

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 000

Brojanje ljudi u sceni

Luka Novak

Zagreb, lipanj 2018.

Umjesto ove stranice umetnite izvornik Vašeg rada.

Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.

SADRŽAJ

1. Uvod	1
2. Računalni vid	2
2.1. Računalni vid	2
2.2. Kategorije računalnog vida	3
3. Detekcija objekata prije konvolucijskih modela	8
3.1. Viola-Jones algoritam	9
4. Detekcija objekata konvolucijskim modelima	14
4.1. Neuronske mreže	14
4.1.1. Neuron	14
4.1.2. Svojstva umjetne neuronske mreže	16
4.1.3. Aktivacijske funkcije	17
4.1.4. Backpropagation algoritam	21
4.2. Konvolucijske neuronske mreže	22
5. Skupovi podataka	27
5.1. Pascal VOC2007	27
5.2. COCO	28
5.3. Video	28
6. Implementacija	30
6.1. Korišteni alati	30
6.2. Viola-Jones algoritam	30
6.3. YOLO	31
6.3.1. Dizajn i svojstva mreže	33
6.3.2. Podijela slike	34
6.3.3. Funkcija gubitka	35

6.3.4. Anchor boxes	38
6.3.5. Backbone	39
6.3.6. Trening	39
6.3.7. Predviđanje	40
7. Zaključak	41
Literatura	42

1. Uvod

Kada se govori o biologiji i medicini, dvije grane koje se bave proučavanjem života i načina funkciranja živih bića, svi se slažu da su izuzetno puno napredovale sa razvojem tehnologije. Ljudi razvijaju sve složenije i sve preciznije alate, strojeve i programe za proučavanje svih sustava u tijelu, od stanice do skupa organa koji rade zajedno, kao jedan, te skupa čine savršeno ugođeni sustav. No u jednom području tih znanosti ne napreduje se tom brzinom kao u ostalim područjima. O tom području i sustavu organa ne zna niti približno mnogo kao o ostalim organizma i sustavima u tijelu. Znanstvenici govore da je to najsloženiji sustav, najjače super-računalo na svijetu. Centar tog sustava je mozak, organ koji može probaviti nevjerojatnu količinu informacija u stvarnom vremenu sa gotovo savršenom prepoznavanjem.

Od početka "modernog doba čovječanstva", javlja se ideja umjetne inteligencije. Znanstvenici iz područja tehničkih znanosti i računarstva teže razvijanju nečeg takvog i prirodno se nameće ljudski mozak kao ideal, razina inteligencije koju se želi dostići. Pretpostavlja se da je najveći dio mozga, oko 2/3, posvećen vidu (Keller et al., 2012). Iz tog razloga, veliki dio proučavanja i razvoja moderne umjetne inteligencije fokusirano je na vid. U nastavku rada dan je pregled najvažnijih algoritama i arhitektura mreža.

2. Računalni vid

2.1. Računalni vid

Računalni vid je veliko područje računarske znanosti koje se bavi algoritmima i metodama vezanim za obradu slike. Kao što je navedeno u uvodu, vid je osjetilo preko kojega ljudi primaju najviše informacija, više nego preko bilo kojeg drugog osjetila. Da bi bilo moguće napraviti stroj ili program koji vidi kao mi, prvo je potrebno shvatiti kako ljudi vide i kako obrađuju informacije primljene preko očiju. Samo shvaćanje kako funkcioniра ljudski vid je izuzetno težak zadatak. Danas donekle razumijemo taj proces, no ima još puno stvari koje su ostale neodgovorene. Recimo, istraživanja pokazuju da ljudi mogu prepoznati što je na slici za samo 13 milisekundi(RSV). Ljudi i dalje ne znaju kako mozak to radi. Postoje nagađanja, no ništa nije dokazano i upitno je hoće li ikada ti koncepti biti do kraja shvaćeni i objasnjeni. Zbog složenosti ovog problema, nemoguće je naći riješenje klasičnim pristupom i napisati program eksplicitno. Zato se u ovakvim problemima pribjegava umjetnoj inteligenciji. Postoji nekoliko dijelova cijelog sustava vida koji ljudima omogućavaju vid:

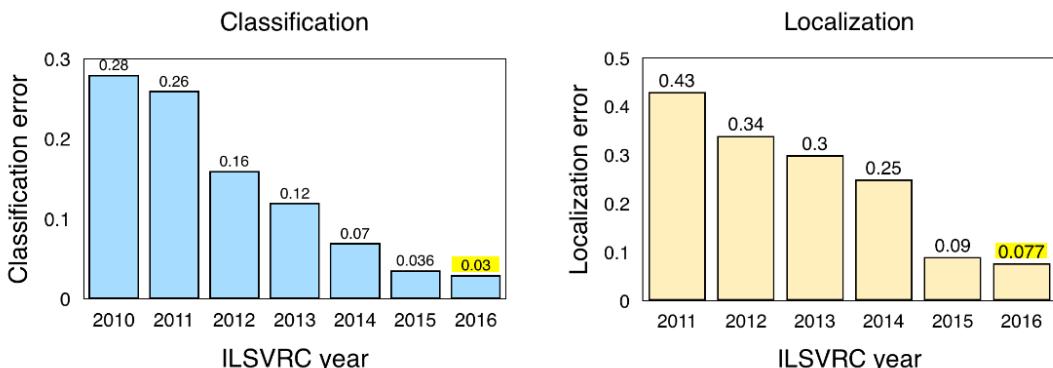
- **Vidjeti** - dati računalu vid znači dati mu oči. U tom području znanost je dosta napredovala tako da danas postoje kamere sa većom rezolucijom od ljudskog oka. Ovaj dio vida odvija se, naravno, na samom oku.
- **Prepoznati** - moći prepoznati različite objekte, maknuti šum sa slike itd. Ukratko, vidjeti što se na slici nalazi. Ovaj aspekt vida odvija se unutar oka i na putu do mozga gdje mozak na kraju posloži sve te informacije u smislenu sliku.
- **Razumijeti** - shvatiti što se na slici nalazi, shvatiti odnose među objektima, propoznati različite predmete iz različitih kuteva... Ljudski mozak sposoban je prepoznati jabuku bila ona crvena, žuta ili zelena. Sposoban je prepoznati jabuku i ako je napola pojedena. Sposoban je prepoznati jabuku neovisno o njenoj orijentaciji, neovisno o tome miruje li ili se kreće. Ovaj dio vida odvija se potpuno u mozgu. Ovdje je trenutno dosegnuta granica računalnog vida i

umjetne inteligencije. No, to ne znači da je nemoguće ići dalje, nego samo da je znanost savladala dosadašnje probleme i trenutno se bavi ovim dok ne nađe riješenje.

2.2. Kategorije računalnog vida

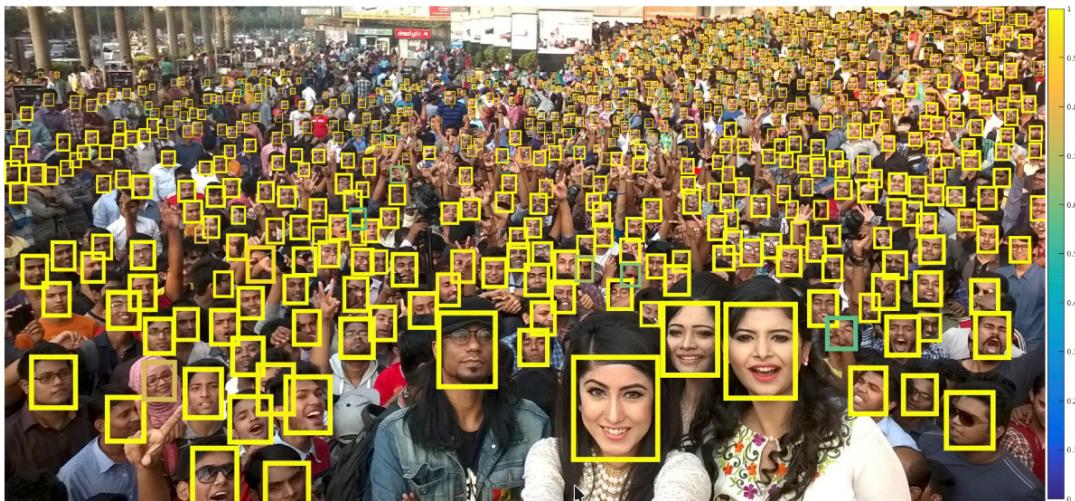
Računalni vid podjeljen je u nekoliko kategorija:

- Klasifikacija i Lokalizacija.** Ovo područje računalnog vida bavi se prepoznavanjem objekata na slici. To znači da algoritam odlučuje kojem razredu (*engl. class*) pripada cijela slika. Pretpostavljajući navedeno, lokalizacija je proces pronaletaženja pozicije objekta na slici. Rezultat procesa lokalizacije obično je crtanje okvira (*engl. bounding box*) oko objekta na slici. Pobjednici natjecanja "Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)" u kategoriji klasifikacije pobjedili su sa rezultatom od 2.251% pogreške, dok su pobjednici u kategoriji lokalizacije postigli rezultat sa greškom od 6.226%. Sa slike 2.1 vidi se da su se rezultati u odnosu na 2016. godinu popravili za 0.8% u kategoriji klasifikacije te 1.5% u kategoriji lokalizacije .



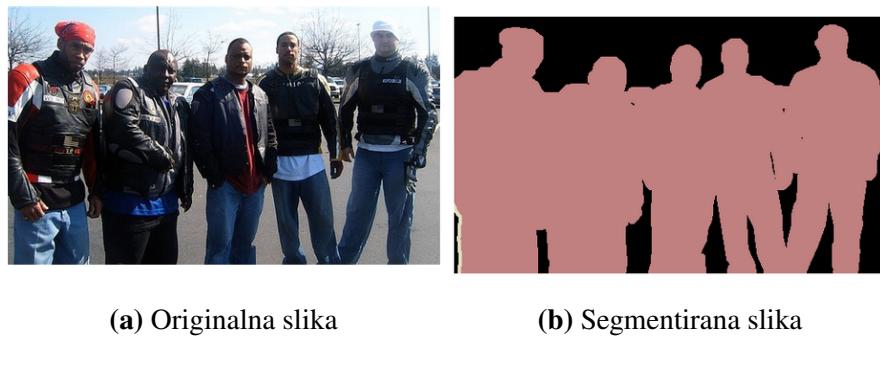
Slika 2.1: Rezultati natjecanja ILSVRC tokom godina (Lab, 2018)

- Detekcija.** Pod pojmom detekcija podrazumijeva se pronaletaženje objekata na slici, odnosno, određivanje prisutnosti objekata na slici. Proces detekcije objekata je malo složeniji od klasifikacije/lokalizacije objekata zbog toga što se ovdje mora provesti više klasifikacija i lokalizacija na jednoj slici te objekti mogu biti iz više domena. Pri klasifikaciji proveravamo pripada li objekt nekom razredu objekata i dobijamo binarni odgovor (*pripada ili ne pripada* dani objekt danom razredu), dok je pri detekciji potrebno provesti taj postupak za sve moguće razrede objekata.



Slika 2.2: Rezultati algoritma za detekciju lica iz rada (Peiyun Hu, 2017)

3. **Praćenje.** Ovaj pojam odnosi se na praćenje određenog objekta ili više njih u sceni. Ova kategorija se, logično, primjenjuje samo u području videa i izuzetno je bitna za sustave koji riješavaju zadatke poput autonomne vožnje.
4. **Segmentacija.** Pojam segmentacije odnosi se na proces grupiranja piksela u skupine s kojima se kasnije može raditi niz drugih procesa, npr. klasifikacija itd.



Slika 2.3: Segmentacija slike

Source: NVIDIA DevBlog

Nadalje, semantička segmenatacija, kako i samo ime govori, pokušava odrediti semantičku ulogu svakog piksela u slici. Dakle, možemo reći da pokušava klasificirati svaki piksel u njegovu semantičku cjelinu.

Instancijska segmentacija (*engl. Instance segmentation*) ide i korak dalje. Ona pokušava segmentirati različite objekte istog razreda objekata. Primjerice, ako se na slici nalaze tri mobitela, svaki mobitelj bit će označen drugom bojom.



Slika 2.4: Semantička segmentacija slike

Source: NVIDIA DevBlog



(a) Originalna slika

(b) Instancijski segmentirana slika

Slika 2.5: Instancijska segmentacija slike

Source: NVIDIA DevBlog

- Ostale, manje zastupljene, kategorije računalnog vida : Super-resolution, Style Transfer, Colourisation



(a) Originalna slika

(b) SRGAN

Slika 2.6: Slika poboljšana metodom Super-resolution GAN (et al., 2016)



Slika 2.7: Prva slika prikazuje originalnu *Mona Lisu*, druga slika prikazuje sliku naslikanu u stilu kubizma/ekspresionizma, treća slika prikazuje stil druge slike prenesen na prvu sliku



(a) Originalna slika

(b) Colourisation

Slika 2.8: Slika poboljšana metodom colourisation

U okviru ovoga rada pažnja je stavljena isključivo na **detekciju objekata**.

