ocjenu performanse modela korištene pri evaluaciji naučenog modela na validacijskom skupu. S obzirom na to da preciznost i odziv ovise od IoU vrijednosti praga, za svaki model imat ćemo različite PR krivulje, a time i različite vrijednosti prosječne preciznosti, ovisno o tome koji smo prag odabrali. Ako povisimo prag, modeli s lošijom lokalizacijom će imati manju prosječnu preciznosti jer će početi proizvoditi više pogrešnih detekcija. Različiti izazovi koriste različite pragove za mjeru uspješnosti modela. Ovdje će ukratko biti opisani samo najvažniji skupovi podataka za zadatak detekcije objekata i njihove mjere za ocjenu modela.

PASCAL VOC (engl. *PASCAL Visual Object Classification*) je jedan od poznatijih skupova podataka za detekciju, klasifikaciju i segmentaciju objekata. Objavljeno je 8 različitih izazova u rasponu 2005. do završne 2012. godine kada su autori počeli surađivati s autorima ImageNet ILSVRC-a. Svaki izazov je specifičan na svoj način. Skup ima oko 10 000 slika za treniranje i validaciju koje sadrže granične okvire objekata. Iako PASCAL VOC sadrži samo 20 različitih klasa, još uvijek se smatra referentnim skupom podataka u problemu detekcije objekata. Važno je napomenuti da PASCAL VOC izazovi uključuju dodatna pravila za definiranje ispravnih i lažnih detekcija. Naime, u slučaju da više detekcija objekata odgovara istoj temeljnoj istini, samo se ona detekcija s najvišom mjerom pouzdanosti smatra ispravnom detekcijom (TP), dok se ostale smatraju pogrešnima (FP). PASCAL VOC izazov definira mAP mjeru uprosječenu preko 20 klasa koristeći jedan IoU prag postavljen na 0.5. Kao što je već spomenuto, iteracija izazova iz 2008. koristi interpolaciju koristeći 11 točaka, dok kasnije iteracije (VOC2010-2012) koriste AUC metodu za računanje prosječne preciznosti. PASCAL VOC mjera uspješnosti se može uzeti kao standardna mjera za evaluaciju modela za detekciju objekata; mjere usvojene u ostala dva izazova mogu se promatrati kao varijante spomenute mjere.

ImageNet je 2009. objavio prvi skup podataka za detekciju objekata s graničnim okvirima. Skup za treniranje danas sadrži preko 14 milijuna slika, od čega nešto više od milijun označenih, te više od 20 000 klasa. Unatoč tome, rijetko se koristi jer veličina skupa podataka zahtjeva ogromnu računalnu snagu za treniranje modela. Također, veliki broj klasa zna zakomplicirati zadatak detekcije objekata. ImageNet je također 2010. pokrenuo prvi izazov pod nazivom ILSVRC (engl. *ImageNet Large Scale Visual Recognition Challenge*) nakon kojeg je slijedilo još sedam izazova objavljivanih na godišnjoj bazi. Uz male preinake, svi su modeli nakon PASCAL VOC2012 izazova nastavili koristiti prosječnu preciznost definiranu u VOC2012. Koristi se uprosječena mjera po svih klasama za IoU = 0.5 koja koristi AUC metodu da računanje prosječne preciznosti za jednu klasu. ILSVRC je promijenio samo IoU prag preklapanja za sitne objekte efektivno povećavajući veličinu označenog graničnog okvira u svrhu bolje ocjene modela.

COCO (engl. *Common Objects in COntext*) skup podataka, razvijen u Microsoftu 2015. godine, koristi se za mnoge izazove kao što su: panoptička detekcija objekata (engl. *panoptic detection*), otkrivanje ključnih točaka (engl. *key point detection*), segmentacija stvari (engl. *stuff segmentation*) te detekcija objekata. Skup podataka se

mijenja svake godine, a trenutno sadrži oko 330 000 slika. Više od 200 000 slika je označeno graničnim okvirima, a svaki granični okvir je označen jednom od 80 različitih klasa. Najnoviji istraživački radovi daju rezultate samo za COCO skup podataka. COCO izazov za detekciju objekata je trenutno najpoznatiji takav izazov. Za razliku od prijašnjih izazova, COCO je definirao mjeru uspješnosti na drugačiji način. Umjesto računanja prosječne preciznosti za samo jedan IoU prag, COCO računa prosječnu preciznost za raspon pragova [0.5 : 0.95 : 0.05] (deset pragova između 0.5 i 0.95 uključivo s pomakom 0.05) i zatim se dobivene vrijednosti uprosječe kako bismo dobili konačni AP za određenu klasu. Postupak se ponavlja za sve klase, a dobivene vrijednosti se ponovno uprosječe po svim klasama dajući na izlazu jedinstven broj za ocjenu performanse modela. Također, za razliku od PASCAL VOC-a i ILSVRC-a, COCO izazov koristi interpolaciju sa 101 točkom za izračun prosječne preciznosti. U tablici ispod prikazane su sve mjere uspješnosti modela korištene u COCO izazovu.

Prosječna preciznost (AP):

AP : AP za $IoU = 0.5 : 0.05 : 0.95$ (primarna mjera uspješnosti)

$AP_{IoU=0.5}$: AP za $IoU = 0.5$ (PASCAL VOC mjera uspješnosti)

$AP_{IoU=0.75}$: AP za $IoU = 0.75$ (stroga mjera uspješnosti)

AP za različite veličine:

AP^{small} : AP za male objekte: $povrsina < 32^2$

AP^{medium} : AP za srednje objekte: $32^2 < povrsina < 96^2$

AP^{large} : AP za velike: $96^2 < povrsina$

Prosječni odziv (AR):

$AR^{max=1}$: AR za 1 detekciju po slici

$AR^{max=10}$: AR za 10 detekcija po slici

$AR^{max=100}$: AR za 100 detekcija po slici

AR za različite veličine:

AR^{small} : AR for small object: $povrsina < 32^2$

AR^{medium} : AR for medium object: $32^2 < povrsina < 96^2$

AR^{large} : AR za velike objekte: $96^2 < povrsina$

5. Implementacija

5.1. Skup podataka

U ovome radu kao eksperiment radila se detekcija objekata u odsječku snimke nogometne utakmice. Točnije, prvih 5 minuta utakmice 1. HNL između Dinama i Hajduka odigrane na stadionu Maksimir. Snimka sadrži 7515 sličica (engl. *frame*) razlučivosti 3260 x 570 koje se koriste kao skup podataka u eksperimentu. Primjer jedne sličice prikazan je na slici 5.1.

