Detekcija i klasifikacija morskog otpada iz podvodnih slika korištenjem konvolucijskih neuronskih mreža

Đuraš, Antun

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

ANTUN ĐURAŠ

DETEKCIJA I KLASIFIKACIJA MORSKOG OTPADA IZ PODVODNIH SLIKA KORIŠTENJEM KONVOLUCIJSKIH NEURONSKIH MREŽA

DIPLOMSKI RAD

Dubrovnik, rujan, 2020.

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

DETEKCIJA I KLASIFIKACIJA MORSKOG OTPADA IZ PODVODNIH SLIKA KORIŠTENJEM KONVOLUCIJSKIH NEURONSKIH MREŽA

DIPLOMSKI RAD

Studij: Diplomski studij
Studijski smjer: Primijenjeno/poslovno računarstvo
Kolegij: Autonomni sustavi
Mentor: izv. prof. dr. sc. Ivana Palunko
Student: Antun Đuraš

Dubrovnik, rujan, 2020.

SAŽETAK

Razorni utjecaj sve većih količina morskog otpada na morski ekosustav predstavlja prijetnju ljudskom zdravlju, ekonomiji i kvaliteti života. Automatizacija procesa sakupljanja i pronalaženja otpada autonomnim podvodnim vozilima omogućila bi značajan napredak prema rješenju ovog problema. Za implementaciju takvog sustava potrebno je semantičko razumijevanje podvodnog okruženja na temelju informacija iz senzora podvodnog robota. Kamera tipično pruža velike količine informacija o okruženju, ipak djelovanje distorzije i visoka varijabilnost uvjeta podvodnog okruženja znači da su potrebni modeli visokog kapaciteta da bi se obuhvatili obrasci prisutni u podacima. Konvolucijske mreže su modeli iz domene strojnog učenja koji posljednje desetljeće dominiraju na zadacima detekcije i klasifikacije. U sklopu ovog rada prikupljene su i označene podvodne slike morskog otpada, formiran je podatkovni skup za treniranje modela za detekciju i klasifikaciju morskog otpada, dana je teorijska pozadina principa rada konvolucijskih modela za detekciju i klasifikaciju, nakon čega su modeli implementirani i evaluirani koristeći *Tensorflow Object Detection* API. Evaluacijom različitih modela, uz usporedbu preciznosti s obzirom na kompleksnost i brzinu inferencije, stečen je uvid u njihovu primjenjivost na ovaj problem.

Ključne riječi: Detekcija i klasifikacija morskog otpada, Duboki konvolucijski modeli, Detekcija objekata, Konvolucijske neuronske mreže, Podvodne fotografije

ABSTRACT

Devastating impact of increasing ammounts of marine litter on the marine ecosystem presents a looming threat to human health, economy and quality of life. Ability to automate the process of marine litter collection utilizing autonomous underwater vehicles represents a significant step towards the problem solution. Implementation of such system requires a semantic understanding of underwater environment based on the information collected by robot sensors. Cameras can typically be used to extract a lot of information about robot surroundings but due to specifics of underwater imagery such as variety of conditions and geometric distorsions, high capacity models are required to learn patterns from the data. During the last decade convolutional neural networks dominate the benchmarks for image classification and object detection tasks. In the scope of this thesis theoretical background of convolutional models for detection and classification was reviewed, underwater images of marine litter were collected from online databases and annotated for litter detection and classification task, finally models were implemented, trained and evaluated using *Tensorflow Object Detection* API. By evaluating performance on models of different complexity and speed of inference we analysed their applicability on this problem.

Key words: Marine litter detection and classification, Deep convolutional models, Object Detection, Convolutional neural networks, Underwater imagery

SADRŽAJ:

1 UVOD

Morskim otpadom smatraju se svi antropogeni objekti koji su bačeni u morsko okruženje, neovisno o veličini. S obzirom da utjecaj otpada ima negativne socijalne, ekonomske i ekološke implikacije [1] sve je više istraživanja posvećenih sustavnom praćenju razmjera zagađenja. Automatiziranje procesa detekcija i klasifikacije morskog otpada pruža važne informacije za semantičko razumijevanje podvodnih slika koje se mogu iskoristiti za mapiranja otpada na morskom dnu, uvid u izvore zagađanja ili industrijske primjene poput sakupljanja otpada korištenjem autonomnih podvodnih vozila AUV (engl. *autonomous underwater vehicle*).

Problem detekcije i klasifikacije morskog otpada dijeli karakteristike sa širim područjem obrade podvodnih slika. Specifični izazovi koji utječu na složenost ovog zadatka su visoka varijabilnost unutar kategorija otpada i uvjeta okoliša u kojim se prikupljaju podvodni podaci.

Okolišni uvjeti kao što su prozirnost medija, morske struje, varijacije prirodnog osvjetljenja, čistoća mora i utjecaj podvodnih organizama čine algoritme bazirane na podacima dohvaćenim u podvodnom okruženju nerobusnim na promjene. Primjer distorzije uzrokovane snimanjem pod vodom je apsorpcija crvene valne duljine svjetlosti, zbog čega su podvodne slike zasićene plavom i zelenom bojom (ovaj efekt se povećava s porastom dubine). Zbog izrazito nelinearne prirode ovih distorzija i visokog kapaciteta neuronskih mreža koji omogućuje modeliranje takvih nelinearnosti, konvolucijske neuronske mreže su u središtu modernog istraživanja ovog područja.

Metode nadgledanog učenja koje se tipično primjenjuju za zadatke klasfikacije i detekcije ovisne su o označenim podacima za učenje. Također visoki kapacitet dubokih modela povlači i zahtjev za velikom količinom varijabilnih podataka kako bi se spriječilo preveliko prilagođavanje podacima za učenje i omogućila generalizacija na eventualnim novim podacima. Količina dostupnih podataka za podvodne primjene je mala u odnosu na kopnene i često nedovoljna za treniranje kompleksnih modela. Kombinacija prethodno navedenih faktora razlog je da se velika količina aktivnog istraživanja u ovom području fokusira na augmentaciju podvodnih skupova podataka, prijenosno učenje i poboljšanje kvalitete podataka.

Slika 1. Primjer varijabilnosti slika morskog otpada ovisno o uvjetima snimanja [2]

1.1. Svrha i ciljevi rada

Ovaj diplomski rad fokusira se na primjenu metoda iz područja dubokog učenja, specifično konvolucijskih neuronskih mreža, s ciljem izgradnje modela koji je sposoban iz fotografija podmorja lokalizirati instance morskog otpada i klasificirati ih u neku od preddefiniranih kategorija koja predstavlja tip otpada po materijalu.

U sklopu ovog rada preuzet je skup fotografija podmorja te označeno 1868 primjeraka s ciljem treniranja dubokih konvolucijskih modela koji čine najsuvremenije metode detekcije i klasifikacije. U fokusu su jednofazni modeli SSD i RetinaNET koji bi omogućili inferenciju u realnom vremenu na ugrađenim sustavima. Spajanjem vlastitog skupa sa javno dostupnim

skupom *TrashCan* [3] srodne primjene dobiven je ukupno najveći podatkovni skup do sad korišten za ovu primjenu od 8688 označenih primjera.

1.2. Struktura rada

Rad je podijeljen na dvije cjeline. U prvoj cjelini dana je teorijska pozadina konvolucijskih modela i procesa učenja neuronskih mreža, uvedeni su osnovni pojmovi, detaljno su opisane arhitekture korištenih modela za detekciju i klasifikaciju te metrike koje se koriste za njihovu evaluaciju. Drugi dio rada sastoji se od opisa formiranja podatkovnog skupa te programske implementacije modela. Modeli su implementirani pomoću *TensorFlow* biblioteke za duboko učenje, trenirani na formiranom skupu podataka i evaluirani prema relevantnim metrikama. S obzirom na malu količinu dostupnih podataka pri treniranju modela korištena je tehnika prijenosnog učenja gdje su modeli predtrenirani na velikim skupovima podataka za druge zadatke poput generičke detekcije objekata.

2 UMJETNE NEURONSKE MREŽE

Princip rada neuronskih mreža može se u najopćenitijem smislu opisati kao aproksimacija neke funkcije $f^*: x \rightarrow y$ (koju je teško eksplicitno modelirati). Aproksimaciju funkcije f^* u i-tom trenutku (stadiju treninga) označavamo sa f_i , a dobiva se prilagođavanjem parametara aproksimacije θ empirijskim podacima. Proces prilagodbe parametara često se naziva i učenjem neuronske mreže, a podaci na temelju kojih se vrše ažuriranja podaci za učenje. Željeni ishod primjene ove metode je dobiti niz aproksimacija $\{f_i\}$ koji posjeduje svojstvo [4]:

$$\lim_{n \to \infty} f_i(x) = f^* \tag{1}$$

Neuronske mreže odlično rješavaju probleme klasifikacije i predviđanja, odnosno općenito sve probleme gdje postoji odnos između prediktorskih (ulaznih) i zavisnih (izlaznih) varijabli, bez obzira na nelinearnost te veze.

U sljedećim poglavljima ukratko je opisan općeniti model umjetnog neurona kao osnovne računske jedinice neuronskih mreža te MLP (engl. *Multilayer Perceptron*) kao povijesno najznačajnijeg predstavnika dubokih unaprijednih arhitektura. Nakon kraćeg opisa MLP arhitekture i uvođenja osnovne terminologije, detaljnije je opisan princip rada konvolucijskih modela čija upotreba je u središtu ovog diplomskog rada. U poglavlju 2.4 postupak učenja neuronskih mreža smješten je u kontekst rješavanja optimizacijskog problema, nakon čega je dan teorijski pregled tipično korištenih algoritama za njegovo rješavanje.

2.1 Matematički model umjetnog neurona

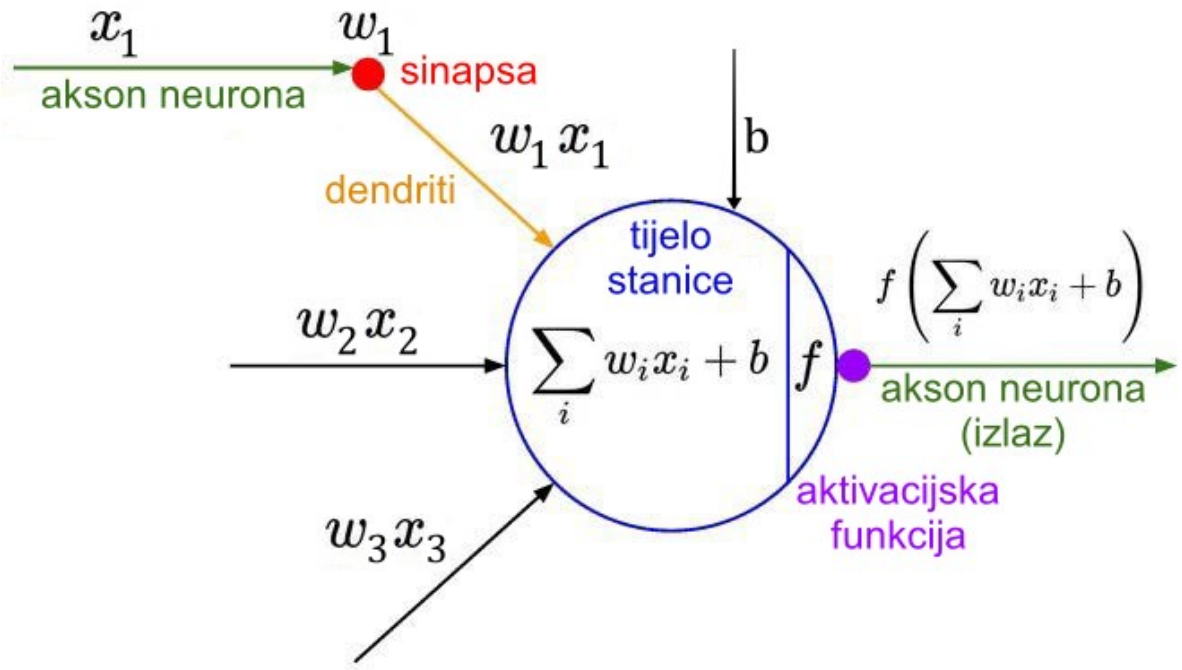
Na slici 2 prikazana je ilustracija općenitog modela umjetnog neurona, s obzirom na to da nije specificirana konkretna aktivacijska funkcija. Numeričke vrijednosti koje čine ulazni signal neurona su izlazne vrijednosti drugih neurona ili dolaze iz nekog vanjskog izvora. Neuronima su pridruženi težinski parametri $\boldsymbol{w}$ i pomak b za ulazni signal $\boldsymbol{x}$. Izlaz neurona izračunava se primjenom aktivacijske funkcije f na linearnu kombinaciju ulaza gdje su koeficijenti težine uz pribrojeni pomak b.

Često se pomak b prikazuje kao dodatna komponenta $w_0 = b$ vektora težina $\boldsymbol{w}$, a ulaznom signalu $\boldsymbol{x}$ se dodaje fiksirana komponenta vrijednosti 1 kako bi se ukupno djelovanje neurona moglo izraziti pomoću skalarnog produkta vektora $\boldsymbol{x}$ i $\boldsymbol{w}$.

$$\boldsymbol{x} = [1, x_1, x_2, \dots, x_d]$$

$$\boldsymbol{w} = [w_0, w_1, w_2, \dots, w_d]$$

$$izlaz = f(\boldsymbol{x} \cdot \boldsymbol{w}) = f\left(\sum_{i=1}^{d} w_i x_i + w_0\right) = f\left(\sum_{i=1}^{d} w_i x_i + b\right) \quad (2)$$



Slika 2. Model umjetnog neurona [5]

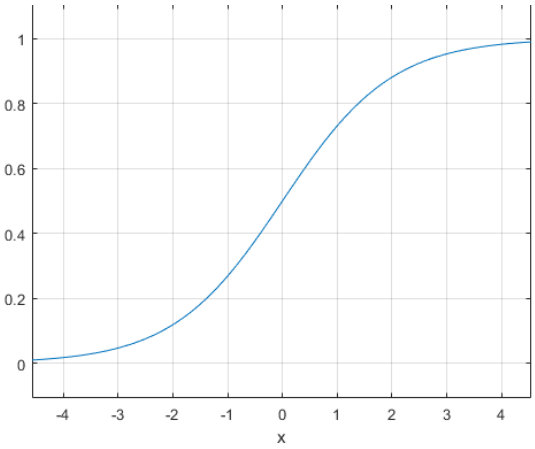
2.2 Aktivacijske funkcije

Odabir konkretnog tipa aktivacijske funkcije ovisi o arhitekturi mreže, tipu problema koji mreža modelira i svojstvima same funkcije. Poželjna svojstva aktivacijskih funkcija su nelinearnost, kontinuiranost, zaglađenost, monotonost i linearnost u blizini ishodišta [6]. U tablici 1 opisana su svojstva nekih od najpopularnijih aktivacijskih funkcija skrivenih slojeva koje se koriste u kontekstu dubokih neuronskih mreža.

Tablica 1. Opis svojstava najčešće korištenih aktivacijskih funkcija i pripadajući grafovi

Sigmoida

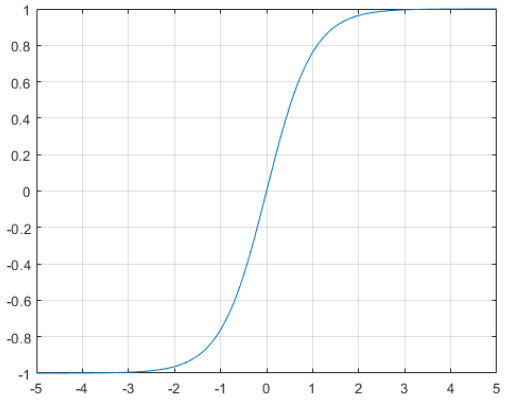
$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$



- „Stisne“ ulaznu vrijednost na interval [0,1]
- Rijetko se koristi u modernim arhitekturama zbog zasićenja u repovima funkcije sa pozitivne i negativne strane domene. Zasićenje uzrokuje probleme u propagaciji gradijenta za duboke arhitekture.
- Drugo nepoželjno svojstvo sigmoide je da vrijednosti funkcije nisu centrirane oko nule što uzrokuje ograničenja na smjerove ažuriranja parametara.