3. Detekcija objekata prije konvolucijskih modela

Povijest prepoznavanja (detekcije) objekata seže puno dalje od konvolucijskih neuronskih mreža. Prvi sustavi detekcije objekata problemu su prilazili geometrijski. Neki od takvih pristupa bili su:

- **poravnanje** gdje se pokušavalo pronaći takvu transformaciju slike koja bi minimizirala pogrešku:

$$\sum_i \text{residual}(T(x_i), x'_i)$$

Ovaj period trajao je od ranih 60ih do početka 90ih godina prošlog stoljeća. Uglavnom se pokušavalo riješiti problem detekcije razlamanjem objekata na slici u manje komponente (*engl. Block world*, L. G. Roberts Machine Perception of Three Dimensional Solids, 1963.).

- **modeli temeljeni na izgledu** (*engl. appearance - based models*) su modeli temeljeni na svojstvenim vrijednostima (Eigenfaces for Recognition, Turk and Pentland, 1991.) te histogramima boja (Swain and Ballard, IJCV 1991.)
- **klizeći prozor** pristup koji se donekle i danas primjenjuje. Ovaj pristup smatra se početkom "modernog doba" računalnogvida, iako se klizeći prozor koristi u najjednostavnijim mogućim oblicima. Ovaj period počinje ranih 90ih godina, a predvode ih Turk and Pentland svojim radom "Eigenfaces for Recognition" izdanim 1991. godine, a vrhunac se potiče 2001. godine objavljanjem algoritma koji su razvili Viola i Jones.
- **lokalne značajke** (*engl. local features*) gdje su oblici predmeta koje je potrebno detektirati djelomično poznati (D. Lowe, 2004.). U tom periodu Google razvija i prvi pretraživač slika.
- **parts-and-shape modeli** koji koriste kombinaciju više vrsta značajki: ponovo razlamaju objekte na dijelove, relativne odnose između tih dijelova te samu

prisutnost dijelova objekta (Weber, Welling and Perona, 2000.).

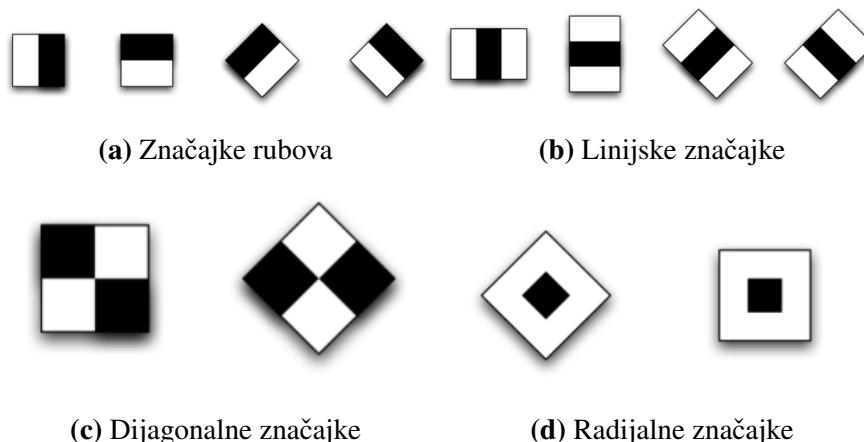
- **bags-of-features modeli** uvođe prepoznavanje tekstura. Klasični bag-of-features modeli imaju standardizirane korake (*engl. pipeline*):

1. Feature extraction
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"

3.1. Viola-Jones algoritam

Prije velikog razvoja konvolucijskih neuronskih mreža, Paul Viola i Michael Jones su 2001. godine u svom radu (Vi) objavili svoj algoritam koji je tada bio prvi algoritam koji je omogućavao detekciju lica u stvarnom vremenu. Iako se može trenirati i na drugim domenama, algoritam je najpoznatiji po detekciji lica pa će na toj domi ovdje biti i opisan.

Naravno, algoritam treba puno pozitivnih (slike lica) i negativnih primjera (slike bez lica). Iz tih slika potrebno je izvući značajke prema kojima će kasnije biti moguće odrediti (detektirati) lice. Za ekstraciju značajki koriste se Haarove značajke.



Slika 3.1: Haarove značajke

Haarove značajke računaju se na temelju razlike u intenzitetu piksela. Primjerice, ako su vrijednosti piksela u rasponu od 0 (bijeli piksel) do 1 (crni piksel), svi pikseli koji imaju vrijednost veću do 0.5 smatraju se tamnim, a svi pikseli koji imaju vrijednost manju od 0.5 smatraju se svijetlima. Na slici 3.1 prikazane su 4 kategorije Haarovih

značajki. Dakle, kako bi bilo moguće odrediti gdje se na slici nalazi (i ako se nalazi) neka od značajki, slika se dijeli na manja područja te se na svakom od tih područja traže Haarove značajke. Naravno, značajke se skaliraju i rotiraju kako bi se pronašli svi dijelovi slike koji zadovoljavaju uvjete značajki. Zatim se odredi gledano područje, kao što to prikazuje slika 3.2, te se računaju vrijednosti za Haarove značajke po formuli:

$$\Delta = dark - white = \frac{1}{n} \sum_{x=1}^n I_{dark}(x) - \frac{1}{n} \sum_{x=1}^n I_{light}(x)$$

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

(a) Idealne vrijednosti piksela

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

(b) Stvarne vrijednosti piksela

Slika 3.2: Izračun za linijske Haarove značajke

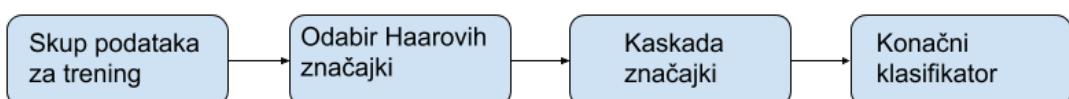
Slika 3.1a prikazuje Haarove značajke rubova. Neke od primjena tih značajki su detekcija obrva, gornje/donje usne i zubi. Slika 3.1b prikazuje linijske Haarove značajke koje služe za detekciju nosa, usta, očiju itd. Nadalje, slika 3.1c prikazuje dijagonalne Haarove značajke koje se koriste u detekciji složenijih karakteristika. Recimo, ako se lice na slici smije, rubovi usana se uvuku u obrazu te su tada slabije osvjetljeni. Tako zajedno sa okom na istoj strani lica tvore tamnije zone, dok obraz i nos čine svijetlijia područja. Zadnja skupina Haarovih značajki su radikalne Haarove značajke koje prikazuje slika 3.1d. Te se značajke koriste u detekciji zjenica, rubova usana i sl. Dakle, Haarove značajke mogu se shvatiti kao preteća konvolucijskih jezgri. Slika 3.3 prikazuje neke od haarovih značajki detektiranih na slici.

Ovaj proces je dosta spor i neefikasan jer se na ovakav način dođe do više od 160000 različitih značajki. Očito je da, ovisno o domeni, neke od značajki odgovaraju više od drugih, no javlja se pitanje kako odabrati najbolje značajke? Odgovor na ovo pitanje je Adaboost algoritam. Adaboost (skraćeno od *engl. Adaptive Boosting*) je algoritam osmišljen 2003. godine koji kombinira izlaze iz ostalih, "slabijih" klasifika-



Slika 3.3: Neke od Haarovih značajki pronađenih na slici

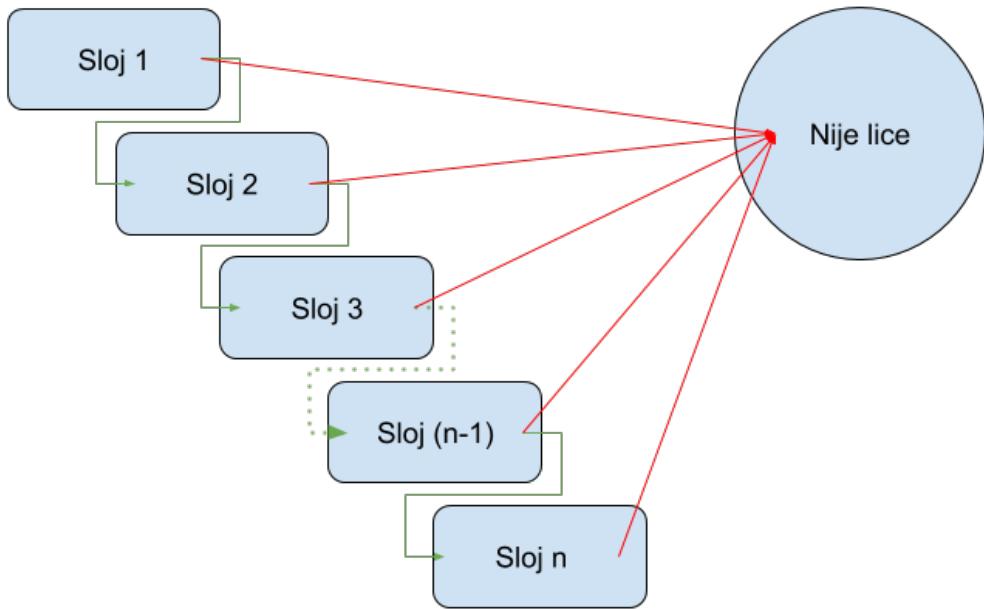
tora, te njihovom težinskom sumom dolazi do optimuma algoritma. Adaboost zapravo uči i optimizira težine za svaki od klasifikatora.



Slika 3.4: Viola-Jones algoritam

Autori su došli do zaključka da je čak sa samo 200 značajki moguće postići točnost pronalaženja lica od 95%. Konačne postavke algoritma iz rada (Vio) sadržavale su oko 6000 značajki. Kako bi optimizirali ovaj složeni proces i izbjegli primjenu svih 6000 značajki na svaku sliku, Viola i Jones odlučili su napraviti strukturu kaskada značajki u kojoj svaka iduća razina ima sve složenije Haarove značajke. Kao što je označeno na slici 3.5, svaka razina ima mogućnost odbaciti sliku ukoliko ne nađe značajke svoje razine na slici. Tako se dijelovi slike koji ne prikazuju lice odbacuju te se oslobađaju računalni resursi za procesiranje slika na kojima se nalazi lice. Autori navode da je

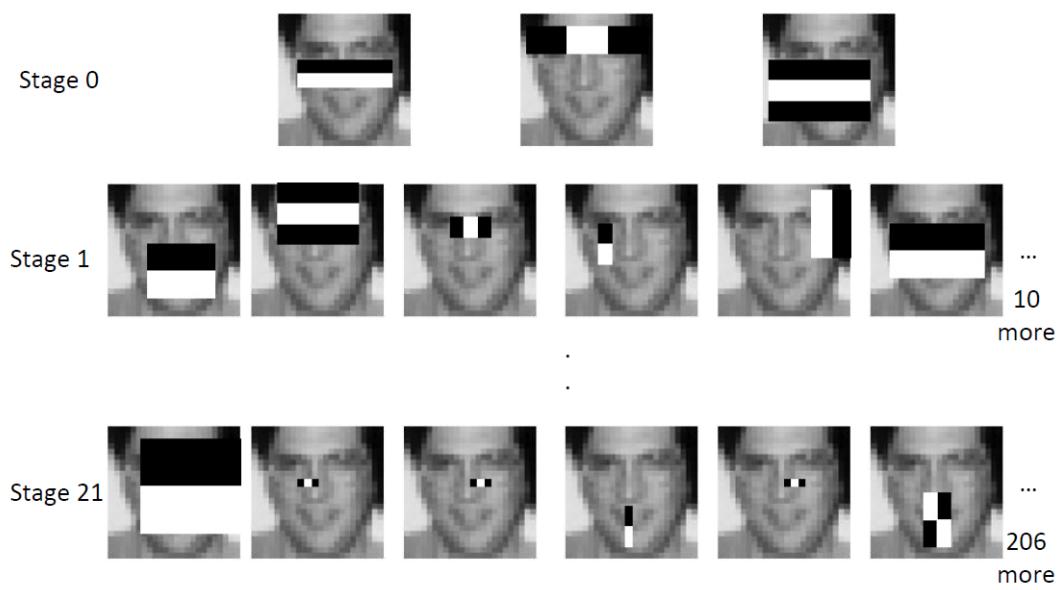
projek značajki evaluiranih pos lici oko 10. Dakle, sa preko



Slika 3.5: Kaskada Haarovih značajki

Rezultati ovih slabijih klasifikatora kombiniraju se u konačni klasifikator koji odlučuje nalazi li se lice na slici.

Viola-Jones algoritam je svojedobno bio vrlo popularan algoritam jer nudi vrlo pouzdane rezultate, obradu u stvarnom vremenu i invarijantnost na skaliranje. No, isto tako, ima i nekoliko nedostataka kao što su netolerantnost na rotaciju, osjetljivost na osvjetljenje objekata na slici i sl. Činjenica da se još i danas koristi govori dovoljno o kvaliteti i važnosti ovog algoritma.



Slika 3.6: Dijelovi Haarovih značajki pronađeni na različitim razinama kaskada

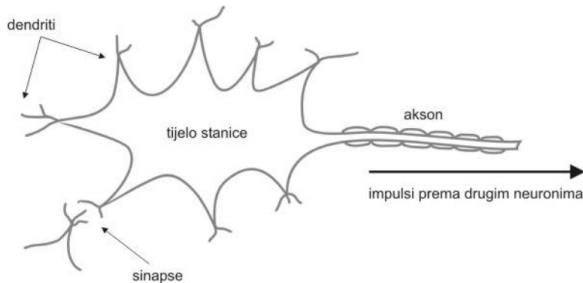
4. Detekcija objekata konvolucijskim modelima

4.1. Neuronske mreže

4.1.1. Neuron

Neuronske mreže nastale su kao rezultat pokušaja reprodukcije rada ljudskog mozga.