Slika 5.1: Primjer sličice iz snimke nogometne utakmice

Svaka sličica u snimci sadrži točno 11 igrača jedne momčadi, 11 druge momčadi i sudca, što znači da je moguće detektirati 3 različite klase: momčad A, momčad B i sudac. Uz snimku je pripremljena *.csv* datoteka s lokacijama svih relevantnih objekata za svaku od sličica u snimci. U daljnjem tekstu, relevantne objekte nazivat ću jednostavno igračima, uključujući i sudca. Svaka sličica ima svoj *frameID*, kao i svaki igrač svoj *playerID*, igrači momčadi A u rasponu $[0, 10]$, igrači momčadi B $[12, 22]$ dok sudac ima broj 11.

Neke situacije na utakmici su lakše za detektirati, primjerice kada su igrači udaljeni jedni od drugih i jasno vidljivi na snimci predikcija će biti bolja. S druge strane, neke situacije će biti izazovnije za detektor i vjerojatnije je da će imati lošiju performansu. Za validacijski skup je izabrana zadnja četvrtina jer najbolje prikazuje različite rasporede igrača na terenu. Osvrnimo se sada na probleme u skupu učenja koji otežavaju detekciju igrača. Postoji nekoliko problema: stapanje igrača s pozadinom (slika 5.2a), preklapanje igrača (5.2b), igrači nisu iste veličine (slika 5.2c). Svi primjeri uzeti

(a) Stapanje s pozadinom **(b)** Preklapanje igrača **(c)** Različite veličine igrača

Slika 5.2: Problemi u skupu podataka

su iz validacijskog skupa. Budući da igrači nisu u velikoj razlučivosti, detektoru je vrlo teško razaznati između pozadine i igrača koji su iste boje ili između dva igrača iz iste momčadi. Detekcija malih objekata može se poboljšati korištenjem FPN-a koji radi ekstrakciju značajki pomoću piramidalnog koncepta opisanog u poglavlju 3.6 i povećanja podataka. Više riječi o povećanju podataka bit će u potpoglavlju 5.3.1.

5.2. Programska podrška

Google Colab

Za ovaj eksperiment koristio sam programski jezik Python u interaktivnom okruženju Colab. Colaboraty, ili skraćeno "Colab", je proizvod Googlovog istraživačkog tima koji bilo kome omogućava pisanje i izvršavanje proizvoljnog Python koda putem preglednika, a posebno je prikladan za strojno učenje, analizu podataka i obrazovanje. Tehnički gledano, Colab je zasnovan na Jupyter bilježnicama (engl. *Jupyter notebooks*) koje ne zahtijevaju nikakvo postavljanje, a pružaju besplatan pristup računalnim resursima, uključujući i grafičke procesorske jedinice - GPU (engl. *Graphical Processing Unit*). Cijeli projekt, uključujući i sve korištene podatke, nalazi se na Google Disku kojeg je potrebno platiti ako se želi koristiti više od 15GB memorije. Razlog zbog kojeg sam koristio Colab je besplatna mogućnost korištenja GPU za brže učenje modela. Budući da je Google Colab potpuno besplatan, ima nekoliko ograničenja. Kao prvo, nije moguće izvršavati kod na GPU više od 12 sati zbog čega sam morao spremati naučene modele periodički tijekom učenja kako bih mogao nastaviti s učenjem sljedeći dan. Također, Colab ne garantira da će korisnik imati najbolju GPU na raspolaganju u svakom trenutku. Vrste GPU-a koje su dostupne u Colabu se mijenjaju s vremenom što je potrebno kako bi Colab mogao besplatno pružiti pristup tim resursima. Najčešće dostupne GPU-u su Nvidiu K80s, T4s, P4s i P100s. Korisnici koji žele

koristiti Colab gotovo bez ograničenja mogu platiti za Colab Pro koji je zasad dostupan samo u SAD-u.

OpenCV

OpenCV biblioteka (engl. *Open Source Computer Vision Library*), originalno razvijena u američkoj tvrtci Intel, je višeplatformska biblioteka otvorenog koda koja omogućuje obradu i analizu slika i videa, uključujući značajke poput detekcije lica i objekata. Uglavnom se usredotočuje na aplikacije računalnog vida u stvarnom vremenu. Nativno je napisan u C++ programskom jeziku, ali pruža sučelja i za Python, Javu i MATLAB zbog čega je široko korišten.

Detectron2

Kao podršku u projektu za sve stvari povezane s dubokim učenjem i detekcijom objekata koristio sam vanjsku biblioteku Detectron2 [22]. Detectron2 je programski sustav razvijen u FAIR-u (engl. *Facebook AI Research*) koji implementira razne algoritme visoke kvalitete za detekciju objekata, segmentaciju i panoptičku segmentaciju, uključujući Mask R-CNN i sve njegove prethodnike. Detectron2 je temeljni prijepis prijašnje verzije Detectron koji je započeo sa *maskrcnn-benchmark* projektom. Nova verzija je modularna, fleksibilna, proširiva i pruža brže učenje modela, zbog čega mnogi istraživači u svojim radovima koriste Detectron2.

Platforma je implementirana uz pomoć PyTorch biblioteke otvorenog koda za duboko učenje koja je također pronikla iz FAIR-a. Cijelo duboko učenje se temelji na provođenju izračuna nad tenzorima, koji su generalizacija matrica s više od dvije dimenzije. PyTorch omogućuje programskim inženjerima računanje s tenzorima uz snažno ubrzanje koristeći GPU-ove te pruža izgradnju dubokih neuronskih mreža na temelju dinamičkih grafova, distribuirano učenje i podršku za oblak (engl. *cloud*). U eksperimentu je korištena verzija PyTorch 1.4.

5.3. Programska implementacija

5.3.1. Priprema skupa podataka

Format podataka

Prvi korak u eksperimentu bio je ekstrahirati sve sličice iz snimke kako bi se mogle koristiti kao skup podatka prilikom učenja i evaluacije modela, za što sam koristio OpenCV biblioteku. Sve sličice su spremljene u jedan direktorij pod nazivom u formatu `frame_{frameID:05}.jpg` . Nakon toga, podijelio sam skup podataka na skup za učenje i skup za validaciju u omjeru 3:1 tako da sam nazive sličica iz podijeljenih skupova spremio u dvije tekstualne datoteke. Nadalje, da bih uopće mogao koristiti vlastiti skup podataka sa Detectronom2, bilo ga je potrebno prilagoditi formatu koji je podržan od strane Detectrona. Sličan format se koristi u COCO izazovu za detekciju objekata. Prilagođeni skup podataka je zapravo JSON (engl. *JavaScript Object Notation*) objekt - u sebi sadrži listu rječnika (engl. *dictionaries*), a svaki rječnik zasebno predstavlja jedan primjer u skupu podataka, odnosno u našem slučaju jednu sličicu iz snimke. Primjer jednog takvog rječnika prikazan je u nastavku.

```
{
  "file_name": "/dataset/images/frame_05635.jpg",
  "width": 3260,
  "height": 570,
  "image_id": 5635,
  "annotations": [
    {
      "bbox": [1141,163,1147,191],
      "bbox_mode": 0,
      "category_id": 0
    },
    ...
  ]
}
```