Tangens hiperbolni

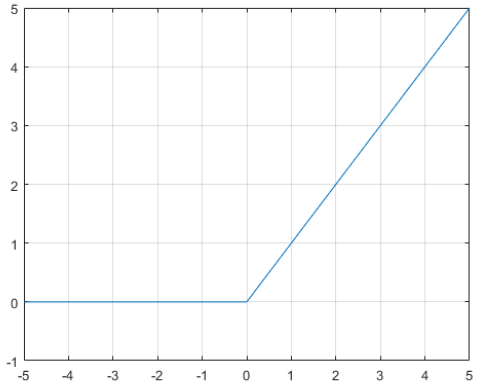
$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



- „Stisne“ ulaznu vrijednost na interval [-1,1]
- Rješava problem sigmoide s necentriranim izlaznim vrijednostima oko 0
- Kao i sigmoida ima problem s propagacijom gradijenta zbog zasićenja u repovima funkcije.

ReLU („*Rectified Linear Unit*“)

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



- Propušta samo ulazne vrijednosti $x > 0$
- Nije derivabilna u $x = 0$, pa se definira $f'(x = 0) = 0$
- Izrazito popularna u kontekstu dubokih arhitektura i konvolucijskih neuronskih mreža.

Nedostaci:

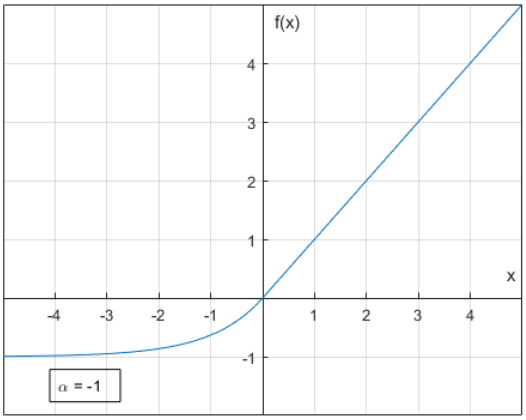
- Nije centrirana oko nule
- Problem „umirućih“ neurona (engl. „*Dying ReLU*“) uzrokovanih lošom inicijalizacijom težina ili uzimanjem prevelikih koraka u smjeru gradijenta pri učenju.
- *Leaky ReLU* varijanta $f(x) = \max(0.01x, x)$ rješava problem „umirućih" neurona

Prednosti:

- Konvergira do 6 puta brže (od neurona koji koriste tangens hiperbolni za aktivaciju) pri optimizaciji algoritamom stohastičkog spuštanja po gradijentu [7]
- Računski efikasna (nema eksponencijalnih računskih operacija)
- Nema problema sa zasićenjem u pozitivnom dijelu domene

ELU („*Exponential Linear Unit*“)

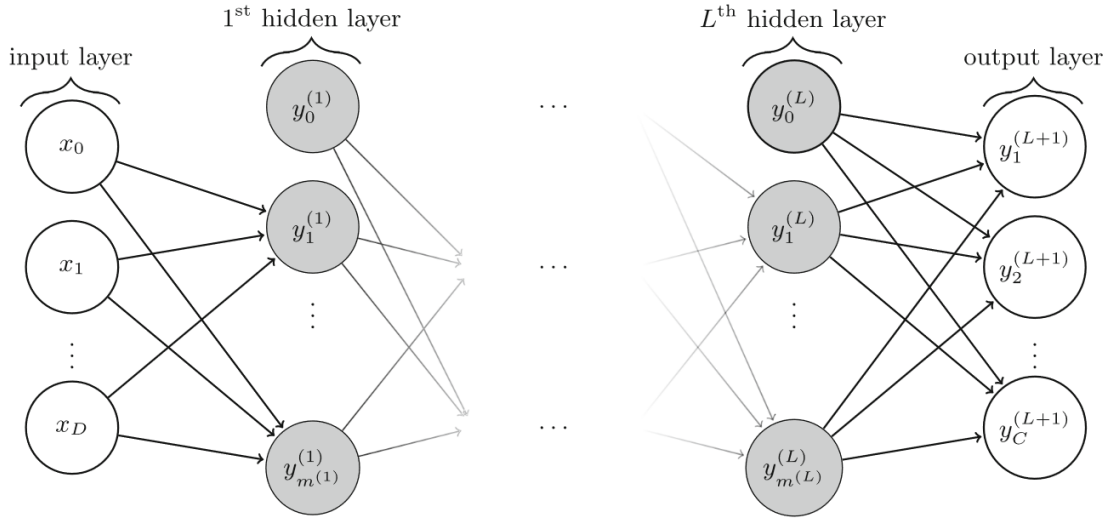
$$f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$$



- Zadržava sva dobra svojstva ReLU funkcije osim što je računski skuplja
- Bliže tome da je centrirana oko 0
- Zasićenje u negativnom dijelu dodaje robusnost u odnosu na ReLU funkciju koja je fiksno definirana 0

2.3 Duboki unaprijedni modeli

Višeslojni perceptron MLP (engl. *Multilayer Perceptron*) je neuronska mreža koja se sastoji samo od potpuno povezanih slojeva i osnovni je predstavnik neuronskih mreža s unaprijednom topologijom. U literaturi se često koriste i nazivi unaprijedne neuronske mreže FNN(engl. *Feedforward neural networks*) i duboke unaprijedne mreže (engl. *Deep feedforward networks*). Termin unaprijedni odnosi se na smjer toka informacija, odnosno nepostojanje veza iz izlaznih slojeva modela natrag u model [8].



Slika 3. Graf MLP sa L skrivenih slojeva , gdje je $D = m^{(0)}$ dimenzija ulaznog sloja, a $C = m^{(L+1)}$ dimenzija izlaznog sloja [9].

$(L+1)$-slojni MLP sastoji se od ulaznog sloja, L skrivenih slojeva i jednog izlaznog sloja (Slika 3). Pojam dubine mreže odnosi se na broj slojeva, a broj umjetnih neurona u pojedinom sloju se naziva širinom. U potpuno povezanim slojevima svi neuroni kao ulazni signal dobivaju sve izlazne vrijednosti neurona prethodnog sloja . Izlaz i-tog neurona za l-ti skriveni sloj definiran je izrazima:

$$y_i^{(l)} = f_i^{(l)}(z_i^{(l)}) \tag{3}$$

gdje je:

$$z_i^{(l)} = \sum_{j=1}^{m^{(l-1)}} w_{ij}^{(l)} y_j^{(l-1)} + w_{i0}^{l} \tag{4}$$

$w_{ij}^{(l)}$ predstavlja otežanu vezu između i-tog neurona l-tog sloja i j-tog neurona $(l-1)$-og sloja, a w_{i0}^{l} je vrijednost pomaka za i-ti neuron l-tog sloja. Iznimno $y_i^{(0)}$ predstavlja i-tu vrijednosti ulaznog sloja. $m^{(l)}$ je širina, odnosno dimenzionalnost l-tog sloja.

2.3.1 Svojstvo univerzalne aproksimacije

Aktivacijska funkcija koja se primjenjuje na izlazu umjetnih neurona skrivenih slojeva ima važnu ulogu u određivanju same reprezentacijske moći neuronske mreže. Korištenjem samo linearne aktivacijske funkcije (funkcije identiteta)

$$f(\boldsymbol{x} \cdot \boldsymbol{w}) = \boldsymbol{x} \cdot \boldsymbol{w} \tag{5}$$

višeslojna neuronska mreža svodi se na jednoslojnu mrežu koja radi linearnu regresiju [10].

Kroz niz radova u razdoblju 90-ih godina dokazano je da, uz blaga ograničenja na svojstva aktivacijske funkcije, MLP sa samo jednim skrivenim slojem i konačnim brojem neurona može aproksimirati bilo koju funkciju tj. da su neuronske mreže univerzalni aproksimator.

U radu [11] dokazano je da svojstvo univerzalne aproksimacije vrijedi ako je aktivacijska funkcija kontinuirana sigmoida, odnosno zadovoljava svojstvo diskriminatornosti. Opseg funkcija koje zadovoljavaju ovo svojstvo može se proširiti i na *ReLU* funkciju [4] koja je najčešće korištena aktivacijska funkcija u modernim arhitekturama. Svojstvo univerzalne aproksimacije može se generalizirati i na širu obitelj aktivacijskih funkcija pa je tako u radu [12] dokazano da neuronske mreže mogu aproksimirati bilo koju funkciju na kompaktnom skupu uz uvjet da aktivacijska funkcija nije polinom.

Svojstvo univerzalne aproksimacije dokazano je za MLP sa jednim skrivenim slojem bez ograničenja na širinu sloja. Takav pristup nije efikasan jer zahtjeva ekstremno velik broj parametara što povećava kapacitet modela i uzrokuje preveliko prilagođavanje modela podacima za učenje (engl. *overfitting*).

Ovaj problem služio je kao glavna motivacija za razvoj arhitektura dubokih neuronskih mreža. Pokazalo se da upravo višestruka kompozicija nelinearnih aktivacijskih funkcija povećava reprezentacijsku moć neuronskih mreža i omogućuje im učenje apstraktnijih koncepata. Dublje mreže najčešće zahtjevaju značajno manji ukupni broj neurona u skrivenim slojevima te djeluju kao forma regularizacije.

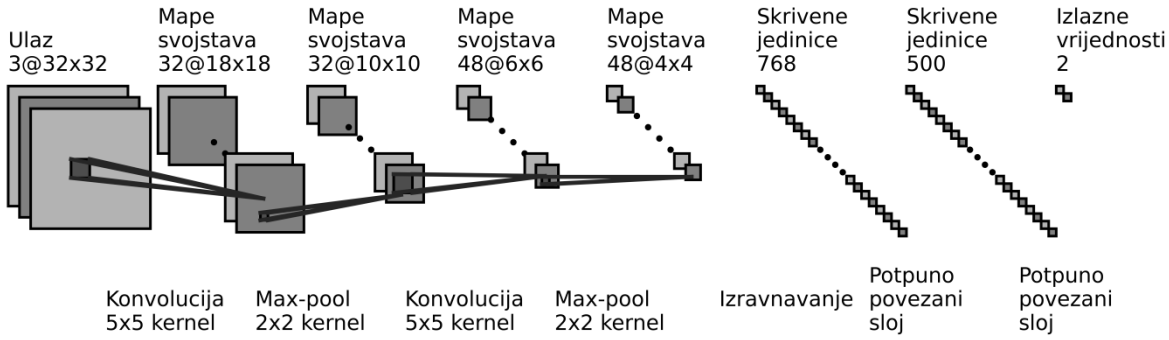
Duboke neuronske mreže sa konačnim brojem slojeva, *ReLU* aktivacijskom funkcijom i ograničenom širinom zadržavaju svojstvo univerzalnog aproksimatora.

2.4 Konvolucijske neuronske mreže

Konvolucijske neuronske mreže CNN (engl. *Convolutional Neural Networks*) su unaprijedni modeli specijalizirani za primjenu na podacima s topologijom rešetke. Model možemo smatrati konvolucijskim ako ima barem jedan konvolucijski sloj [13].

U memoriji računala slike su pohranjene kao 3D struktura s dvije prostorne i jednom semantičkom dimenzijom ($visina \times širina \times broj\ kanala = H \times W \times C$). Svaki od $H \times W$ piksela za slike u boji tipično je predstavljen sa $C = 3$ numeričke vrijednosti. Interpretacija pojedinačnih vrijednosti ovisi o prostoru boja (engl. *color space*) koji se koristi (RGB, HSV, CIELAB i sl.). Matematički ova struktura odgovara tenzorima 3. reda $\mathbb{R}^{H \times W \times C}$. Iznimno je u slučaju crnobijelih (engl. *grayscale*) slika $C = 1$.

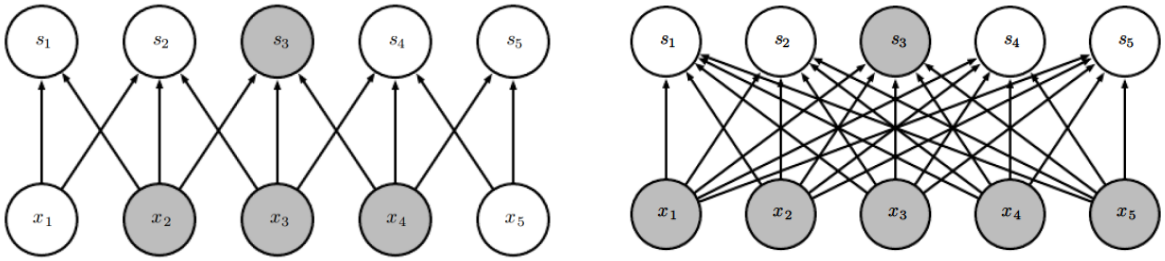
Slojevi konvolucije sa *ReLU* aktivacijama, povremeni slojevi sažimanja (engl. *pooling layers*) te normalizacijski slojevi su tipični sastavni blokovi CNN arhitektura (Slika 4). Često se u posljednjim slojevima modela koriste potpuno povezani slojevi čiji ulaz čini 1D vektor svojstava. CNN arhitekture kroz svojstva slojeva konvolucije i slojeva sažimanja u modele ugrađuju brojne pretpostavke o strukturi ulaznih podataka koje ih čine pogodnim za primjenu na zadatcima iz područja računalnog vida.



Slika 4. Primjer konvolucijske arhitekture [14]

Osnovna svojstva ugrađena u konvolucijske modele koja ih čine robusnim za ove primjene su [8]:

- **Manji broj parametara –** Lokalna povezanost, dijeljenje parametara konvolucijskih slojeva za pojedine mape značajki i smanjivanje dimenzionalnosti slojevima sažimanja reducira broj parametara i omogućuje bolju generalizaciju. Na slici 5 prikazana je razlika u broju veza (parametara) između potpuno povezanog sloja i lokalno povezanog konvolucijskog sloja.

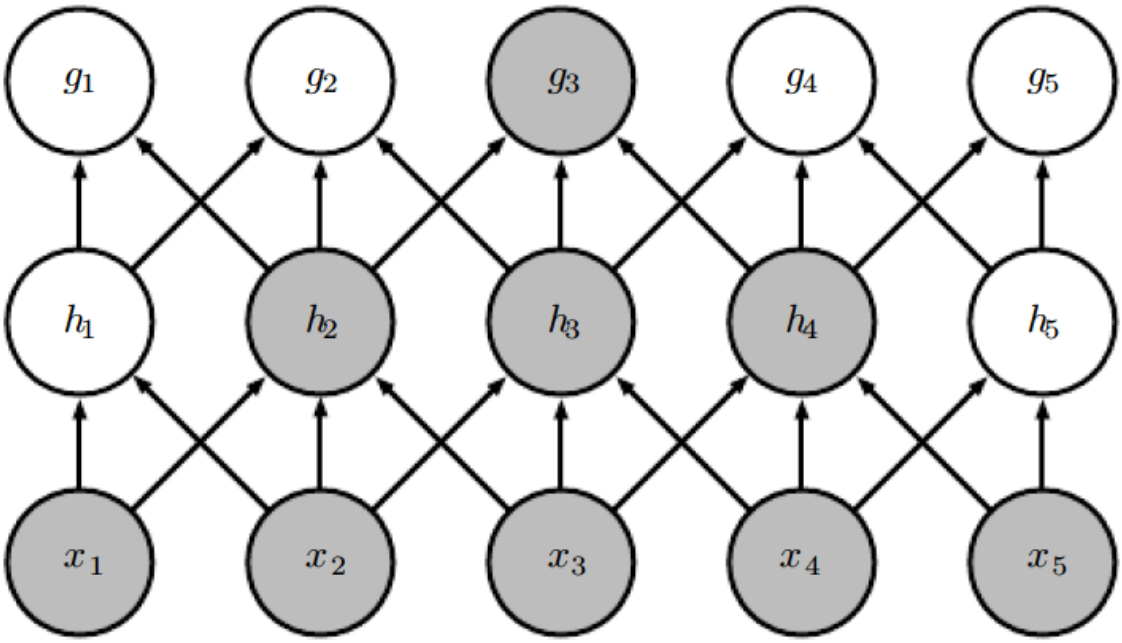


(a) Lokalno povezani neuroni konvolucijskog sloja

(b) Neuroni potpuno povezanog sloja

Slika 5. Ilustracija smanjenog broja parametara za konvolucijski sloj (a) u odnosu na potpuno povezani sloj (b) [8]

- **Translacijska ekvivarijantnost** – za slične regije slike mreža bi u ranijim slojevima trebala generirati slične vrijednosti neovisno o poziciji regije unutar slike. Ovo svojstvo ostvaruje se kroz dijeljenje parametara i lokalnu povezanost u konvolucijskim slojevima.
- **Translacijska invarijantnost –** slojevi sažimanja daju modelu robusnost s obzirom na male pomake unutar slike.
- **Receptivno polje raste s dubinom –** Iako su neuroni samo lokalno povezani, povećanjem dubine CNN arhitektura i poduzorkovanjem slojevima sažimanja te korištenjem pomaka $S > 1$ za konvolucijske slojeve, povećava se receptivno polje značajki dubljih slojeva. Ovaj princip ilustriran je na slici 6. Značajke ranijih konvolucijskih slojeva imaju manje receptivno polje pa njihovi filteri uče općenite značajke kao što su rubovi i teksture. Značajke dubljih slojeva sažimaju veće količine informacija što omogućuje prepoznavanje apstraktnijih koncepata na razini čitave ulazne slike.