Osnovna gradivna jedinica ljudskog mozga jest neuron. Ljudski mozak sastavljen je od oko 10^{11} neurona kojih ima više od 100 vrsta i koji su shodno svojoj funkciji raspoređeni prema točno definiranom rasporedu. Svaki je neuron u prosjeku povezan s 10^4 drugih neurona. Četiri su osnovna dijela neurona: tijelo stanice (soma), skup dendrita (ogranaka), aksona (dugačke cijevčice koje prenose električke poruke) i niza završnih članaka



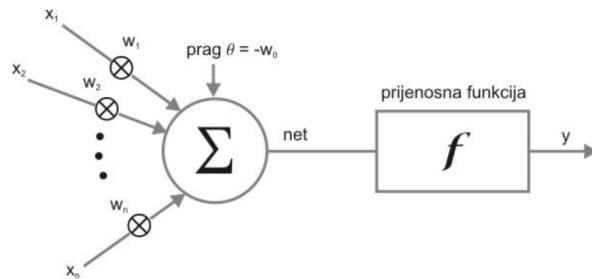
Slika 4.1: Građa neurona

Tijelo stanice sadrži informaciju predstavljenu električkim potencijalom između unutrašnjeg i vanjskog dijela stanice (oko -70 mV u neutralnom stanju). Na sinapsama, spojnom sredstvu dvaju neurona kojim su pokriveni dendriti, primaju se informacije od drugih neurona u vidu post-sinaptičkog potencijala koji utječe na potencijal stanice povećavajući (hiperpolarizacija) ili smanjivajući ga (depolarizacija). U tijelu stanice sumiraju se post-sinaptički potencijali tisuća susjednih neurona, u ovisnosti o

vremenu dolaska ulaznih informacija. Ako ukupni napon pređe određeni prag, neuron "pali" i generira tzv. akcijski potencijal u trajanju od 1 ms. Kada se informacija akcijskim potencijalom prenese do završnih članaka, onda oni, ovisno o veličini potenijala, proizvode i otpuštaju kemikalije, tzv. neurotransmitere. To zatim ponovno inicira niz opisanih događaja u dalnjim neuronima. Propagacija impulsa očigledno je jednosmjerna.

Funkcionalnost biološkog neurona imitira McCulloch-Pitts model umjetnog neurona, tzv. *Threshold Logic Unit* (TLU). Model koristi slijedeću analogiju: signali su opisani numeričkim iznosom i na ulazu u neuron množe se težinskim faktorom koji opisuje jakost sinapse; signali pomnoženi težinskim faktorima zatim se sumiraju analogno sumiranju potencijala u tijelu stanice; ako je dobiveni iznos iznad definirana praga, neuron daje izlazni signal.

U općenitom slučaju, umjetni neuron umjesto funkcije praga može imati i neku drugu funkciju, tzv. aktivacijsku funkciju (transfer funkcija, prijenosna funkcija). Općeniti model umjetnog neurona nalazi se na slici 4.2. U nastavku ćemo za pojam umjetni neuron ravnopravno koristiti i istovjetne pojmove: čvor ili jedinica.



Slika 4.2: Umjetni neuron

Ulagane signale (njih ukupno n) označavamo sa $x_1, x_2, x_3, \dots, x_n$, a pripadajuće težine označavamo sa $w_1, w_2, w_3, \dots, w_n$. Ulagani signali općenito su realni brojevi u intervalu $[-1, 1]$, $[0, 1]$ ili samo elementi iz $\{0, 1\}$, kada govorimo o Booleovom ulazu. Težinska suma *net* dana je formulom 4.1. Zbog kompaktnosti se često dogovorno uzima da je vrijednost praga $\theta = -w_0$ te se dodaje ulazni parametar x_0 s fiksiranim vrijednošću 1, te tada 4.1 izgleda:

$$net = w_1x_1 + w_2x_2 + \dots + w_nx_n - \theta \quad (4.1)$$

$$net = w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n = \sum_{i=0}^n w_i x_i \quad (4.2)$$

Izlaz y je rezultat aktivacijske funkcije te ga zapisujemo:

$$y = f\left(\sum_{i=0}^n \omega_i x_i\right) = f(\text{net}) \quad (4.3)$$

4.1.2. Svojstva umjetne neuronske mreže

Umjetna neuronska mreža (*engl.* Artificial Neural Network, ANN) u širem je smislu riječi umjetna replika ljudskog mozga kojom se nastoji simulirati postupak učenja. Nešto stroža definicija bila bi skup međusobno povezanih jednostavnih procesnih elemenata, *jedinica* ili *čvorova*, čija se funkcionalnost temelji na biološkom neuronu. Pri tome je obradbena moć mreže pohranjena u snazi veza između pojedinih neurona tj. težinama do kojih se dolazi postupkom prilagodbe odnosno učenjem iz skupa podataka za učenje. Neuronska mreža obrađuje podatke distribuiranim paralelnim radom svojih čvorova.

To je paradigma kojom su implementirani pojednostavljeni modeli što sačinjavaju biološku neuronsku mrežu. Ova je analogija vrlo poopćena jer, naravno, postoje još mnogi fenomeni živčanog sustava koji nisu modelirani ovim pristupom. Također, postoje i karakteristike umjetnih neuronskih mreža koje se ne poklapaju sa onima živčanog sustava.

Prednosti neuronskih mreža nad standardnim (simboličkim) načinom obrade podataka:

- Vrlo su dobre u procjeni nelineranih odnosa uzoraka
- Mogu raditi s nejasnim ili manjkavim podacima tipičnim za podatke iz različitih senzora, poput kamera i mikrofona, i u njima raspoznavati uzorke
- Robusne su na pogreške u podacima, za razliku od konvencionalnih metoda koje prepostavljaju normalnu raspodjelu obilježja u ulaznim podacima
- Stvaraju vlastite odnose između podataka koji nisu zadani na ekplicitan simbolički način
- Mogu raditi s velikim brojem varijabli ili parametara
- Prilagodljive su okolini
- Moguća je jednostavna VLSI implementacija (*engl.* Very-large-scale integration)
- Sposobne su formirati znanje učeći iz iskustva (tj. primjera)

Neuronske mreže odlično rješavaju sve probleme kod kojih postoji odnos između prediktorskih (ulaznih) i zavisnih (izlaznih) varijabli, bez obriza na visoku složenost te veze (nelinearnost) – *klasifikacija i regresija (predviđanje)*. Neuronske mreže uključuju se u sve više pordučja, a primjeri domena na kojima se već široko primjenjuju su:

- raspoznavanje uzoraka
- obrada slike
- obrada teksta
- obrada govora (zvuka)
- problemi optimizacije
- nelinearno upravljanje
- obrada nepreciznih i nekompletnih podataka
- simulacije i mnogi drugi

Način na koji su neuroni međusobno organizirani i povezani u mreži određuju njenu arhitekturu. Razlikujemo četiri osnovne arhitekture:

- aciklička unaprijedna (*engl. feedforward net*) mreža
- mreža s povratnom vezom (*engl. recurrent net*)
- lateralno povezana mreža
- hibridna mreža

U ovom radu proučavat će se konvolucijske neuronske mreže koje spadaju u kategoriju unaprijednih acikličkih mreža. U konvolucijskim mrežama kao aktivacijske funkcije koriste se ReLu i sigmoidalna funkcija koje su opisane u nastavku.

4.1.3. Aktivacijske funkcije

Adaline

Adaline (*engl. Adaptive Linear Element*) aktivacijska funkcija je prva aktivacijska funkcija ikada te dijeli ime sa neuronom koji ju koristi. Zbog svojeg ranog razvoja, naravno da je i najjednostavnija:

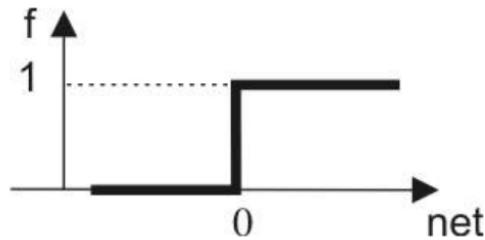
$$f(\text{net}) = \text{net} \quad (4.4)$$

Izlaz iz takve jedinice upravo je težinska suma njegovih ulaza. Dakle, izlaz odgovara općenitom modelu umjetnog neurona prikazanom na slici 4.2, a dan je izrazom 4.3.

Funkcija skoka

Funkcija skoka ili praga (*engl. Threshold Logic Unit, TLU*) na izlazu daje Booleov izlaz ($\{False, True\}, \{0, 1\}$). Graf TLU dan je slikom 4.3.

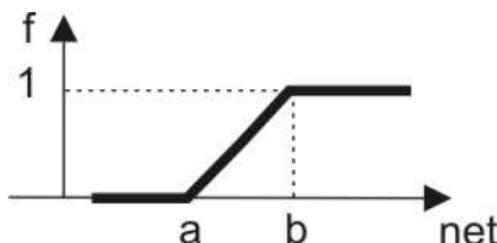
$$f(\text{net}) = \begin{cases} 0 & \text{za } \text{net} < 0, \\ 1 & \text{inače} \end{cases} \quad (4.5)$$



Slika 4.3: Funkcija skoka

Znak nejednakosti u funkciji skoka, dodatno može u nekim slučajevima sadržavati i znak jednakosti čime se funkcija ?? mijenja u ??

$$f(\text{net}) = \begin{cases} 0 & \text{za } \text{net} \leq a, \\ \text{net} & \text{za } a < \text{net} < b, \\ 1 & \text{za } \text{net} \geq b \end{cases} \quad (4.6)$$

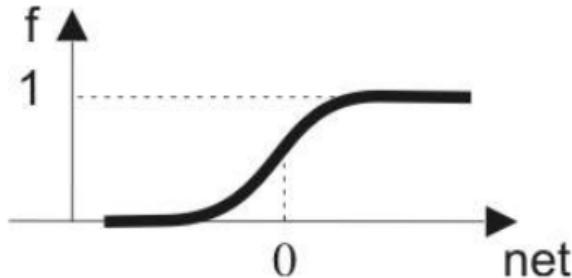


Slika 4.4: Na dijelovima linearna TLU

Najčešća aktivacijska funkcija jest sigmoida. Vrlo važno svojstvo ove funkcije koje ju razlikuje od dosad navedenih funkcija jest da je derivabilna. To će se pokazati kao

vrlo važna značajka za proces učenja neuronske mreže. Sigmoida je definirana kao:

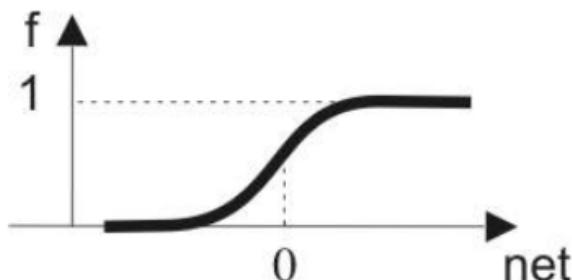
$$f(\text{net}) = \frac{1}{1 + e^{-a\text{net}}} \quad (4.7)$$



Slika 4.5: Sigmoidlana (logistička) funkcija

gdje parametar a određuje nagib funkcije. Vrlo važno svojstvo sigmoide je da sve brojeve iz bilo kojeg intervala preslikava na raspon $[0, 1]$. Zbog toga vrlo često izlaz iz logističke funkcije ima vjerojatnosnu interpretaciju, odnosno koristi se kada je na izlazu potrebno predvijeti vjerojatnosti. Važno je napomenuti da se sigmoida u području strojnog učenja koristi kada model ima binarni izlaz (predviđa 2 klase), dok se za višeklasnu regresiju/klasifikaciju koristi softmax:

$$f(\text{net}) = \frac{e^{\text{net}_i}}{\sum_{j=1}^k e^{\text{net}_j}} \quad (4.8)$$

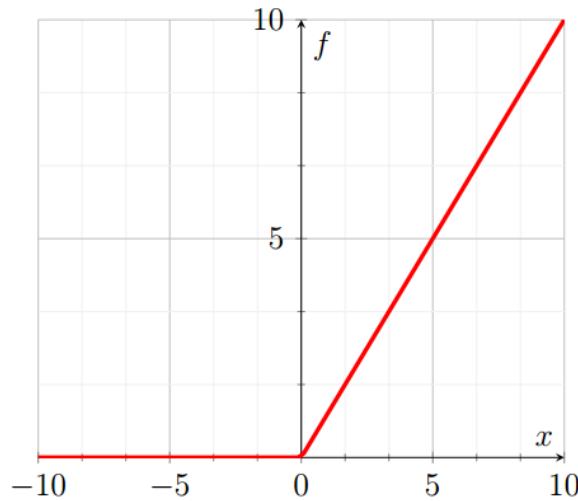


Slika 4.6: Sigmoidlana (logistička) funkcija

Softmax je generalizirana sigmoida. Izlaz iz funkcije softmax također vrlo često ima vjerojatnosnu interpretaciju.

Zadnja aktivacijska funkcija koju ovdje valja spomenuti je ReLU (*engl. Rectified Linear Unit*).

