SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6349

Konvolucijski modeli za lokalizaciju sportaša

Dominik Stipić

Zagreb, srpanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 15. ožujka 2019.

ZAVRŠNI ZADATAK br. 6349

Pristupnik: **Dominik Stipić (0036501052)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Konvolucijski modeli za lokalizaciju sportaša**

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 primjene takvih pristupa za lokaliziranje sportaša u snimkama nogometnih susreta.
U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće metode dubokog učenja za lokalizaciju objekata u slikama. Predložiti arhitekturu dubokog modela koja bi bila prikladna za pronalaženje osoba. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele na snimkama nogometnih susreta. 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.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 14. lipnja 2019.

Mentor:
Prof. dr. sc. Siniša Šegvić

Djelovođa:
Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Doc. dr. sc. Marko Čupić

Zahvaljujem se mentoru, prof. dr. sc. Siniši Šegviću na ukazanoj pomoći, strpljenju i znanju kojim mi je pomogao napraviti ovaj rad. Također se zahvaljujem svojoj obitelji i bližnjima na danoj podršci.

SADRŽAJ

1. Uvod

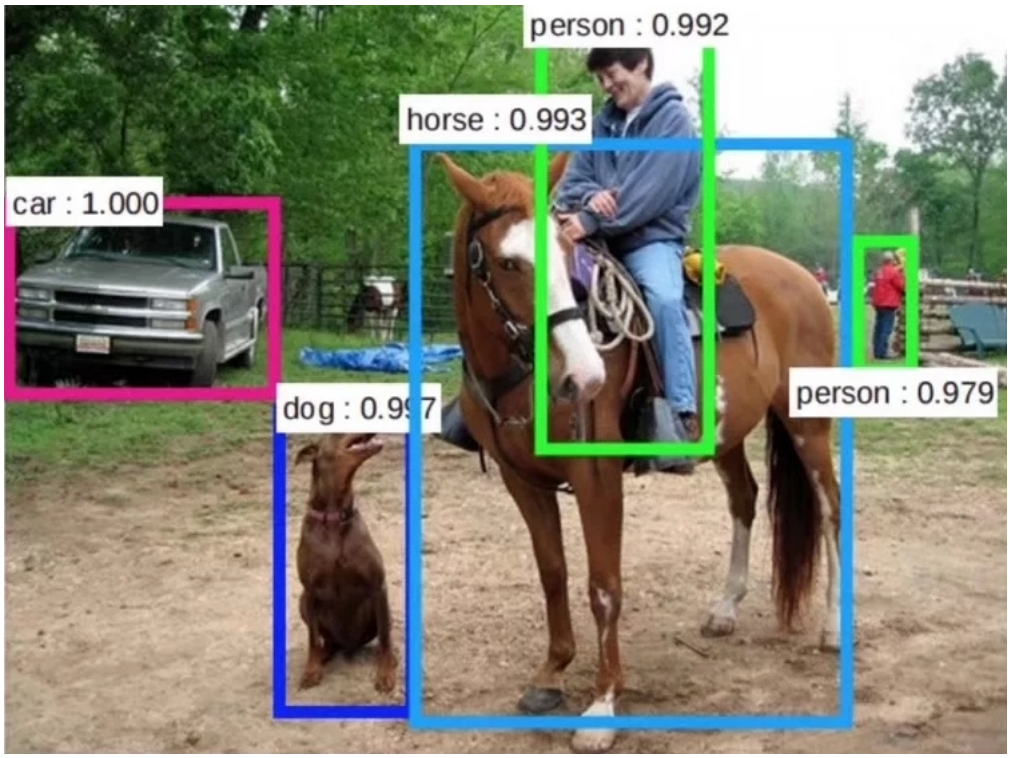
Računalni vid je područje koje nastoji automatizirati proces razumijevanja i interpretiranja slike i videa. Računalni vid pripada u jednu od grana umjetne inteligencije.

Cijelo područje računalnog vida je u ovom desetljeću napravilo veliki iskorak i njegove metode se redovito koriste u današnjoj industriji poput tehnologije autonomnih vozila, sportske analitke, interpretacije teksta, prepoznavanje osoba i tako dalje. Jedna od mogućih primjena se obrađuje u ovom radu, lokalizacija sportaša u stvarnom vremenu.

Cilj lokalizacije je pronalaženje koordinata ciljnog objekta. Neki od najpoznatijih zadataka koji se rješavaju računalnim vidom su: klasifikacija, klasifikacija s lokalizacijom, detekcija objekata, semantička segmentacija, segmentacija instanci.

U prošlosti su se navedeni zadaci rješavali klasičnim metodama strojnog učenja i računalnog vida. Danas ovim područjem dominiraju duboki konvolucijski modeli, koji se u navedenim zadacima mogu mjeriti i s čovjekom. Glavni preokret koji se desio početkom ovog desetljeća je mogućnost korištenja grafičkih kartica koje uvelike ubrzavaju proces učenja te veliki broj kvalitetno označenih podataka. Zbog toga su dotadašnji modeli strojnog učenja postajali sve dublji i sofisticiraniji. Takve modele nazivamo dubokim modelima, a područje je postalo duboko učenje (*engl. Deep learning*). Brojni kvalitetni modeli razvijeni su u okviru natjecanja koje organizira zajednica koja se bavi dubokim učenjem. Primjer jednog takvog natjecanja je "ImageNet Large Scale Visual Recognition Challenge" [2]. U okviru tog natjecanja dobivena je ogromna baza podataka ImageNet s označenim slikama na kojoj su naučeni brojni modeli. Takvi naučeni modeli postali su dijelovi standardnih biblioteka.

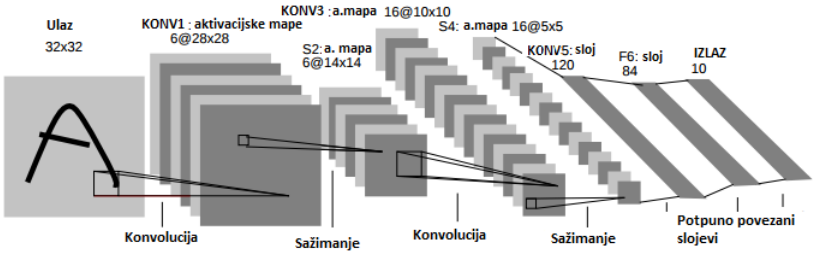
U radu će se opisati općenita arhitektura dubokih modela. Prikazati će se poznate arhitekture detektora koje se danas koriste, te će se opisati dobiveni rezultati u zadatku prepoznavanja sportaša.



Slika 1.1: Primjeri detekcije objekata pomoću Faster R-CNN detektora na skupu podataka PASCAL VOC 2007 [5]. Uz detekcijski okvir predviđen je razred kojem objekt pripada u obliku vjerovatnosti [14].

2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže su temelj velikog broja modela dubokog učenja i postali su *de facto* standard u području računalnog vida. Konvolucijske neuronske mreže (skraćeno CNN) mogu se shvatiti kao nadogradnja nad običnim neuronskim mrežama, te se primjenjuju u situacijama gdje se želi uzeti u obzir topološka organizacija podataka, npr. slika. Osnovni elementi konvolucijskih neuronskih mreža su: konvolucijski sloj, sloj sažimanja i potpuno povezani sloj. Konvolucijski i sloj sažimanja zaduženi su za ekstrakciju značajki iz slike dok je potpuno povezani sloj implementiran kao tradicionalna neuronska mreža s varijabilnom dubinom i klasificira sliku na temelju prethodno ekstrahiranih značajki.



Slika 2.1: Arhitektura LeNet-a, prve konvolucijske neuronske mreže koja klasificirala rukom napisane simbole [12]

Na slici 2.1 prikazana je jedna od prvih implementiranih CNN-ova koja je primijenjena na raspoznavanje dokumenata. Vidljivo je da se ulazna slika postupno smanjuje i povećava volumen kako prolazi kroz niz konvolucijskih slojeva i slojeva sažimanja sve dok se u potpunosti ne vektorizira. Nakon vektorizacije, dobivene značajke se propuštaju kroz potpuno povezani sloj.

Konvolucijske neuronska mreže kombiniraju nekoliko ideja koje su se pokazale izuzetno korisnim:

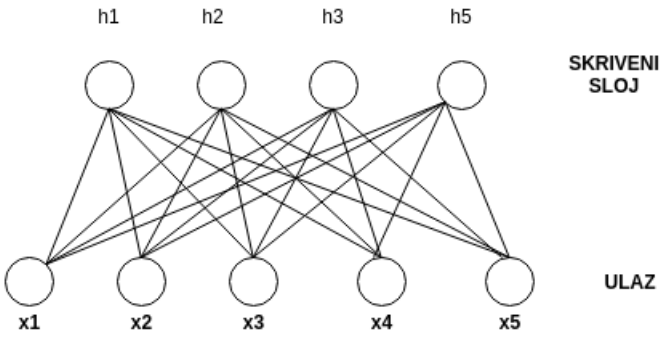
- Lokalne interakcije tj. manja povezanost između slojeva
- Dijeljenje parametara između značajki
- Ekvivarijantnost s obzirom na pomak

Lokalne interakcije smanjile su broj potrebnih parametara koje je potrebno učiti, te tako spriječile problem prenučenosti.

Dijeljenje parametra između značajki omogućilo je učenje niz specijaliziranih filtra čime je omogućeno prepoznavanje različitih elemenata slike. Filtri plićih slojeva prepoznavaju općenite elemente poput rubova i okruglih predmeta. Dok filtri dubljih slojeva prepoznaju specijaliziranije elemente poput očiju, usta, vrata, itd. Time dobivamo prepoznavanje na različitim nivoima apstrakcije.

Funkcija $f(x)$ je ekvivarijantna funkciji $g(x)$ ako vrijedi $f(g(x)) = g(f(x))$. Ako je f konvolucija, a g pomak onda se izlaz pomiče s pomicanjem ulaza. To jest ako pomaknemo objekta na ulazu, tada će se njegove značajke također pomaknuti za isti omjer.

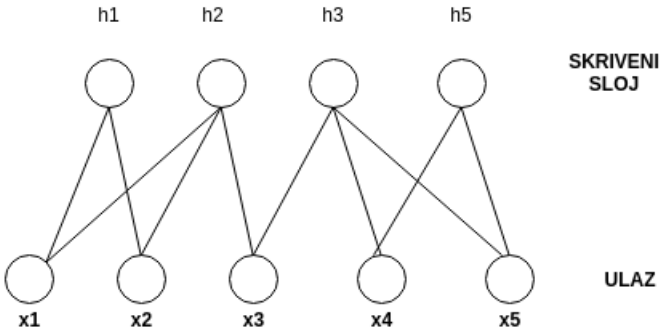
Obične neuronske mreže su u praksi ne koriste pri radu sa slikama i ostalim složenijim podacima. Jedan od bitnih razloga je povećani broj parametara koje takve mreže održavaju.



Slika 2.2: Slika prikazuje lokalnu povezanost neuronskih mreža. Svaki skriveni neuron dijeli parametre sa svim značajkama na ulazu

Na slici 2.2 prikazan je primjer jednog dijela neuronske mreže. Na ulazu mreže imamo vektor značajki $\vec{x}$. Iznad ulaza vidimo neurone skrivenog sloja. Svaki neuron skrivenog sloja povezan je bridom sa značajkama, gdje bridovi predstavljaju parametre u mreži. Takav pristup je neodrživ u slučaju ako je na ulazu slika. Treniranje takvih mreža trajalo bi jako dugo, a u praksi se pokazalo da modeli s velikim brojem parametra imaju svojstvo prenaučenosti, odnosno nemaju sposobnost generalizacije izvan skupa podataka na kojem su učili.