Slika 6. Ilustracija povećanja receptivnog polja na dubljim slojevima, gdje značajka g_3 trećeg sloja ima receptivno polje veličine 5 tj. sažima informaciju $x_1, \dots, x_5$ ulaznih značajki iako je direktno povezana sa samo 3 značajke drugog sloja h_2, h_3, h_4 [8].

2.4.1 Konvolucijski slojevi

Konvolucijski sloj određen je ulaznim volumenom dimenzija $W_1 \times H_1 \times D_1$ i sljedećim hiperparametrima [15]:

- Broj filtera (jezgri, engl. *kernel*) $\boldsymbol{K}$
- Prostorni raspon filtera $\boldsymbol{F} \times \boldsymbol{F}$ (visina i širina su tipično iste vrijednosti $F = 1$, $F = 3$ ili $F = 5$)
- Veličina pomaka (engl. *stride*) $\boldsymbol{S}$
- Količina nadopunjavanja nulama **P** (engl. *zero padding*)

Izlaz konvolucijskog sloja čini volumen $W_2 \times H_2 \times D_2$, gdje ovisno o dimenzijama ulaza i hiperparametrima sloja, dimenzije izlaza poprimaju sljedeće vrijednosti:

$$W_2 = \left\lfloor \frac{W_1 - F + 2P}{S} \right\rfloor + 1 \qquad H_2 = \left\lfloor \frac{H_1 - F + 2P}{S} \right\rfloor + 1 \qquad D_2 = K$$

Uzastopnim slojevima konvolucije smanjuju se dimenzije ulaznog volumena kada je $F > 1$ ili $S > 1$. Kako bi se kontrolirala brzina smanjivanja mapa značajki i spriječio gubitak

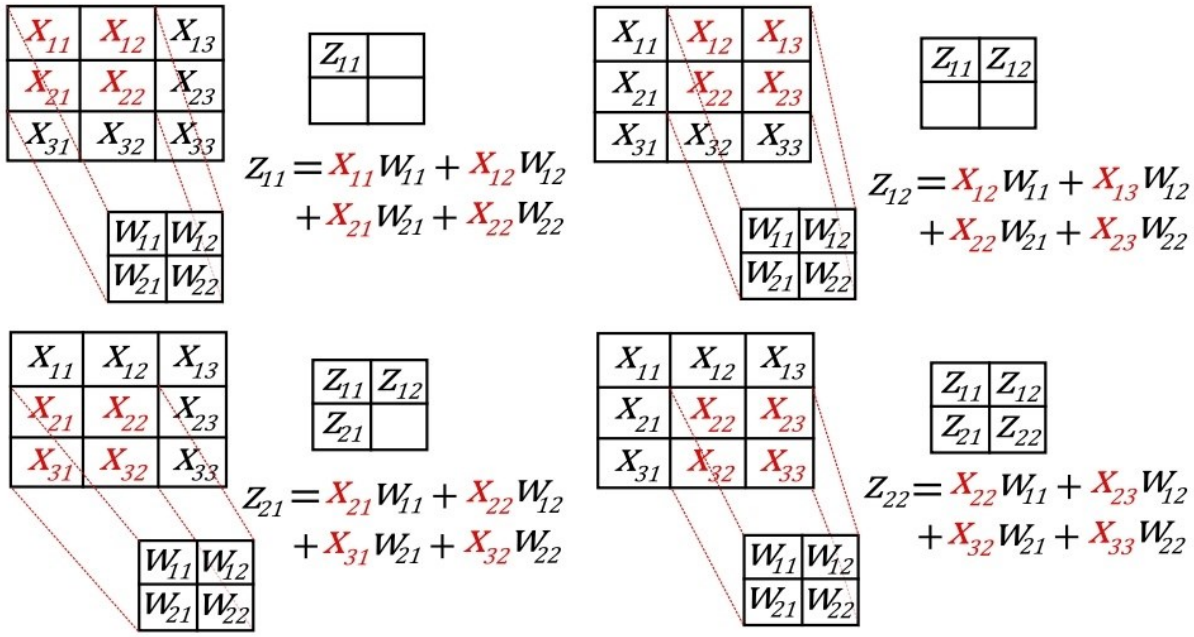
informacija na rubovima dodaje se proširivanje nulama. Nadopunjavanje nulama nazivamo proces povećavanja dimenzije H_1 ili W_1 za P gdje dodane vrijednosti poprimaju vrijednost 0. Najčešće korištene strategije nadopunjavanja su *valid* i *same* čiji je učinak opisan u tablici 2.

Tablica 2. Strategije nadopunjavanja [15]

Strategija	*valid*	*same*
Vrijednost	$P = 0$	$P_{start} = \left\lfloor \frac{\left(S\left\lceil \frac{I}{S} \right\rceil - I + F - S\right)}{2} \right\rfloor$ $P_{end} = \left\lceil \frac{\left(S\left\lceil \frac{I}{S} \right\rceil - I + F - S\right)}{2} \right\rceil$
Rezultat	Ulazna mapa značajki se ne nadopunjava	Izlazna mapa značajki je $\left\lceil \frac{I}{S} \right\rceil$ veličine. Izlaz je dimenzije ulaza I ako je $S = 1$.

Operaciju konvolucije možemo zamisliti kao klizeći prozor (engl. *sliding window*) gdje filter pomičemo po širini i visini ulaznog volumena s pomakom S. Svaki od K filtera dimenzija $F \times F \times D_1$ sadrži parametre modela. Dio ulaznog volumena obuhvaćen filterom čini ulaz umjetnog neurona, čija se izlazna vrijednost izračunava kao linearna kombinacija ulaza gdje su težinski koeficijenti parametri filtera. Ilustracija operacije konvolucije je prikazana na slici 7 za 2D primjer uz napomenu da se operacija za 3D primjere proteže kroz čitavu dubinu ulaznog volumena. Na rezultate konvolucije dodaje se pomak $b \in \mathbb{R}$ i primjenjuje aktivacijska funkcija f čime se dobiva konačna vrijednost izlaza. Aktivacijska funkcija u slučaju konvolucijskih mreža najčešće je $ReLU$ funkcija, odnosno neka od srodnih varijanti ili $tanh$ funkcija [16].

Za svaki filter na izlazu konvolucijskog sloja dobiva se po jedna dvodimenzionalna aktivacijska mapa $H_2 \times W_2$ koja predstavlja odziv filtera na različitim pozicijama ulaza. Aktivacijske mape se još nazivaju i mape značajki (engl. *feature maps*) jer svaka mapa odgovara odzivu za pojedinačni filter koji uči prepoznati neku specijaliziranu značajku (primjerice, rubove ili teksture na nižim slojevima).



Slika 7. 2D primjer operacije konvolucije 2×2 filterom, 3×3 ulazom i pomakom $S = 1$

2.4.2 Slojevi sažimanja

Osim korištenjem pomaka $\boldsymbol{S} > 1$ u konvolucijskim slojevima, smanjenje dimenzionalnosti postiže se i korištenjem slojeva sažimanja. Slojevi sažimanja (engl. *pooling*) djeluju na ulazni volumen i za 2D regije značajki računaju statistički deskriptor. Regije značajki su kao i za operaciju konvolucije određene pomicanjem klizećeg prozora po ulaznom volumenu s tim da se operacija vrši posebno za svaki pojedini dubinski presjek.

Slojevi sažimanja određeni su ulaznim volumenom $W_1 \times H_1 \times D_1$ i hiperparametrima sloja:

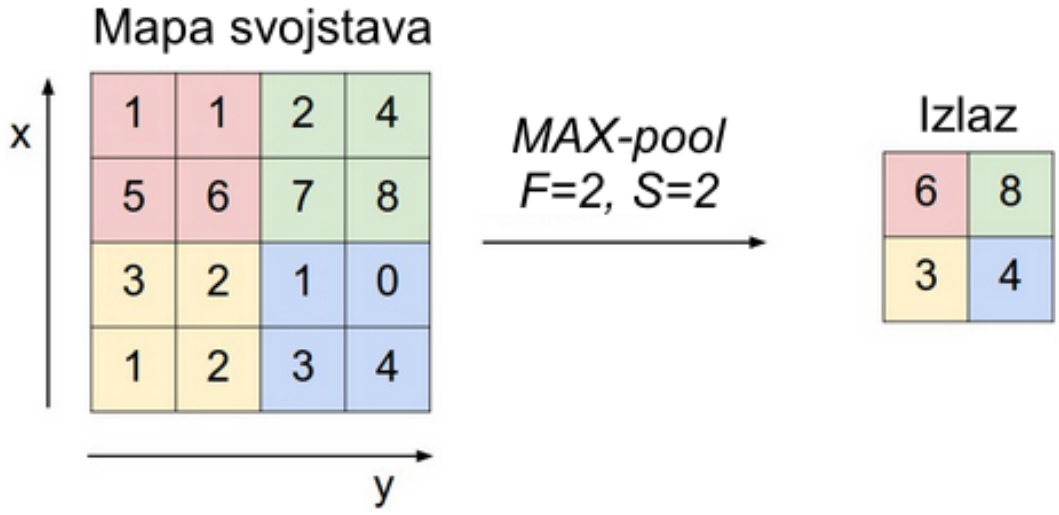
- Prostorni raspon prozora $\boldsymbol{F}$ (visina i širina)
- Veličina pomaka $\boldsymbol{S}$

Dimenzije izlaznog volumena $W_2 \times H_2 \times D_2$ za slojeve sažimanja definirane su sljedećim izrazima:

$$W_2 = \left\lfloor \frac{W_1 - F}{S} \right\rfloor + 1 \qquad H_2 = \left\lfloor \frac{H_1 - F}{S} \right\rfloor + 1 \qquad D_2 = D_1$$

Konkretan tip sloja sažimanja određen je agregatnom funkcijom koja se izvršava nad regijama značajki. Najčešće korištena funkcija je maksimalna vrijednost (MAX-*pooling*) gdje se pojedina regija reducira na značajku sa najvećim iznosom. Druge agregacijske funkcije koje

se ponekad primjenjuju za slojeve sažimanja su srednja vrijednost (AVG-*pooling*) i L2 norma(L2 *norm pooling*) [17]. Princip rada MAX-*pooling* sloja ilustriran je na primjeru 4×4 mape značajki na slici 8. Uobičajne postavke za slojeve sažimanja su $F = 3, S = 2$ (naziva se još i sažimanje s preklapanjem) i $F = 2, S = 2$. Korištenje većih receptivnih polja se ne preporučuje jer je previše destruktivno, odnosno uzrokuje gubitak prevelike količine informacija [18].



Slika 8. MAX-*pooling* sloj s prozorom dimenzija 2×2 i pomakom $S = 2$ [18]

2.4.3 Prijenosno učenje

Duboki modeli zahtjevaju veliku količinu podataka za učenje kako bi postigli prihvatljivu razinu generalizacije pa prikupljanje podataka za specifične zadatke može predstavljati izazov. Prijenosno učenje jedna je od metoda kojom se može smanjiti količina potrebnih podataka iskorištavanjem parametara modela treniranih na velikim skupovima podataka. Neki od velikih skupova podataka koji se koriste na zadacima klasifikacije i detekcije su *ImageNet* [19] (15 milijuna označenih slika za 22 000 kategorija) i COCO [20] (330 000 slika sa preko 1 500 000 instanci objekata označenih za detekciju).

Razlog zašto je ova metoda posebno značajna za konvolucijske modele je postojanje hijerarhijskog odnosa između općenitih značajki nižih slojeva i zadatku specifičnih značajki dubljih slojeva. Uočeno je da svi konvolucijski modeli na najnižem sloju nauče filtere za detekciju orijentiranih rubova (Gabor filteri) i filtere osjetljive na boje. Ova svojstva iskoristiva su u svim zadacima koji spadaju u domenu raspoznavanja sadržaja slike pa se za konvolucijske modele gotovo uvijek koristi predtreniranje na nekom od ranije navedenih

velikih skupova [21]. Modeli predtrenirani na velikim skupovima podataka dio su eko sustava većine modernih knjižnica za duboko učenje.

Primjena prijenosnog učenja na novim zadacima svodi se na dvije najčešće korištene strategije:

1. **Izvlačenje svojstava (engl. *feature extraction*) –** konvolucijska mreža, predtrenirana na velikom skupu podataka, koristi se za dobivanje vektora značajki na izlazu mreže bez dodatnog treniranja za specifični zadatak. Za parametre konvolucijske mreže u ovom slučaju kažemo da su „zamrznuti" odnosno ne ažuriraju se. Značajke dobivene pomoću CNN prosljeđuju se daljnjim slojevima čiji se parametri treniraju za neki specifični zadatak . Najjednostavniji primjer ovog pristupa bio bi prosljeđivanje značajki linearnom klasifikatoru čiji bi se parametri trenirali za klasifikaciju.
2. ***Fine-tuning* parametara –** parametri dobiveni treniranjem na nekom drugom podatkovnom skupu koriste se za inicijalizaciju modela. Ovisno o veličini i sličnosti novog podatkovnog skupa u odnosu na podatke s kojima je model predtreniran moguće je trenirati čitav model ili zamrznuti niže slojeve te trenirati samo dublje slojeve modela. Problem treniranja svih parametara modela na novom skupu je što često vodi prevelikom prilagođavanju podacima za učenje ako je podatkovni skup mali u odnosu na broj parametara modela. Zamrzavanjem većeg broja slojeva smanjuje se broj parametara modela, ali uvodi mogućnost lošijih performansi ako specifična svojstva baznog modela nisu primjenjiva na novi zadatak. U radu [21] ispitivana je prenosivost svojstava u neuronskim mrežama i uočeno je da osim specifičnosti svojstava na višim slojevima, zamrzavanje na razini srednjih slojeva negativno utječe na performanse zbog koadaptacije među svojstvima. Također, pokazano je da modeli koji su inicijalizirani korištenjem prijenosnog učenja postižu bolju generalizaciju u odnosu na nasumično inicijalizirane modele čak i kada se *fine-tuning* parametara specifičnom zadatku radi za čitav model.

2.4.4 Augmentacija podataka

Zadatci vezani uz raspoznavanje sadržaja slika kompliciranu su jer je teško prikupiti skup podataka dovoljno velik da obuhvaća varijabilnosti pozadine, osvjetljenja, skale, zaklonjenosti objekata, točke gledišta i sl. Metode augmentacije podataka služe kako bi se postiglo povećanje robusnosti modela na različite transformacije. Osim ograničene količine podataka

čest problem u ovoj domeni je neravnoteža zastupljenosti različitih kategorija zbog čega modeli budu naklonjeni predviđanju većinski zastupljene katagorije. Tipičan pristup rješavanju ovog problemu je preuzorkovanje (engl. *oversampling*) na podacima manjinske kategorije što također spada u metode augmentacije podataka. Metode za augmentaciju možemo podijeliti na jednostavne manipulacije slikama koje se svode na primjenu geometrijskih i fotometrijskih transformacija te metode bazirane na dubokom učenju [22].