Slika 5.3: Primjer formata korištenog u Detectronu2

Prva četiri ključa u rječniku predstavljaju naziv, veličinu i identifikator sličice, dok ključ `annotations` sadrži listu rječnika koji predstavljaju anotacije (oznake) na promatranom sličici. Svaka oznaka sadrži tri vrijednosti. `bbox` predstavlja lokaciju graničnog okvira na slici u obliku liste s 4 broja koji se mogu prikazati na nekoliko načina, ovisno o tome koji se `bbox_mode` koristi. U ovome projektu, korišten je `BoxMode .XYXY_ABS` način gdje prva dva broja predstavljaju poziciju donje lijeve točke, a zadnja dva poziciju gornje desne točke graničnog okvira. Konačno, `category_id` pokazuje kojoj klasi pripada promatrana oznaka. Kako skup podataka sadrži tri razli-

čite klase, može poprimiti vrijednosti u rasponu [0,2]. Tako prilagođeni skup podataka mora se registrirati u razredu `DatasetCatalog`. Funkcija koja to radi prikazana je u isječku kôda 5.4.

Kôd 5.1: Funkcija za registriranje skupova podataka

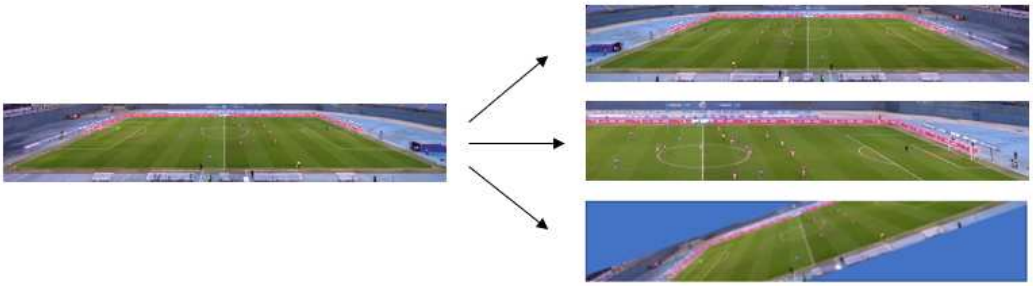
```
def register_datasets(self):
   get_dicts = lambda d = d: self.get_players_dicts(d, load=True)
   for d in ['train', 'val']:
     set_name = "players_" + d
     DatasetCatalog.register(set_name, get_dicts(d))
     MetadataCatalog.get(set_name)
       .set(thing_classes=["A", "B", "Ref"],
            thing_colors=[BLUE, RED, YELLOW])
```

Funkcija `get_players_dicts` pretvara skup podataka u format koji Detectron2 podržava. Kako bih ubrzao postupak dohvaćanja rječnika, implementirao sam mogućnost spremanja i učitavanja dobivenih rječnika u obliku JSON objekta. Rezultat se sprema u predefiniranu *.json* datoteku u strukturi projekta ako *.json* datoteku ako se argument `save` postavi na `True`. Ako vrijedi `load=True`, funkcija će učitati skup podataka iz te iste datoteke. Time je ubrzano i učenje modela jer se funkcija za dohvaćanje validacijskog skupa poziva svakih n iteracija prilikom periodičke evaluacije modela. Konačno, preko razreda `MetadataCatalog` postavljamo nazive klasa i odgovarajuću boju graničnog okvira.

Učitavanje i povećanje podataka

Sada kada imamo pripremljen skup podataka u željenom formatu, potrebno je još definirati na koji način će se podaci učitavati za potrebe učenja i evaluacije. Detectron2 pruža ugrađeni cjevovod za učitavanje podataka s dvije funkcije `build_detection_[train,test]_loader`. Navedene funkcije primaju ime registriranog skupa podataka i učitavaju listu rječnika u kanoničkom obliku. Ovakvi primjeri još nisu učitani u memoriju i nisu spremni za korištenje od strane modela. Potrebno je mapirati svaki rječnik iz liste funkcijom *mapper* koja se koristi prilikom učitavanja. Uloga preslikivača (engl. *mapper*) je transformirati kanoničku reprezentaciju skupa podataka u oblik koji model može konzumirati, uključujući čitanje slika, izvršavanje slučajnih povećanja nad podacima i pretvaranje u tenzore korištene u PyTorchu. Detectron2 nudi zadani preslikivač `DatasetMapper` koji radi sve navedene stvari. Za kraj se povratne vrijednosti iz preslikivača podijele u skupine koje se obično predaju `model.forward()` funkciji.

Ako korisnik želi druge funkcionalnosti, moguće je prilagoditi funkciju *mapper* vlastitim potrebama i predati je kao argument funkciji za učitavanje podataka. Za potrebe ovog rada, korišten je ugrađeni `DatasetMapper` s parametrima prilagođenim našem skupu. Prije nego objasnim koje su povećanja korištena u eksperimentu, bitno je shvatiti što je povećanje podataka i zašto se provodi.



Slika 5.4: Primjer povećanja podataka: horizontalno zrcaljnje, rastresanje i rotiranje

Povećanje podataka (engl. *data augmentation*) je uobičajena tehnika koja omogućuje istraživačima da značajno povećanje raznolikosti dostupnog skupa podataka bez prikupljanja novih podataka, u svrhu poboljšanja rezultata i izbjegavanja prenaučenosti modela. Postoje razne vrste povećanja slika: promjena veličine, promjena boja - *color jitter* (zasićenje, kontrast, nijanse), rotiranje, zrcaljenje, rezanje, proširivanje slika itd. Povećanja uglavnom nisu deterministička što znači da za isti ulaz mogu dati različit izlaz. Drugim riječima, povećanja u sebi sadrže stupanj nasumičnosti koji se realizira na različite načine. Na primjer, zrcaljenje slike će se izvršiti nad primjerom sa zadanom vjerojatnošću, dok se kod rotacije nasumično bira kut pod kojim će slika biti zarotirana. Na taj način se uvodi šum u podatke kako bi model mogao bolje generalizirao. U praksi se najčešće koristi horizontalno zrcaljenje, nadopuna i rezanje slike. Rezanjem pa promjenom veličine slike se postiže rastresanje podataka (engl. *data jittering*), što je jednostavno uvećanje jednog dijela slike koji se daje na ulaz mreži. Razred `DatasetMapper` koristi dva ugrađena povećanja: promjena veličine po kraćem rubu i horizontalno zrcaljenje slike, uz mogućnost rezanja slike prije navedenih povećanja, što ćemo vidjeti u sljedećem odlomku.