Usporedimo lokalnu povezanost neuronske mreža s lokalnom povezanošću konvolucijskih mreža na slici 2.3. Kod konvolucijskih mreža budući slojevi su povezani s malim brojem značajki. Ovakav pristup uvelike smanjuje složenost mreže, omogućava bržu evaluaciju i time daje izvrsne rezultate u radu s kompleksnim tipovima podataka



Slika 2.3: Slika prikazuje lokalnu povezanost konvolucijskih neuronskih mreža. Neuroni dijele parametre samo s nekoliko značajki na ulazu.

kao što su slike. Time je moguće stvoriti dublje modele koji su se u praksi pokazali kao puno bolji izbor od plićih modela.

2.1. Konvolucijski sloj

Konvolucijski sloj je sloj u kojem se koristi operacija konvolucije, pojam koji dolazi iz teorije obrade signala. Operacija konvolucije realne neprekinute funkcije je definirana sljedećim izrazom:

$$h(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau \tag{2.1}$$

U strojnom učenju koristi se diskretna operacija konvolucije:

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.2}$$

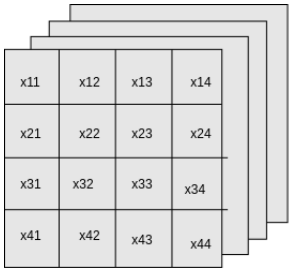
Ulaz u sloj konvolucije su tipično slike, pa se zbog toga koristi diskretna konvolucija po dvodimenzionalnoj slici:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n) \tag{2.3}$$

U kontekstu strojnog učenja:

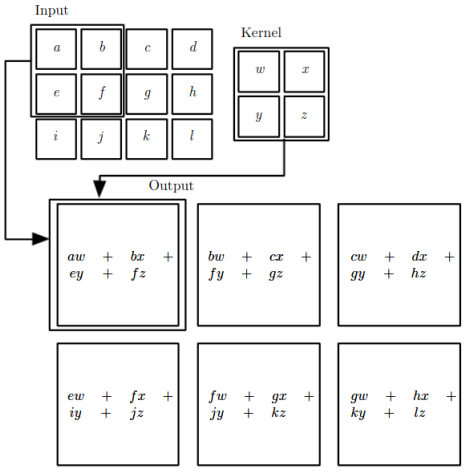
- I je ulaz
- K je jezgra, kernel ili filtar koji posjeduje slobodne parametre koji se mogu učiti
- S je rezultat koji se naziva mapa značajki

U strojnom učenju ulaz je najčešće multi-dimenzionalna matrica tj. tenzor. Tenzor predstavlja višekanalnu sliku nad kojom se primjenjuje operacija konvolucije, a dimenzije tenzora ćemo označavati sa $(width, heigth, depth)$.



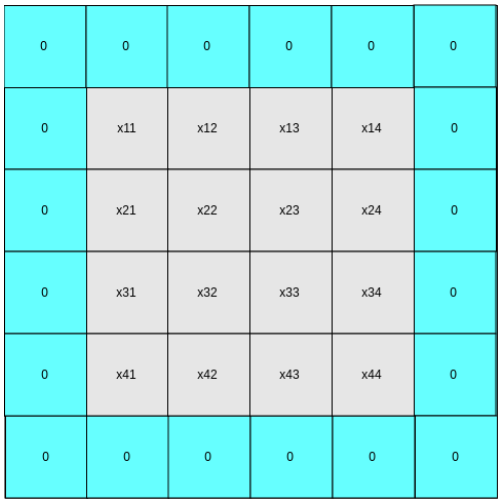
Slika 2.4: Slika prikazuje tenzor dimenzija (4,4,4)

Implementacijski, kernel bi bio tenzor koji "klizi" po ulaznom tenzoru (slici) i računa skalarni produkt. Grafička reprezentacija konvolucije vidljiva je na slici 2.7. Vidimo (2,2) matricu jezgre koja se postavlja nad (3,4) matricom ulaza i uzastopno radi skalarni produkt te "klizi" korakom veličine jedan po ulazu. Općenito ulaz će biti tenzor dimenzija (W, H, D), a jezgra će biti tenzor dimenzija (F, F, D). Bitno je primijetiti da se dubina tenzora jezgre i tenzora ulaza mora poklapati. Ako imamo ovakvu dimenzionalnost ulaznih podataka, konvolucija se primjenjuje i po dubini.



Slika 2.5: Operacija konvolucije nad dvodimenzionalnom ulazu [8]. Filtar dimenzija (2,2) izvodi operaciju konvolucije nad ulazom dimenzije (3,4)

Nekada će biti dobro popuniti ulazni okvir nadopunama (*eng. padding*), kao na slici 2.6. Time je moguće kontrolirati dimenzije izlaza konvolucijskog sloja.



Slika 2.6: Nadopuna nulama *eng. padding* ulaza u konvolucijski sloj. Na slici je korištena nadopuna veličine jedan, P = 1.

Hiperparametri koji se zadaju u konvolucijskom sloju su:

- broj jezgara koji mogu vršiti konvoluciju nad ulazom, **K**
- dimenzija kernela, **F**
- pomak **S**, označava broj piksela za koji se jezgra pomakne u svakom koraku konvolucije
- popunjenje: **P**

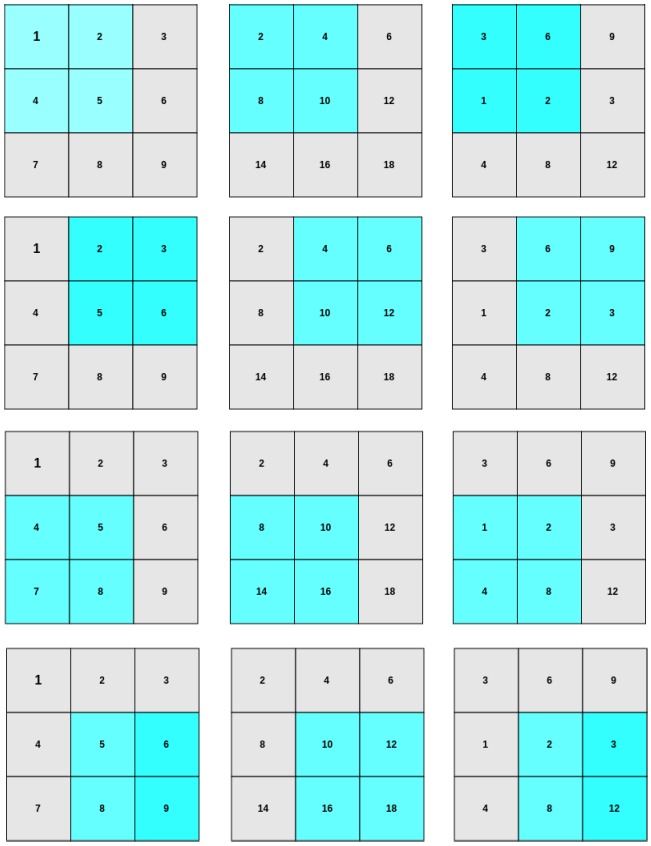
Pretpostavimo da imamo ulaz dimenzija (W_1, H_1, D_1). Taj ulaz dovodimo u konvolucijski sloj s pomakom S, nadopunom P i brojem jezgara K. Dimenzionalnost izlaza (W_2, H_2, D_2) iz konvolucijskog sloja računa se formulama:

$$\begin{pmatrix} W_2 = \frac{W_1 - F + 2P}{S} + 1 \\ H_2 = \frac{H_1 - F + 2P}{S} + 1 \\ D_2 = K \end{pmatrix} \tag{2.4}$$

Ilustrirajmo operaciju konvolucije na jednostavnom primjeru. Recimo da imamo ovakav (2,2,3) filtar:

$$\begin{bmatrix} 5 & 3 \\ 10 & 1 \end{bmatrix} \begin{bmatrix} 6 & 1 \\ 4 & 8 \end{bmatrix} \begin{bmatrix} 2 & 5 \\ 2 & 3 \end{bmatrix}$$

Na slici 2.7 demonstrirane su različite etape prilikom izračuna mape značajki, ona će nam poslužiti kao pomoć u izračunu.



Slika 2.7: Konvolucija filtra (2,2,3) s tenzorom (3,3,3)

U primjeru imamo ulazni tenzor čije su dimenzije $(W = 3, H = 3, D = 3)$ i imamo filtar koji ima dimenzije $(W = 2, H = 2, D = 3)$. Ne koristimo nadopunu i pomak filtra je $S = 1$. Može se vidjeti, ali i provjeriti pomoću formula 2.4, da će izlazni tenzor biti dimenzija $(1, 2, 2)$. Pa pogledajmo izračun element po element za matricu značajki

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

Elementi izlazne matrice značajke računaju se na sljedeći način, gdje je * skalarni produkt:

$$x_{11} = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} 5 & 3 \\ 10 & 1 \end{bmatrix} + \begin{bmatrix} 2 & 4 \\ 8 & 10 \end{bmatrix} * \begin{bmatrix} 6 & 1 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 3 & 6 \\ 1 & 2 \end{bmatrix} * \begin{bmatrix} 2 & 5 \\ 2 & 3 \end{bmatrix}$$

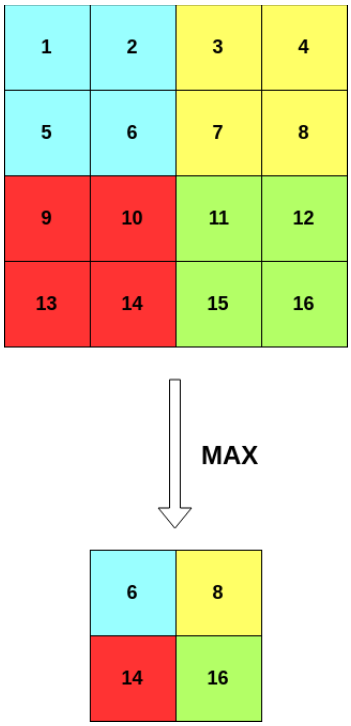
$$x_{12} = \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix} * \begin{bmatrix} 5 & 3 \\ 10 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 6 \\ 10 & 12 \end{bmatrix} * \begin{bmatrix} 6 & 1 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 6 & 9 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} 2 & 5 \\ 2 & 3 \end{bmatrix}$$

$$x_{21} = \begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix} * \begin{bmatrix} 5 & 3 \\ 10 & 1 \end{bmatrix} + \begin{bmatrix} 8 & 10 \\ 14 & 16 \end{bmatrix} * \begin{bmatrix} 6 & 1 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 4 & 8 \end{bmatrix} * \begin{bmatrix} 2 & 5 \\ 2 & 3 \end{bmatrix}$$

$$x_{22} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix} * \begin{bmatrix} 5 & 3 \\ 10 & 1 \end{bmatrix} + \begin{bmatrix} 10 & 12 \\ 16 & 18 \end{bmatrix} * \begin{bmatrix} 6 & 1 \\ 4 & 8 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 8 & 12 \end{bmatrix} * \begin{bmatrix} 2 & 5 \\ 2 & 3 \end{bmatrix}$$

2.2. Sloj sažimanja

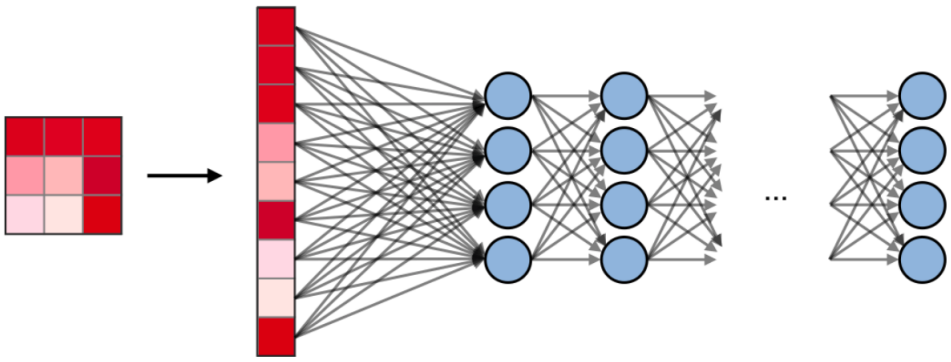
Sloj sažimanja mapira skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu. Cilj sloja sažimanja je reduciranje dimenzionalnosti značajki, te povećanje invarijantnosti na pomak. Prostorno bliske značajke mapiraju se u jednu značajku na izlazu. Ovaj se sloj tipično dodaje nakon konvolucijskog sloja. Kod implementacije ovog sloja uzima se okvir koji, klizi po ulaznim značajkama i izvršava operacija sažimanja. Tipično se kao operacija sažimanja uzima sažimanje maksimalnom vrijednošću ili sažimanje srednjom vrijednošću. Na slici 2.8 dan je primjer sažimanja maksimalnom vrijednošću. Različitim bojama su označene regije nad kojima djeluje operacija max.