Najčešće korištene metode koje spadaju u jednostavne manipulacije slikama su:

- **Okretanje** (engl. *flipping*) vertikalne ili horizontalne osi slike
- **Nasumično obrezivanje** (engl. *random cropping*) generira nove primjere nasumičnim uzorkovanjem regije u originalnoj slici i proširivanjem na veličinu originalne slike.
- **Translacija** – Slika se pomiče po X i Y osi, prazan prostor nastao pomicanjem popunjava se nulom ili Gaussovim šumom. Cilj je ukloniti pozicijsku pristranost u podacima.
- **Rotiranje –** ulazna slika se rotira za neku vrijednost između 0° i 360°.
- **Manipulacije prostorom boja** – izoliranje R,G ili B kanala, pojačavanje ili smanjivanje intenziteta piksela za određene kanale.
- **Miješanje slika –** dobivanje novih primjera usrednjavanjem vrijednosti piksela više slika.
- **Nasumično brisanje** (engl. *random erasing*) – postavljanje vrijednosti piksela u nasumično odabranoj pravokutnoj regiji ulazne slike na 0, 255 ili nasumične vrijednosti iz Gaussove distribucije. Ideja ove metode je spriječiti ovisnost modela o pojedinačnim svojstvima, čime se povećava robusnost na zaklonjenost i različite točke pogleda.
- **Dodavanje šuma –** empirijski je pokazano da dodavanje nasumičnih vrijednosti dobivenih iz Gaussove distribucije može povećati robusnost naučenih svojstava.

Od metoda baziranih na dubokom učenju najveću raširenost i primjenu ostvarili su generativni modeli. GAN-ovi (engl. *Generative Adversial Network*) generiraju nove umjetne primjere za postojeći podatkovni skup koristeći suparničko učenje. Princip rada ove metode svodi se na treniranje dva suprotna modela, generatora i diskriminatora. Generator je model zadužen za generiranje umjetnih primjera koji se pomiješani sa stvarnim primjerima prosljeđuju diskriminatoru čija je zadaća razlikovanje umjetnih i stvarnih primjera. Osim GAN-ova za

augmentaciju podataka mogu se primjeniti i druge metode kao što neuronski prijenos stila (engl. *neural style transfer*) i meta-učenje. Čitatelja upućujem na [22] za detaljniji pregled primjena ovih metoda.

2.5 Učenje neuronskih mreža

Postupak učenja neuronskih mreža je optimizacijski problem gdje se minimizira funkcija empirijskog rizika $J(\theta)$.

$$J(\theta) = \mathbb{E}_{(x,y)\sim\hat{p}_{podaci}}[L(f(x;\theta),y)] = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta),y_i) \tag{6}$$

Funkcija empirijskog rizika definirana je kao očekivanje izraženo s obzirom na funkciju gubitka L na podatkovnom skupu za treniranje, gdje su (x, y) podaci iz empirijske vjerojatnosne razdiobe $\hat{p}_{podaci}$ [8].

Funkcija gubitka L (engl. *loss function*) kvantificira „nezadovoljstvo“ (pogrešku) dobivenom predikcijom $f(x_i;\theta)$ u odnosu na stvarnu vrijednost y_i za jedan primjer podatkovnog skupa za treniranje. Očekivana vrijednost računa se kao prosječna vrijednost gubitaka za N trening podataka.

Učenje neuronskih mreža razlikuje se od klasičnog pristupa optimizaciji jer funkcija $J(\theta)$ nije dobivena s obzirom na stvarnu distribuciju p_{podaci} nego s obzirom na distribuciju $\hat{p}_{podaci}$ empirijskog skupa podataka. Funkcija koju želimo indirektno optimizirati minimizirajući $J(\theta)$ je nepoznata funkcija $J^*(\theta)$ definirana kao:

$$J^*(\theta) = \mathbb{E}_{(x,y)\sim p_{podaci}}[L(f(x;\theta),y)] \tag{7}$$

Pretpostavka od koje polazimo pri minimizaciji empirijskog rizika je da će parametri θ dobiveni minimizacijom funkcije J dobro generalizirati na sve podatke iz stvarne distribucije p_{podaci}.

Problem sa optimizacijom empirijskog rizika je što se za optimizaciju pretežno koriste metode gradijentnog spusta, a mnoge korisne funkcije gubitka nije moguće efikasno optimizirati jer nisu derivabilne ili im je derivacija 0. Primjer takve funkcije gubitka je $0-1$ gubitak definiran kao:

$$L_{0-1}(y, y_i) = \begin{cases} 0, & ako\ je\ y = y_i \\ 1, & inače \end{cases} \tag{8}$$

Zbog navedenih problema najčešće se kao funkcija gubitka odabire neka zamjenska (engl. *surrogate*) funkcija. Primjer funkcije koja se koristi kao zamjena za $0-1$ gubitak je negativna logaritamska tranformacija izglednosti (engl. *negative log likelihood*).

$$L_{MLE} = -\sum \log p(y = y_i|x;\theta) \tag{9}$$

Prednost ovog oblika funkcije gubitka je općenitost primjene na probleme klasifikacije i regresije [23]. Za regresiju (uz pretpostavku normalne razdiobe) dobivamo srednju kvadratnu pogrešku:

$$p(y = y_i|x;\theta) = \mathcal{N}(y|\mu = f_{y_i}(x_i;\theta), \Sigma = I) \tag{10}$$

$$L_{MSE} = (y_i - y)^2 \tag{11}$$

A za klasifikaciju, uz pretpostavku kategoričke razdiobe, dobivamo izraz:

$$p(y = y_i|x;\theta) = f_{y_i}(x_i;\theta) \tag{12}$$

$$L_{CE} = -\log f_{y_i}(x_i;\theta) \tag{13}$$

Gubitak za klasifikaciju interpretiramo kao unakrsnu entropiju (engl. *cross entropy*) između stvarne distribucije $\boldsymbol{y}$ i predviđene distribucije modela.

2.5.1 Gradijentni spust

Za funkciju $J: \mathbb{R}^n \to \mathbb{R}$ vektor gradijenta $\boldsymbol{\nabla_\theta J(\theta)} = [\frac{J(\theta)}{\partial\theta_1}, \frac{J(\theta)}{\partial\theta_2}, \ldots, \frac{J(\theta)}{\partial\theta_n}]$ sadrži parcijalne derivacije funkcije J, odnosno stopu promjene izlazne vrijednosti J(skalara) s obzirom na infinitezimalne promjene pojedinačnih komponenti vektora $\boldsymbol{\theta} \in \mathbb{R}^n$. Stopa promjene vrijednosti funkcije u smjeru proizvoljnog jediničnog vektora $\vec{\boldsymbol{u}}$ izražava se usmjerenom derivacijom:

$$D_u J(\theta) = \boldsymbol{\nabla_\theta J(\theta)} \cdot \vec{\boldsymbol{u}} = \frac{\partial J(\theta)}{\partial\theta_1} u_1 + \frac{\partial J(\theta)}{\partial\theta_2} u_2 + \cdots + \frac{\partial J(\theta)}{\partial\theta_n} u_n \tag{14}$$

Smjer vektora gradijenta daje smjer najbrže stope rasta funkcije J, a negativni gradijent smjer smjer najbrže stope pada što vidimo iz sljedećeg izraza:

$$\min_{\vec{u}} \nabla_{\theta} J(\theta) \cdot \vec{u} = \min_{\vec{u}} \|\nabla_{\theta} J\| \|\vec{u}\| cos\beta \tag{15}$$

S obzirom da $\|\vec{u}\| = 1$, a član $\|\nabla_{\theta} J(\theta)\|$ nije ovisan o odabiru $\vec{u}$ gornji izraz svodi se na pronalaženje $\min_{\vec{u}} cos\beta$.

Uzimanjem malih koraka u smjeru negativnog gradijenta algoritam predlaže nove točke $\theta' = \theta - \eta \cdot \nabla_{\theta} J(\theta)$ gdje je η hiperparametar stope učenja (engl. *learning rate*) koji određuje veličinu koraka. Kretanjem u smjeru točke θ' krećemo se u smjeru lokalne informacije o najvećoj stopi pada funkcije.

Gradijentni spust je temeljni algoritam za učenje neuronskih mreža, koji se koristi za minimizaciju empirijskog rizika. Ovisno o veličini uzorka iz skupa za učenje na kojem se izračunava gradijent razlikujemo tri varijante gradijentnog spusta:

Grupni gradijentni spust (engl. *batch gradient descent*) – gradijent se računa koristeći sve primjere skupa za učenje. Ovaj pristup tipično vodi preciznijem kretanju u smjeru lokalnog minimuma. Problem grupnog pristupa je nedostatak regularizacije i sublinearno poboljšanje aproksimacije s obzirom na povećanje veličine uzorka [24]. Za jedno ažuriranje parametara potrebno je napraviti unaprijedni i unatražni prolaz za sve primjere skupa za učenje. Osim što je računski neefikasno, u dubokim arhitekturama često je i neizvedivo zbog broja parametara koje je potrebno pohranjivati u memoriji računala.

Stohastički gradijentni spust (engl. *stohastic gradient descent SGD*) – varijanta gradijentnog spusta u kojoj se parametri modela ažuriraju s obzirom na gradijent za pojedinačni primjer iz skupa za učenje. S obzirom da je gradijent aproksimiran na samo jednom primjerku uvodi se komponenta šuma koja djeluje regularizacijski što algoritam čini robusnijim na zapinjanje u lokalnim minimumima. Da bi SGD konvergirao potrebno je prilagoditi stopu učenja, odnosno postupno je smanjivati čime se smanjuje utjecaj šuma za aproksimaciju gradijenta.

Gradijentni spust s mini-grupama (engl. *minibatch gradient descent*) – poseban slučaj stohastičkog gradijentnog spusta gdje je veličina uzorka na kojem se aproksimira gradijent

veća od 1 (tipično potencija broja 2 u rasponu od 16 do 256 ovisno o broju parametara modela). Veličina uzorka je hiperparametar koji se naziva veličina grupe (engl. *batch size*). Što je veličina grupe veća gradijenti su precizniji (može se uzimati veće korake za stopu učenja), manje veličine grupe imaju više šuma i djeluju regularizacijski. Ova varijanta gradijentnog spusta najčešća je u kontekstu optimizacije neuronskih mreža jer ujedinjuje dobre strane gornjih metoda, ali uvodi dodatne izazove po pitanju konvergencije:

- Odabir stope učenja značajno utječe na brzinu konvergencije, premale stope učenja usporavaju konvergenciju, a previsoke stope učenja mogu uzrokovati fluktuacije oko lokalnog minimuma ili divergenciju.
- Uvođenjem prilagođavanja stope učenja (kaljenje, engl. *annealing*) tijekom optimizacije na temelju predodređenog rasporeda (engl. *learning rate scheduling*) proces učenja ne prilagođava se karakteristikama podatkovnog skupa [25].
- S obzirom da je u neuronskim mrežama funkcija koja se optimira izrazito nekonveksna, gradijentni spust ima problem zapinjanja u lokalnim minimumima i točkama sedla (što je češće u visokodimenzionalnom prostoru prema [26])

Gore navedenim izazovima pristupa se kroz napredne varijante algoritama temeljenih na gradijentnom spustu s mini grupama koji su opisani u idućim poglavljima. Konkretno, fokus je stavljen na dvije skupine koje su najčešće korištene u kontekstu dubokog učenja, algoritme koji koriste moment i algoritme sa adaptivnim stopama učenja.

2.5.2 Algoritmi učenja s momentom

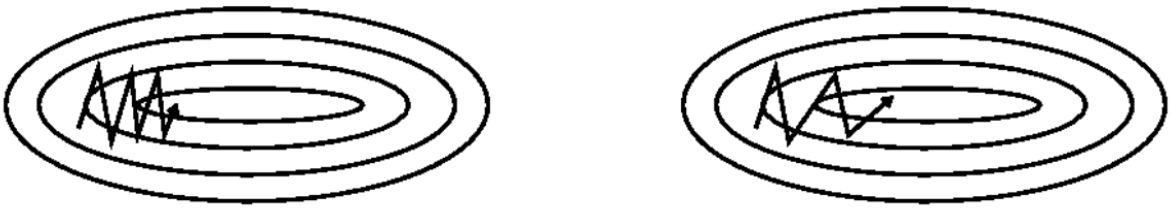
Za visokodimenzionalne funkcije kakve se optimiziraju u kontekstu neuronskih mreža funkcije oko lokalnih minimuma često poprimaju oblik usjeka (engl. *ravine*) [25]. Ovo svojstvo karakterizira velika razlika u zakrivljenosti (engl. *curvature*) s obzirom na različite dimenzije, odnosno kažemo da je zakrivljenost loše uvjetovana (engl. *bad conditioning*). U takvom području potrebno je uzimati veće korake u smjeru gdje je zakrivljenost mala kako bi se ubrzala konvergencija i manje korake u smjeru visoke zakrivljenosti kako bi se spriječile oscilacije. Zbog načina na koji SGD ažurira parametre, događa se suprotno od željenog, algoritam uzima velike korake u smjeru visoke zakrivljenosti i male korake u smjeru niske zakrivljenosti.

Moment je metoda koja omogućuje ubrzavanje kretanja SGD algoritmom prema minimumu i smanjuje oscilacije u područjima loše uvjetovane zakrivljenosti. Vektor ažuriranja parametara za trenutni korak računa se kao zbroj dijela prethodnih vektora ažuriranja. Udio prethodnog vektora određen je hiperparametrom γ čija se vrijednost naziva i iznosom momenta. Često korištena vrijednost za član $\gamma = 0.9$. Vektor ažuriranja u trenutku t računa se prema sljedećim izrazima:

$$\boldsymbol{v_t} = \gamma \boldsymbol{v_{t-1}} + \eta \, \boldsymbol{\nabla_\theta J(\theta)} \tag{16}$$

$$\boldsymbol{\theta'} = \boldsymbol{\theta} - \boldsymbol{v_t} \tag{17}$$

Utjecaj momenta uzrokuje povećanje iznosa ažuriranja za dimenzije gdje su gradijenti istog smjera, a smanjuje iznos ažuriranja za dimenzije gdje gradijenti mijenjaju smjer. Rezultat primjene ove metode ilustriran je na slici 9 i prikazuje prigušenje oscilacija i bržu konvergenciju za područja loše uvjetovane zakrivljenosti.



Slika 9. Klasični SGD algoritam (lijevo) i brža konvergencija zbog primjene SGD + momenta (desno) [27]

Kako bi dodatno ubrzali konvergenciju često se koristi „pametnija“ verzija algoritma s metodom momenta **NAG (engl. *Nesterov accelerated gradient*)** koja iskorištava činjenicu da u trenutku kada se računa vektor ažuriranja algoritam već ima grubu aproksimaciju $\boldsymbol{\theta - \gamma v_{t-1}}$ za sljedeće vrijednosti parametara. Umjesto računanja vektora ažuriranja na temelju gradijenta za trenutnu vrijednost $\boldsymbol{\nabla_\theta J(\theta)}$, iskorištava se ova aproksimacija i računa iznos ažuriranja pomoću gradijenta $\boldsymbol{\nabla_\theta J(\theta - \gamma v_{t-1})}$ [25].

$$\boldsymbol{v_t} = \gamma \boldsymbol{v_{t-1}} + \eta \, \boldsymbol{\nabla_\theta J(\theta - \gamma v_{t-1})} \tag{18}$$

$$\boldsymbol{\theta'} = \boldsymbol{\theta} - \boldsymbol{v_t} \tag{19}$$

Ovom prilagodbom algoritam dobiva mogućnost „gledanja unaprijed“ čime se postiže dodatna robusnost i spriječava oscilacije pri usporavanju u minimum uzrokovane prevelikim nakupljanjem momenta.

2.5.3 Algoritmi s adaptivnim stopama učenja

U metodama koje koriste moment stopa učenja je fiksna što može dovesti do problema u konvergenciji kada određena svojstva imaju malu frekvenciju u podatkovnom skupu jer se parametri modela vezani uz ta svojstva rijeđe ažuriraju. Algoritmi s adaptivnim stopama učenja prilagođavaju stopu učenja tako da paramatre povezane sa svojstvima koja se često pojavljuju ažuriraju manjom stopom učenja, a parametre povezane sa rijetkim svojstvima većim stopama učenja.

Adagrad algoritam realizira ovu ideju tako da pri računanju vektora ažuriranja koristi različite stope učenja za svaki parametar θ_i u koraku t. Informaciju o prethodnim ažuriranjima održava se u obliku dijagonalne matrice G_t koja sadrži sume kvadrata svih prethodnih gradijenata u trenutku t i koristi kako bi se prilagodila stopa učenja pojedinačnih parametara. Član matrice G_t indeksa (i, i) je vrijednost za parametar θ_i i u daljnjem opisu označavati ćemo ga sa $G_{t,ii}$.