5.3.2. Konfiguracija modela

Ono što čini Detectron2 iznimno fleksibilnim i proširivim je mogućnost nadjačavanja metoda i proširivanja već postojećih razreda prema potrebi. Također, parametri

korišteni tijekom učenja i evaluacije mogu se postaviti kroz konfiguracijski razred `CfgNode` koji je zasnovan na principu ključ-vrijednost (engl. *key-value based*). Navedeni konfiguracijski sustav koristi *.yaml* i *.yacs* datoteke za spremanje i učitavanje konfiguracije. Prije pokretanja učenja modela, potrebno je dohvatiti konfiguraciju i težine baznog modela nad kojim će biti pokrenut postupak učenja. Detectron2 pruža različite konfiguracijske datoteke prednaučenih modela za detekciju i segmentaciju objekata, dostupnih u `model_zoo` modulu.

Budući da datoteka s lokacijama igrača u skupu podataka ne sadrže podatke o segmentaciji (lista točaka koji predstavljaju obrub objekta), nije bilo moguće koristiti običan Mask R-CNN model koji uključuje i segmentaciju objekta. U ovome radu je napravljen eksperiment prvo s Faster R-CNN model kao baznim modelom, a zatim s Mask R-CNN kojemu je isključena segmentacijska glava (maska), što se kontrolira parametrom `MODEL.MASK_ON`. Korištene su implementacije modela s FPN ekstraktorom značajki i rezidualnom neuronskom mrežom (ResNet) koja ima 101 sloj. Rezidualna neuronska mreža je vrsta mreže koja se temelji na konstrukcijama poznatim iz piramidalnih stanica u moždanoj kori, tako da se neki slojevi u mreži preskaču. Motivacija za preskakanje slojeva je izbjegavanje prenaučenosti i problema nestajućih gradijenata (engl. *vanishing gradients*). Primjer dijela konfiguracijske postavke za Mask R-CNN model prikazan je u nastavku.

Kôd 5.2: Konfiguracijska postavka

```
# Naziv konfiguracijske datoteke baznog modela
MODEL = r"COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml"
# Osnovna konfiguracija
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file(MODEL))
cfg.DATASETS.{TRAIN,TEST} = ("players_{train,test}",)
# Konfiguracija modela
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(MODEL)
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3
cfg.MODEL.MASK_ON = False
# Konfiguracija za trening
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001
cfg.SOLVER.LR_SCHEDULER_NAME = "WarmupCosineLR"
cfg.SOLVER.MAX_ITER = 15000
cfg.SOLVER.WARMUP_ITERS = 5000
```

Kao što se može vidjeti, nazivi i značenje većine konfiguracijskih parametara su jasni sami po sebi. Parametri koji se postavljaju i za učenje i za ispitivanje su prikazani u jednom redu radi jednostavnosti. Detectron2 pruža mogućnost korištenja zadane konfiguracije funkcijom `get_cfg`, dok funkcija `get_config_file` dohvaća konfiguracijske vrijednosti iz predane datoteke. Za dohvaćanje težina prednaučenog modela koristi se funkcija `get_checkpoint_url`. Za ovaj skup podataka, broj područja interesa po slici postavljen je na 128, dok je broj slika u seriji (engl. *batch*) jednak 4, što daje $128 \cdot 8 = 512$ područja interesa po jednoj seriji prilikom učenja. Broj slika u seriji se morao smanjiti za veće dimenzije jer inače ne bi stale u memoriju. Za manje dimenzije korišteno je 8 slika po seriji.

Kôd 5.3: Konfiguracija povećanja podataka

```
# Konfiguracija povecanja podataka
cfg.INPUT.MIN_SIZE_{TRAIN,TEST} = (570, )
cfg.INPUT.MAX_SIZE_{TRAIN,TEST} = 3260
cfg.INPUT.CROP.ENABLED = True
cfg.INPUT.CROP.TYPE = 'relative_range'
cfg.INPUT.CROP.SIZE = [0.5, 0.5]
```

Konfiguracija povećanja omogućava nasumično rezanje slike kod učitavanja skupa podataka. Parametar `CROP.SIZE` predstavlja relativnu veličinu izrezane slike u odnosu na originalnu sliku i može biti u rasponu $[0, 1]$. Točnije, uniformno se bira relativni udio ulazne slike koji će biti izrezan u rasponu između $[\texttt{CROP.SIZE[x]}, 1]$ gdje x predstavlja visinu ($x = 0$) ili širinu ($x = 1$). Ako je parametar postavljen na 0.4, najviše 60% visine ili širine slike će biti odrezano. Također, podešavao sam i parametre `INPUT.[MIN,MAX]_[TEST,TRAIN]_SIZE` koji definiraju dimenzije za promjenu veličine, što će biti pokazano u eksperimentalnim rezultatima.

5.3.3. Učenje i inferencija

Učenje

S definiranim modelom i učitavačem podataka (engl. *data loader*), može se pokrenuti postupak učenja. Detectron2 pruža standardiziranu apstrakciju "trenera" sa sustavom kuka (engl. *hooks*) koji omogućava dodavanje novih funkcionalnosti. Razred `SimpleTrainer` implementira jednostavan postupak učenja koristeći predane argumente: model, učitavač podataka i optimizator. Također, dostupan je i zadani `DefaultTrainer` razred koji nasljeđuje `SimpleTrainer`, a inicijalizira se iz predane `CfgNode` konfiguracije. Koristi zadanu konfiguraciju optimizatora i "ras-

poređivača" stope učenja (engl. *learning rate scheduler*), kao i ostale dodatke poput logiranja, evaluacije i spremanja modela korištenjem kontrolnih točaka (engl. *checkpoint*). Za učenje našeg modela, korištena je zadana funkcionanost `DefaultTrainer` razreda prilagođena potrebama projekta preko konfiguracijske postavke.

Osvrnimo se sada na parametre za učenje u konfiguracijskoj postavci danoj u prošlom potpoglavlju. Bazna stopa učenja je 0.001, koja je eksperimentalno dala najbolje rezultate. Kao što je već spomenuto, stopa učenja nije konstantna tijekom učenja, već se postepeno povećava prema zadanom linearnom raspoređivaču. S parametrom `WARMUP_ITERS` kontroliramo koliko iteracija će se stopa učenja povećavati, dok s parametrom `WARMUP_FACTOR`, čija je zadana vrijednost 10^{-3}, kontroliramo rast stope učenja u tom periodu. Kada dođe do bazne vrijednosti, raspoređivač će stopu učenja držati na toj vrijednosti do kraja učenja, što može biti problem jer su nam u kasnijim fazama učenja potrebni manji pomaci da bismo dobili bolje rezultate. Iz tog razloga sam eksperimentirao i s drugim raspoređivačem `WarmupCosineLR` koji nakon početnog povećanja, počinje postepeno smanjivati stopu učenja kako se približavamo maksimalnoj iteraciji. Kao optimizator kod učenja koristi se stohastički gradijentni spust s momentom postavljenim na uobičajenu vrijednost od 0.9. Tijekom učenja potrebno je pratiti validacijsku grešku periodičkom evaluacijom na ispitnom skupu kako bismo izbjegli prenaučenost modela. U našem slučaju, evaluacijski period (`TEST.EVAL_PERIOD`) postavljen je na 500, što znači da će se evaluacija pozivati svakih 500 iteracija. Također, kako bismo mogli koristiti modele koji daju najbolji rezultat prilikom evaluacije, potrebno je spremiti težine tih modela. Modeli se spremaju u obliku *.pth* datoteke, a frekvencija spremanja kontrolira se parametrom `SOLVER.CHECKPOINT_PERIOD`.