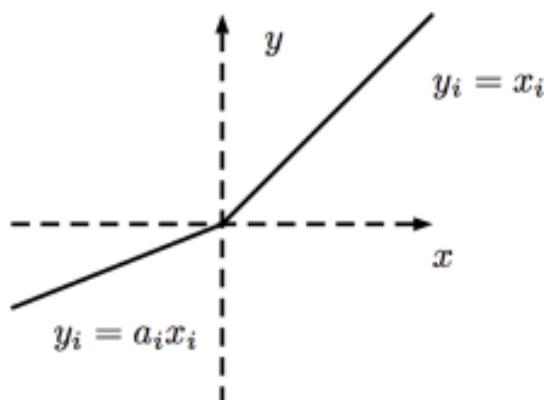
$$f(\text{net}) = \max(0, \text{net}) \quad (4.9)$$



Slika 4.7: ReLU

Ova aktivacijska funkcija je vrlo raširena u računalnom vidu i vrlo popularna kod konvolucijskih neuronskih mreža. Problem je što ova funkcija sve negativne vrijednosti stavlja u 0 što smanjuje mogućnost modela da potpuno izvuče znanje iz podataka. Zato je osmišljena još jedna funkcija koja ne preslikava negativne vrijednosti u 0, nego u neku vrijednost blizu 0 (što je vrijednost negativnija, preslikava se u vrijednost dalje od 0). Ta funkcija naziva se "Leaky ReLU":

$$f(\text{net}) = \begin{cases} a\text{net} & \text{za } \text{net} < 0, \\ \text{net} & \text{za } \text{net} \geq 0 \end{cases} \quad (4.10)$$



Slika 4.8: Leaky ReLU

4.1.4. Backpropagation algoritam

Kako bi neuronska mreža mogla predstaviti visoko nelinearne funkcije, potrebno je da prijenosna funkcija njezinih procesnih elemenata i sama bude nelinearna funkcija svojih ulaza. Nadalje, radi primjene gradijentne metode pri postupku učenja mreže, potrebno je da prijenosna funkcija bude derivabilna funkcija težinskih faktora. Backpropagation algoritam (Rumelhart et al., 1988) nastao je 1970ih godina te omogućuje učenje višeslojnih neuronskih mreža.

Algoritam koristi metodu gradijentnog spusta kako bi minimizirao nastalu pogrešku. Kod višeslojne mreže izlazni sloj može sačinjavati veći broj neurona, te je potrebno proširiti definiciju pogreške za višestruke izlaze:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{kd} - o_{kd})^2 \quad (4.11)$$

pri čemu su t_{kd} i o_{kd} ciljna i stvarna izlazna vrijednost za k -ti neuron izlaznog sloja dobivene s primjerom za učenje d . Učenje višeslojne mreže backpropagation algoritmom svodi se na pretraživanje u n -dimenzionalnom prostoru hipoteza, gdje je n ukupan broj težinskih faktora u mreži. Pogrešku u takvom prostoru možemo vizualizirati kao hiper-površinu koja, za razliku od paraboličke površine jednog procesnog elementa, može sadržavati više lokalnih minimuma u kojima gradijentni spust može zaglaviti. No, u praksi se pokazalo da algoritam daje vrlo dobre rezultate. Također, postoje i poboljšanja algoritma dodavanjem momenta inercije (Sutskever et al., 2013).

Algorithm 1: Backpropagation algoritam

inicijalizacija težina na male slučajne vrijednosti;

while $\Delta\varepsilon < \varepsilon_{thresh}$ **do**

izračunaj o_i za svaku ulaznu jedinicu x_i ulaza \mathbf{X} ;

za svaki izlazni čvor k izračunaj pogrešku δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \quad (4.12)$$

za svaku skrivenu jedinicu h izračunaj pogrešku δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{s \in \text{Downstream}(h)} (\omega_{hs} \delta_s) \quad (4.13)$$

ažurirati sve težine ω_{ij}

$$\omega_{ij} \leftarrow \omega_{ij} + \Delta\omega_{ij} \quad (4.14)$$

gdje je

$$\Delta\omega_{ij} \leftarrow \eta \delta_j x_{ij} \quad (4.15)$$

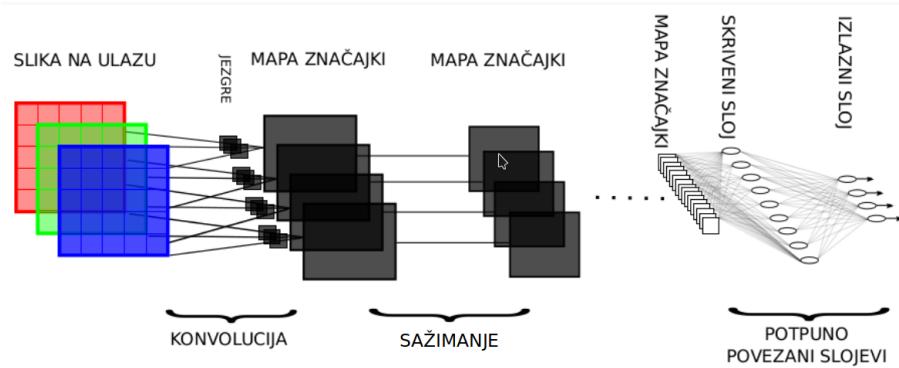
end

Zato što primjeri za učenje određuju ciljne vrijednosti samo izlaznog sloja neurona, poznata nam je jedino pogreška izlaznog sloja 4.11. Javlja se pitanje: kako ažurirati težine u skrivenom sloju? Backpropagation algoritam računa pogrešku bilo kojeg skrivenog neurona h tako da zbraja pogreške δ s svih onih neurona s na koje utječe izlaz neurona h , uz dodatno množenje težinskim faktorom ω_{hs} . Faktor ukazuje na to u kojoj je mjeri skriveni neuron h pridonio nastanku pogreške na izlazu jedinice s . Skup $\text{Downstream}(h)$ jest skup svih neurona "nizvodno" od neurona h , tj. svi oni neuroni čiji ulazi uključuju izlaz iz neurona h .

4.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže razvile su se početkom ovog tisućljeća (LeCun et al.). One su prilagođene za tipove podataka s topologijom rešetke (1D - vremenski slijed, 2D - slika/video). Standardna struktura konvolucijskog modela prikazana je na slici 4.9.

Konvolucijske neuronske mreže dobile su ime po matematičkoj operaciji koja se naziva **konvolucija**. Konvolucija je matematički operator koji od dvije funkcije x i w prizvodi treću, h , koja predstavlja količinu preklapanja između funkcije x i okrenute i prevedene verzije funkcije w . Tu operaciju možemo zapisati kao:



Slika 4.9: Klasična struktura konvolucijske neuronske mreže

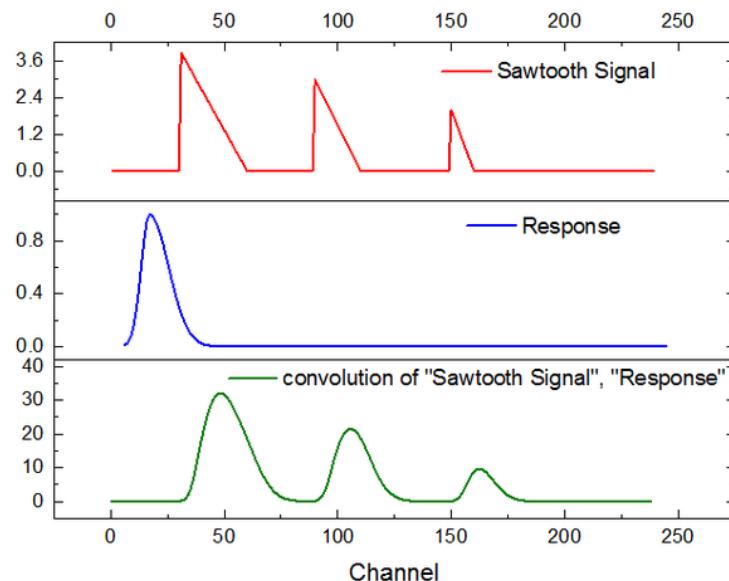
$$h(t) = \int_{-\infty}^{\infty} x(\tau)w(t - \tau)d\tau \quad (4.16)$$

što se još zapisuje kao:

$$h(t) = (x * w)(t) \quad (4.17)$$

gdje su:

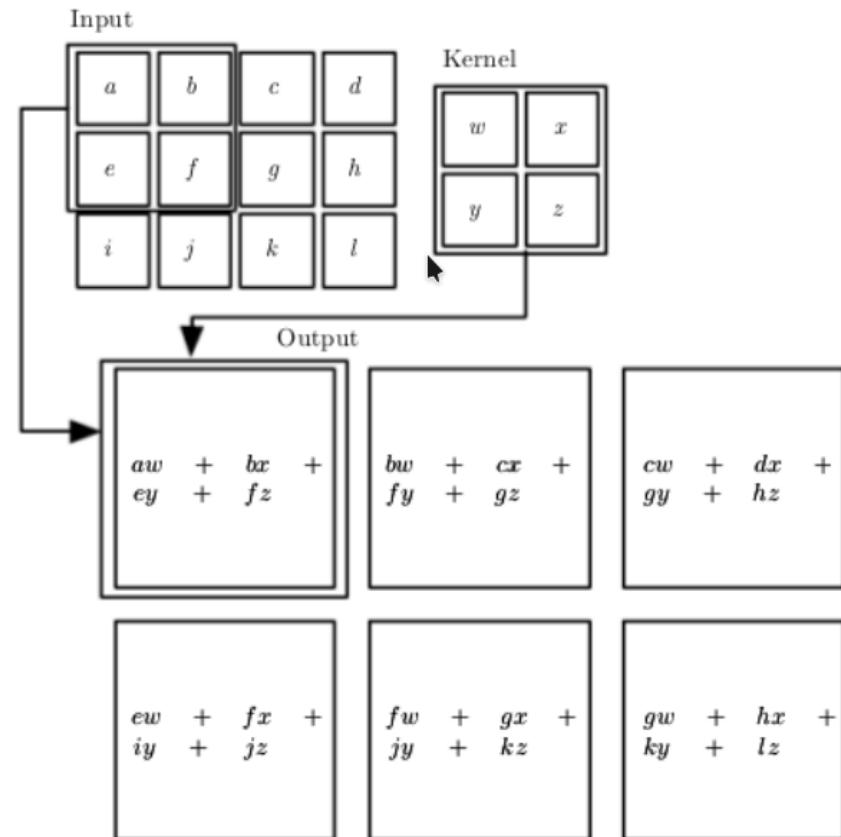
- funkcija x - ulaz
- funkcija w - jezgra
- funkcija h - **mapa značajki**, rezultat konvolucije



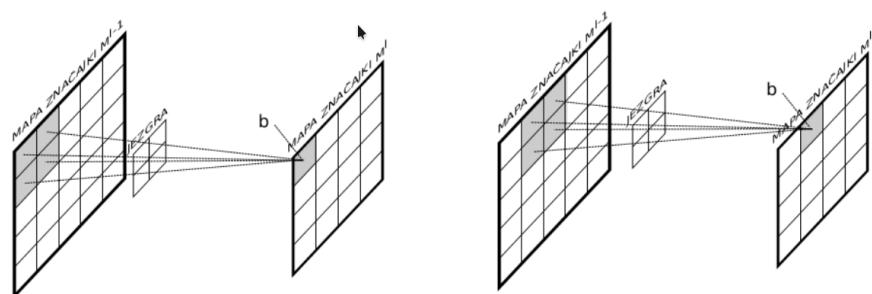
Slika 4.10: Konvolucija dviju funkcija

U primjenama su x i w obično višedimenzionalne funkcije, $\mathbf{x}(t), \mathbf{w}(t) \in \mathbb{R}^d$. Konvolucija se može primjenjivati i više dimenzija, npr. slika (dvondimenzionalan i diskretan ulaz), tada konvolucija izgleda kao što je prikazano jednadžbom 4.18.

$$h(i, j) = (\mathbf{X} * \mathbf{W})(i, j) = \sum_{m_{min}}^{m_{max}} \sum_{n_{min}}^{n_{max}} \mathbf{X}(m, n) \mathbf{W}(i - m, j - n) \quad (4.18)$$