Slika 2.8: Primjer operacije sažimanja maximalnom vrijednošću.

2.3. Potpuno povezani sloj

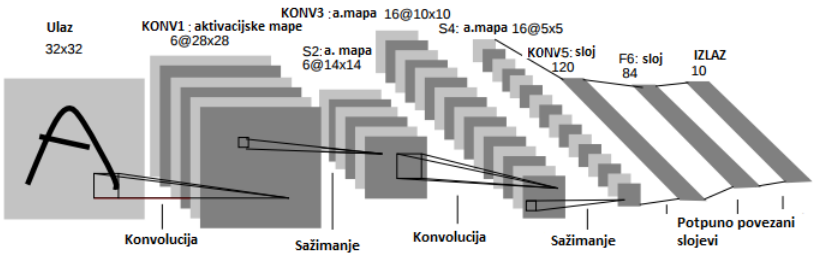
Potpuno povezani sloj je zadnji sloj kod konvencionalnih konvolucijskih neuronskih mreža. Svrha ovog sloja je klasifikacija ili regresija na temelju do tada ekstrahiranih značajki. Djeluje nad vektoriziranim podacima gdje svaka značajka dijeli parametre sa svakim neuronom skrivenog sloja. Ulaz u ovaj sloj su ekstrahirane značajke, a na izlazu se dobiva vektor identične veličine kao i broj klasa. Broj skrivenih slojeva je proizvoljan.



Slika 2.9: Primjer potpuno povezanog sloja. Na ulazu potpuno povezanog sloja dobivamo značajke koje se vektoriziraju i povezuju sa svim neuronima sljedećeg sloja. Dubina potpuno povezanog sloja je proizvoljna, a izlaz je četvero-dimenzionalan.

2.4. Arhitektura konvolucijskih neuronskih mreža

Kombiniranjem prethodno opisanih slojeva dobivaju se različite arhitekture konvolucijskih neuronskih mreža. Opisat će se dvije arhitekture: LeNet i AlexNet arhitekture. Arhitktura prve konvolucijske neuronske mreže LeNet prikazana je na slici 2.10.



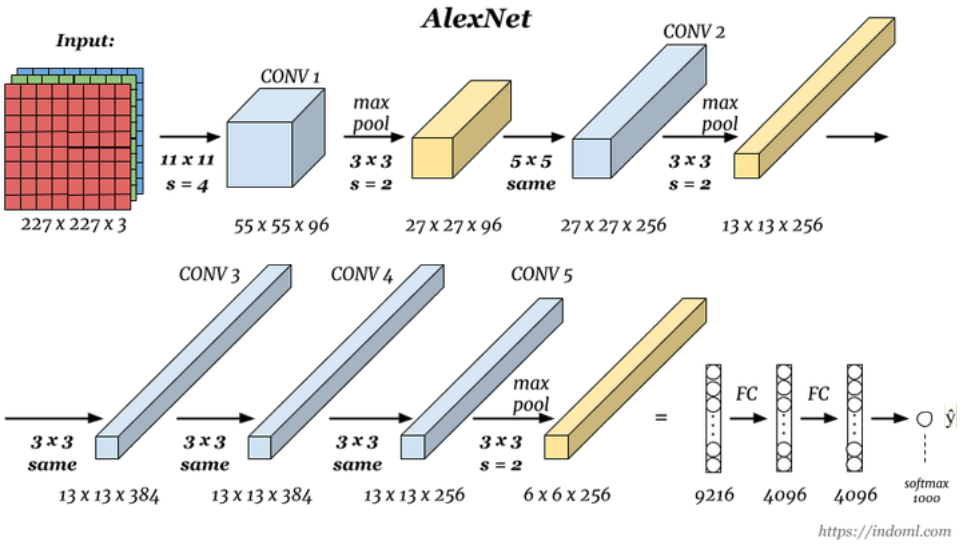
Slika 2.10: Arhitektura LeNet [12]

Na ulazu mreže postavljaju se (32,32,1) slike rukom pisanih simbola. Arhitektura mreže pisana u programskom okviru PyTorch izgledala bi ovako:

```
convNet = nn.Sequential([
    nn.Conv2d(1,6,kernel_size = (5,5)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(2,2),stride = 2),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(2,2),stride = 2),
    nn.Conv2d(16,120,kernel_size=(5,5)),
    nn.ReLU()
    ])
fc = nn.Sequential([
    nn.Linear(120,84),
    nn.ReLU(),
    nn.Linear(84,10),
    nn.LogSoftMax()
])
```

LeNet arhitektura je osmišljena 1998. godine. Do 2012. godine područje računalnog vida je doživjelo stagnaciju sve do pojave AlexNet arhitekture. Novosti u toj arhitekturi su *dropout* sloj, augmentacija podataka, treniranje na GPU kartici i mnogo dublja struktura. Model je treniran 6 sati na GPU kartici, čime je masovno ubrzan postupak treniranja.

Augmentacija podataka je postupak u kojem se generiraju podaci na temelju transformiranih podataka iz skupa za učenje. Time se povećava veličina skupa za učenje, te se povećava sposobnost generalizacije modela. Tipične transformacije koje se primjenjuju prilikom augumentacije slika su rotiranje, skaliranje, rezanje slike, zrcaljenje, itd.



Slika 2.11: Arhitektura AlexNet. Model na ulazu očekuje tenzor dimenzija (227,227,3). Koriste se 5 konvolucijskih slojeva i 3 POOL sloja. Na kraju modela imamo potpuno povezane slojeve koji klasificiraju značajke u 1000 kategorija

Dropout sloj služi za prevenciju prenaučenosti modela. Izradom sve kompleksnijih modela, postoji rizik da model neće imati sposobnost generalizacije. Droupout je dizajniran tako da prilikom treniranja modela, svaki neuron ima neku vjerovatnost p da bude aktiviran tj. da sudjeluje u unaprijednom prolazu. Implementacijski mapa značajki se množi binarnom maskom gdje je vjerovatnost da pojedini element poprimi vrijednost jedan jednaka p. Također pospješuje generalizaciju modela.

2.5. Optimizacijski postupci

Algoritmima optimizacije pokušavamo pronaći optimalne parametre $\vec{\theta}$ koji minimiziraju neku mjeru performanse $J(\vec{\theta})$ nad generalizirajućoj distribuciji podataka $p(\vec{x}, y)$.

U strojnom učenju ne poznajemo distribuciju $p(\vec{x}, y)$, nego poznajemo empirijsku distribuciju podataka $\hat{p}(\vec{x}, y)$, definiranu nad našim uzorkom za učenje. Nad uzorkom za učenje veličine m definiramo procjenu generalizirane mjere performanse kao :

$$\hat{J}(\vec{\theta}) = E_{(\vec{x},y)\sim\hat{p}(\vec{x},y)}[L(f(\vec{x}, \vec{\theta}), y)] = \frac{1}{m}\sum_{i=1}^{m} L(f(\vec{x}_i, \vec{\theta}), y_i) \tag{2.5}$$

,gdje je $L(f(\vec{x}_i, \vec{\theta}), y_i)$ funkcija gubitka nad jednim primjerkom za učenje, a procjenitelj se naziva empirijski rizik ili ukupni gubitak nad uzorkom za učenje.

U praksi postoji jako puno funkcija gubitka i odabiru se na temelju zadataka koji

model pokušava riješiti. U problemu regresije tipično se kao funkcija gubitka koristi srednje kvadratno odstupanje (*engl. mean squared error loss*), definirano kao:

$$L(\vec{\theta}) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

.

Prilikom klasifikacije ulaza $\vec{x}$ u C klasa tipično se koristi negativna log-izglednost:

$$L(\vec{p}, C) = -log(p_c) \tag{2.7}$$

, gdje je $\vec{p}$ izlaz softmax sloja. Jedna komponenta softmax sloja opisana je sa:

$$p_j = softmax(j, \vec{s}) = \frac{e^{s_j}}{\sum_i e^{s_i}} \tag{2.8}$$

Pri postupku minimizacije empirijskog rizika nadamo se da ćemo također minimizirati generalizirani rizik $J(\vec{\theta})$ iz izvorne distribucije podataka. Pri optimizaciji empirijskog rizika postoji opasnost od prenaučenosti (*engl. overfitting*). Model je prenaučen u trenutku kada ima malen empirijski rizik nad uzorkom za učenje, ali nema dobra generalizirajuća svojstva, odnosno mjera performanse $J(\vec{\theta})$ nad izvornom distribucijom podataka je visoka. Prenaučenost se tipično događa kada imamo nepotrebno kompleksan model.