Osim navedenih prilagodbi, moguće je proširiti funkcionalnost implementacijom "kuka" što je izvedeno nasljeđivanjem razreda `HookBase`. Budući da Detectron2 nema ugrađenu funkcionalnost koja prati grešku na validacijskom skupa tijekom učenja, bilo je potrebno dodati je u implementaciju razreda `DefaultTrainer`. To je napravljeno nasljeđivanjem navedenog razreda u čijoj je implementaciji nadjačana funkcije `build_hooks`. Osim toga, dodan je i ugrađeni COCO evaluator nadjačavanjem funkcije `evaluator`. Za korištenje vlastitog evaluatora, potrebno je naslijediti razred `DatasetEvaluator` i implementirati apstraktne metode razreda. Ugrađene funkcionalnosti uz mogućnost proširivanja i konfiguracije parametara znatno olakšavaju rad s Detectronom2 i smanjuju vrijeme potrebno za postavljanje eksperimenta.

Inferencija

Konačno, sada kada imamo naučeni model možemo testirati njegovu performansu na skupu za validaciju. Prag za mjere pouzdanosti (`SCORE_THRESH_TEST`) i IoU prag za potiskivanje višestrukih odziva (`NMS_THRESH_TEST`) postavljeni su na uobičajene vrijednosti od 0.5. Za predikciju je korišteno zadano ponašanje zapakirano u razred `DefaultPredictor`. Model se kreira iz konfiguracijskog razreda, ali se postavlja u evaluacijski način, što znači da na izlazu neće dati gubitke kao kod učenja, već predikcije za predanu sliku. Slikama se prije evaluacije promjeni veličina na dimenzije postavljene u konfiguraciji. Takvi podaci se pretvaraju u tenzore i predaju modelu. Prediktor na ulazu prima jednu sliku, a na izlazu vraća listu detekcija koja sadrži sve relevantne podatke poput lokacije graničnih okvira, mjere pouzdanosti i rednog broja klase detektiranih objekata. Za vizualizaciju dobivenih podataka se koristi razred `Visualizer` čija metoda `draw_instance_predictions` vraća ulaznu sliku s nacrtanim prediktiranim okvirima.

Kôd 5.4: Inferencija

```
def evaluate(cfg, models, dataset_name):
  for m in models:
    output_dir = path.join(OUT_DIR, f'{m}_output/')
    model_file = path.join(output_dir, 'model_best.pth')
    infr_dir = path.join(output_dir, 'inference')

    model = build_model(cfg)
    DetectionCheckpointer(model).load(model_file)
    evaluator = COCOEvaluator(dataset_name, cfg, False, infr_dir)
    val_loader = build_detection_test_loader(cfg, dataset_name)
    inference_on_dataset(model, val_loader, evaluator)
```

Detectron2 nudi i funkciju `inference_on_dataset` za evaluaciju koja prima model, učitavač podataka i evaluator. U isječku kôda 5.4 prikazana je funkcija koja pokreće inferenciju nad svim predanim modelima. Navedena funkcija dohvaća težine najboljeg modela pomoću `DetectionCheckpointer` razreda. Kao evaluator je korišten COCO evaluator, a kao učitavač podataka funkcija opisana u odlomku 5.3.1. Detektirane instance su zapisane su u datoteku `coco_instances_results.json` koja sadrži sve potrebne podatke (granični okviri, mjere pouzdanosti i klase) za crtanje krivulje preciznosti i odziva. U tu svrhu sam koristio gotovi kôd [1] koji računa prosječnu preciznost sa samo jednu klasu. Za potrebe eksperimenta, morao sam prilagoditi kôd kako bi radio za više klasa.

6. Eksperimentalni rezultati

Eksperimenti su provedeni na skupom slika ekstrahiranih iz isječka snimke nogometne utakmice. Svi eksperimenti koriste isti skup za učenje i validaciju. Skup podataka je podijeljen tako da skup za treniranje sadrži prve 3/4 ukupnog skupa podataka (5637 sličica), dok skup za validaciju sadrži preostalu 1/4 (1878 sličica). Bitno je napomenuti da podaci za učenje i validaciju ne smiju imati visoku korelaciju. Kako su slike izvučene iz snimke, mnogo slika će biti vrlo slične jedna drugoj. Ako se skup podataka podijeli nasumično, velika je vjerojatnost da ćemo dobiti vrlo slične slike u oba skupa za učenje i validaciju, što vodi do pogrešno dobivene visoke točnosti modela.

Svi naučeni modeli su koristili baznu stopu učenja postavljenu na 0.001. Modeli su se učili 15000 iteracija na Nvidia Tesla T4 i Tesla P100 GPU-ovima. Nakon početnih eksperimenata sa zadanom rezolucijom slike 1333x233, naučio sam modele s većim ulaznim dimenzijama: 2048x358 i 3260x570. Iste dimenzije koristile su se i kod testiranja. Vrijeme učenja povećalo se s dimenzijom slike, što je očekivano jer su mape značajki veće i potrebno je obaviti više operacija. Dobivena vremena prikazana su u tablici 6.1. Pri završetku učenja, odabran je model koji je dao najvišu srednju prosječnu preciznost, što znači da nisu svi konačni modeli odabrani na istoj iteraciji. Modeli uglavnom imaju bolju prosječnu preciznost u ranijim fazama učenja, pa je vjerojatnije da će ti modeli imati lošiju performansu na skupu za učenje.

Rezolucija	1333x333	2048x358	3260x570
Vrijeme učenja (s/iter)	0.625	0.821	1.425
Vrijeme inferencije (s/img)	0.0524	0.0825	0.1621

Tablica 6.1: Vremena izvođenja na GPU Nvidia Tesla T4.