Radi sažetosti zapisa i-ti član gradijenta $\boldsymbol{g_t = \nabla_\theta J(\theta)}$ u trenutku t, odnosno parcijalnu derivaciju $\frac{\partial J(\theta)}{\partial \theta_i}$ označavati ćemo sa $g_{t,i}$.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}}\, g_{t,i} \tag{20}$$

ϵ u ovom izrazu predstavlja konstantu koja se koristi za izbjegavanje numeričkih nestabilnosti (tipično se postavlja na malu vrijednost poput 10^{-8}). Ovu matematičku operaciju možemo vektorizirati za sve parametre sljedećim izrazom, gdje $\odot$ označava množenje između matrice i vektora po elementima:

$$\boldsymbol{\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t} \tag{21}$$

Problem **Adagrad** algoritma je što se akumuliranjem gradijenata stopa učenja agresivno monotono smanjuje pa nakon nekog vremena stopa učenja postane premala zbog čega

algoritam prestaje učiti. Algoritmi **Adadelta i RMSProp** (razvijeni u približno isto vrijeme) ovaj problem rješavaju tako što informaciju o prošlim gradijentima ne održavaju kao akumulirajuću sumu kvadrata gradijenta, nego kao eksponencijalni pomični prosjek EMA (engl. *Exponential Moving Average*) kvadrata gradijenta. Na ovaj način s odmakom vremena utjecaj starijih gradijenata opada, ali nikada ih se potpuno ne odbacuje.

Za **RMSProp** algoritam, vektor ažuriranja dobivamo sljedećim izrazima:

$$EMA[\boldsymbol{g}^2]_t = \gamma EMA[\boldsymbol{g}^2]_{t-1} + (1-\gamma)\boldsymbol{g}_t^2 \tag{22}$$

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \frac{\eta}{\sqrt{EMA[\boldsymbol{g}^2]_t + \epsilon}}\boldsymbol{g_t} \tag{23}$$

Tipične početne vrijednosti hiperparametara [25] su $\gamma = 0.9$ i $\eta = 0.001$, prema preporuci autora algoritma.

Adam algoritam kombinira RMSProp i učenje s momentom što također rezultira adaptivnim stopama učenja za svaki parametar. Ovaj algoritam pohranjuje EMA za gradijente (moment) i kvadrate gradijenata (RMSProp) parametara.

$$\boldsymbol{m_t} = \beta_1 \boldsymbol{m_{t-1}} + (1-\beta_1)\boldsymbol{g_t} \tag{24}$$

$$\boldsymbol{v_t} = \beta_2 \boldsymbol{v_{t-1}} + (1-\beta_2)\boldsymbol{g_t^2} \tag{25}$$

$\boldsymbol{m_t}$ je estimacija srednje vrijednosti gradijenta (prvi moment), a $\boldsymbol{v_t}$ varijance gradijenata (drugi moment) za parametre $\boldsymbol{\theta}$. Konačni izraz za vektor ažuriranja dobivamo uz primjenu korekcija $\widehat{\boldsymbol{m}}_{\boldsymbol{t}}$ i $\widehat{\boldsymbol{v}}_{\boldsymbol{t}}$ kako bi se kompenzirala naklonjenost algoritma prema vrijednosti 0 kada su hiperparametri opadanja β_1 i β_2 blizu vrijednosti 1. Preporuke autora za početne vrijednosti su $\beta_1 = 0.9$ i $\beta_1 = 0.999$ [28].

$$\widehat{\boldsymbol{m}}_{\boldsymbol{t}} = \frac{\boldsymbol{m_t}}{1-\beta_1} \qquad \widehat{\boldsymbol{v}}_{\boldsymbol{t}} = \frac{\boldsymbol{v_t}}{1-\beta_2}$$

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \frac{\eta}{\sqrt{\widehat{\boldsymbol{v}}_{\boldsymbol{t}} + \epsilon}}\widehat{\boldsymbol{m}}_{\boldsymbol{t}} \tag{26}$$

U originalnom radu [28] u kojem je predstavljen Adam algoritam autori navode empirijske rezultate koji ukazuju da Adam kao i NAG metoda konvergiraju puno brže od Adagrad

algoritma za konvolucijske neuronske mreže. Kao razlog autori navode da vrijednost estimacije drugog momenta $\boldsymbol{v_t}$ nestaje nakon par epoha, odnosno poprima vrijednosti blizu 0 što znači da je drugi moment loša estimacija geometrije funkcije troška CNN u odnosu na mrežu sa FC slojevima [28].

Iako autori navode da je korištenje Adam algoritma marginalno bolje od NAG metode postoje pokazatelji da adaptivne metode konvergiraju u značajno drugačije minimume od SGD i SGD + moment metoda. Autori rada [29] izložili su hipotezu i priložili empirijske dokaze da optimizacija adaptivnim metodama za CNN postiže lošije rezultate od korištenja SGD + moment metoda, te lošiju generalizaciju na testnom skupu čak i kada postiže bolje rezultate na skupu za učenje. Iako u ranijim fazama treniranja (do 50 epoha) izgleda kao da su adaptivne metode superiorne promatrajući grešku na skupu za učenje do 100. epohe SGD + moment metode su postigle značajno bolje rezultate pri evaluaciji na oba skupa.

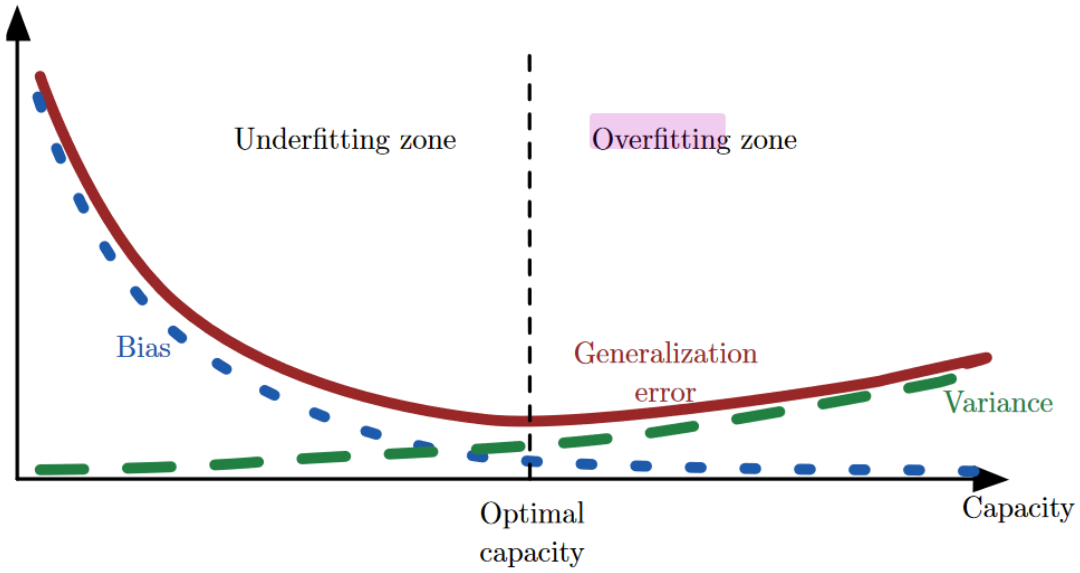
2.5.4 Regularizacija

Konačni cilj procesa učenja je dobiti model koji ostvaruje dobre rezultate na novim podacima. Ovo svojstvo povezano je sa minimizacijom greške generalizacije, zadane izrazom (7). Za potrebe evaluacije modela grešku generalizacije aproksimiramo na manjem dijelu izdvojenih empirijskih podataka koje ne koristimo u procesu učenja.

Ako model ne može naučiti obrasce prisutne u podacima za učenje, kažemo da je model podnaučen (engl. *underfitting*) ili da ima visoku pristranost (engl.*bias*). Rješenje ovog problema je tipično povećati kapacitet modela ili izdvajanje svojstava sa većom prediktivnom moći. Duboki modeli imaju velik broj parametara i shodno tome visok kapacitet pa je češći slučaj da je model prenaučen (engl. *overfitting*). Za ove modele kažemo da su previše prilagođeni podacima za učenje odnosno da imaju visoku varijancu. Prenaučeni modeli ne generaliziraju dobro pa ih karakterizira značajna razlika između performanci na podacima za učenje i evaluacijskim podacima [30].

Regularizacija obuhvaća metode koje se koriste da bi se proces učenja prilagodio i dobio jednostavniji model. Primjenom ovih metoda tipično se povećava pristranost modela, ali i smanjuje varijanca pa kažemo da optimalan model pronalazimo na temelju kompromisa pristranosti i varijance (engl. *bias-variance tradeoff*) ilustriranog na slici 10.

Zbog ranije navedenih specifičnosti procesa učenja u odnosu na klasičnu optimizaciju , postupak učenja ne zaustavlja se u lokalnom minimumu nego na osnovu kriterija konvergencije ranog zaustavljanja (engl. *early stoppage*). Ovaj kriteriji temelji se na povremenoj evaluaciji modela na validacijskom skupu podataka kako bi se proces učenja prekinuo prije nego se model previše prilagodi podacima za učenje što ga također uvrštava u metode regularizacije [8].



Slika 10. Graf ilustrira da se optimalni model nalazi kao kompromis između pristranosti i varijance modela, odnosno između zone prenaučenosti i podnaučenosti [8]

Jedna od jednostavnijih metoda regularizacije je uvođenje dodatnog člana $\lambda R(\theta)$ u izraz empirijskog rizika:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta), y_i) + \lambda R(\theta) \tag{27}$$

gdje je λ hiperparametar koji se naziva snaga regularizacije, a $R(\theta)$ funkcija koja penalizira parametre modela. U kontekstu neuronskih mreža često se koriste $L1$ i $L2$ norma parametara.

Druge često korištene metode za regularizaciju su slučajno izostavljanje neurona (engl. *dropout*) i *batch* normalizacija. *Dropout* je metoda gdje se hiperparametrom $p \in [0, 1]$ zadaje vjerojatnost odbacivanja (ili zadržavanja) neurona pri računanju izlaznih vrijednosti modela čime se postiže efekt treniranja ansambla podmreža koje se mogu dobiti isključivanjem pojedinih jedinica. Ovom metodom uvodi se destruktivna komponenta maskirajućeg šuma

koja djeluje uklanjajući naučena svojstva što intuitivno kažnjava modele koji ovise o pojedinim svojstvima i poboljšava generalizaciju [8].

Batch normalizacija je metoda kojom se ulazne vrijednosti sloja normaliziraju na temelju statistika mini-grupe. Za ulaz $\boldsymbol{x}$ koji pripada mini-grupi B izlazna vrijednosti normalizacijskog sloja je:

$$BN(\boldsymbol{x}) = \boldsymbol{\gamma}\frac{\boldsymbol{x} - \boldsymbol{\mu_B}}{\boldsymbol{\sigma_B}} + \boldsymbol{\beta} \tag{28}$$

$\boldsymbol{\mu_B}$ je srednja vrijednost, a $\boldsymbol{\sigma_B}$ standardna devijacija dobivena na uzorku grupe. Parametar skaliranja $\boldsymbol{\gamma}$ i pomaka $\boldsymbol{\beta}$ su učivi parametri modela [16].

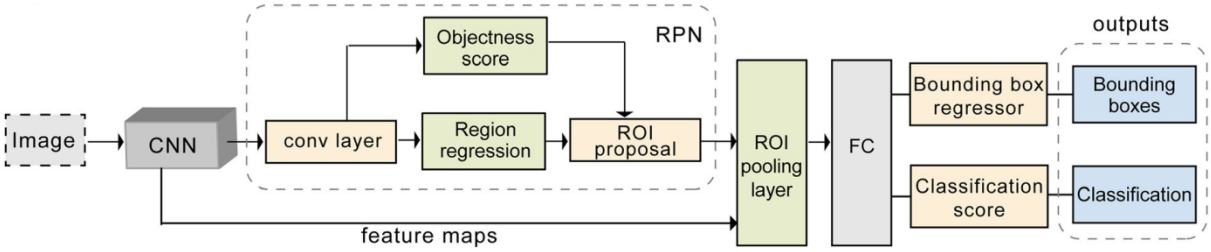
Osim navedenih matematičkih metoda i ranije spomenuta augmentacija podataka spada u regularizacijske metode [30]. Odabirom određenih hiperparametara kao što su više stope učenja i manja veličina mini-grupa također se postiže regularizacijski efekt.

2.6 Modeli za detekciju i klasifikaciju objekata

2.6.1 *Faster* RCNN arhitektura

Princip rada RCNN (engl. *Region-based Convolutional Neural Networks*) metoda može se podijeliti na dvije faze:

1. Generiranje regija kandidata
2. Klasifikacija i regresija okvira na temelju predloženih regija



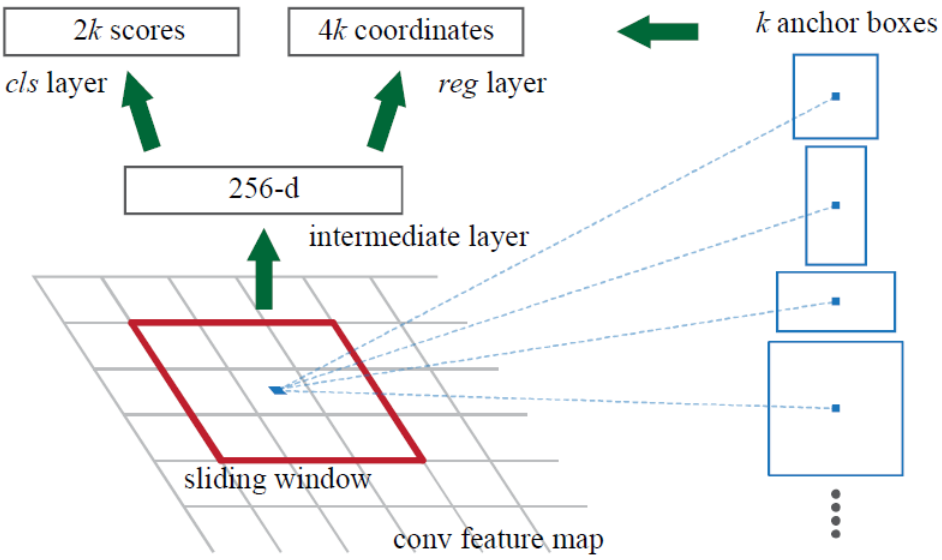
Slika 11. Arhitektura *Faster* RCNN mreže [31]

Faza generiranja regija kandidata je u ranijim RCNN modelima (RCNN, *Fast* RCNN) predstavljala usko grlo. *Faster* RCNN model uvodi promjenu na razini dizajna mreže (Slika

11) zamjenom funkcije za prijedlog regija koja je radila na originalnog ulaznoj slici sa potpuno konvolucijskom mrežom RPN (engl. *Region Proposal Network*). RPN na temelju mapa značajki dobivenih *backbone* CNN mrežom uči generirati prijedloge za regije. Prijedlozi regija zatim se prosljeđuju u *Fast* RCNN detektor.

RPN se $n \times n$ klizećim prozorom kreće po ulaznim mapama značajki, za svaki prozor generira 1D vektor (256-d za ZF mrežu ili 512-d za VGG mrežu) koji se prosljeđuje u dva potpuno povezana sloja: sloj za klasifikaciju okvira *cls* i sloj za regresiju okvira *reg*. Ova arhitektura je potpuno implementirana konvolucijskim slojevima. Odnosno jednim $n \times n$ konvolucijskim slojem za dobivanje odgovarajućeg 1D vektora i 1×1 konvolucijskim slojevima za *cls* i *reg* slojeve. Princip rada RPN mreže ilustriran je na slici 12.

Faster RCNN koristi bazne okvire (engl. *anchor boxes*) na svakoj prostornoj lokaciji mape značajki. Broj baznih okvira na svakoj lokaciji označvamo sa k, a svaki bazni okvir definiran je sa skalom i omjerom stranica. Faster RCNN u originalnom radu upotrebljava 3 skale i 3 omjera stranica što daje ukupno $k = 9$ kombinacija baznih okvira na jednoj prostornoj lokaciji. Korištenje više skala za bazne okvire je ključna komponenta ovog pristupa jer omogućuje korištenje istih mapa značajki za detekciju objekata različite veličine. Za mapu značajki veličine $W \times H$ ima ukupno $W \cdot H \cdot k$ baznih okvira [32].