Većina današnjih optimizacijskih postupaka u strojnom učenju se temelji na gradijentnom spustu. Cilj gradijentnog spusta je kretati se u suprotnom smjeru od gradijenta kako bi konvergirali i našli lokalni optimum. Postupak ažuriranja parametara u postupku gradijentnog spusta bio bi sljedeći:

$$\theta_{k+1} = \theta_k - \alpha \bigtriangledown \hat{J}(\theta_k) = \theta_k - \alpha \bigtriangledown \frac{1}{m}\sum_{i=1}^{m} L(f(\vec{x}_i, \vec{\theta_k}), y_i) \tag{2.9}$$

, gdje je parametar α stopa učenja (*engl. learning rate*). Da bismo dobili egzaktni gradijent funkcije $\hat{J}(\theta_k)$, moramo izračunati gradijent srednje vrijednost funkcije gubitka nad cijelim uzorkom. Duboki modeli se treniraju nad ogromnim uzorkom za učenje i računanje gornjeg gradijenta je vremenski vrlo zahtjevan postupak. Općenito, postoji nekoliko varijanti optimizacije ovisno o broju primjera koji se koristi za računanje gradijenta,a to su:

- Grupni optimizacijski postupci
- Stohastički optimizacijski postupci
- Optimizacijski postupci temeljeni na mini-grupama

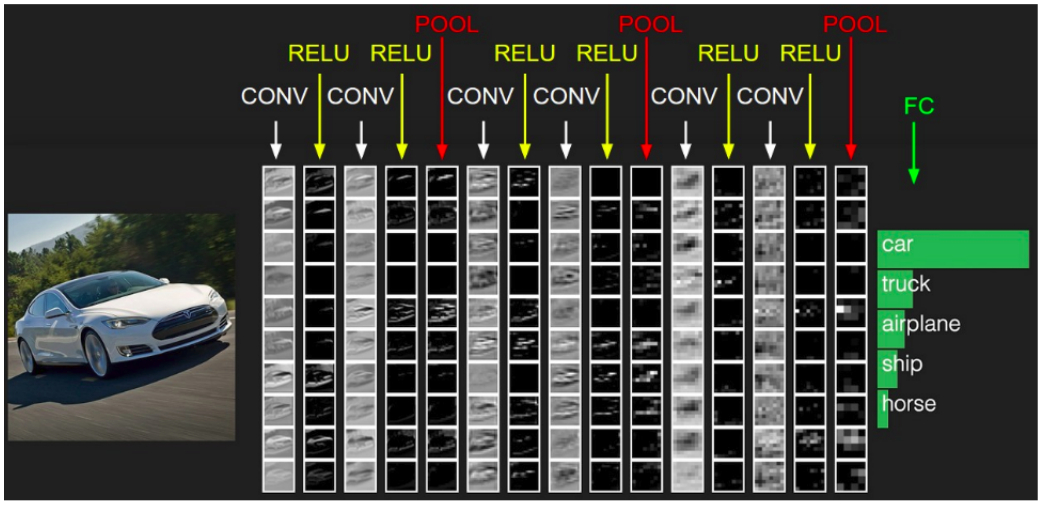
Grupni optimizacijski postupci računaju egzaktni gradijent, računajući gradijent na temelju cijelog uzorka. Ažuriranje parametara na temelju egzaktnog gradijenta empirijskog rizika $\hat{J}(\theta_k)$ je opisano jednadžbom 2.9
Stohastički optimizacijski postupci računaju gradijent na temelju jednog nasumično odabranog primjerka, pa ažuiranje parametara postaje:

$$\theta_{k+1} = \theta_k - \alpha \bigtriangledown L(f(\vec{x}_{random}, \vec{\theta_k}), y_{random}) \tag{2.10}$$

Time se dobiva aproksimacija egzaktnog gradijenta i uvelike se ubrzava postupak konvergencije cijelog postupka, te se smanjuje vjerovatnost zapinjanja u lokalnom minimumu.
Optimizacijski postupak temeljen na mini-grupama koristi nasumično odabrani podskup primjera iz cijelog uzorka za učenje kako bi izračunao aproksimaciju gradijenta empirijskog rizika. U praksi se ova varijanta najčešće koristi jer dobivamo najbolje od obje krajnosti.

3. Metode računalnog vida

U ovoj sekciji biti će opisani neki od zadataka na koje se može naići u literaturi. Vjerojatno prva i osnova metoda jest klasifikacija slika. Zadatak te metode je predvidjeti razred u koji slika pripada za neviđene primjere slika.

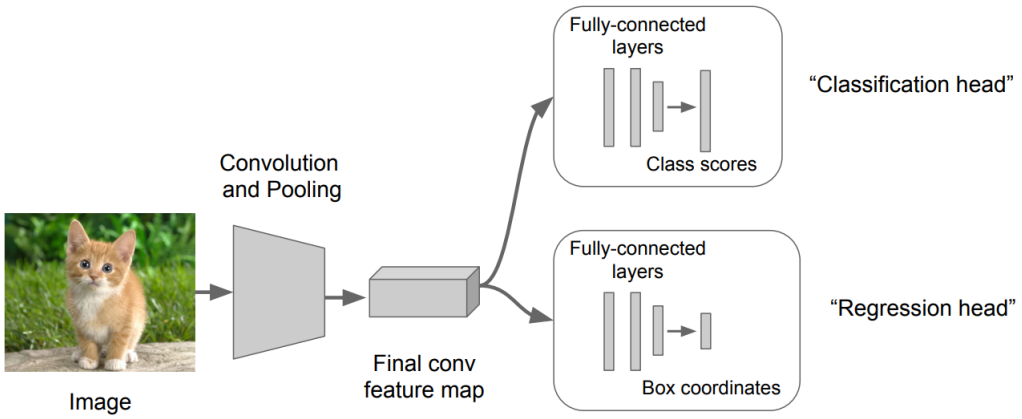


Slika 3.1: Klasifikacija auta u pet različita razreda [15].

U trenutku pisanja rada najbolji današnji model u klasifikaciji ImageNet slika [2] predviđa točan razred(*engl. Top 1 Accuracy*) ulazne slike u 84.4 posto slučajeva, a točan razred se nalazi u 5 najboljih predikcija u 97.1 posto slučajeva (*engl. Top 5 Accuracy*) [3].

Lokalizacija je postupak u kojem algoritam nastoji predvidjeti poziciju fiksnog broja razreda na ulaznoj slici i također predvidjeti njegovu klasu. Može se pokazati da se arhitektura dubokih modela za lokalizaciju ne razlikuje previše od onih za klasifikaciju. Dovoljno je u već postojeću arhitekturu konvolucijskih mreža dodati potpuno povezani sloj koji će na izlazu imati 4 neurona koji će predviđati koordinate objekta. Takav bi model imao jedan klasifikator i jedan regresor za predviđanje koordinata detekcijskih okvira. Na slici 3.2 imamo primjer jednog takvog lokalizatora.

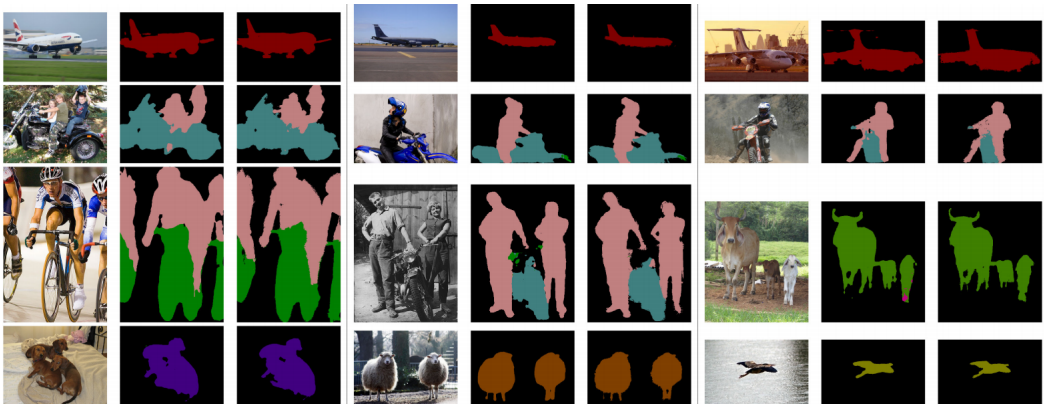
Kod modela koji je sposoban predvidjeti klasu i lokalizirati objekt, problem dolazi



Slika 3.2: Arhitektura za lokalizaciju i klasifikaciju [15]. Na kraju modela postoje dvije "glave" sa različitim funkcijama. Jedna služi za klasifikaciju, a jedna za regresiju koordinata. Postojeća arhitektura neće moći predvidjeti okvir za više od jednog objekta

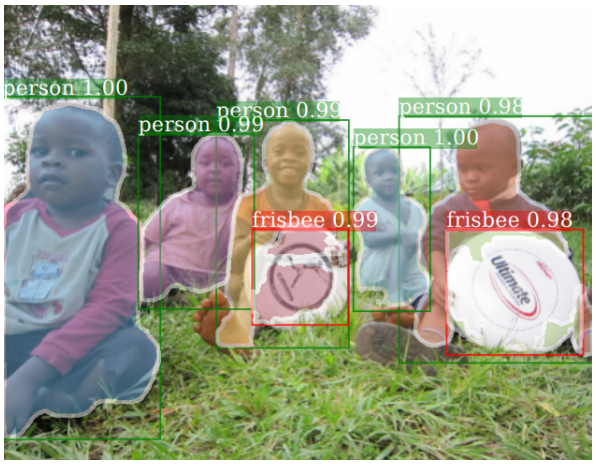
u trenutku kada imamo više objekata na ulaznoj slici ili ako uopće nemamo objekt. Ovakav problem zahtijeva kompleksniju arhitekturu i metode. Naivan pristup koristio bi klizajući "prozor" koji bi uzastopno izrezivao komadiće ulazne slike i njih postavljao na ulaz dubokog modela. Ako bi model prepoznao objekt, na regiju u kojoj se nalazi klizajući prozor postavili bi detekcijski okvir. Mana tog pristupa bila bi velika vremenska složenost i mogućnost da klizajući prozor ne zahvati najbolje objekt. Mjera performanse koja se koristi u zadatku detekcije objekta jest AP mjera (*engl. Avarage precision*). Definira se kao površina ispod krivulje odziva i preciznosti (*engl. RP curve*). Mjera je detaljnije opisana u nastavku rada.

Zadatak semantičke segmentacije *engl. Semantic segmentation* je klasificirati svaki piksel s obzirom na pripadnost kategoriji. Rezultat semantičke segmentacije vidljiv je na slici 3.3. Mjera performansa koja se koristi u ovom zadatku jest mIoU (*engl. mean Intersection-Over-Union*) metrika. IoU je udio presjeka između sematičke maske i oznake te unije maske i oznake. Za svaku klasu izračunali bismo IoU između predviđene segmentacije i oznake, tada bismo izračunali prosjek po svim razredima.



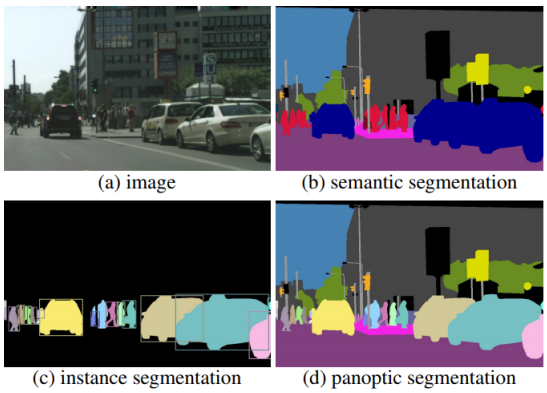
Slika 3.3: Primjeri rezultata semantičke segmentacije na PASCAL VOC-u 2012 korištenjem DeepLab modela. Svakom razredu je pridružena različito obojena maska [3]

Semantička segmentacija na razini instanci (*engl. Instance segmentation*) predviđa oznake za svaki piksel pri čemu su instance unutar jednog razreda različito označene. Semantička segmentacija na razini instanci prikazan je na slici 3.4.



Slika 3.4: Primjer rezultata semantičke segmentacije na razini instanci uporabom Mask R-CNN modela. Slika prikazuje 5 osoba za koje je točno predviđena kategorija te je svakoj osobi pridijeljena različito obojena maska. Semantička segmentacija ne bi razlikovala osobe, nego bi svakoj pridijelila masku iste boje [10].

Panoptička segmentacija (*engl. panoptic segmentation*) je metoda koja kombinira semantičku segmentaciju i segmentaciju na razini instanci. Svakom pikselu slike potrebno je pridijeliti semantičku oznaku i oznaku instance. Na slici 3.5 najbolje se uočava razlika.