Prije nego što predstavim dobivene rezultate, treba napomenuti kako su rezultati prikazani za prag mjere pouzdanosti i IoU prag za potiskivanje višestrukih odziva pos-

tavljen na uobičajenu vrijednost od 0.5, ako nije navedeno drugačije. Sve mjere uspješnosti iz tablica izražene su u postotcima (%). Oznaka P je preciznost, R odziv, AP_X prosječna preciznost za klasu X, mAP srednja prosječna preciznost po klasama, a $mAP_{[50,95]}$ srednja prosječna preciznost korištena u COCO izazovu. Sve ostale mjere su iskazane za IoU prag postavljen na 0.5 koji se koristi u PASCAL VOC izazovu.

Faster R-CNN

Prvi model koji sam naučio bio je Faster R-CNN s FPN-om i ResNet mrežom od 101 sloja. Navedeni model koristi ugrađena povećanja za promjenu veličine po kraćem rubu i horizontalno zrcaljenje slike, ali ne i rastresanje podataka. Veličina je promijenjena na dimenziju 1333x233. Manja veličina omogućila je manje vrijeme izvođenja učenja i inferencije. Ovakvi modeli mogli bi se koristiti u stvarnom vremenu, ali to nije napravljeno u ovome radu. Za navedene postavke dobili smo $mAP = 53.4\%$.

Sljedeći Faster R-CNN model s kojim sam eksperimentirao ima istu konfiguraciju kao i prvi, ali uz dva ugrađena povećanja koristi i rastresanje slike s parametrima $[0.5, 0.8]$. Način na koji se slike režu objašnjen je u potpoglavlju 5.3.2. U tablici 6.2 su prikazani su rezultati evaluacije oba modela nad skupom za validaciju. Ponovno je dobivena $mAP = 53.4\%$ uz malo poboljšanje AP vrijednosti za sudca i igrače momčadi B, ali i pogoršanje detekcije igrača momčadi A.

Metoda	Rastresanje	P	R	AP_A	AP_B	AP_R	mAP_{50}	$mAP_{[50,95]}$
Faster R-CNN	-	**68.2**	68.2	**47.2**	49.6	63.3	53.4	16.5
	$[0.5, 0.8]$	61.5	**72.1**	44.0	**49.8**	**66.3**	53.4	**16.7**

Tablica 6.2: Rezultati evaluacije Faster R-CNN modela

U skupu podataka koji sadrži velike objekte nije zgodno rezati veće dijelove slike jer bi neki objekt mogao biti izrezan što bi dovelo do gubitka njegovih značajki. Budući da su objekti u našem skupu podataka manji i ima ih mnogo, rezanje prije promjene veličine slike može biti korisno jer povećava objekte i omogućuje ekstrakciju korisnijih značajki. Možemo vidjeti kako je rezanjem povećan odziv, a smanjena preciznost. Za naš skup podataka smanjena preciznost znači veći broj višestrukih detekcija istog igrača, pogotovo kada igrač nije jasno vidljiv. Primjer višestruke detekcije igrača prikazan je na slici 6.7. U ovome primjeru model je detektirao sve ostale objekte ispravno osim onih koji djelomično prekrivaju jedan drugog. Navedeni problem nije lako riješiti

jer su objekti mali i imaju nisku razlučivost.

Slika 6.1: Primjer višestruke detekcije. Model često detektira igrače više puta, pogotovo kad su igrači prekrivaju jedan drugoga. Na ovoj slici je detektirano 30 primjeraka.

Da bismo poboljšali performanse ovakvih modela, potrebno je odbaciti detekcije koje imaju manje preklapanje sa stvarnim graničnim okvirom objekta, a da pritom ne izgubimo detekcije drugih stvarnih objekata koji prekrivaju promatrani objekt. Navedeni problem je uobičajen u detekciji objekata, a obično se rješava snižavanjem IoU praga za potiskivanje višestrukih odziva (u daljnjem tekstu: NMS-prag). To znači da će sve detekcije s IoU vrijednošću iznad praga biti odbačene. IoU vrijednost se računa samo s detekcijama veće mjere pouzdanosti od promatrane detekcije.

Smanjivanjem praga odbacit ćemo neke višestruke detekcije, ali se isto tako može dogoditi da se ispravna detekcija odbaci. Ovom metodom se pokušava povećati preciznost modela, ali budući da su preciznosti i odziv obrnuto proporcionalni, odziv će se obično smanjiti. Povećanje odziva postiže se snižavanjem praga pouzdanosti. Potrebno je pronaći optimalne vrijednosti navedena dva praga koji daju najbolje rezultate uz visoku preciznost i odziv. Eksperimentalno sam došao do vrijednosti 0.4 za oba praga koji daju najbolje rezultate za ove modele. Uz navedene preinake dobivamo $mAP = 53.7\%$ za prvi model, a $mAP = 53.8\%$ za drugi uz poboljšanje preciznosti za 1.0%, odnosno 2.6%. Odziv se čak povećao u prvom slučaju za 0.2%, a u drugom slučaju je pao za 3.7%. Budući da su veličine slike smanjene prije ulaska u mrežu i Faster R-CNN modeli koriste manje precizan sloj za sažimanje područja od interesa (*RoIPool*), nije čudno što ovi modeli daju slabije rezultate.

Mask R-CNN

Sljedeća skupina modela s kojima sam eksperimentirao su Mask R-CNN koji također koriste ResNet-101 i FPN. Mask R-CNN modeli znaju raditi bolje od Faster R-CNN-a zbog usklađivanja područja od interesa (*RoIAlign*) koje omogućuje precizniju detekciju na razini piksela i ojačava dijeljene značajke. Rezultati evaluacije svih relevantnih modela prikazani su u tablici 6.3.

Metoda	Rastresanje	P	R	AP_A	AP_B	AP_R	mAP_{50}	$mAP_{[50,95]}$
Mask R-CNN	-	63.3	72.1	47.8	56.8	**68.3**	57.6	**21.2**
	$[0.8, 0.8]$	**65.8**	71.4	46.2	**59.6**	66.0	57.3	20.6
	$[0.3, 0.3]$	52.9	**76.7**	**52.8**	58.2	66.5	**59.2**	21.0

Tablica 6.3: Rezultati evaluacije Mask R-CNN modela

Može se primijetiti da svaki Mask R-CNN model daje višu srednju prosječnu preciznost u usporedbi s Faster R-CNN modelima, što je bilo i za očekivati. U skupu podataka u kojemu su objekti mali i često loše vidljivi, preciznost na razini piksela je vrlo bitna. Od modela koji nisu prikazani u tablici, naučio sam Mask R-CNN model sa ResNet-50 mrežom koji ima visok odziv, ali mu je preciznost jako loša - detektirao je 68797 objekta, od 43194 mogućih u validacijskom skupu, s čak 19.2 pogrešnih detekcija (FP) po slici. Slične rezultate dobio sam učenjem modela koji koriste rezanje s parametrima $[0.5, 0.7]$ i $[0.7, 0.7]$. Prvi model daje $mAP = 56.1\%$, a drugi $mAP = 55.4\%$, ali imaju više od 15 pogrešnih detekcija po slici. Oba učenja su dala model s najvišom prosječnom preciznošću u ranijim fazama što, uz lošu rezoluciju ulazne slike, može biti jedan od razloga zbog kojeg imaju nisku preciznost. Zbog svega navedenog, ovi modeli nisu uzeti u obzir i prikazani u tablici.