Slika 12. Ilustracija principa rada RPN mreže gdje se klizećim prozorom generira 256-d vektor na temelju kojeg se dobiva *score* objektnosti i računa regresija koordinata za k baznih okvira [32]

Na izlazu klasifikacijskog sloja dobiva se predikcija 2 *score*-a koji se *softmax* slojem pretvaraju u vjerojatnosti da je bazni okvir objekt ili pozadina. Ukupni broj *score*-ova na izlazu klasifikacijskog sloja je $2k$. Za regresijski sloj radi se predikcija 4 parametra $(\Delta cx, \Delta cy, \Delta h, \Delta w)$ koji određuju posmak okvira relativno na bazni okvir za konačni prijedlog kandidata.

Podudaranje između baznih okvira i *ground truth* okvira radi se dodjelom labela (pozitivnih i negativnih) na osnovu sljedećih kriterija:

A. Pozitivna labela dodjeljuje se baznim okvirima koji:
   i. Ostvare najviše IoU podudaranje s obzirom na *ground truth* okvir
   ili
   ii. Imaju IoU preklapanje više od 0.7 u odnosu na *ground truth* okvir
B. Negativna labela dodjeljuje se baznim okvirima koji nisu pozitivni i zadovoljavaju uvjet IoU preklapanja manjeg od 0.3 u odnosu na sve *ground truth* okvire

Bazni okviri kojima nije dodjeljena ni pozitivna ni negativna labela ne uzimaju se u obzir pri računanju funkcije gubitka. Funkcija gubitka koju minimiziramo u kontekstu RPN mreže dana je kao otežana suma gubitka klasifikacije L_{cls} i gubitka lokalizacije L_{reg} :

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{29}$$

$$L_{reg}(t_i, t_i^*) = smooth_{L1}(t_i - t_i^*) \tag{30}$$

p_i – predviđena vjerojatnost da i-ti bazni okvir predviđa objekt

p_i^* - *ground truth* vjerojatnost, 1 za pozitivne i 0 za negativne bazne okvire

t_i – predstavlja odstupanje 4 koordinate $i \in (x, y, w, h)$ predviđenog okvira u odnosu na bazni okvir

t_i^* - predstavlja odstupanje 4 koordinate *ground truth* okvira $i \in (x, y, w, h)$ s obzirom na bazni okvir. Množenjem elementom p_i^* negativni okviri se ne uzimaju u obzir pri računanju gubitka za regresiju.

Gubitak koji je zadan izrazom (29) naziva se gubitak više zadataka (engl. *multi-task loss*) jer daje jednu izlaznu vrijednost (skalar) za kombinaciju zadataka klasifikacije i regresije okvira. Hiperparametrom λ regulira se utjecaj zadatka regresije okvira na ukupni gubitak. Za RPN mrežu L_{cls} je gubitak za binarnu klasifikaciju (objekt ili pozadina), a za *Fast* RCNN detektor

L_{cls} je gubitak višeklasne klasifikacije. Zbog jednostavnosti implementacije obje funkcije gubitka računaju se kao unakrsna entropija između diskretnih vjerojatnosnih distribucija.

S obzirom da se arhitektura *Faster* R-CNN mreže sastoji od RPN mreže i *Fast* R-CNN mreže za detekciju potrebno je osmisliti način na koji će se cjelokupni model trenirati odnosno kako postići dijeljenje parametara pri optimizaciji za generiranje regija kandidata i detekciju. U originalnom radu u kojem je predstavljena ova arhitektura autori predlažu alternirajući pristup treniranju od 4 koraka:

1. RPN mreža se trenira za generiranje regija kandidata. Parametri za *backbone* CNN mrežu su inicijalizirani na vrijednosti predtrenirane na *ImageNet* skupu podataka. Parametri za konvolucijske slojeve specifične RPN mreži inicijaliziraju se iz normalne distribucije $N\sim(0, 0.01^2)$.
2. Koristeći regije kandidate dobivene treniranom RPN mrežom kao ulaz, trenira se *Fast* RCNN mreža za detekciju. Parametri za *backbone* CNN mrežu (odvojeno od CNN mreže koja se koristi u prvom koraku) inicijaliziraju se na vrijednosti predtrenirane na *ImageNet* skupu podataka. Potpuno povezani slojevi *Fast*-RCNN detektora se inicijaliziraju iz normalne distribucije $N\sim(0, 0.01^2)$.
3. Fiksiraju se vrijednosti *backbone* CNN mreže nakon čega se radi *fine-tuning* konvolucijskih slojeva specifičnih za RPN mrežu. CNN mreža je inicijalizirana na vrijednosti parametara dobivenim treniranjem *Fast*-RCNN detkecijske mreže drugog koraka čime se uvodi dijeljenje parametara *backbone* CNN mreže za RPN i detekcijsku mrežu.
4. U posljednjem koraku radi se *fine-tuning* potpuno povezanih slojeva detekcijske mreže pri čemu su dijeljeni CNN parametri fiksirani na vrijednosti dobivene prethodnim koracima.

2.6.2 SSD arhitektura

SSD (engl. *Single Shot Multibox Detector*) je metoda detekcije objekata koja koristi skup unaprijed zadanih baznih okvira, različitih omjera stranica. Ovom diskretizacijom prostora mogućih okvira eliminira se potreba za fazom generiranja regija kandidata, koja zbog računske složenosti predstavlja ograničavajući čimbenik.

Princip rada SSD metode se temelji na predviđanju *score*-ova za kategorije $(c_0, \dots, c_K)$ i potrebnu promjenu oblika (cx, cy, h, w) za svaki pojedini bazni okvir. *Score* za kategoriju c_0 se definira kao pozadina, odnosno predstavlja neprisutnost objekta.

Broj baznih okvira za sloj ulaznog volumena $H_1 \times W_1 \times C_1$ je

$$broj\ okvira = H_1 \cdot W_1 \cdot B \tag{31}$$

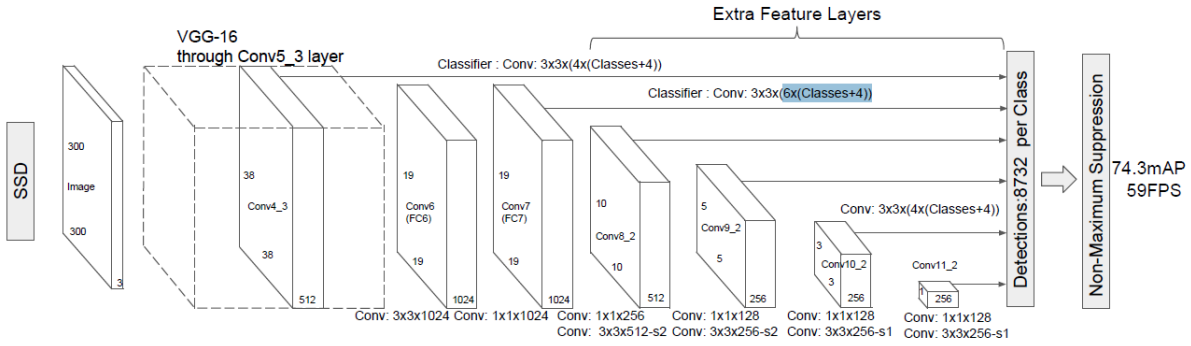
, a broj predikcija

$$H_1 \cdot W_1 \cdot B((K+1)+4) \tag{32}$$

Gdje je B hiperparametar broja baznih okvira, K broj kategorija koji se predviđa, a $K+1$ se dobiva uključujući i pozadinsku kategoriju koja označava odsutnost objekta.

Ulazni volumen mapa značajki dobivenih iz *backbone* CNN propušta se kroz dodatne konvolucijske slojeve radi poduzorkovanja kako bi omogućilo predviđanje objekata na različitim skalama. Značajke dubljih slojeva imaju veće receptivno polje pa su bolje prilagođene za predviđanje većih objekata. Na ulaznom volumenu svakog konvolucijskog sloja primjenjuje se $3 \times 3 \times B(K+4)$ konvolucija čime se generiraju predikcije. Ukupan broj baznih okvira u SSD arhitekturi prikazanoj na slici 13 je 8732 [33].

Pri treniranju prvo se određuju bazni okviri koji se podudaraju (engl. *matching*) sa *ground truth* okvirima. Podudaranje se vrši na temelju Jaccardovog indeksa, odnosno omjera presjeka i unije IoU (engl. *Intersection over Union*) između dva okvira. Ako je IoU veći od granice ($\text{IoU} > 0.5$) onda kažemo da se okviri podudaraju.



Slika 13. SSD arhitektura sa VGG-16 *backbone* mrežom [33]

Funkcija gubitka koja se koristi definirana je kao otežana suma gubitka lokalizacije L_{loc} i gubitka pouzadnosti (klasifikacije) L_{conf} .

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right) \tag{33}$$

- N je broj baznih okvira koji se podudaraju
- $L_{loc}\ (x, l, g)$ je funkcija gubitka za lokalizaciju koja kvantificira pogrešku u predikciji potrebnog pomaka l za bazni okvir d i stvarnog pomaka između baznog i *ground truth* okvira g.

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, h, w\}} x_{ij}^{k} smooth_{L1}\left(l_i^m - \hat{g}_j^m\right) \tag{34}$$

$$\hat{g}_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w} \qquad \hat{g}_j^{cy} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h} \qquad \hat{g}_j^h = log(\frac{g_j^h}{d_i^h}) \qquad \hat{g}_j^w = log(\frac{g_j^w}{d_i^w})$$

- $L_{conf}(x, c)$ funkcija gubitka pouzdanosti koja kvantificira nezadovoljstvo sa dobivenim *scoreovima* pouzdanosti za kategorije c. Korištenjem *softmax* funkcije *scoreovi* se pretvaraju u diskretnu vjerojatnosnu distribuciju, nakon čega se računa unakrsna entropija sa *ground truth* distribucijom.

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \tag{35}$$

$$\hat{c}_i^p = \frac{\exp\left(c_i^p\right)}{\sum_p \exp(c_i^p)} \tag{36}$$

Omjer stranica (engl. *aspect ratio*) baznih okvira zadaje se kao hiperparametar $a_r = \{1,2,3,\frac{1}{2},\frac{1}{3}\}$, a skala određuje u ovisnosti o dubini. Skala se povećava sa dubinom mreže jer su dublji slojevi bolje prilagođeni detekciji većih objekata. Skala s_k za mape značajki na kojima se radi predikcija k-tog sloja $k \in \{1, \dots, m-1]$ izračunava se izrazom:

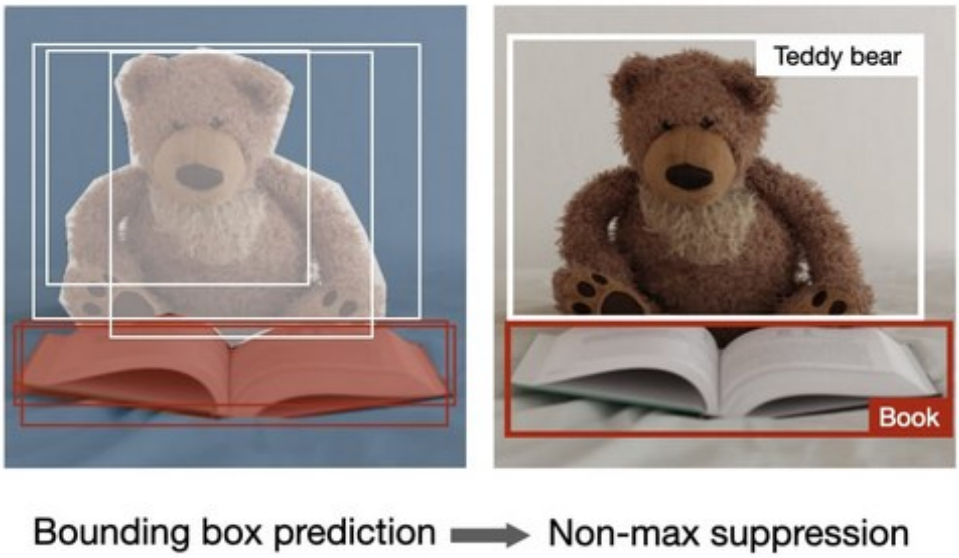
$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1) \tag{37}$$

$$s_{min} = 0.2, s_{max} = 0.9$$

Vrijednosti širine w i visine h za bazne okvire zatim se izračunavaju:

$$w = s_k\sqrt{a_r} \qquad h = \frac{s_k}{\sqrt{a_r}} \tag{38}$$

S obzirom da SSD generira veliki broj baznih okvira većina se ne podudara sa objektima. Ovaj nesrazmjer između broja negativnih i pozitivnih podudaranja uzrokuje da negativni primjeri dominiraju funkcijom gubitka pa model favorizira neprepoznavanje objekata (odnosno uči prepoznavati pozadinu umjesto objekata). Kako bi se izbjegao ovaj problem koristi se *hard negative mining*. Negativne primjere se filtrira te zadržava one koji imaju najveći gubitak za pouzdanost tako da omjer negativnih i pozitivnih primjera bude najviše 3:1.



Slika 14. Primjer uklanjanja redundantnih okvira primjenom *non-maximum supression* metode [15]

Posljednji korak, koji spada u naknadnu obradu dobivenih detekcija je NMS (engl. *non-maximum supression*), kojim se za detekcije čiji se okviri preklapaju više od neke IoU granice (primjerice $IoU > 0.6$) zadržava samo one koji imaju najviši *score* pouzdanosti. Ovim postupkom se uklanja redundantne detekcije koje su posljedica velikog broja baznih okvira što je ilustrirano na slici 14.

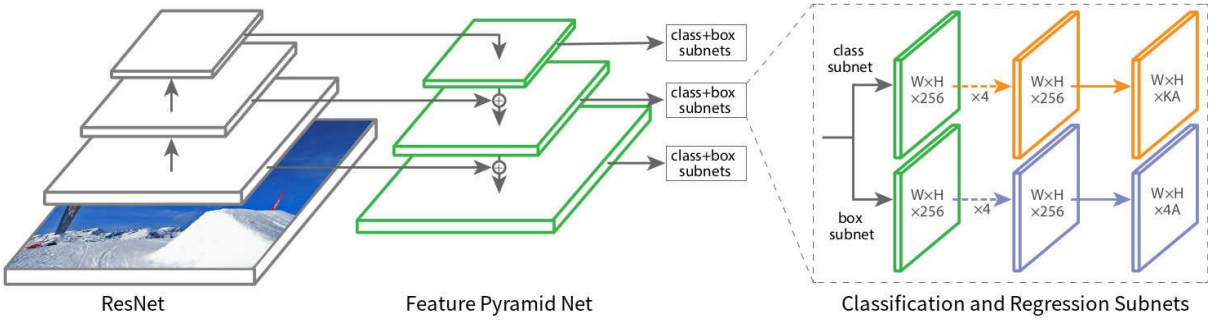
2.6.3 RetinaNet arhitektura

Problem neuravnoteženosti između količine negativnih primjera u odnosu na pozitivne značajno utječe na efikasnost učenja jednofaznih detektora. RetinaNet uvodi alternativni

pristup za rješavanje ovog problema na razini funkcije gubitka, koji je efektivniji od klasičnih metoda kao što je ranije spomenuti *hard negative mining*. Uvođenjem faktora skaliranja u izraz za unakrsnu entropiju dinamički se smanjuje vrijednost gubitka prema 0 s povećanjem pouzadnosti za predviđenu klasu. Ovim se smanjuje doprinos primjera sa visokom pouzdanosti za klasifikaciju (lagani primjeri) i postiže da se model fokusira na učenje teških primjera.

Razmatrajući RetinaNet na visokoj razini, arhitekturu možemo podijeliti na tri podmreže prikazane na slici 15:

1. *Backbone* FPN (engl. *Feature Pyramid Network*) mreža s *ResNet* osnovicom za dobivanje mapa značajki različitih skala iz ulazne slike.
2. Podmreža za klasifikaciju baznih okvira
3. Podmreža za regresiju baznih okvira



Slika 15. Arhitektura RetinaNet sa *ResNet backbone* mrežom i FPN-om [34]

Dobivanje mapa značajki različitih skala uvođenjem bočnih veza je unapređenje koncepta korištenog u SSD detektoru. Podsjećamo, SSD je predikcije generirao za mape značajki više slojeva s obzirom da značajke dubljih slojeva imaju veće receptivno polje i bolje su prilagođene za predviđanje većih objekata. Ovaj prolaz ulazne slike kroz ResNet mrežu nazivamo prolazom „odozdo prema gore“ (engl. *bottom-up pathway*). Problem ovog pristupa je što mape značajki dubljih slojeva sadrže više semantičkih informacija, koje onda nije moguće iskoristiti za predikcije plićih slojeva.