Slika 3.5: Slika prikazuje razlike između panoptičke, semantičke i segmentacije instanci. Semantička segmentacija pridjeljuje svakom pikselu oznaku kategorije kojoj taj piksel pripada. Segmentacija na razini instanci pridjeljuje svakom objektu kategoriju i različitu masku. Panoptička segmentacija kombinira te dvije metode i **svakom** pikselu u sceni pridjeljuje vlastitu masku [11].

4. Detekcija objekta

4.1. Arhitekture modela za detekciju objekta

U ovoj sekciji opisat će se poznate arhitekture modela za detekciju objekata. Kronološki će se opisati razvoj detektora iz R-CNN familije, a to su:

- R-CNN
- Fast R-CNN
- Faster R-CNN
- Mask R-CNN

Svaka sljedeća inačica donosi niz poboljšanja u vidu brzine učenja, predikcije i točnosti. Mask R-CNN donosi i novu funkcionalnost: semantičku segmentaciju instanci. Za razliku od semantičke segmentacije objekata, segmentacija instanci razlikuje objekte u istoj kategoriji.

Slika 4.1: Rezultati Mask R-CNN modela na COCO uzorku. Uz semantičke maske prikazani su detekcijski okviri, kategorije i sigurnost u pripadnost objekta u klasificirani razred [9]

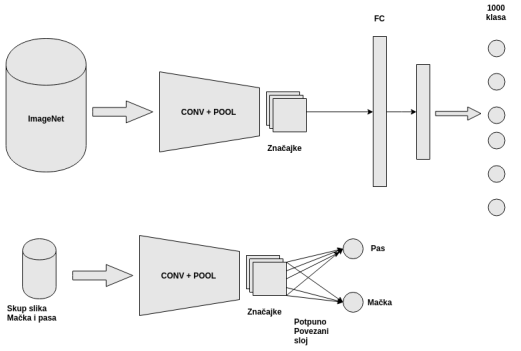
4.2. R-CNN

R-CNN arhitektura je jedna od prvih modela dubokog učenja koja je bila namijenjena detekciji objekta. Ideja je bila svesti problem detekcije objekta na problem klasifikacije.

Prvi korak prilikom stvaranja R-CNN detektora jest preuzimanje pretreniranog konvolucijskog modela (engl. Backbone model) nad nekim velikim skupom podataka. Tada se radi postupak prenesenog učenja (*engl. Transfer learning*). Preneseno učenje je općeniti postupak kojim dobiveno znanje iz jednog područja iskorištavamo u drugom sličnom području. Ako posjedujemo model koji uspješno klasificira mačke, možemo taj isti model iskoristiti za klasifikaciju psa. Preuzeti pretrenirani model je tipično dizajniran za klasifikaciju slika u veliki broj razreda, dok naš skup podataka sadrži nekoliko razreda. Cilj nam je prilagoditi pretrenirani model tako da prepoznaje isključivo razrede koje se nalaze u našem skupu podataka. Postoje dvije metode prenesenog učenja kojim prilagođujemo pretrenirani model našem skupu podataka.

Kod prve metode uklonili bi potpuno povezane slojeve koje služe za klasifikaciju. Nakon toga bismo dodali potpuno povezane slojeve koji su prilagođeni za naš skup podataka. To jest klasificiraju slike u onoliko kategorija koliko ima naš uzorak za učenje. Tada bismo naučili parametre isključivo tog posljednjeg sloja. Ova metoda prilagodbe se tipično izvodi u slučaju kada imamo malen i sličan skup podataka originalnom skupu.

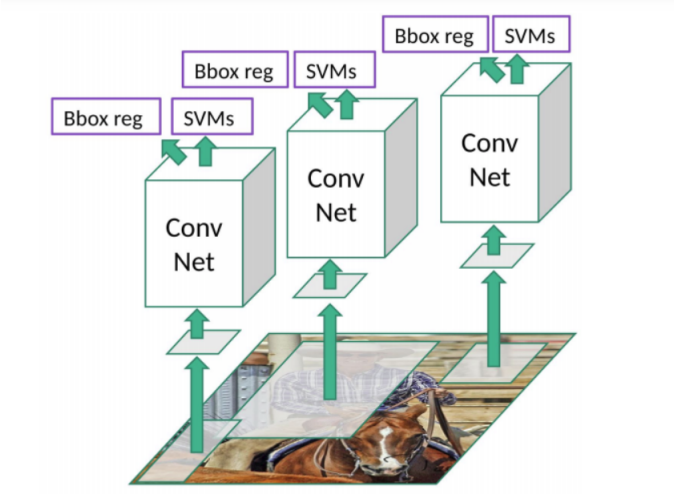
Druga metoda ne bi samo zamijenila i naučila klasifikator pretreniranog modela, nego bismo također prilagodili i naučili ostale slojeve mreže. Ova metoda se koristi kada imamo skup podataka koji nije sličan originalnom skupu.



Slika 4.2: Prilagodba pretreniranog modela na ImageNet-u, skupu označenih slika mačka i psa. Originalnom modelu je zamijenjen klasifikator za 1000 razreda s klasifikatorom za 2 razreda

Nakon prilagodbe modela našem skupu podataka, morali bismo za svaku sliku odrediti neke regije interesa (*engl. Region of Interest(RoI)*), smanjiti ih na ulazne dimenzije modela i spremiti ih negdje na disk. U originalnom radu, korišten je algoritam selektivne pretrage (*engl. Selective Search*) koji je po slici davao 2000 regija interesa. Nakon toga bi se za svaku klasu naučio binarni stroj s potpornim vektorima (SVM) koji klasificira sliku kao objekt ili okolinu. Također, algoritam bi predviđao 4 vrijednosti koje bi predstavljale detekcijski okvir objekta. Koordinate okvira naučile bi se uz pomoć linearne regresije.

Mana ovakvog detektora je dugo učenje detektora, kompliciran postupak učenja te se model ne može koristiti za detekciju u stvarnom vremenu jer je potrebno oko 47 sekundi za predikciju.



Slika 4.3: Prikaz arhitekture R-CNN. Unaprijedni prolaz slike sastoji se od: uzorkovanja dijelova slike(RoI-a), prilagodba RoI-a ulazu modela te određivanje razreda i koordinata objekta na RoI-u. [7]

4.3. Fast R-CNN

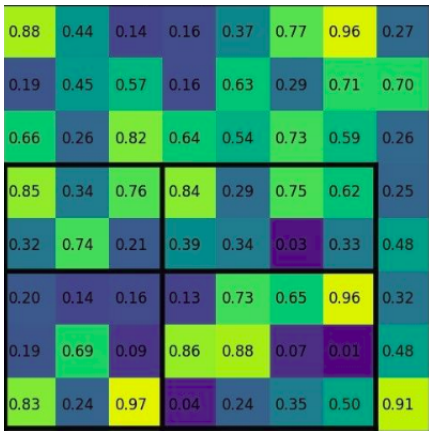
Problem kod R-CNN detektora je taj što se za svaku regiju interesa mora izvesti unaprijedni prolaz kroz model. U slučaju algoritma selektivne pretrage imali bismo oko 2000 unaprijednih prolaza samo za jednu sliku. Fast R-CNN rješava taj problem na način da uzorkuje regije interesa na izlazu pretreniranog modela, tj. uzorkuje mape značajki. Time, umjesto 2000 unaprijednih prolaza imamo samo jedan prolaz po slici. Fast R-CNN također uvodi jedan dodatni sloj u arhitekturu - sloj sažimanja regije interesa(*ROI pooling layer*). Potpuno povezani sloj zahtijeva određene dimenzije značajki

na ulazu, što predstavlja problem jer uzorkovane regije interesa mogu biti varijabilnih dimenzija. Sloj sažimanja regije interesa sažima dimenzije RoI-a u dimenzionalnost kakvu zahtijeva potpuno povezani sloj.

Rad RoI Pool sloja biti će objašnjen uz pomoć slike 4.4. Pretpostavimo da je izlaz RoI Pool sloja matrica dimenzija $(2, 2)$. Nakon unaprijednog prolaza slike dobivamo mapu značajki dimenzija $(8, 8)$. Nakon uzorkovanja regija interesa dobivamo RoI definiran koordinatama $(0, 4)$, $(6, 0)$(gornji lijevi ugao i donji desni ugao kvadrata). Pošto je izlaz RoI Pool sloja $(2, 2)$, podijelimo RoI u 4 regije. Regije ne trebaju biti jednakih veličina. Nad svakom regijom izvedemo neku sažimajuću operaciju, npr. uzimamo najveću vrijednost iz svake regije ili izračunamo srednju vrijednost. Izlaz RoI Pool sloja uzimanjem maksimalne vrijednosti iz svake regije tada će biti:

$$\begin{bmatrix} 0.85 & 0.84 \\ 0.97 & 0.96 \end{bmatrix}$$

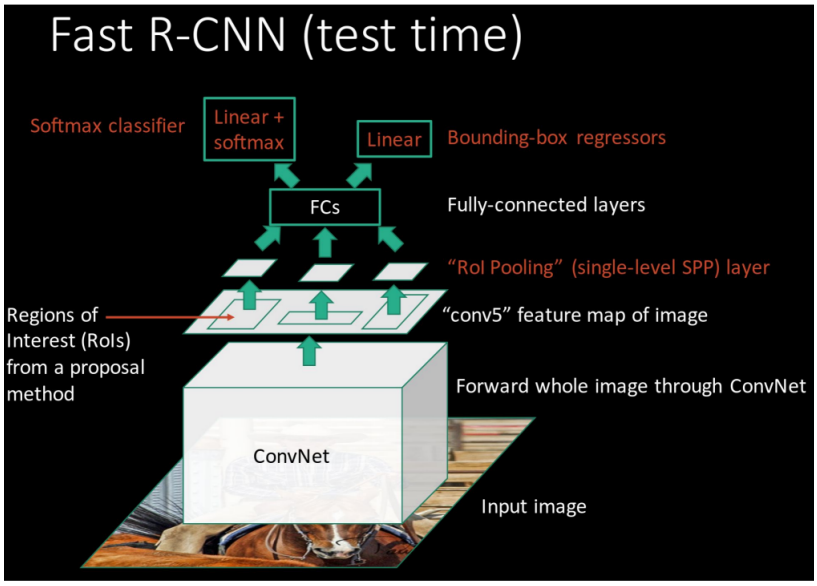
. Takav izlaz može se proslijediti potpuno povezanom sloju koji će klasificirati te predvidjeti detekcijski okvir objekta na slici.



Slika 4.4: Slika prikazuje matricu zančajki dimenzija (8,8) i (5,7) RoI. Izlaz RoI Pool sloja je matrica dimenzija (2,2) [1]

Slika 4.5 prikazuje cjelovitu arhitekturu Fast R-CNN detektora. Unaprijedni prolaz se sastoji od:

- Unaprijednog prolaza slike kroz pretrenirani model.
- Stvaranje regija interesa nad mapama značajki uz pomoć algoritma selektivne pretrage (*engl. Selective search*).
- Sažimanje RoI-a na definiranu dimenzionalnost uz pomoć RoI Pool sloja.
- Klasifikacija i predikcija koordinata objekta



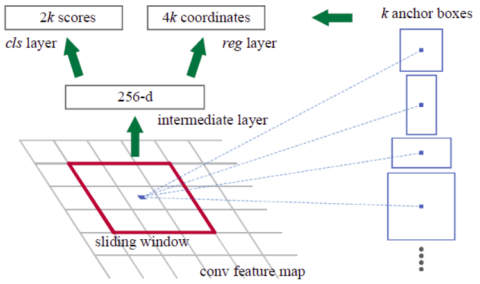
Slika 4.5: Fast R-CNN [6]

4.4. Faster R-CNN

Analizom performansi Fast R-CNN arhitekture uviđa se da predviđanje regija interesa algoritmom selektivne pretrage (*engl. Selective search algorithm*) dominira u cjelokupnom postupku treniranja i predviđanja. Faster R-CNN zamjenjuje algoritam selektivne pretrage s mrežom za predviđanje regija interesa (*engl. Region proposal network*) ili RPN mreža. RPN kao ulaz uzima mape značajki bilo kakvih dimenzija, a kao izlaz daje listu RoI-a s mjerom sigurnosti. Mjera sigurnosti je vjerovatnost da se objekt nalazi u zadanom RoI-u.