Model koji ne koristi rastresanje podataka dao je dosta dobre rezultate ako uzmemo u obzir veličinu slika. Unatoč niskoj rezoluciji, model je uspio naučiti korisne značajke i dati $mAP = 57.6\%$. Štoviše, uz model sa slabim rastresanjem ($[0.8, 0.8]$) dao je najbolji omjer preciznosti i odziva. Budući da su slike na ulazu male, agresivnije rastresanje slike s parametrima $[0.3, 0.3]$ je dalo bolje rezultate uz relativno nisku preciznost. Ovaj model ima jako dobar odziv, ali problem i dalje predstavljaju višestruke detekcije igrača zbog čega nije upotrebljiv za ozbiljnije korištenje. Bolje rezultate od navedenih možemo dobiti povećavanjem dimenzija ulaznih slika.

Mask R-CNN s povećanim dimenzijama

Budući da su igrači su na originalnoj veličini slike široki maksimalno 8 piksela, na smanjenim slikama bit će široki maksimalno 2-3 piksela što jako otežava detekciju, pogotovo u problematičnim situacijama. Da bi mreža dobila više značajki od igrača potrebno joj proširiti dimenzije ulaza. Veća ulazna dimenzija slike provlačenjem kroz sljedove konvolucijskih slojeva daje na izlazu veću mapu značajki iz koje je moguće pročitati više podataka. Iz tog razloga, povećao sam dimenzije ulaznih slika Mask R-CNN korištenih tijekom učenje i evaluaciju. Koristio sam parametar rastresanja postavljen na 0.5 jer su slike veće, pa nije bilo potrebno agresivnije rastresanje. Dobiveni rezultati su prikazani u tablici 6.4.

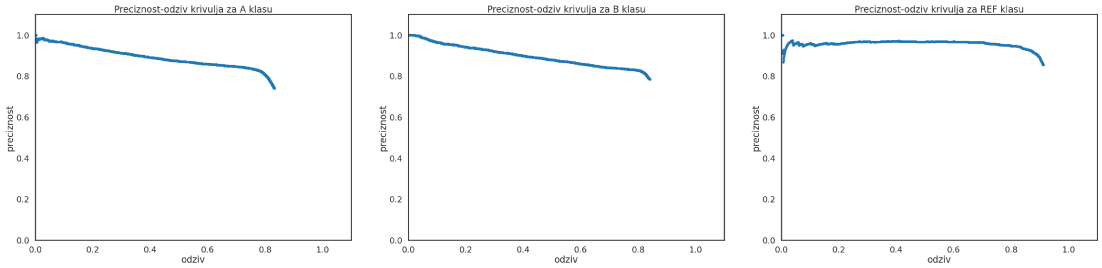
Rezolucija	Rastresanje	P	R	AP_A	AP_B	AP_R	mAP_{50}	$mAP_{[50,95]}$
1333x233	$[0.3, 0.3]$	52.9	76.7	52.8	58.2	66.5	59.2	21.0
2048x358	$[0.5, 0.5]$	66.2	82.1	62.4	65.2	77.2	68.3	24.3
3260x570	$[0.5, 0.5]$	**76.6**	**84.0**	**68.5**	**70.0**	**82.8**	**73.8**	**27.1**

Tablica 6.4: Rezultati evaluacije Mask R-CNN modela za različite ulazne dimenzije

Možemo vidjeti kako su se povećanjem dimenzija slike rezultati znatno poboljšali. Konačni model koji koristi veličinu originalne slike (3260x570) daje $mAP = 73.8\%$ što je zadovoljavajuć rezultat. Ovaj model je uspio ekstrahirati korisnije značajke koje su mu omogućile bolje detekciju u problematičnim situacijama. Značajno je smanjen broj višestrukih detekcija što je rezultiralo povećanjem preciznosti. Također, povećao se i odziv te model sad puno bolje pronalazi preklapajuće igrače, kao i igrače koji su stopljeni s pozadinom. Budući da vratari nemaju istu boju dresa kao i momčad, zna se dogoditi da vratar bude pogrešno nepronađen (FN) što bi se moglo riješiti stavljanjem vratara u zasebnu klasu.

Slika 6.2: Rezultat detekcije igrača. Model puno bolje detektira igrače u problematičnim situacijama, uz manji broj višestrukih detekcija istog igrača (na ovoj slici samo jedna).

Model daje 19.3 ispravnih detekcija (TP) i 5.9 neispravnih detekcija (FP) po slici. Preciznost na skupu za validaciju iznosi $P = 76.6\%$, a odziv $R = 84.0\%$. Pokrenuo sam evaluaciju i na skupu za učenje gdje su rezultati očekivano bili bolji - preciznost na skupu za učenje je $P = 83.4\%$, odziv $R = 89.6\%$, dok je $mAP = 80.8\%$. Model koji daje najveću prosječnu preciznost na skupu za validaciju je pronađen nakon 5000 iteracija, što je razlog zbog kojeg rezultati na skupu za učenje nisu i bolji. Krivulje preciznosti i odziva za svaku klasu dobivene na skupu za validaciju prikazane su na slici 6.6. Na slici 6.7 su prikazane neke pozitivne i negativne situacije kod predikcije.



Momčad A: $AP = 68.5\%$ Momčad B: $AP = 70.0\%$ Sudac: $AP = 82.8\%$

Slika 6.6: Krivulje preciznosti i odziva. Prosječna preciznost sudca je najviša od svih klasa jer je sudac samo jedan, pa je skup za ispitivanje manji od igrača momčadi. Prosječne preciznost igrača obje momčadi su podjednake. Problemi koji smanjuju prosječnu preciznost momčadi su uglavnom višestruke detekcije, pogrešne detekcije vratara i ponekad preklapanje igrača.

Slika 6.7: Primjeri ispravnih i pogrešnih detekcija u problematičnim situacijama. Gornje tri slike prikazuju slučajeva kada je model uspješno detektirao preklapajuće igrače i igrače koji se stapaju s pozadinom. Donje tri slike prikazuju redom: pogrešnu detekciju sudca (FP), pogrešnu ne-detekciju vratara (FN) i višestruku detekciju istog igrača (FP).