FPN pristupa ovom problemu dodavanjem prolaza „odozgo prema dolje“ (engl. *top-down pathway*). Počevši od posljednje dobivene mape značajki, naduzorkovanjem (metodom najbližih susjeda) dobiva se skala koja odgovara pretposljednjoj mapi značajki prolaza „odozdo prema gore“. Odgovarajuće mape se zatim zbrajaju po elementima čim se dobiva

nova mapa značajki koja se prosljeđuje podmrežama za klasifikaciju i regresiju okvira. Na slici 15 ova operacija je predstavljena bočnim vezama. Proces se iterativno nastavlja dok svakoj mapi značajki generiranoj tijekom prolaza „odozdo prema gore“ nije pridružena nova mapa značajki. Ovime se uz mali dodatni trošak po pitanju računske složenosti dobiva semantički i prostorno jake mape značajki na osnovu kojih se rade predikcije.

Klasifikacijska podmreža je potpuno konvolucijska mreža čije ulaze čini svaka razina FPN mreže. Sastoji se od 4 konvolucijska sloja 3×3 sa C filtera i ReLU aktivacijskim funkcijama te posljednjeg konv. sloja 3×3 sa $K \cdot A$ filtera gdje je K broj kategorija, a A broj baznih okvira. Na izlaznu mapu značajki posljednjeg sloja primjenjuje se sigmoida kao aktivacijska funkcija. Izlazni volumen je dimenzija $W \times H \times (K \cdot A)$. Regresijska podmreža razlikuje se od klasifikacijske samo u posljednjem $3 \times 3 \times 4A$ konvolucijskom sloju. Izlazni volumen je dimenzija $W \times H \times 4A$.

Svakoj značajki $W \times H$ presjeka volumena pridružena je odgovarajuća regija ulazne slike i vrijednosti za jedan od A baznih okvira pridruženih toj regiji. Za klasifikacijsku mrežu vrijednosti treće dimenzije su vjerojatnosti postojanja objekta kategorije K za neki od A baznih okvira, a za regresijsku mrežu pridružene su vrijednosti relativni pomaci 4 parametra baznog okvira kojim se predviđa oblik objekta. Parametri mreže su dijeljeni po razinama, ali zasebni za svaku podmrežu.

Za gubitak regresije L_{loc} koristi se *smooth* L1 između podudaranih okvira kao i u prethodno opisanom SSD detektoru. Gubitak klasifikacije koji se naziva i fokalni gubitak (engl. *focal loss*) definiran je sljedećim izrazom:

$$L_{cls} = -\sum_{i=1}^{K} (y_i \log(p_i)\,(1-p_i)^{\gamma}\alpha_i + (1-y_i)\log(1-p_i)\,p_i^{\gamma}(1-\alpha_i)) \qquad (39)$$

gdje K predstavlja broj objektnih kategorija, y_i definira zastavicu pripadanja i-toj kategoriji ($y_i = 1$ kada je *ground-truth* okviru dodijeljena i-ta kategorija, a u suprotnom $y_i = 0$), p_i je predviđena vjerojatnost za i-tu kategoriju. Fokalni gubitak određen je sa dva hiperparametra $\gamma \in (0, +\infty)$, koji se naziva parametar fokusa te $\alpha \in [0,1]$, koji je parametar otežavanja. Povećanjem γ smanjuje se utjecaj primjera kojima je pridružena visoka vjerojatnost za točnu kategoriju kako bi se spriječilo da lagani pozadinski primjeri dominiraju funkcijom gubitka.

Vrijednost α se tipično postavlja na inverznu frekvenciju kategorija ili se određuje unakrsnom validacijom.

RetinaNet tipično ima 5 razina na kojima radi predikcije pa je ukupni broj predviđenih okvira $\sum_{l=3}^{7} W_l H_l A$. Naknadnom obradom odabire se samo prvih 1000 predviđenih okvira za svaku razinu, sortiranih po pouzdanosti s donjom granicom od 0.05. Posljednji korak je primjena NMS kako bi se uklonilo redundantne okvire.

2.6.4 Evaluacijske metrike

Za uspoređivanje kvalitete dobivenog modela na zadataku detekcije objekata pretežno korištena metrika je prosječna preciznost AP (engl. *average precision*). Kako bi mogli definirati i opisati AP potrebno je uvesti pojmove koji omogućuju razlikovanje točnih i netočnih detekcija.

Pojedinačne instance objekata određene su koordinatama pravokutnih okvira koji opisuju prostorni raspon objekta i labelom koja klasificira objekt u određenu kategoriju. Svakom označenom primjeru evaluacijskog skupa pridružen je određeni broj *ground truth* pravokutnih okvira $\boldsymbol{B_{gt}}$, odnosno okvira koji predstavljaju stvarnu prisutnost objekata od interesa. Model za detekciju objekata generira okvire koji predstavljaju predikciju modela za instance objekata. Koncepti koje koristimo kako bi opisali odnos između stvarnih i predviđenih okvira su sljedeći:

- *True Positive* (TP): Predviđeni okvir podudara se sa *ground truth* okvirom, odnosno detekcija je točna.
- *False Positive* (FP): Predviđen je okvir za nepostojeći objekt ili je pogrešno predviđen okvir za postojeći objekt (predviđeni okvir se ne podudara sa *ground truth* okvirom)
- *False Negative* (FN): Ne postoji predviđeni okvir koji se podudara sa *ground truth* okvirom, odnosno objekt nije detektiran iako je prisutan

True Negative (TN) nije značajan koncept za zadatak detekcije objekata jer teorijski postoji neograničen broj okvira koji ne bi trebali biti predviđeni.

Da bi gornje definicije bile iskoristive za evaluaciju potrebno je definirati što podrazumijeva podudaranje dva okvira. Pristup koji se koristi za zadatak detekcije objekata je podudaranje pomoću IoU (engl. *Intersection over Union*), vrijednosti koja se temelji na Jaccardovom

indeksu (mjeri sličnosti dva skupa). IoU između stvarnog okvira B_p i predviđenog okvira B_{gt} definiramo kao omjer površine presjeka dva okvira i površine njihove unije:

$$J(B_p, B_{gt}) = IoU = \frac{površina(B_p \cap B_{gt})}{površina(B_p \cup B_{gt})} \tag{40}$$

Dobivena vrijednost uspoređuje se s proizvoljno odabranom graničnom vrijednosti t (engl. *threshold*). Ako je $IoU \geq t$ kažemo da se okviri podudaraju (detekcija je točna), a ako je $IoU < t$ da se ne podudaraju, odnosno da je detekcija netočna.

Dvije metrike koje se koriste za evaluaciju detektora i koje se temelje na gore opisanim konceptima su preciznost P (engl. *precision*) i odziv R (engl. *recall*).

$$P = \frac{TP}{TP + FP} = \frac{TP}{sve\ detekcije} \tag{41}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{svi\ ground\ truth\ okviri} \tag{42}$$

Preciznost se računa kao postotak točnih predikcija s obzirom na ukupan broj predikcija, a interpretira se kao sposobnost modela da identificira samo objekte od interesa. Odziv je postotak točno predviđenih okvira s obzirom na sve *ground truth* okvire, a interpretira se kao sposobnost modela da predvidi okvir za sve relevantne objekte.

Krivulja preciznost-odziv (engl. *precision-recall curve*) sažima informaciju o balansu između preciznosti i odziva s obzirom na pouzdanost detekcija modela. Ako odaberemo granicu pouzdanosti (engl. *confidence threshold*) na temelju koje odbacujemo detekcije za koju je broj FP nizak, preciznost modela će biti visoka. Isto tako moguće je da granica koja daje visoku preciznost ne obuhvaća velik broj pozitivnih primjera što povećava broj FN i uzrokuje nizak odziv modela. Vrijedi i obrnuto, ako model prihvaća više pozitivnih primjera (niže pouzdanosti) odziv će se povećati ili ostati isti, ali i broj FP primjera će narasti zbog čega se pojavljuju fluktuacije preciznosti [35].

Idealan detektor bi pronalazio sve objekte od interesa i samo relevantne objekte (FN=0 i FP=0), odnosno imao visoku preciznost bez obzira na visoki odziv. Ova svojstva odgovaraju visokoj vrijednosti površine ispod krivulje AUC (engl. *area under curve*) za krivulju

preciznosti-odziva. S obzirom da je izgled PR krivulje često zig-zag oblika potrebno je estimirati AUC vrijednost numeričkom vrijednosti koja se naziva prosječna preciznost.

Dva korištena pristupa za ovu estimaciju su:

1. **Interpolacija kroz 11 točaka odziva**

Za 11 jednako razmaknutih točaka $r \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$ $P_{interp}(r)$ vrijednost se interpolira maksimalnom preciznosti za odziv veći ili jednak toj točki. Vrijednost prosječne preciznosti AP_{11} dobiva se računanjem prosjeka interpoliranih vrijednosti.

$$AP_{11} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} P_{interp}(r) \tag{43}$$

$$P_{interp}(r) = \max_{\tilde{r} \geq r} P(\tilde{r}) \tag{44}$$

2. **Interpolacija za sve točke odziva**

Vrijednosti interpolirane krivulje se računa za n točaka kao maksimalna preciznost za odziv veći ili jednak toj točki. Prosječnu preciznost računamo kao površinu ispod interpolirane krivulje:

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) P_{interp}(r_{i+1}) \tag{45}$$

$$P_{interp}(r_{i+1}) = \max_{\tilde{r} \geq r_{i+1}} P(\tilde{r}) \tag{46}$$

Konačno, s obzirom da se evaluacija prosječne preciznosti radi na razini pojedinih kategorija, računa se srednja prosječna preciznost mAP (engl. *mean average precision*) kao srednja vrijednost prosječnih preciznosti za pojedinačne kategorije. Za K kategorija u podatkovnom skupu mAP računamo sljedećim izrazom:

$$mAP = \frac{1}{K} \sum_{i \in K} AP_i \tag{47}$$

Pascal VOC izazov za evaluaciju koristi prosječnu preciznost sa IoU granicom $t = 0.5$, dok se u MS COCO izazovu usrednjava vrijednost AP tako da granica t poprima vrijednosti iz

raspona $[0.5: 0.95: 0.05]$ što povećava robusnost metrike na lošu lokalizaciju. MS COCO koristi i evaluaciju za 3 skale veličine objekata: male (površina $< 32^2$ piksela), srednje (32^2 piksela $<$ površina $< 96^2$ piksela) i velike (96^2 piksela $<$ površina).

3 DETEKCIJA I KLASIFIKACIJA MORSKOG OTPADA

3.1 Formiranje podatkovnog skupa za treniranje i evaluaciju

U trenutku početka rada na ovom diplomskom zadatku nije postojao javno dostupan skup označenih podataka namijenjen treniranju modela za detekciju i klasfikaciju morskog otpada iz podvodnih fotografija. Iz tog razloga formiran je novi skup podataka na tragu rada [36] korištenjem slika iz baze podataka *Deep sea debris database* [2] , koja je samo dio J-EDI baze podataka japanske agencije za pomorsku znanost i tehnologiju (engl. *Japan Agency for Marine-Earth Science and Technology*). Autori istog rada su 23.07.2020 objavili prvi javno dostupan veliki skup podataka namijenjen detekciji i semantičkoj segmentaciji morskog otpada na temelju slika iz J-EDI baze podataka. Popratni rad [3] u kojem je opisan podatkovni skup trenutno je u fazi pripreme za objavljivanje.

S obzirom da je u vrijeme objave *TrashCan* skupa podataka već napravljen vlastiti skup za treniranje od ~2000 označenih fotografija formiran je novi podatkovni skup spajanjem dva prethodno navedena. U potpoglavlju 3.1.1. opisan je proces označavanja vlastitog podatkovnog skupa za detekciju, u potpoglavlju 3.1.2. kratko su sažete karakteristike *TrashCan* skupa i u završnom potpoglavlju 3.1.3. opisano je spajanje i predobrada dva prethodno opisana skupa kako bi se dobio konačni skup za treniranje.

Autori „Vodiča za identifikaciju morskog otpada“ iz AWARE projekta *Dive Against Debris* [24] razradili su klasifikacijski sustav koji morski otpad kategorizira u 8 osnovnih kategorija koje su detaljnije podijeljene u čak 100 različitih kategorija otpada. 8 osnovnih kategorija koje vodič definira su: plastika, staklo i kermaički materijali, metalni materijali, drveni materijali, tkanine, papirnati/kartonski materijali, gumeni materijali i miješani otpad. Za potrebe formiranja podatkovnog skupa u ovom radu također je korištena kategorizacija otpada po tipu materijala. Uzimajući u obzir količine podataka dostupnih za pojedine kategorije neke su ignorirane zbog nedovoljne zastupljenosti.

3.1.1 Stvaranje vlastitog podatkovnog skupa

Slike i video uradci za stvaranje vlastitog skupa podataka su preuzeti korištenjem *Selenium* skripti s obzirom da službena stranica J-EDI baze podataka ne nudi opciju grupnog preuzimanja materijala.

Tablica 3. Detalji o svim podacima preuzetim iz JAMSTEC *Deep-sea Debris* baze podataka

Podaci	**Format**	**Rezolucija**	**Količina**
Video uradci niske kvalitete	mp4	480×360, 480×270	5283
Slike dobivene uzorkovanjem videa (rezolucija 1 sec)	jpg	480×360, 480×270	293 845
Fotografije visoke kvalitete	png	1192×894, 1284×722, 1284×856	3123

Korišteni algoritmi za detekciju i klasifikaciju pripadaju domeni nadgledanog učenja (engl. *supervised learning*) podatke za treniranje i evaluaciju modela je potrebno označiti na odgovarajući način. Za proces označavanja korišten je softver *labelme,* koji omogućuje dodjeljivanje pravokutnih *ground truth* okvira s pridruženim kategorijama instancama objekata pomoću grafičkog sučelja. Primjeri slika označenih okvirima vidljivi su na slikama 16 i 17. Anotacije kreirane ovom metodom pohranjuju se u *.json* formatu. Tijekom procesa označavanja objektima su dodjeljivane kategorije prema materijalu na sličan način kao u [37]:

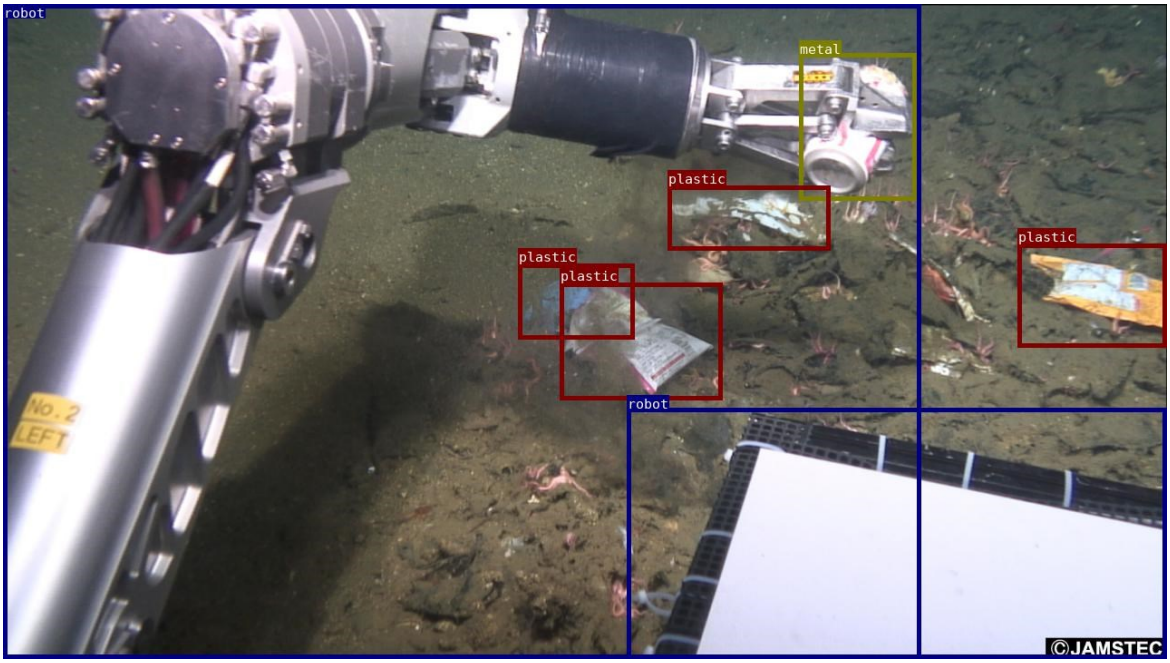
- *plastic* – plastični predmeti (PET boce, plastične vrećice, drugi plastični predmeti...)
- *cloth* – materijali od tkanine (mreže, odjevni predmeti, pletene vreće...)
- *metal* – metalni predmeti
- *other debris* – otpad koji nije moguće kategorizirati bez dodatnih informacija
- *rubber tire* – gume (sa vozila, kolica itd.)