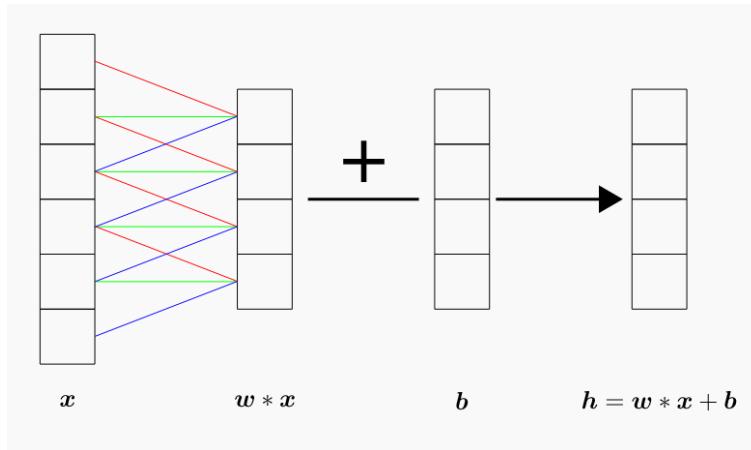
(a) Računanje mape značajki sa slike



(b) Računanje mape značajki iz druge mape značajki

Slika 4.11: Računanje mapa značajki

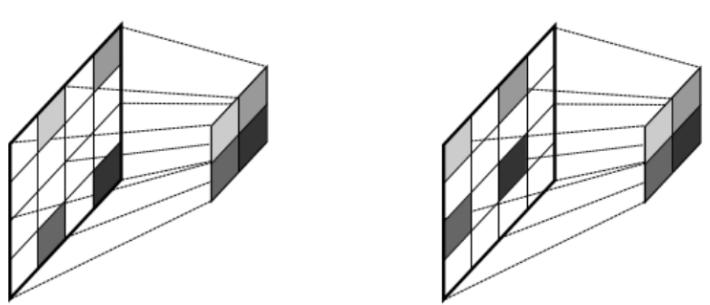
Slika 4.11.a prikazuje kako jezgra (*engl. kernel*) "klizi" po slici i računa pojedini element mape značajki. Slika 4.11.b prikazuje računanje mape značajki iz druge mape značajki. Na ovoj slici vidljivo je da elementi izlazne mape značajki ovise o lokalnom susjedstvu (čija veličina ovisi o veličini jezgre) elemenata ulazne mape značajki. Važno je napomenuti da se na obje slike svi elementi mape značajki računaju uz pomoč istog skupa parametara. Kod konvolucije, važno je napomenuti da modelira samo lokalne interakcije (ponovno, ovisno o veličini jezgre) te da dijeli parametre. Dijeljenje parametara, prikazano na slici 4.12, važno je svojstvo konvolucijskih neuronskih mreža jer im omogućuje evivariantnost obzirom na pomak. Takvo dijeljenje težina omogućava da mreža nauči relevantne i diskriminativne značajke. Jezgre se specijaliziraju za određenu funkciju, primjerice detekcija horizontalnih i vertikalnih bridova, odziv na različite uzorke i slično. Time postaju slične npr. Haarovim i drugim značajkama. Bez dijeljenja težina dijelovi neuronske mreže bili bi pretrenirani na određeni detalj podataka. Kod potpuno povezanog sloja svaki izlaz je povezan sa svim ulazima, a kod konvolucije samo sa malim dijelom ulaza što jako smanjuje i broj parametara koje je potrebno naučiti te omogućuje bržu evaluaciju. To dakle znači i manje podataka potrebno za trening da bi se postigla jednaka točnost modela, a istovremeno smanjuje opasnost pretreniranja modela. Problemi za konvoluciju nastaju kod transformacija ulaza koje nisu translacijske, na primjer, rotacija i skaliranje.



Slika 4.12: Dijeljenje parametara. Plava, crvena i zelena crta predstavljaju parametre jezgre koji klize po ulaznom nizu i stvaraju mapu značajki $w * x$

Sloj sažimanja (*engl. pooling layer*) ima zadaću mapirati skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu. Obično se za funkciju sažimanja uzima neki statistički pokazatelj ulaznih značajki kao što su srednja ili maksimalna vrijednost. Ovim postupkom dodatno se povećava invarijantnost na pomak, invarijantnost

na pomak je vrlo korisna ako pretpostavljamo da je za raspoznavanje bitnije detektirati prisutnost određene značajke, nego njezinu točnu lokaciju. Veličina regije sažimanja regulira dozu invarijantnosti: što je regija sažimanja veća, to je mreža invarijantija na veće pomake. Sažimanje se može obaviti ne samo preko susjednih značajki u istoj mapi, nego i preko različitih mapa: tada mreža može naučiti invarijantnost i na druge transformacije. Slika 4.13 prikazuje sažimanje maksimalnom vrijednosti (*engl. max-pooling*).



Slika 4.13: Max-pooling

Sažimanje kao posljedicu ima sažimanje mape značajki k puta, gdje je k veličina regije koja se sažima. Sažimanje je izuzetno korisno i zbog toga što omogućava procesiranje ulaza (npr. slika) različitih veličina.

Korištenjem konvolucijskog sloja i sloja sažimanja, u model se ugrađuju određene pretpostavke. Pretpostavke uvedene konvolucijom su:

- pretpostavljamo topologiju podataka
- predikcija je kovarijantna s pomakom

Pretpostavka uvedena slojem sažimanja jest da je predikcija invarijantna na male pomake. Ove pretpostavke povećavaju pristranost i smanjuju varijancu. U teoriji, to dovodi do podnaučenosti mreže te time i bolje generalizacije. No, u praksi nema podnaučenosti, ali pokazalo se da konvolucijski modeli generaliziraju bolje od potpuno povezanih slojeva. Konvolucijski slojevi mogu se zamisliti kao specijalni slučaj potpuno povezanih slojeva u kojima su sve težine koje nisu unutar područja jezgre postavljene na nulu.

Slika 4.9 prikazuje klasičnu strukturu konvolucijske neuronske mreže. Na ulazu može biti jedna monokromatska slika ili višekanalna slika u boji. Zatim slijede naizmjenice konvolucijski slojevi i slojevi sažimanja. Na samom kraju nalazi se nekoliko potpuno povezanih slojeva (perceptron) koji su jednodimenzionalni, uključujući i izlazni sloj.

5. Skupovi podataka

5.1. Pascal VOC2007

Ovaj skup podataka (Everingham et al.) također sadrži slike objekata u svakidašnjem okruženju. Ovdje se nalazi 9963 slike sa 24640 objekata podijeljenih u 20 kategorija:

- **Person:** person
- **Animal:** bird, cat, cow, dog, horse, sheep
- **Vehicle:** aeroplane, bicycle, boat, bus, car, motorbike, train
- **Indoor:** bottle, chair, dining table, potted plant, sofa, tv/monitor

U ovom skupu podataka, slike su označene kao okviri (koordinate okvira).



Slika 5.1: Primjeri slika iz skupa Pascal VOC2007

5.2. COCO

COCO (*Common Object in Context*) (Lin et al.) je skup podataka nastao 2014. godine. Kako i samo ime nalaže, COCO sadrži preko 330 tisuća označenih slika sa 1.5 milijuna označenih objekata u svakidašnjem okruženju i scenama te na taj način tim objektima pridružuje "kontekst". Ovaj skup podataka služi za treniranje algoritama čija je zadaća detekcija i klasifikacija objekata. COCO sadrži objekte podijeljene u 80 klasa. Na slikama se nalazi više objekata od kojih su svi označeni i segmentirani tako da se na ovom skupu podataka može vršiti i trening mreže za segmentaciju slika.

5.3. Video

U ovom radu, istraživanje je provedeno na videu 'Esquina Democratica.mp4' preuzetom sa web lokacije:

<https://www.videezy.com/people/4966-people-and-commerce-in-a-street-one-point-perspective>

Ovaj video je rezolucije 1920×1080 te sadrži 30 slika u sekundi videa (30 FPS).



Slika 5.2: Primjer slika i oznaka iz COCO skupa podataka

6. Implementacija

Ovaj zadatak potrebno je razlomiti na nekoliko podzadataka:

- detekcija osoba
 - lokalizacija
 - klasifikacija
- praćenje objekata na videu

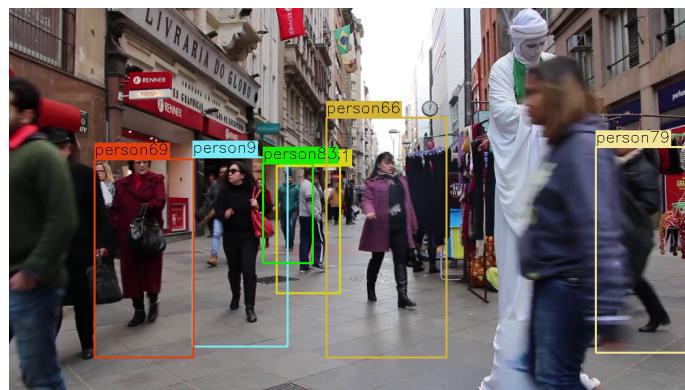
6.1. Korišteni alati

U ovom radu, algoritmi su implementirani u programskom jeziku **Python** verzije 3.6.5. na operacijskom sustavu Ubuntu Linux 18.04. Za konstrukciju neuronskih mreža korištena je biblioteka **Keras**. Keras je relativno nova biblioteka te je svojevrsna apstrakcija *Tensorflow-a* jer pojednostavljuje konstrukciju mreže na način da se slojevi "sami brinu" o dimenzijama. Programer mora samo definirati ulaznu dimenziju podataka te izlaznu dimenziju. Ostalo biblioteka napravi sama. Iz tog razloga je vrlo popularna te je primarno namjenjena brzom prototipiziranju mreža, dok su performanse u drugom planu. U Kerasu, autor može birati nekoliko biblioteka za neuronske mreže koje u pozadini odraduju posao kao što su **Tensorflow**, **CNTK** i **Theano**. U implementaciji se u pozadini koristi **Tensorflow**. Keras u pozadini koristi i standardne biblioteke kao što su *NumPy*, *Scipy* i sl. za numeričke i ostale operacije. Za Viola-Jones algoritam, kao i učitavanje slika i videa, spremanje i crtanje okvira korištena je biblioteka *OpenCV*.

6.2. Viola-Jones algoritam

Kao primjer algoritama za detekciju objekata implementiran je i algoritam Viole i Jonesa opisan u poglavljju 3.2. Algoritam je implementiran korištenjem biblioteke *OpenCV*.

Algoritam prima nekoliko ulaznih parametara od kojih je obavezan samo parametar "- -video" kojim se zadaje video na kojem da se vrši detekcija i praćenje osoba. Parametar "- -win-stride" je uređeni par vrijednosti (x, y) koji određuje korak u smjeru x i y osi po kojima se kreće klizeći prozor. Zadana vrijednost je (8, 8). Idući bitan parametar je "--scale". Ovaj parametar omogućuje detekciju ljudi različitih veličina na slici (i ljudi koji su blizu i ljudi koji su daleko). Povećanjem ovog parametra povećava se i rizik da ne budu detektirani svi objekti, ali istovremeno se algoritam i ubrzava. Obrnuto, smanjenjem ovog parametra algoritam detektira više objekata i usporava mu se rad. Posljedično, javlja se i više false-positivea. U implementaciji je ovaj algoritam postavljen na vrijednost 1.05. Ostavljena je i mogućnost spremanja svake slike u videu (svakog *framea*) parametrom "- -save" radi kasnije evaluacije.



Slika 6.1: Primjer detekcije Viola-Jones algoritmom

6.3. YOLO

U okviru ovog rada implementiran je i algoritam za detekciju objekata pod nazivom **YOLO - You Only Look Once** (?). Ovaj algoritam je predvodnik novih revolucionarnih algoritama za detekciju objekata. Trenutno najnovija, treća, verzija ovog algoritma nastala je 2018. godine, dok se algoritam, u svojoj osnovnoj inačici, pojavio 2015. godine. Također, postoje dvije inačice svake verzije algoritma: *full-YOLO* i *tiny-YOLO*. Tiny-YOLO ima manju mrežu i, naravno, slabije sposobnosti, no ima bolje performanse.

Prije ovog algoritma, detekcija objekata se vršila pomoću klasifikatorskih ili lokализacijskih mreža. Takve mreže primjenjuju se na različita područja slike te na različito skaliranu sliku. One regije slike za koje su mreže vrlo sigurne smatraju se detekcijom. YOLO algoritam koristi potpuno novu paradigmu, novi pristup detekciji objekata.

Ideja je da se kroz mrežu provuće jedna slika, bez skaliranja ili dovođenja različitih regija slike na ulaz mreže. Dakle, postupak detekcije objekata u mreži YOLO jest da mreža sama podijeli sliku na regije te predviđa okvire objekata (*engl. bounding-box*) te njihove vjerojatnosti za svaku regiju. Odnosno, klasifikacija i lokalizacija objekata na slici te predviđanje okvira objekata su sve zadaće jedne mreže.