Da bismo generirali regije interesa, RPN mreža uzima okvir dimenzija (n, n) i s njim klizi po mapama značajki. Oko središnje lokacije klizećeg prozora ili sidra (*engl. Anchor*) generiramo k novih okvira (*engl. Anchor boxes*). Autori[14] su oko svakog sidra generirali k novih okvira s različitim dimenzijama i formatima prikaza(*engl. aspect ratio*).

Na Slici 4.6 prikazan je postupak kreiranja okvira oko sidra u Faster R-CNN arhitekturi. Oko sidra klizećeg okvira dimenzija $(3, 3)$ generiramo k novih okvira, različitih dimenzija. Generirani okviri se preslikavaju na 256 dimenzionalne značajke, koji su pak istovremeno povezane s dva potpuno povezana sloja ispred sebe: *cls* slojem i *reg* slojem. *cls* sloj sadrži $(2k)$ neurona i svaki okvir oko sidra klasificira kao objekt ili kao pozadinu. Drugi sloj sadrži $4k$ neurona i predviđa koordinate objekta za svaki okvir oko sidra.



Slika 4.6: mreža za predviđanje RoI-a. Oko svakog sidra generiramo k novih okvira. Okvir su različitih dimenzija i formata prikaza. Značajke okvira se preslikavaju na 256 dimenzionalne značajke prolaskom kroz konvolucijski sloj. Dalje se na temelju tih značajki određuje koordinate objekta i sigurnost u postojanje objekta unutar okvira [6]

Prije svega, RPN je potrebno naučiti da predviđa regije koje sadrže objekte. Regije koje sadrže objekte prosljeđuju se dalje na završnu klasifikaciju i regresiju koordinata. Na početku, potrebno je svakoj predikciji okvira oko sidra pridijeliti binarnu oznaku na temelju toga dali se u njoj nalazi objekt ili pozadina. Autori izvornog rada[14] pridijelili su oznake predikcijama na temelju sljedeće jednadžbe:

$$IoU(anchorBox, gtBox) = \begin{cases} > 0.7 = objekt \\ < 0.3 = pozadina \end{cases} \tag{4.1}$$

U jednadžbi 4.1, anchorBox predstavlja okvir oko sidra, a gtBox je oznaka. Predikcije za koje ne vrijedi niti jedan slučaj u 4.1 ne dobivaju oznaku i ne doprinose ukupnom gubitku prilikom učenja. Funkcija gubitka jedne slike definirana je kao:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{4.2}$$

Opis članova u formuli 4.2 je sljedeći:

- i = indeks predikcije u mini-grupi
- p_i = vjerovatnost da je i-ta predikcija objekt
- p_i^* = oznaka klase i-tog okvira s vrijednostima 0,1
- t_i = koordinate i-tog predviđenog okvira
- t_i^* = koordinate i-te oznake
- L_{cls} = klasifikacijska pogreška izražena kao binarni log gubitak
- L_{reg} = pogreška regresije
- N_{cls} = predstavlja veličinu mini-grupe

– N_{reg} = broj sidra

– λ = parametar koji određuje težinu ukupnog gubitka regresije u mini-grupi

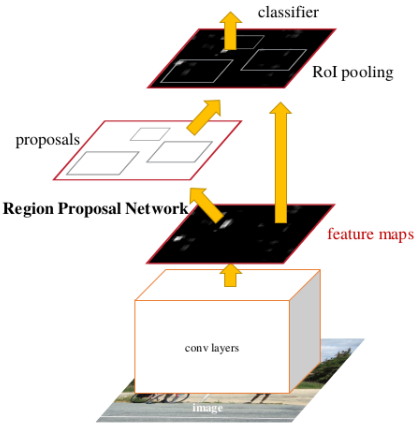
Jednadžba 4.2 kaže da je ukupni gubitak mini-grupe jednak prosječnom gubitku klasifikacije i regresije koordinata.

RPN se može optimizirati uz pomoć stohastičkog gradijentnog spusta. U [14] mini-grupe su dobivene uzorkovanjem 256 pozitivnih i 256 negativnih predikcija jedne slike.

Na slici 4.7 prikazan je tok podataka unutar Faster R-CNN detektora. Nakon naučenog RPN sloja, oznake i okviri sa najvećim vjerovatnostima u postojanje objekta prosljeđuju se RoI Pool sloju. Nakon toga značajke se mapiraju na potpuno povezani sloj, koji dalje vrši klasifikaciju i regresiju. Te slojeve je također potrebno naučiti. Funkcija gubitka u tom slučaju bi bila jednaka prosječnom gubitku regresije i klasifikacije u N klasa. Ukupni gubitak Faster R-CNN detektora je:

$$L = L_{class}^{rpn} + L_{reg}^{rpn} + L_{class} + L_{reg} \tag{4.3}$$

Dakle, Faster RCNN detektor ima 4 funkcije gubitka: funkciju gubitka predlaganja RoI-a, gubitak binarne klasifikacije RoI-a (objekt vs. pozadina), gubitak klasifikacije u N klasa te gubitak regresije objekta unutar RoI-a. Autori rada [14] prvo su naučili RPN modul, pa su potom koristeći RoI-e iz tog modula naučili dio modela odgovornog za klasifikaciju objekta i regresiju koordinata.



Slika 4.7: Faster R-CNN arhitektura [14]. RPN nad mapama značajki pretreniranog modela predviđa koordinate RoI-a. RoI-i se sažimaju u RoI Pool sloju te se prosljeđuju na klasifikaciju i regresiju koordinata.

4.5. Mask R-CNN

Mask R-CNN arhitektura unaprijeđuje Faster R-CNN arhitekturu i dodaje dvije značajne promijene. Zamjenjuje sloj sažimanja regije interesa (*RoI pooling layer*) sa poravnatim sažimanjem regije interesa (*RoI Align*) i dodaje novu granu na potpuno povezane slojeve zaduženu za predikciju semantičke maske objekta.

Uporabom RoI Pool sloja, maske objekata nisu dobro poklapale predviđene objekte. Problem je prikazan na slici 4.8. Pretpostavimo da na ulazu RoI Pool sloja imamo RoI dimenzija $(5, 7)$, a izlaz RoI Pool sloja treba biti matrica dimenzija $(2, 2)$. Zbog toga što na ulazu možemo dobiti RoI različitih dimenzija ne možemo uvijek podijeliti RoI na regije jednakih dimenzija. Ovaj slučaj je već objašnjen i prikazan je na slici 4.4. RoI Align sloj dijeli RoI na regije jednakih veličina. Na slici 4.8 dijelimo RoI na način da podijelimo dimenzije RoI-a s dimenzijama traženog izlaza:

$$bin_W = \frac{roi_W}{output_W} = \frac{7}{2} = 3.5 \tag{4.4}$$

$$bin_H = \frac{roi_H}{output_H} = \frac{5}{2} = 2.5 \tag{4.5}$$

, gdje su (bin_W, bin_H) dužina širine i visine regije unutar RoI-a, (roi_W, roi_H) širina i visina RoI-a, a $(output_W, output_H)$ dužina širine i visine izlaza RoI Align sloja. Svaku regiju podijelimo na 4 subregije kao na trećoj slici i provodimo bilinearnu interpolaciju da bismo dobili jedinstvenu vrijednost za svaku regiju, kao na četvrtoj slici. Tada konačno možemo izvesti neku sažimajuću operaciju nad svakom regijom i dobiti željeni izlaz.

Bilinearna interpolacija je postupak kojim izračunavamo vrijednosti u nepoznatim točkama, pritom koristeći poznate uzorke i koristeći linearnu interpolaciju. Pretpostavimo da je potrebno izračunati vrijednost funkcije $f(x, y)$ u točki $P = (x, y)$ prikazanu na slici 4.9. Najprije izračunavamo interpoliranu vrijednost u točki $R_1 = (x, y_1)$ i točki $R_2 = (x, y_2)$, koristeći linearnu interpolaciju:

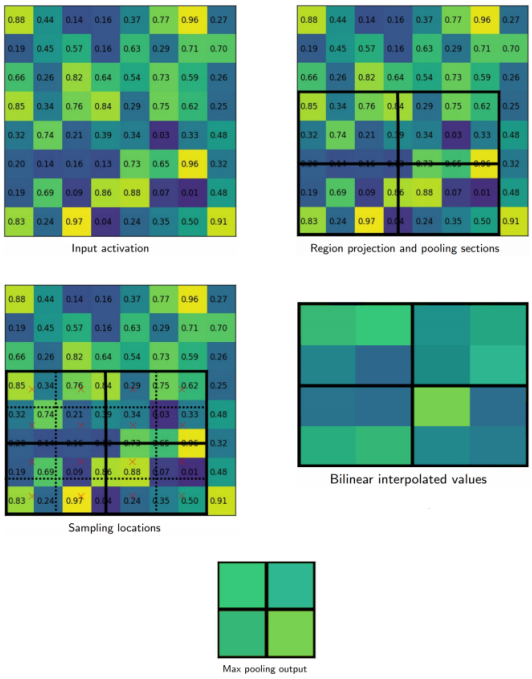
$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{4.6}$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{4.7}$$

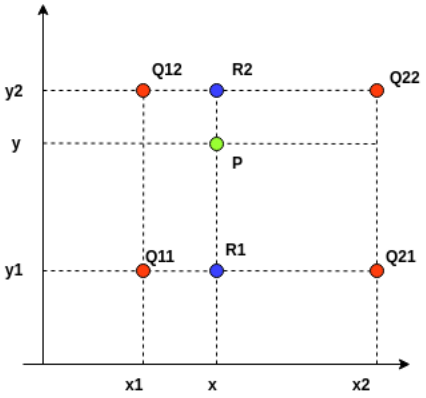
Potom bismo opet napravili linearnu interpolaciju između vrijednosti u točkama R_1 i R_2. Tako bismo dobili procjenu vrijednosti funkcije u točki P.

Tijekom učenja, funkcija gubitka nad uzorkovanim regijama interesa jednaka je:

$$L = L_{cls} + L_{box} + L_{mask} \tag{4.8}$$



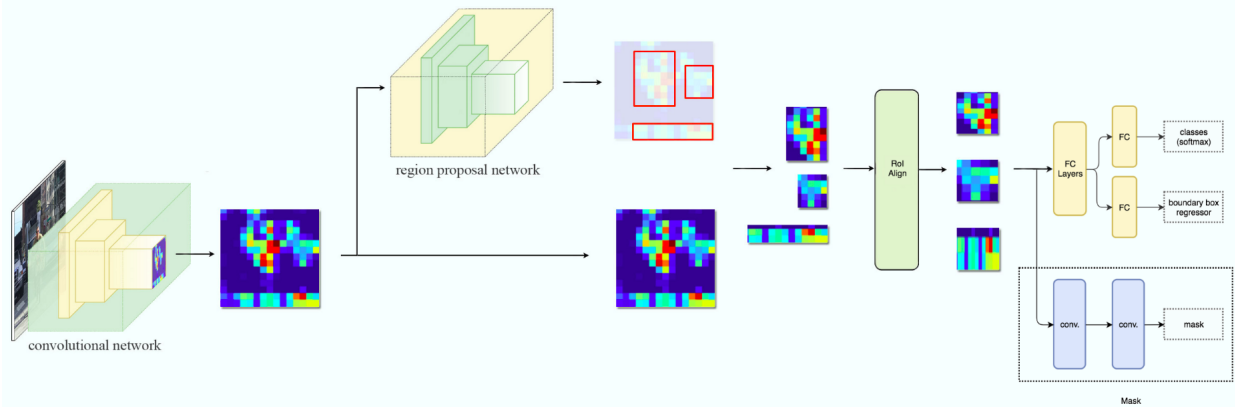
Slika 4.8: Slika prikazuje proceduru pri izračunu RoI Align sloja. Nakon podijele RoI-a u regije jednakih dimenzija, u svakoj regiji provodimo bilinearnu interpolaciju, te provodimo sažimajuću operaciju nad regijama.