7. Zaključak

Ovaj rad se bavi suvremenim metodama za detekciju objekata zasnovanim na predlaganju područja. Počevši od prvog R-CNN modela, inkrementalnim poboljšanjima došli smo do vrlo snažnog modela Mask R-CNN koji može istovremeno i detektirati i segmentirati objekt. Rad Mask R-CNN modela poboljšavao se i nadograđivao kroz godine. Tako danas postoji programski sustav Detectron2 koji nudi razne algoritme dubokog učenja u domeni detekcije objekata. Detectron2 je iznimno koristan alat koji istraživačima omogućuje lakše učenje modela i eksperimentiranje sa podacima.

U radu su napravljeni eksperimenti s Faster i Mask R-CNN modelima nad skupom podataka izvađenim iz snimke nogometne utakmice. Budući da su objekti u skupu podataka su mali i niske razlučivosti, za ekstrakciju značajki korištena je mreža piramidalnih značajki (FPN) koja pospješuje detekciju malih objekata. U usporedbi dva modela, Mask R-CNN je očekivano dao bolje rezultate od Faster R-CNN zbog usklađivanje područja od interesa koje je omogućilo precizniju detekciju na razini piksela što je jako bitno kada su objekti sitni. Povećanjem dimenzija ulaznih slika značajno se poboljšala performansa modela, ali se i usporilo učenje i predikcija. Pokazalo se kako je jače rastresanje podataka za manje dimenzije slika dalo bolje rezultate, dok je za veće dimenzije bolju performansu davalo slabo do umjereno rastresanje. Krajnji model koji je naučen nad slikama visoke rezolucije (3260x570) vrlo dobro prolazi igrače ($mAP = 73.8\%$ na validacijskom skupu), ali ih nekad zna višestruko detektirati.

Daljnjim povećanjem dimenzija ulaznih slika možda bismo dobiti bolje rezultate, ali bi se brzina smanjila. Također, bolji rezultati bi se mogli dobiti nastavkom eksperimentiranja s različitim parametrima povećanja podataka. U ovome radu nije napravljena detekcija u stvarnom vremenu, ali bi daljnje nadogradnje mogle uključivati navedeni koncept. Također, mogla bi se ubaciti i detekcija lopte, što bi predstavljalo veći problem, pogotovo u snimkama loše rezolucije gdje je lopta često loše vidljiva. U snimkama veće rezolucije mogli bismo s višom preciznošću detektirati i igrače i loptu. Uz veći skup podataka, fino ugođen Mask R-CNN model mogao bi se koristiti kao oslonac za automatsku analizu nogometnih utakmica.

LITERATURA

[1] Timothy Arlen. Calculate mean AP (mAP). URL https://gist.github.com/tarlen5/008809c3decf19313de216b9208f3734.

[2] Leonardo Araujo dos Santos. Object Localization and Detection. URL https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html.

[3] Ross Girshick. Fast R-CNN. 2015. URL https://arxiv.org/pdf/1504.08083.pdf.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, i Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014. URL https://arxiv.org/pdf/1311.2524.pdf.

[5] Tomasz Grel. Region of interest pooling explained. 2017. URL https://blog.deepsense.ai/region-of-interest-pooling-explained/.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, i Ross Girshick. Mask R-CNN. 2018. URL https://arxiv.org/pdf/1703.06870.pdf.

[7] Jonathan Hui. mAP (mean Average Precision) for Object Detection. 2018. URL https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

[8] Lars Hulstaert. A Beginner's Guide to Object Detection. 2018. URL https://www.datacamp.com/community/tutorials/object-detection-guide.

[9] Andrej Karpathy. Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. URL http://cs231n.github.io/.

[10] Josip Krapac, Marko Čupić, i Siniša Šegvić. *Duboko učenje - nastavni materijali*. 2018.

[11] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, i Serge Belongie. Feature pyramid networks for object detection. U *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[12] Jonathan Long, Evan Shelhamer, i Trevor Darrell. Fully convolutional networks for semantic segmentation. U *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. URL https://arxiv.org/pdf/1411.4038.pdf.

[13] Damjan Miko. Duboki konvolucijski modeli za lokalizaciju objekata. http://www.zemris.fer.hr/~ssegvic/project/pubs/miko18ms.pdf, 2018.

[14] Jan Šnajder i Bojana Dalbelo Bašić. *Strojno učenje - skripta*. 2014.

[15] Arthur Ouaknine. Review of Deep Learning Algorithms for Object Detection. 2018. URL https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852.

[16] Dhruv Parthasarathy. A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN. 2017. URL https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, i Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. U *Advances in Neural Information Processing Systems (NIPS)*, 2015. URL https://arxiv.org/pdf/1506.01497.pdf.

[18] Javier Rey. Faster R-CNN: Down the rabbit hole of modern object detection. 2018. URL https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/.

[19] Marko Čupić, Bojana Dalbelo Bašić, i Marin Golub. *Neizrazito, evolucijsko i neuroračunarstvo*. 2013.

[20] Rodrigo Verschae i Javier Ruiz-del Solar. Object Detection: Current and Future Directions. 2015. URL https://www.frontiersin.org/articles/10.3389/frobt.2015.00029/full.

[21] Lilian Weng. Object Recognition for Dummies. 2017. URL https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html#r-cnn.

[22] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, i Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

Fino ugađanje konvolucijskih modela za lokalizaciju objekata

Sažetak

U okviru ovog rada razmatrane su metode za detekciju objekata zasnovane na predlaganju područja. Na početku rada ukratko se prolazi kroz osnovne koncepte korištene u konvolucijskih mrežama. Zatim su detaljnije opisane četiri metode za detekciju objekta zasnovanih na predlaganju područja - R-CNN modeli. Svaki od modela dao je bolje rezultate od svojih prethodnika u pogledu brzine i preciznosti. Nadalje, objašnjene su osnovne mjere korištene kod evaluacije modela za detekciju objekata. Provedeni su eksperimenti nad vlastitim skupom podataka koji koriste Faster i Mask R-CNN kao bazne modele. Za skup podataka korištene su sličice iz snimke nogometne utakmice gdje su objekti igrači obje momčadi i sudac. Za kraj, modeli su evaluirani osnovnim mjerama uspješnosti, prikazana su poboljšanja i krajnji rezultati detekcije.

Ključne riječi: duboko učenje, detekcija objekata, konvolucijske neuronske mreže, Faster R-CNN, Mask R-CNN

Fine-tuning convolutional models for object localization

Abstract

This paper deals with object detection methods based on region proposal. At the start of the paper, we briefly go through the basic concepts used in convolutional networks. Then we describe four object detection methods in more detail known as R-CNN family. Each of the models gave better results than its predecessors in terms of speed and precision. Furthermore, we explain basic metrics used in object detection models. Experiments were performed using Faster and Mask R-CNN as base models over our own dataset. Extracted frames from a football match footage are used as dataset in which objects for detection are players of both teams and the referee. Finally, we evaluate models using basic metrics and present its improvements and final detection results.

Keywords: deep learning, object detection, convolutional neural networks, Faster R-CNN, Mask R-CNN