6.3.1. Dizajn i svojstva mreže

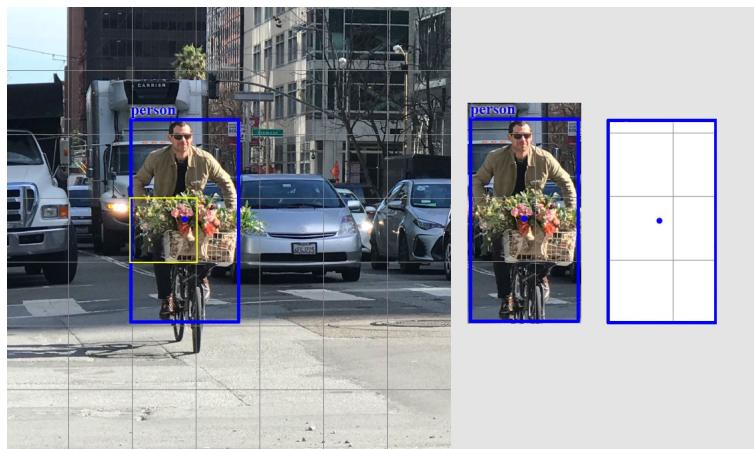
Name	Filters	Stride	Input Dimension	Output Dimension
Conv1	3 x 3 x 32	1 x 1	416 x 416 x 3	416 x 416 x 32
MaxPool1	2 x 2	2 x 2	416 x 416 x 32	208 x 208 x 32
Conv2	3 x 3 x 64	1 x 1	208 x 208 x 32	208 x 208 x 64
MaxPool2	2 x 2	2 x 2	208 x 208 x 64	104 x 104 x 64
Conv3	3 x 3 x 128	1 x 1	104 x 104 x 64	104 x 104 x 128
Conv4	1 x 1 x 64	1 x 1	104 x 104 x 128	104 x 104 x 64
Conv5	3 x 3 x 128	1 x 1	104 x 104 x 64	104 x 104 x 128
MaxPool2	2 x 2	2 x 2	104 x 104 x 128	52 x 52 x 128
Conv6	3 x 3 x 256	1 x 1	52 x 52 x 128	52 x 52 x 256
Conv7	1 x 1 x 128	1 x 1	52 x 52 x 256	52 x 52 x 128
Conv8	3 x 3 x 256	1 x 1	52 x 52 x 128	52 x 52 x 256
MaxPool3	2 x 2	2 x 2	52 x 52 x 256	26 x 26 x 256
Conv9	3 x 3 x 512	1 x 1	26 x 26 x 256	26 x 26 x 512
Conv10	1 x 1 x 256	1 x 1	26 x 26 x 512	26 x 26 x 256
Conv11	3 x 3 x 512	1 x 1	26 x 26 x 256	26 x 26 x 512
Conv12	1 x 1 x 256	1 x 1	26 x 26 x 512	26 x 26 x 256
Conv13	3 x 3 x 512	1 x 1	26 x 26 x 256	26 x 26 x 512
MaxPool4	2 x 2	2 x 2	26 x 26 x 512	13 x 13 x 512
Conv14	3 x 3 x 1024	1 x 1	13 x 13 x 512	13 x 13 x 1024
Conv15	1 x 1 x 512	1 x 1	13 x 13 x 1024	13 x 13 x 512
Conv16	3 x 3 x 1024	1 x 1	13 x 13 x 512	13 x 13 x 1024
Conv17	1 x 1 x 512	1 x 1	13 x 13 x 1024	13 x 13 x 512
Conv18	3 x 3 x 1024	1 x 1	13 x 13 x 512	13 x 13 x 1024
Conv19	3 x 3 x 1024	1 x 1	13 x 13 x 1024	13 x 13 x 1024
Conv20	3 x 3 x 1024	1 x 1	13 x 13 x 1024	13 x 13 x 1024
Route	from	Conv13		
Conv21	1 x 1 x 64	1 x 1	26 x 26 x 512	26 x 26 x 64
Space2Depth	reshape			
Concatenate	Route from Conv20	and	Space2Depth	
Conv22	3 x 3 x 1024	1 x 1	13 x 13 x 1280	13 x 13 x 1024
Conv23	1 x 1 x 425	1 x 1	13 x 13 x 1024	13 x 13 x 425

Tablica 6.1: Yolo arhitektura

U tablici 6.1 prikazana je arhitektura mreže Yolo. Radi preglednosti tablice, pod slojem Conv podrazumijeva se konvolucijski sloj dimenzija kako su napisane, Batch Normalization sloj koji, kako mu samo ime kaže, služi na normalizaciju *batcha*, poboljšava konvergenciju modela dok istovremeno služi i kao regularizator. Na kraju je dodan i Leaky ReLu sloj.

6.3.2. Podijela slike

Kada se na ulaz mreže dovede slika, prva stvar koju mreža napravi jest da podijeli sliku u $C \times C$ čelija kao što to prikazuje slika 6.2. Svaka od tih čelija predviđa točno **jedan** objekt. Primjerice, žuta čelija na slici pokušava predvidjeti objekt klase osoba čiji se centar (plava točka na slici desno) nalazi unutar te čelije.



Slika 6.2: Podijela slike na $C \times C$ čelija

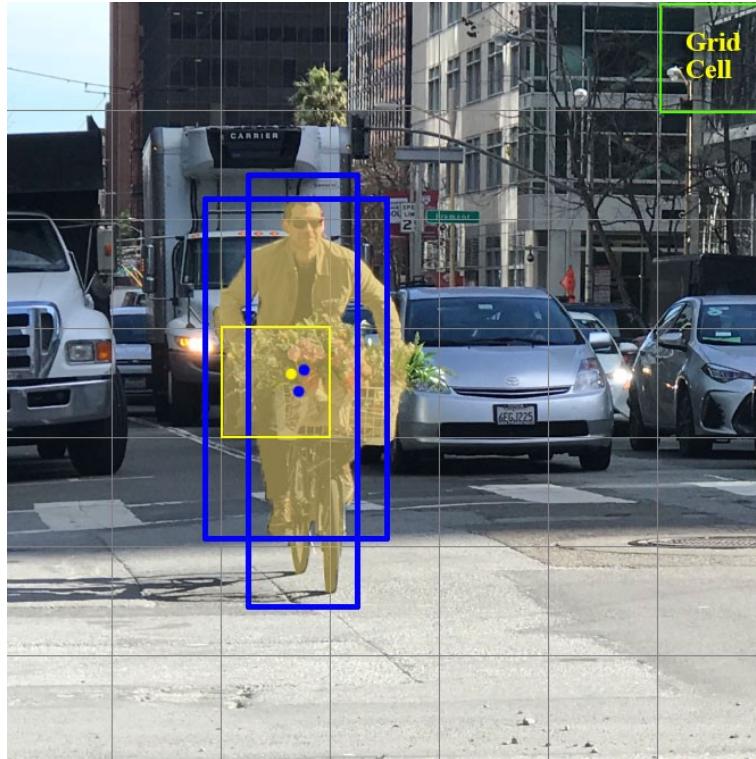
Svaka čelija predviđa fiksani broj okvira (*engl. bounding boxes*). Na slici ?? žuta čelija predviđa dva okvira kako bi locirala osobu na slici.

Međutim, mogućnost predviđanja samo jednog objekta po čeliji donekle ograničava algoritam u smislu blizine objekata koje može detektirati. Odnosno, ako se u čeliji nalazi više od 1 objekta, algoritam mora odabratи samo 1 koji od njih koji će detektirati.

Za svaku čeliju C algoritam predviđa:

- B okvira te za svaki okvir njegovu pouzadnost (*engl. box confidence score*)
- samo jedan objekt
- P vjerojatnosti, po jednu za svaku klasu C

Ulez u mrežu može biti proizvoljne veličine, no mreža YOLO preoblikuje ulaz u dimenzije $416 \times 416 \times 3$ kako bi ga mogao dovesti na ulaz u mrežu. Mreža dijeli sliku



Slika 6.3: Predviđanje okvira

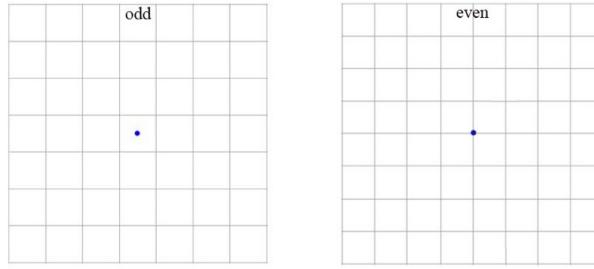
na 13×13 čelija. Svaka čelija predviđa 5 okvira za objekt koji se nalazi u toj čeliji (ako se nalazi), te svaki od okvira ima 85 parametara. Izlaz iz mreže je oblika $(13, 13, 425)$. U svrhu razumijevanja izlaza iz mreže, zgodno ga je zamisliti kao kompozit tri tenzora:

- pouzdanost okvira (*engl. box_confidence*) kao tenzor dimenzije $(13 \times 13, 5, 1)$ koji sadrži p_c pouzdanost postojanja objekta u svakom od 5 predviđenih okvira u čeliji C za svaku od 13×13 čelija
- okviri (*engl. boxes*) kao tenzor dimenzije $(13 \times 13, 5, 4)$ koji sadrži $(x_{top}, y_{top}, x_{bottom}, y_{bottom})$ za svaki od 5 okvira u čeliji
- vjerojatnosti klasa za okvire (*engl. box_class_prob*) kao tenzor dimenzije $(13 \times 13, 5, 80)$ koji čuva informaciju o pouzdanosti za svaku od 80 klasa za svaki od 5 okvira u čeliji.

Kao što prikazuje slika 6.4, neparan broj čelija po dimenziji (npr. 13×13) je pogodan jer je jednostavnije odrediti u kojoj se čeliji objekt nalazi.

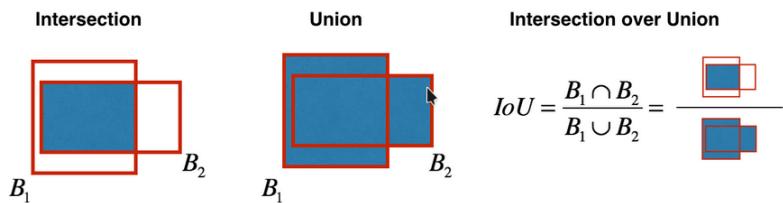
6.3.3. Funkcija gubitka

Yolo algoritam predviđa više okvira po jendoj čeliji na slici. Od svih tih čelija, potrebno je zadržati samo one u kojima se zaista nalazi objekt. U tu svrhu, izabire se



Slika 6.4: Neparan i paran broj čelija po dimenzij

ona čelija koja ima naveći IoU (*engl. Intersection over Union*) sa stvarnim objektom (*engl. ground truth*). Za računanje gubitka, Yolo koristi sumu kvadrata pogrešaka predviđanja algoritma i stvarnih objekata. Zbog ideje Yolo algoritma da se klasifikacija i lokalizacija rade istom mrežom u jednom prolasku, funkcija gubitka sastavljena je od nekoliko dijelova.



Slika 6.5: Intersection over union

• Pogreška klasifikacije

Ako je objekt detektiran, pogreška klasifikacije u svakoj čeliji računa se kao kvadrat pogreške uvjetne vjerojatnosti za svaku klasu.

$$\sum_{i=1}^{S \times S} 1_i^{obj} \sum_{c \in klase} (p_i(c) - \hat{p}_i(c))^2 \quad (6.1)$$

gdje

- $1_i^{obj} = 1$ ako se objekt pojavljuje u čeliji i , inače 0
- $\hat{p}_i(c)$ označava uvjetnu vjerojatnost klase c u čeliji i

• Pogreška lokalizacije

Ovaj segment funkcije pogreške mjeri grešku u lokaciji i veličini predviđenih okvira. U obzir se uzimaju samo oni okviri u kojima se objekt nalazi. Ideja je da se jednak greška u velikom i u malom predviđenom okviru ne kažnjavaju jednakom (npr. greška od 2px u velikom i malom okviru). Zbog toga Yolo

predviđa **korjene kvadrata** visine i širine okvira. Nadalje, kako bi još točije predviđao okvire, pogreška okvira se množi paramterom λ_{coord} .

$$\begin{aligned} & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \\ & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \end{aligned} \quad (6.2)$$

gdje

- $1_{ij}^{obj} = 1$ ako se objekt nalazi u j -tom okviru i -te čelije, inače 0
- λ_{coord} parametar za ugađanje težine za pogrešku u koordinatama okvira

• Pogreška pouzdanosti

Ovdje se javljaju dva slučaja:

1. Objekt je detektiran u okviru

$$\sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \quad (6.3)$$

gdje

- $1_{ij}^{obj} = 1$ ako se objekt nalazi u j -tom okviru i -te čelije, inače 0
- \hat{C}_i jest iznos pouzdanosti za okvir j u čeliji i

2. Objekt nije detektiran u okviru

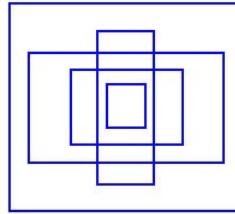
Većina okvira ne sadrži objekt što dovodi do problema neravnomjernosti klase, odnosno, model se trenira da puno češće detektira pozadinu nego sami objekti. Kako bi se ovaj problem riješio, gubitak se množi faktorom λ_{no-obj} koji ima vrijednost između 0 i 1.

$$\lambda_{no-obj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{no-obj} (C_i - \hat{C}_i)^2 \quad (6.4)$$

gdje

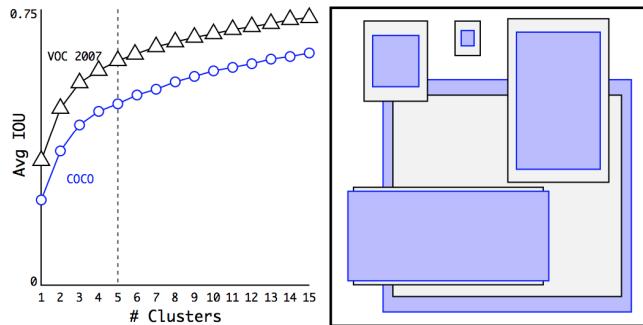
- 1_{ij}^{no-obj} jest komplement gornjeg izraza 1_{ij}^{obj}
- \hat{C}_i jest iznos pouzdanosti za okvir j u čeliji i
- λ_{no-obj} je faktor kojim se smanjuje iznos gubitka pri detekciji pozadine

6.3.4. Anchor boxes



Slika 6.6: Klasteri dimenzija za anchor boxeve

Anchor boxes ili *priors* jest još jedan pojam uveden ovim algoritmom. Kako u početku treninga mreža ne bi predviđala okvire nasumično, predloženo je da se, ovisno o domeni i podacima, izabere n najčešćih okvira koji najbolje opisuju klase u tom skupu podataka. Primjerice, pješake se uglavnom može označiti uspravnim pravokutnim okvirom, dok je kod automobila okvir obično polegnuti pravokutnik. Na ovaj način uzeto je 5 standardnih okvira (*anchor boxes*) koji su izračunati K-means klasteringom ($k = 5$) iz COCO i VOC2007 skupova podataka kako bi se pronašle centoride k najboljih klastera. Da bi se odredile točne dimenzije standardnih okvira, korišena je IoU mjeru. Izračunati anchor boxevi prikazani su slikom 6.7. Ljubičasti okviri uzeti su iz COCO skupa podataka, a bijeli sa crnim rubom iz VOC2007 skupa podataka.