Slika 4.9: Izračunavanje vrijednosti funkcije $f(x, y)$ u točki P koristeći bilinearnu interpolaciju između uzorka $(Q_{11}, Q_{12}, Q_{21}, Q_{22})$

,odnosno gubitak na jednom primjeru za učenje jednak je zbroju gubitku klasifikacije, regresije koordinata okvira i regresije maske.

Na slici 4.10 vidljiva je cjelokupna arhitektura Mask RCNN-a detektora. RPN mreža predviđa koordinate RoI-a na mapama značajki. Lista RoI-a koji najvjerojatnije sadrže objekt šalju se u RoI Align sloj gdje se sažimaju na očekivane dimenzije. Takve sažete značajke se prosljeđuju potpuno povezanom sloju koji dalje vrši klasifikaciju, regresiju koordinata okvira i predviđanje maske okvira.



Slika 4.10: Prikaz Mask R-CNN arhitekture. [9]

5. Detekcija sportaša

U ovom radu glavni je zadatak bio detekcija sportaša na jednoj nogometnoj utakmici. Korišten je skup podataka u kojem se nalazi oznake detekcijskih okvira za svakog pojedinog igrača i suca za svaki prikaz videa. Snimka je statična i snimana je iz ptičje perspektive. Razlučivost je 1632 x 288 piksela i snimka traje 5 minuta, te u snimci postoji 7515 slika. Pošto je slika statična i na njoj su uvijek prikazani objekti, zadatak se sveo na problem lokalizacije objekata. Preuzeta je Facebook-ova implementacija Mask R-CNN [13] napisana u PyTorch-u. Korištene su evaluacijske mjere proizašle iz PASCAL VOC natjecanja [5]. Programsku podršku za evaluaciju rada modela smo sami napisali, a implementacijski detalji su opisani u nastavku.

Primjer jednog prikaza videa vidljiv je na slici 5.1. Može se primijetiti da su osobe u daljini jako male, što bi moglo stvarati problem prilikom detekcije. Također, na slici se mogu vidjeti objekti koje nisu sportaši, a koje bi model mogao detektirati. U nekim trenucima igrači mogu biti prikriveni, čime model gubi uvid u jednog od igrača. Navedeni problemi prikazani su na slici 5.2.

Slika 5.1: Isječak videa koji se koristi za detekciju sportaša

Mana ovakvih modela je ta što ne mogu prepoznati kontekst scene u kojoj se objekti nalaze. Ljudi bi imali problema prilikom prepoznavanja objekata ako im se samo pokažu pojedinačne nisko kvalitetne slike, ali ako im se ukaže cijela scena i kontekst u kojoj se objekt nalazi mogu bez problema prepoznati objekte. Modeli strojnog učenja svaki ulazni podatak promatraju kao nezavisne i nepovezane jedinke pa samim time imaju problema prilikom prepoznavanja objekta u nisko kvalitetnih snimaka.

Slika 5.2: Slika prikazuje neke moguće situacije i probleme s kojima se detektor može susresti. Slika 1 prikazuje sportaša koji se nalazi blizu nas, Slika 2 sportaša koji je daleko od nas, a slika 3 preklapanje igrača pri čemu detektor gubi uvid u jednog od igrača.

5.1. Programska potpora

5.1.1. Python

Python je interpreterski programski jezik visoke razine. Kreiran od strane Guida van Rossuma i prvi put objavljen 1991. godine, Pythonova filozofija dizajna naglašava čitljivost koda. Njegovi jezični konstrukti i objektno-orijentirani pristup pomažu programerima da pišu jasan, logičan kod za male i velike projekte.

Podržava više paradigmi programiranja, uključujući proceduralno, objektno orijentirano i funkcionalno programiranje.

5.1.2. PyTorch biblioteka

PyTorch [4] je računalni softver, specifično biblioteka strojnog učenja za programski jezik Python. Prvenstveno ga je razvila Facebook-ova istraživačka skupina za umjetnu inteligenciju. To je besplatan softver otvorenog koda. Osnovni elementi ove biblioteke su tenzori koji predstavljaju multidimenzionalnu strukturu podataka. Tenzori u PyTorch-u slični su Numpy nizovima, uz dodatak da se tenzori mogu koristiti i na GPU-ima koje podržava CUDA. U PyTorch biblioteci može se naći mnoštvo pretreniranih modela, skupova podataka i implementiranih algoritama strojnog učenja.

5.2. Eksperimentalni Rezultati

5.2.1. Evaluacijske mjere

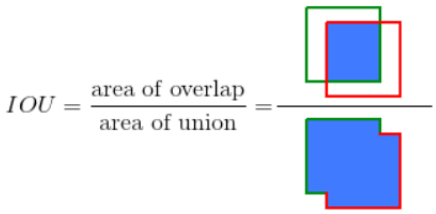
Evaluacijske mjere nam pomažu kvantitativnom opisu i usporedbi modela prilikom obavljanja nekog zadatka. Kao evaluacijsku mjere korištene su mjere proizašle iz Pascal VOC natjecanja [5].

Glavne evaluacijske mjere koje se koriste za detekciju objekata su preciznost, odziv, krivulja preciznosti i odziva, prosječna preciznost (*engl. Avarage precision* ili AP) i srednja prosječna preciznost (*engl. Mean Avarage Precision* ili mAP).

Za početak slijede definicije ključnih pojmova u kontekstu evaluacije modela za detekciju.

IoU(*engl. Intersection Over Union*) je mjera koja nam pokazuje udio preklapanja dva detekcijska okvira. Tipično je koristimo kada želimo vidjeti koliki je udio u preklapanju predikcijskog okvira i označenog okvira. Primjenom IoU-a možemo utvrditi koje detekcije su dobre, a koje su promašaj. Definiramo IoU kao:

$$IOU = \frac{Povrsina(box_1 \cap box_2)}{Povrsina(box_1 \cup box_2)} \tag{5.1}$$



Slika 5.3: Grafički prikaz računanja IoU mjere. IoU mjera je omjer presjeka dvaju okvira i njihove unije.

Klasificiramo detekcije u usporedbi sa stvarnim oznakama kao:

True Positive(TP) Detekcija je točna, $IoU > \tau$.

False Positive(FP) Detekcija je netočna, $IoU < \tau$.

False Negative(FN) Detektor nije uspio detektirati objekt .

True Negative(TN) Detekcije koje nisu detektirane, a nisu ni trebale biti detektirane. Irelevantno u zadatku detekcije objekata.

τ je u broj imeđu $[0, 1]$ koji predstavlja neku kritičnu vrijednost. Tipično se uzima $\tau = 0.5$.

Preciznost detektora je udio točnih detekcija u uzorku svih detekcija, tj:

$$Preciznost = \frac{TP}{TP + FP} = \frac{TP}{sveDetekcije} \tag{5.2}$$

Odziv detektora je udio prepoznatih objekata u uzorku svih objekata, tj:

$$Odziv = \frac{TP}{TP + FN} = \frac{TP}{oznaceniOkviri} \tag{5.3}$$

.

Krivulju odziva i preciznosti dobivamo tako da odredimo preciznost modela za različite vrijednosti odziva.

Da bismo dobili kvantitativnu mjeru kvalitete uspješnosti detekcije jedne klase koristimo prosječnu preciznost (AP). Opća definicija AP mjere jest površina ispod krivulje odziva i preciznosti:

$$AP = \int_0^1 p(r)dr \tag{5.4}$$

Postupak kojim se dobiva AP vrijednost je sljedeći:

- Poredamo silazno predikcije detektora prema sigurnosti (*engl. confidence*) u postojanje objekta u njima
- Za svaku predikciju izračuna se maksimalna IoU vrijednost sa označenim okvirom
- Klasificiraju se predikcije u TP ili FP na temelju IoU vrijednosti i definirane kritične vrijednosti τ.
- Ako postoji više predikcija za jednu oznaku, predikciju s najvećom IoU vrijednosti klasificiraj kao TP, a ostale kao FP.

Nakon što su provedeni gornji koraci, računanje preciznosti i odziva demonstrirat će se primjerom 5.1. Pretpostavimo da se naš skup za učenje sastoji od N = 3 slike, te da se na svakoj slici nalaze po 2 objekta. Preciznost i odziv računali smo pomoću formula 5.2 i 5.3.

Slike	**Sigurnost**	**TP**	**FP**	**Brojač TP**	**Brojač FP**	**Preciznost**	**Odziv**
Slika 3	90 %	True	False	1	0	1	1/6
Slika 1	80 %	False	True	1	1	1/2	1/6
Slika 3	60 %	True	False	2	1	2/3	2/6
Slika 1	50 %	True	False	3	1	3/4	3/6
Slika 2	30 %	False	True	3	2	3/5	3/6
Slika 2	25 %	False	True	3	3	3/6	3/6

Tablica 5.1: Izračun preciznosti i odziva

Da bismo izbjegli prevelike oscilacije u krivulji odziva i preciznosti. Vrijednosti preciznosti za određeni odziv računaju se formulom:

$$p(r) = max_{\hat{r} \geq r} p(\hat{r}) \tag{5.5}$$

U gornjoj formuli p je oznaka za preciznost, r je oznaka za odziv, dok je $\hat{r}$ skup vrijednosti koje su veće ili jednake r. Time dobivamo monotono padajuću krivulju.

Ukoliko naš skup za učenje ima više klasa koje je potrebno detektirati, tada koristimo srednju prosječnu preciznost mAP *(eng. mean average precision)*. Ona jednim brojem opisuje rezultate detekcije nekog modela nad skupom slika sa K klasa. Izračun mAP vrijednosti je sljedeći:

$$mAP = \frac{1}{K} \sum_{i=1}^{K-1} AP_i \tag{5.6}$$

5.2.2. Eksperimenti

Za eksperimentiranje odabrana je inačica Mask RCNN-a temeljena na ImageNet pretreniranim modelima resnet 50 i FPN-u (*engl. Feature Pyramid Networks*). Službeni naziv je R-50-FPN. Neke specifikacije tog modela, preuzete sa stranice projekta [13] dane su u tablici 5.2

Model	R-50-FPN
Skup za učenje	ImageNet
Vrijeme učenja (sec/iter)	0.3530
Ukupno vrijeme učenja(sati)	8.8
GPU	8 NVIDIA V100 GPUs
Vrijeme predikcije (s/predikcija)	0.12580
AP okvira	36.8

Tablica 5.2: Specifikacije korištenog Mask-RCNN modela [13].