Slika 6.7: Standardni okviri dobiveni K-means klasteringom i IoU mjerom (?)

Tako sada Yolo mreža predviđa koordinate okvira objekta direktno, no relativno u odnosu na poziciju promatrane čelije od gornjeg lijevog ruba. Nadalje, okviri se predviđaju kao odstupanje od nekog od standardnih okvira. To odstupanje je ograničeno kako nebi bilo koji okvir mogao prijeći u bilo koji drugi jer se onda gubi smisao ovog cijelog postupka. Ograničavanjem odstupanja omogućeno je kako bi se svako predviđanje moglo opredjeliti za točno jedan oblik iz skupa standardnih okvira.

6.3.5. Backbone

Mreža kojom se inicijalizira Yolo mreža naziva se *DarkNet* (Redmon). Mreža je prikazana na slici 6.8

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Slika 6.8: Originalna DarkNet-19 mreža (?)

Zadnji slojevi (koji su prekriženi na slici 6.8) izbačeni su za potrebe Yolo mreže. Ova arhitektura zahtjeva oko 5.8 milijardi operacija što je, u usporedbi sa VGG16 mrežom koja ima 3 sloja manje i zahtjeva oko 30.69 milijardi operacija za samo jednu sliku veličine 224×224 , izuzetno malo i efikasno.

6.3.6. Trening

Prvo se trenira mreža DarkNet-19 za klasifikaciju na ImageNet skupu podataka sa 1000 klasa objekata u 160 epoha koristeći stohastički gradijentni spust (*SGD*). Stopa učenja inicijalno je postavljena na vrijednost 0.1 sa polinomijalnim raspadom potencije 4, *weight decay* parametrom postavljenim na 0.0005 te momentom vrijednosti 0.9. Tijekom treninga koriste se standardne metode izmjene podataka kao što su nasumično izrezivanje, rotacija i izmjene boje.

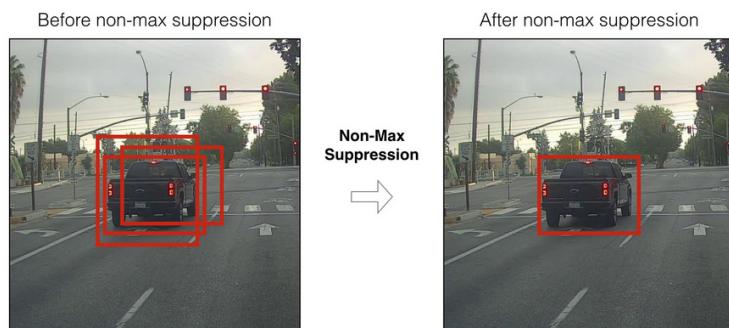
Zatim se izgradi mreža YOLO, prenesu se težine iz DarkNet mreže u početne slojeve mreže YOLO, te se ponovo trenira. Stopa učenja kreće od vrijednosti 10^{-3} , a

smanjuje se nakon 10, 60 i 90 epoha na isti način kao i kod treniranja DarkNeta. Parametar *weight decay* postavljen je na vrijednost od 0.0005, a moment na vrijednost 0.9. Koriste se i metode izmjene boja, nasumičnog izrezivanja itd. kako bi algoritam bio što robusniji. Mreža se na taj način trenira na COCO i VOC2007 skupovima podataka.

Yolo mreža može primati slike različitih veličina. Svakih 10 *batcheva* algoritam nasumično odabere neku veličinu slike za treniranje modela (veličina slike mora biti višekratnik broja 32). Na taj način, prisiljava se mrežu da predviđa dobro za slike različitih dimenzija i skala. Na taj način omogućava se prilagodba mreže za određenu svrhu. Primjerice, ako je potrebno raditi predviđanja u stvarnom vremenu, moguće je davati manje slike algoritmu kako bi predviđanja bila brža. Ako je pak cilj da predviđanja budu što točnija, mogu se uzimati slike veće rezolucije.

6.3.7. Predviđanje

Način na koji YOLO detektira objekte i predviđa okvire objašnjeni su u podoglavlјima 6.3.2 i 6.3.4. No još nije spomenuto na koji način YOLO zaključuje koje okvire treba odbaciti a koje zadržati. Dakle, za neki objekt mreža predviđi nekoliko okvira. Metoda kojom se odabiru okviri za odbacivanje naziva se **Non-Maximal Suppression**. Način na koji ta metoda radi jest da poreda okvire po pouzdanosti, iterira kroz njih i odbacuje sve okvire koji predviđaju istu klasu kao neki od prethodnih i imaju IoU veći od 0.5 sa tim prethodnim okvirom iste klase. Taj proces se ponavlja dok svi okviri ne budu ili odbačeni ili zadržani.



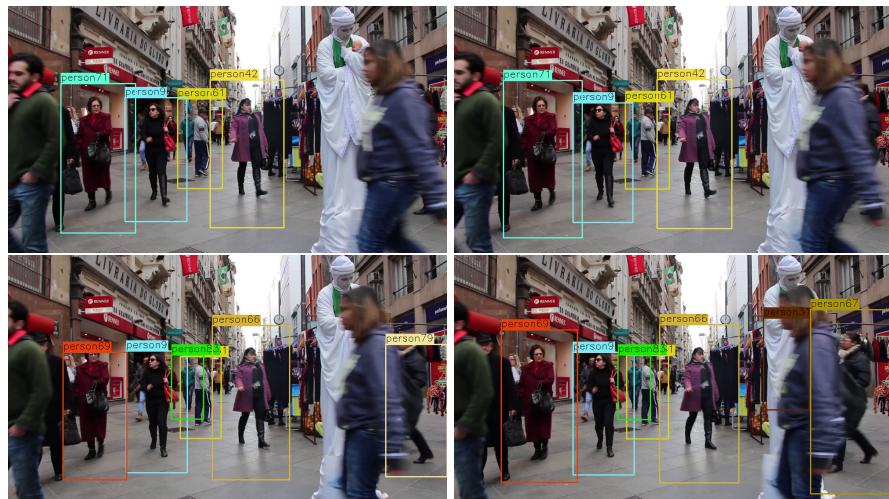
Slika 6.9: Non-Maximal Suppression

7. Eksperimentalni rezultati

Postoje dvije vrste podataka na kojima je moguće provesti ovo istraživanje: slika i video. Nije potrebno navoditi da koristeći sliku, zbog svojstva statičnosti, eliminiramo zadatku praćenja objekata.

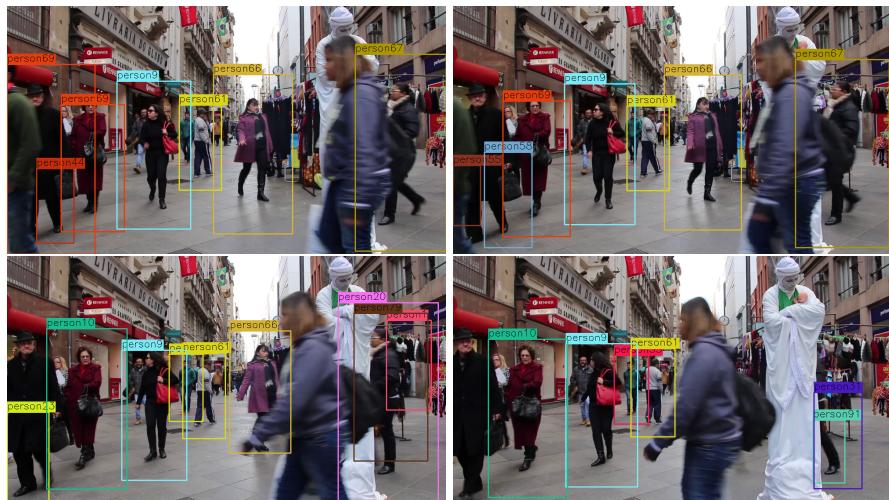
7.1. Viola-Jones algoritam

Algoritam Viole i Jones-a, iako osmišljen davno, koristi se i danas. Izuzetno je brz, dovoljno točan i zahtjeva malo računalnih resursa. Sami algoritam opisan je u poglavljju 3.2.



Slika 7.1: Primjer detekcije Viola-Jones algoritmom

Viola-Jones algoritam vrlo je pouzdan i dolazi sa većinom biblioteka specijaliziranih za računalni vid (*OpenCV*). Za razliku od YOLO-a, ovaj algoritam ima nešto više false-positivea. Nekoliko slika sa pogrešnim označavanjem prikazano je na slici ???. Još jedan od nedostataka ovog algoritma jest osjetljivost na različite fotometrijske značajke (promjene svijetlosti).



Slika 7.2: Primjer detekcije Viola-Jones algoritmom

7.2. YOLO

Ovaj je algoritam, u originalnom obliku, implementiran u programskom jeziku C te je izuzetno brz. U radu se spominje da na grafičkoj kartici NVidia Titan X postiže brzine od 45 do 90 FPS (*engl. Frames Per Second*) što je izuzetno brzo, uzimajući u obzir i izvanrednu točnost modela. Implementacija u ovom radu je nešto sporija zbog nekoliko razloga: algoritam je implementiran u programskom jeziku Python, a što je još važnije, izvršava se na procesoru (CPU).

U ovoj implementaciji za procesiranje slike potrebno je vrijeme od 3.5 do 6 sekundi, ovisno o slici. Tu je uračunato nekoliko komponenti čija se vremena izvođenja nalaze u tablici ?? te slici ??.

Listing 7.1: Poziv programa za jednu sliku

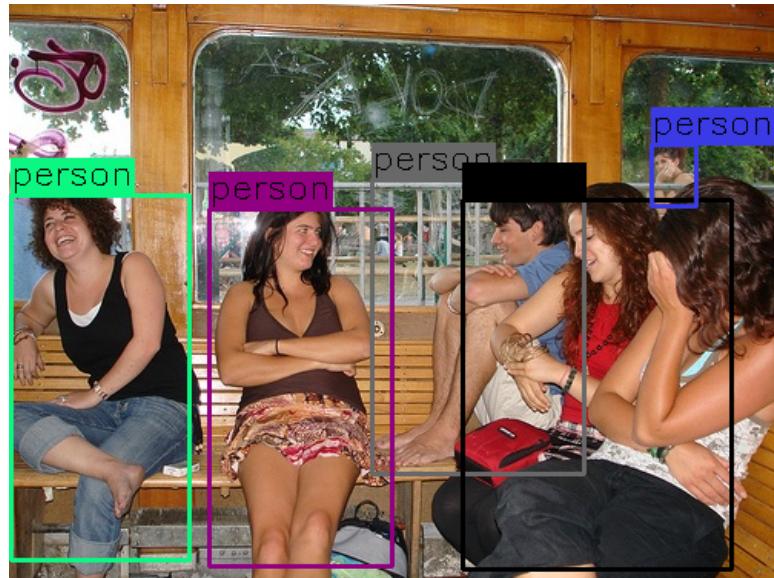
```
python3 eval.py image='path/to/image'
```

Obrada slike	3.9219 s
Stvaranje modela	2.717 s
Predviđanje sa slike	1.201 s
Postprocesiranje	0.0003 s
Iscrtavanje okvira	0.0002 s

Tablica 7.1: Vrijeme izvođenja pojedinih dijelova programa za obradu slike

```
-/Workspaces/Python/diplomski/yolo/yolo_video ➜ master pipenv run python eval.py --image='images/2007_001284.jpg'  
/home/luka/.local/share/virtualenvs/yolo_video-cLZT34q0/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: compiletime version 3.5 of module 'tensorflow.python.framework.tensor_util' does not match runtime version 3.6  
    return f(*args, **kwargs)  
2018-06-20 19:29:30.153551: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA  
Absolute execution time: 3.9219  
Model creation time: 2.717  
Prediction time: 1.201  
Postprocessing time: 0.0003  
Boundary boxes drawing time: 0.0002  
-/Workspaces/Python/diplomski/yolo/yolo_video ➜ master
```

(a) Vrijeme izvođenja pojedinih dijelova programa za obradu slike



(b) Slika sa označenim detektiranim objektima

Slika 7.3: Rezultat pokretanja algoritma na slici

Danas je detekcija objekata na slikama vrlo dobro riješena. Kao što je bilo govora na početku poglavlja o detekciji konvolucijskim mrežama, problem slike odlično riješavaju modeli koji imaju klasifikaciju i lokalizaciju odvojenu, koji izrežu dijelove slike i ponovo ih dovode na ulaze mreža. Primjer jednog takvog algoritma je i *Tiny Face Detector* opisan u istoimenom radu (?). No, ti algoritmi su izuzetno spori zbog gore navedenih razloga. Njima nije naglasak na performansama, nego isključivo na točnosti i preciznosti.

Ovaj algoritam je, kao predvodnik u kategoriji algoritama za detekciju objekata, zapravo dizajniran za brzo izvođenje i primjenu na videu, uz minimalne gubitke točnosti i preciznosti.

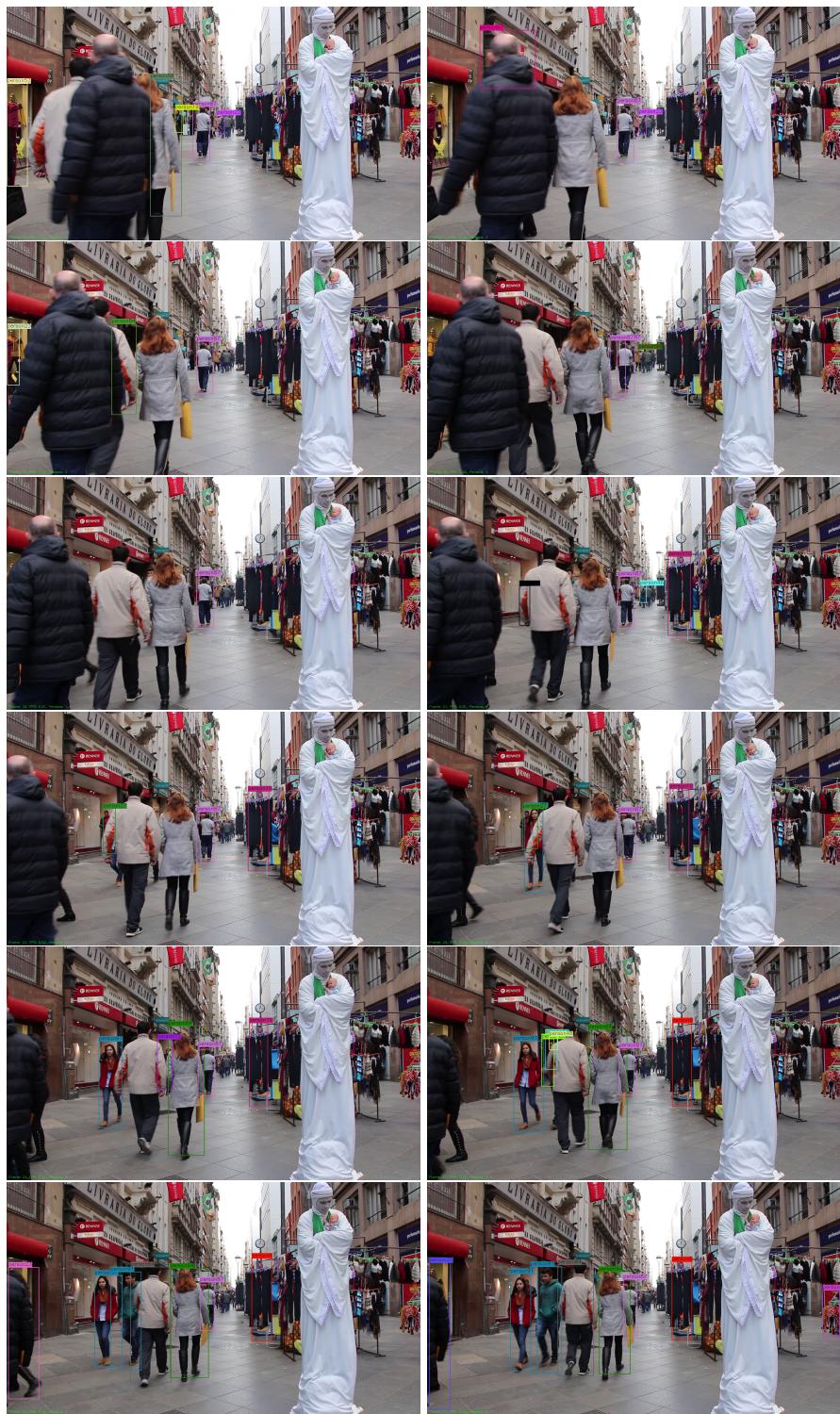
Video je zapravo niz slika pa se tada ovaj problem svodi na provođenje ovog algoritma za svaku sliku u videu (24 slike u sekundi). Kao što je već navedeno, u eksperimentima se algoritam izvodi na CPU, pa je stoga uvedena zastavica pri pozivu algoritma koja određuje koliko će slika u videu algoritam preskočiti prije nego uzme iduću sliku na obradu i predviđanje. Tako je moguće ubrzati izvođenje programa, te ako se osobe na slici kreću sporije, u nekoj koloni, moguće je, i čak poželjno, postaviti ovaj parametar na neki veći broj. Tada se standardni poziv programa u terminalu mijenja na sljedeći način:

Listing 7.2: Poziv programa sa opcijom preskakanja zadanog broja (n) slika u videu

```
python3 eval.py --skip_n_frames=n --video='path/to/video'
```

Podatak da su grafičke kartice za ove primjene 100 puta brže od procesora, daje podatak da bi se ovaj algoritam mogao izvršavati brzinom oko 25 FPS-a što je dovoljno i za video u stvarnom vremenu jer je standardna brzina videa 24 FPS-a. Dodatno, model se učita jednom i vrši se predikcija na svakoj slici sa tim, već učitanim, modelom. Dakle, prema tablici ?? za predviđanje i postprocesiranje slike dovoljno je oko 1.2 sekunde po slici što daje brzinu od oko 80 FPS-a na grafičkoj kartici, što je vrlo blizu originalnog rada. Ova brzina postignuta je jer se u pozadini Kerasa izvodi Tensorflow koji je maksimalno optimiziran i koristi biblioteke kao što su NumPy i SciPy koje su napisane u C-u.

Nadalje, u poglavlju o implementaciji algoritma YOLO, bilo je govora o Non-Maximal Suppression metodi koju ovaj algoritam koristi za predviđanje i evaluaciju kako bi maknuo redundantne okvire. No, tamo je naveden prag pouzdanosti koji utječe na to koji od okvira uopće dolaze u obzir pri predviđanju objekata. Taj prag je također parametar koji je moguće mijenjati. Naravno, tako će algoritam prikazivati sve više

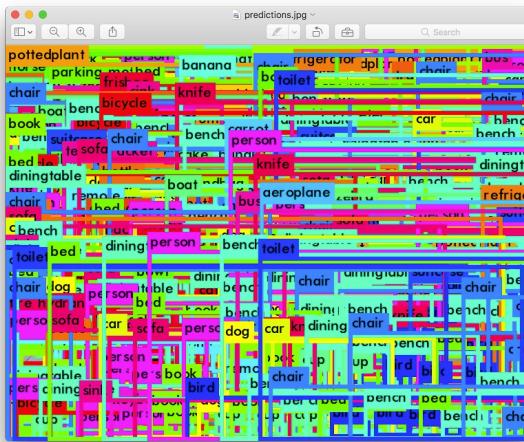


Slika 7.4: Primjer konzistentnog praćenja osobe označene sa *person87*

šuma, te je stoga potrebno dobro istražiti kako se algoritam ponaša na danom skupu podataka. Na slici ?? prokazana je ekstremna situacija kada se ovaj prag postavi na vrijednost 0 kada algoritam isctrava sve okvire. U ovoj implementaciji, konkretno, prag je izabran eksperimentalno i postavljen na 0.3. Ako korisnik želi postaviti neki drugi prag, to mora biti broj između 0 i 1. Korištenje ove zastavice omogućeno je sljedećom naredbom:

Listing 7.3: Poziv programa sa prozvoljnim pragom pouzdanosti okvira

```
python3 eval.py --threshold=n --video='path/to/video'
```

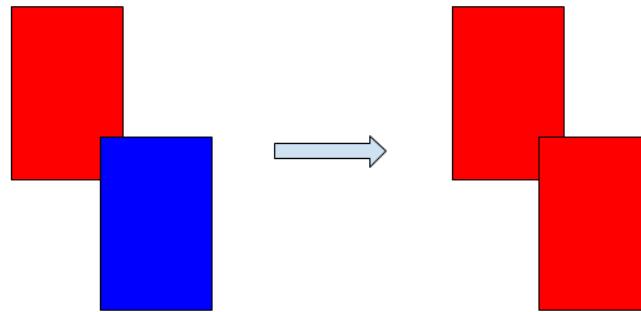


Slika 7.5: Rezultat ako se postavi prag na vrijednost 0 (?)

Ovime je završen zadatak detekcije objekata (klasifikacija i lokalizacija). Diničnost videa u odnosu na sliku donosi sa sobom još jedan problem, a to je praćenje kretanja objekata.

Ovaj zadatak može biti riješen na više načina, no u ovoj implementaciji odabrana je već spomenuta metoda *Intersection over Union*. Ova metoda prati trajektorije objekata na način da zapamti predviđene okvire iz prošle slike (*engl. frame*) te nakon predviđanja okvira iduće slike iterira kroz okvire nove slike i uspoređuje ih sa svim okvirima prošle slike. Tada okvir iz nove slike koji ima najveći IoU rezultat sa okvirom stare slike, postaje taj okvir. To se manifestira u videu na način da novi okvir preuzme boju okvira sa prošle slike s kojim ima najveće poklapanje. Ova metoda, kao i sve druge, ima i svojih nedostataka. Primjerice, ako je okvir objekta sa prošle slike imao vrlo malo preklapanje sa okvirom potpuno drugog objekta sa nove slike, a okvir sa nove

slike nema doticaja sa nijednim drugim okvirom sa stare slike, preuzet će boju okvira sa stare slike. Situacija je prikazana slikom ?? To je neželjeno ponašanje jer to nije taj objekt. To je riješeno na način da svaki okvir, pri stvaranju (predviđanju) dobija neku svoju nasumičnu boju. Zatim, kada dođe do gore opisane situacije, gleda se rezultat IoU metode te se stavlja prag, odnosno, postotak preklapanja, za koji možemo reći da je dovoljan da odredimo da je to isti objekt iz prošle slike. U implementaciji je taj prag postavljen na vrijednost 0.5.



Slika 7.6: Situacija kada okvir novog objekta preuzeće boju starog objekta na temelju zanemarivo malog poklapanja

Jedan od primjera takvih pogrešaka prikazan je na slici ???. Vidi se da na prvoj slici u lijevom rubu slike, postoje dvije osobe: *person10* i *person69*. Na idućoj slici, viljivo je da je algoritam promijenio ime osobe 69 u osobu 10. To se događa jer se promijenila veličina okvira te sada okvir osobe 69 ima veći IoU sa osobom 10 iz prethodnog okvira nego sa osobom 69.



Slika 7.7: Primjer pogreške kod praćenja trajektorija metodom IoU

Kod ove metode javlja se još jedan problem, a to je da ako se objekt kreće uz objektiv kamere i ide slijeva nadesno ili obrnuto, svi okviri konvergiraju u okvir tog objekta. Ovaj problem se javlja zbog načina računanja trajektorija objekta metodom IoU. Kada objekt prođe ispred objektiva, njegov okvir je jako velik, gotovo kao cijeli

objektiv. Kada se u tom koraku računa trajektorija ostalih objekata, svi ostali okviri imaju najveći IoU sa tim okvirom te se algoritam dalje ponaša kao da su svi oni ista osoba. Ovaj problem riješen je na način da je postavljen prag veličine okvira koji se crta. Što je objekt bliže objektivu, on je veći, pa je na ovaj način odabrana blizina objekata koji se detektiraju. U implementaciji je ovaj prag postavljen na vrijednost od 100000.



Slika 7.8: Primjer kretanja objekta uz objektiv kamere

8. Zaključak

Zaključak.

LITERATURA

Christian Ledig et al. Photo-realistic single image super-resolution using a generative adversarial network. 2016. URL http://openaccess.thecvf.com/content_cvpr_2017/papers/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.pdf.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, i A. Zisserman. The Georg B. Keller, Tobias Bonhoeffer, i Mark Hübener. Sensorimotor mismatch signals in primary visual cortex of the behaving mouse. *Neuron*, 74(5):809 – 815, 2012. ISSN 0896-6273. doi: <https://doi.org/10.1016/j.neuron.2012.03.040>. URL <http://www.sciencedirect.com/science/article/pii/S0896627312003844>.

Stanford Vision Lab. Large scale visual recognition challenge (ILSVRC), 2018. URL <http://www.image-net.org/challenges/LSVRC>.

Yann LeCun, Patrick Haffner, i L Bottou.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll, i C. Lawrence Zitnick. Microsoft.

Deva Ramanan Peiyun Hu. Finding tiny faces. 2017. URL <https://www.cs.cmu.edu/~peiyunh/tiny/>.

Joseph Redmon. Darknet: Open source neural networks in c. \bibitem[Redmon et al(2015)]{Redmon2015} Redmon, Divvala, Girshick, i Farhadi]YOLO Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, i Ali Farhadi. You only

look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL <http://arxiv.org/abs/1506.02640>.

David E. Rumelhart, Geoffrey E. Hinton, i Ronald J. Williams. Neurocomputing: Foundations of research. stranice 696–699, 1988. URL <http://dl.acm.org/citation.cfm?id=65669.104451>.

Ilya Sutskever, James Martens, George Dahl, i Geoffrey Hinton. On the importance of initialization and momentum in deep learning. stranice 1139–1147, 2013.

Brojanje ljudi u sceni

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.