Modelu je uklonjena glava za predikciju maske objekta te je time model predviđao samo klase i detekcijske okvire objekata. Model je prvo testiran na nekoliko različitih slika. Primjer rada detektora vidi se na slici 5.4. Model je dao 73 predikcije.

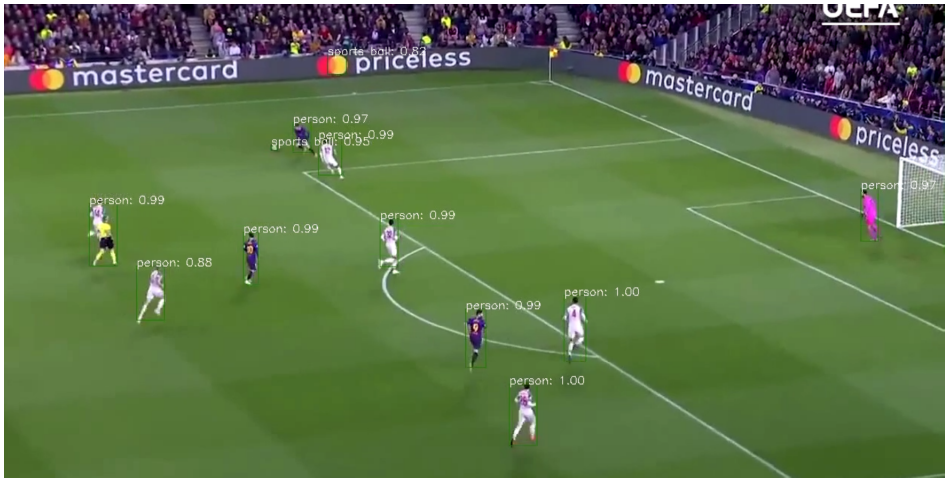
Nakon toga uklonjene su predikcije čiji je postotak sigurnosti manji od 70 %, te smo time uklonili šum i nesigurne predikcije. Ovako konfiguriran detektor u mogućnosti je klasificirati i detektirati 81 klasu. To će predstavljati problem ako ga iskoristimo za lokalizaciju sportaša na našem skupu podataka jer će detektirati nama irelevantne objekte.

Slika 5.4: Primjer rada Mask R-CNN detektora na jednoj slici. Detektor je za svaki objekt na slici predvidio koordinate njegovog okvira i razred u koji taj objekt pripada.

Nakon toga testirali smo detektor u domeni sporta. U ovom slučaju koristili smo kvalitetne nogometne snimke. Jedna od scena prikazana je na slici 5.5. Detektor vrlo kvalitetno detektira osobe i loptu. Prvi problem koji je uočen je da model detektira objekte koje nemaju veze s nogometnom utakmicom, poput publike. Drugi problem je da detektor ima problema s detektiranjem brzo putujuće lopte. Rješenje prvog problema bilo bi uklanjanje dijela snimke koje nema veze s nogometnom igrom. Rješenje drugog problema izgleda malo teže jer brzo putujuća lopta nije jasno vidljiva na slici. Čovjek u takvom slučaju loptu prepoznaje jedino povezivanjem nekoliko slika i zaključivanjem da je brzo putujući objekt u kontekstu nogometne utakmice najvjerojatnije lopta.

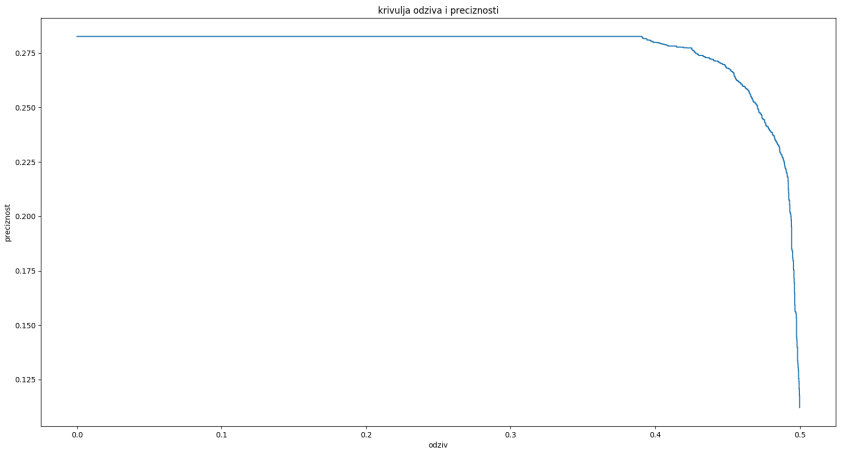
Model je testiran na danom nogometnom skupu podataka i rezultati su prikazani na slici 5.6. Na izlazu modela dobili smo 100 detekcijskih okvira. Neki od tih okvira uspješno su uokvirili sportaše i objekte iz ostalih kategorija dok neki izgledaju kao šum. Takvim detekcijskim okvirima model je predao malenu vjerovatnost u postojanje objekta unutar njih.

Napravljena je evaluacija nad takvim detektorom i pripadna krivulja odziva i preciznosti prikazana je na slici 5.7. AP vrijednost detekcije osoba ovakvog modela iznosila je 0.1385.



Slika 5.5: Primjer rada Mask R-CNN detektora u domeni sporta. Detektor je uspješno detektirao sve osobe, osim one koja se preklapa sa sucem. Detektor je također uspješno detektirao loptu.

Slika 5.6: Izlaz detektora se sastojao od 100 predikcijskih okvira i sigurnosti u postojanje objekta u okviru. Vidimo da je detektor imao dobar odziv za igrače, ali također ima jako puno promašaja.



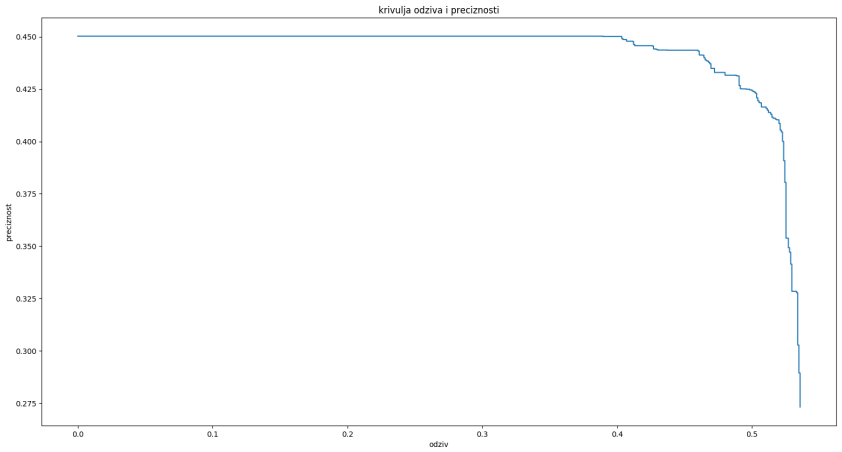
Slika 5.7: Krivulja odziva i preciznosti za detekciju nogometaša. Primjer jedne detekcije prikazan je na slici 5.6. AP vrijednost ovakvog modela ispala je 0.1385

Nakon uklanjanja nesigurnih predikcija, tj. predikcija čija je sigurnost manja od 0.7 dobiven je rezultat prikazan na slici 5.8. Time smo u potpunosti uklonili detek-

cijske okvire koji predstavljaju šum te dobivamo izlaz od oko 40 detekcijskih okvira. Ovdje vidimo da je model dobro uokvirio sportaše, ali također vidimo da je uokvirio irelevantne objekte, poput vozila na lijevoj strani slike.

Slika 5.8: Rad detektora s uklanjanjem nesigurnih predikcijskih okvira. Okviri čija je sigurnost manja od 0.4 su odbačeni što je dalo puno bolje rezultate. Iako možemo vidjeti u lijevom gornjem kutu kako je detektor prepoznao vozilo

Pripadna krivulja odziva i preciznosti nakon uklanjanja nesigurnih predikcija prikazana je na slici 5.9. Vidljivo je povećanje preciznosti detekcije, jer smo uklonili nepotrebne okvire.



Slika 5.9: Rad detektora nakon uklanjanja predikcija čija sigurnost je manja od 0.4. Za točne predikcije (TP) uzete su one za koje je IoU veći od 0.5. Dobivena AP vrijednost je 0.23

6. Zaključak

U okviru ovog rada proučeni su modeli dubokog učenja koji se primjenjuju pri detekciji i lokalizaciji objekata. Napravljeni su eksperimenti s Mask R-CNN arhitekturom za detekciju objekta i semantičku segmentaciju na razini instanci. Uz pomoć Mask R-CNN arhitekture napravljena je detekcija sportaša na nogometnoj utakmici. Mask R-CNN postiže dobre rezultate, uz ne tako kvalitetne video snimke, u lokalizaciji nogometaša i ostalih relevantnih objekata. Uz doradu i optimalno podešavanje parametara modela svakako se može očekivati kvalitetna primjena ovakvog modela u sportskoj industriji. Također, pretrenirani modeli za detekciju objekata postali su dijelovi standardnih biblioteka, poput PyTorch-a [4]. Time je olakšan pristup tim modelima i olakšano eksperimentiranje s opisanim arhitekturama.

LITERATURA

[1] Deepsense.ai. URL https://deepsense.ai/region-of-interest-pooling-explained/.

[2] Imagenet. URL http://www.image-net.org/.

[3] paperswithcode. URL https://paperswithcode.com/.

[4] Pytorch. URL https://pytorch.org/tutorials/.

[5] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, i Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, Lipanj 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4. URL http://dx.doi.org/10.1007/s11263-009-0275-.

[6] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

[7] Ross B. Girshick, Jeff Donahue, Trevor Darrell, i Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

[8] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, i Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[10] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, i Ross B. Girshick. Learning to segment every thing. *CoRR*, abs/1711.10370, 2017. URL http://arxiv.org/abs/1711.10370.

[11] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, i Piotr Dollár. Panoptic segmentation. *CoRR*, abs/1801.00868, 2018. URL http://arxiv.org/abs/1801.00868.

[12] Yann Lecun, Léon Bottou, Yoshua Bengio, i Patrick Haffner. Gradient-based learning applied to document recognition. stranice 2278–2324, 1998.

[13] Francisco Massa i Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: [Insert date here].

[14] Shaoqing Ren, Kaiming He, Ross B. Girshick, i Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[15] Stanford University. Cs231n convolutional neural networks for visual recognition. URL http://cs231n.github.io/.

Konvolucijski modeli za lokalizaciju sportaša

Sažetak

Modeli dubokog učenja temeljeni na konvolucijskim neuronskim mrežama postižu vrhunske rezultate u području računalnog vida. U ovom radu razmatramo arhitekture i algoritme za detekciju i lokalizaciju objekta. Kasnije primjenjujemo Mask R-CNN arhitekturu za lokalizaciju sportaša na video snimci nogometne utakmice. Nakon toga, mjerimo performanse Mask R-CNN detektora prilikom detekcije sportaša na zadanoj video snimci nogometne utakmice.

Ključne riječi: Strojno učenje, duboko učenje , računalni vid, detekcija objekata, lokalizacija objekata, Mask R-CNN

Convolutional Models for Person Localization in Sport Broadcasts

Abstract

Deep learning models based on convolutional neural networks achieve top results in the field of computer vision. In this paper, we consider architecture and algorithms for detecting and localizing objects. Later we apply the Mask R-CNN architecture to localize the athletes on the videotape of the football match. We then measure the performance of the Mask R-CNN detector on detecting athletes on the given football match videotape.

Keywords: Machine learning, deep learning, computer vision, object detection, object localization, Mask R-CNN