Osim kategorija koje se odnose na tip morskog otpada, u procesu označavanja dodane su i sljedeće 2 kategorije koje daju korisne informacije za razumijevanje sadržaja slika:

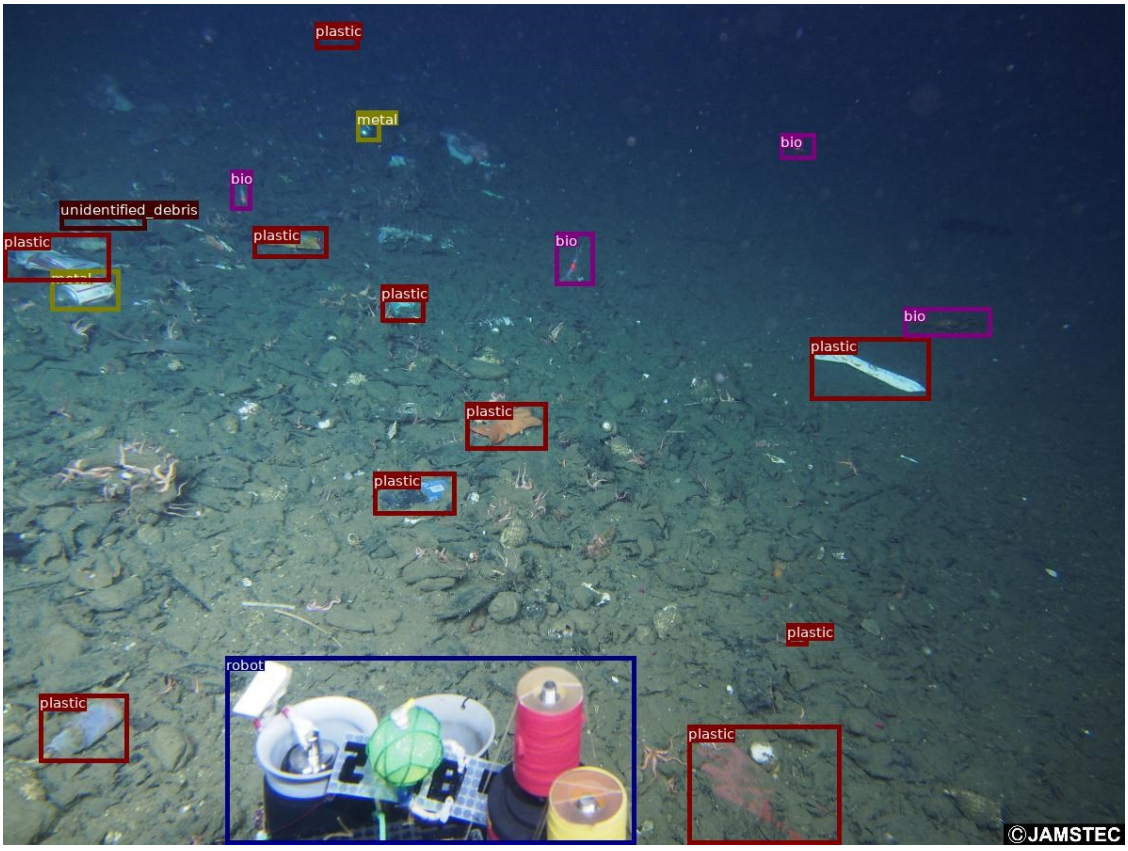
- *bio* – biološki organizmi
- *robot* – dijelovi podvodnog robota (vozila) koji se koristio za prikupljanje podataka

Dodavanje informacija iz ove dvije kategorije čini algoritam robusnijim za primjene kao što je sakupljanje otpada.

S obzirom da *labelme* koristi vlastiti *.json* format za pohranu oznaka napravljena je konverzija u *Pascal* VOC format koji pohranjuje informacije o označenim slikama u XML datoteci. Razlog za konverziju je što većina knjižnica namijenjenih dubokom učenju radi sa oznakama *Pascal* VOC ili COCO formata.



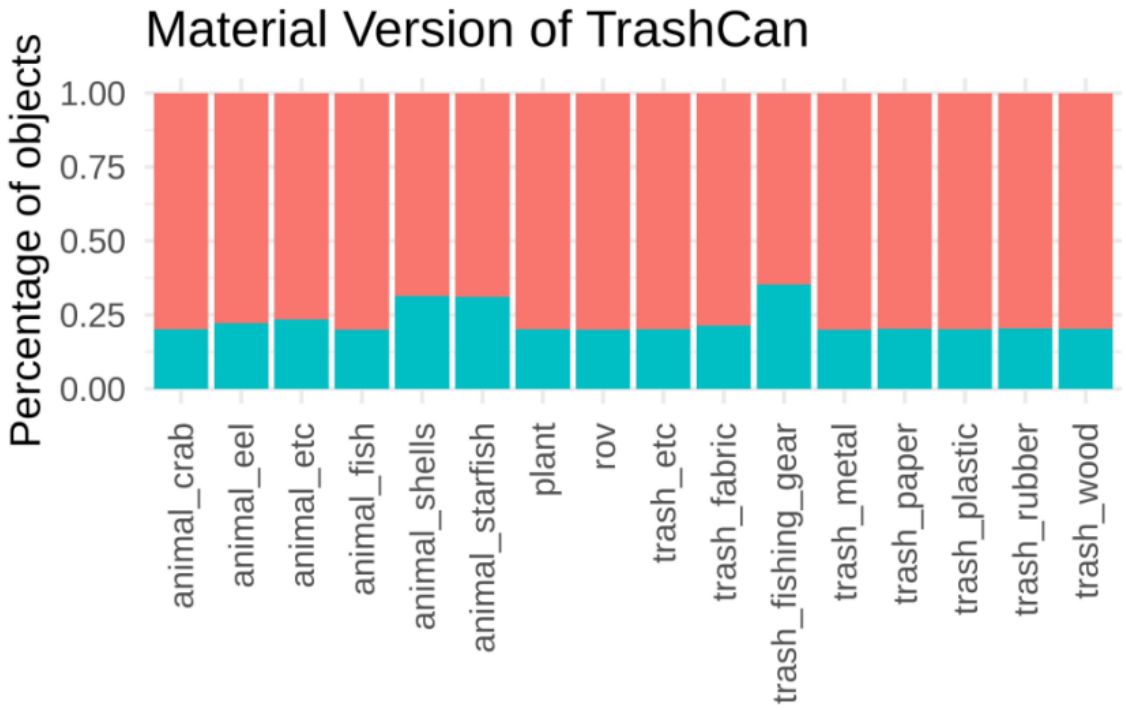
Slika 16. Primjer 1. slike označene *ground truth* okvirima [2]



Slika 17. Primjer 2. slike označene *ground truth* okvirima [2]

3.1.2 *TrashCan* podatkovni skup

TrashCan podatkovni skup sastoji se od 7212 slika označenih za zadatke detekcije i segmentacije instanci morskog otpada. Autori navode da je u procesu označavanja sudjelovala 21 osoba i da je uloženo preko 1500 radnih sati. Oznake podatkovnog skupa pohranjene su u COCO *.json* formatu.



Slika 18. Raspodjela označenih primjera za *TrashCan material* varijantu skupa [3]

Označeni objekti od interesa dijele se u 4 superkategorije: životinje (*animal*), biljke (*plant*), dijelovi robota (*rov*) i otpad (*trash*). Skup dolazi u dvije varijante gdje je u *instance* varijati otpad detaljnije klasficiran u kategorije po tipu konkretne instance otpada (limenka, konop, cijev, torba i sl.), a u *material* varijanti je klasficiran po materijalu od kojeg se sastoji otpad (plastika, metal, papir, guma i sl.). Raspodjela svih označenih objekata po kategorijama na trening i validacijski skup prikazana je grafu na slici 18 za *material* varijantu skupa.

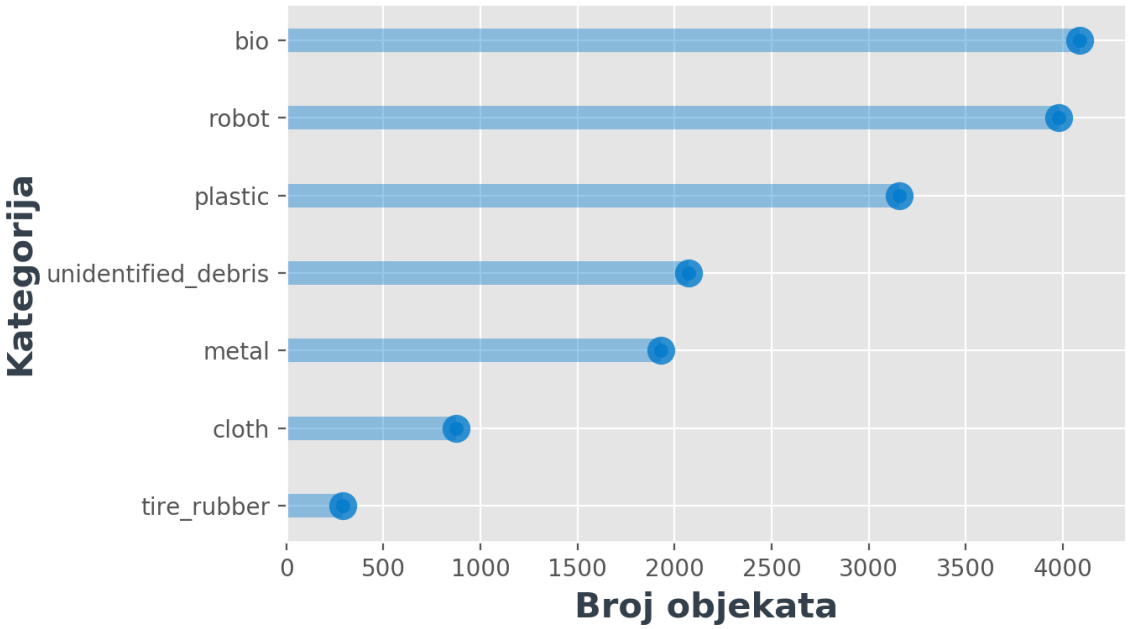
Autori su za detekciju trenirali *Faster* RCNN model sa *ResNeXt-101-FPN backbone* mrežom predtreniran na COCO skupu podataka. Rezultati su dobiveni korištenjem COCO evaluacijskih metrika. Objavljeni rezultati (Tablica 4) postignuti su za *fine-tuning* modela na 10 000 iteracija i možemo ih koristiti kao osnovicu u ovom području.

Tablica 4. Rezultati autora *TrashCan* skupa podataka na zadatku detekcije i klasifikacije morskog otpada pomoću *Faster RCNN ResNeXt-101-FPN* modela [3]

Varijanta skupa	AP	AP_{50}	AP_{75}	AP_s	AP_M	AP_L
Instance	34.5	55.4	38.1	27.6	36.2	51.4
Material	29.1	51.2	27.8	28.2	30.2	40.0

3.1.3 Konačni podatkovni skup

Konačni podatkovni skup formiran je spajanjem vlastitog i *TrashCan* podatkovnog skupa. S obzirom da podaci oba skupa potiču iz J-EDI baze podataka bilo je potrebno ukloniti sve duplikate. Kao metrika udaljenosti za uspoređivanje slika korištena je srednja kvadratna pogreška MSE(engl. *mean squared error*), a granica udaljenosti je postavljena na 0%. Za identifikaciju i uklanjanje duplikata je korišten je softver *AntiDupl.NET*-2.3.9 i uklonjeno je ukupno 96 primjeraka. Konačni podatkovni skup sastoji se od 8688 označenih slika sa distribucijom instanci objekata po kategorijama prikazanom na slici 19.

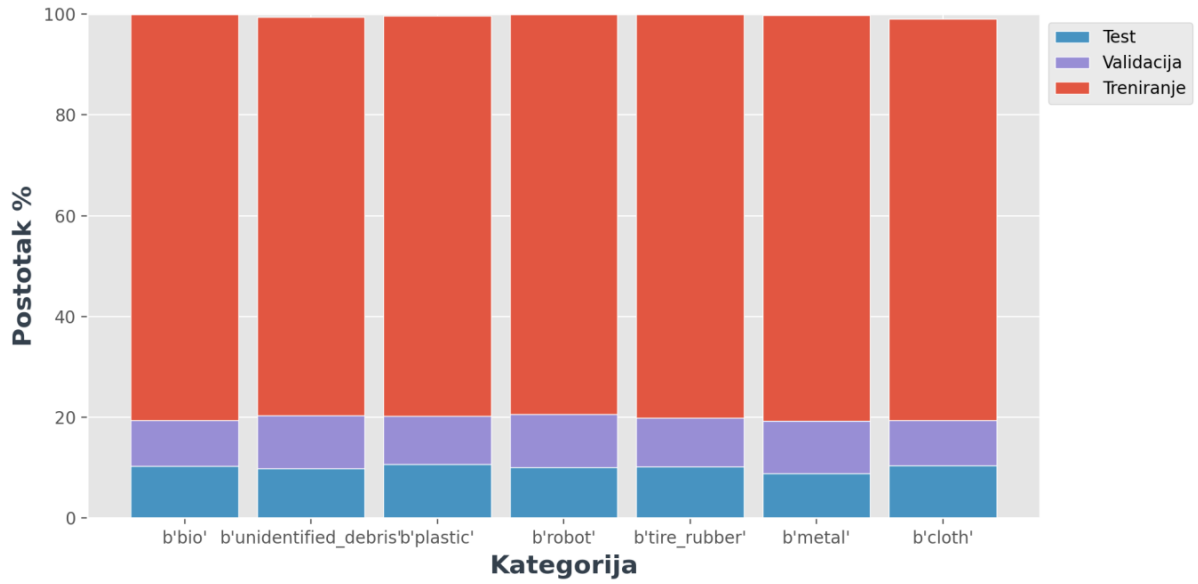


Slika 19. Broj instanci objekata za pojedinačne kategorije konačnog podatkovnog skupa

Konačni podatkovni skup podijeljen je na 3 podskupa (Slika 20):

1. **Skup za treniranje (engl. *training set*) –** podaci na temelju kojih se optimizacijom funkcije gubitka uče parametri modela (6950 slika ~80% podataka)

2. **Skup za validaciju (engl. *validation set*)** – koristi se za evaluaciju modela tijekom procesa učenja, donošenje odluka o arhitekturi modela i podešavanje hiperparametara (869 slika ~10% podataka)
3. **Testni skup (engl. *test set*)** – koristi se za dobivanje konačne nepristrane procjene kvalitete naučenog modela (869 slika ~10% podataka)



Slika 20. Raspodjela instanci izražena u postotcima za sva 3 generirana podskupa

3.2 Programska implementacija modela za detekciju

U ovom poglavlju opisani su detalji implementacije modela za detekciju i klasifikaciju morskog otpada, treniranih na podatkovnom skupu opisanom u poglavlju 3.1.3. Modeli su implementirani koristeći *Object Detection API* bibilioteke *TensorFlow* programskog jezika *Python*, a pristup treniranju je fokusiran na *fine-tuning* modela predtreniranih na velikim skupovima podataka zbog male veličine podatkovnog skupa s obzirom na složenost zadatka.

Za incijalizaciju modela korišteni su parametri modela predtrenirani na COCO podatkovnom skupu. U tablici 5 sažeti su detalji o odabranim modelima poput brzine inferencije, ostvarene preciznosti na COCO skupu i broja učenih parametara koji bi trebali služiti kao indikator složenosti i kapaciteta modela. Mjerenja brzine izvedena su na Nvidia GeForce GTX TITAN X i ulazima veličine 600×600.

Arhitekture detektora odgovaraju onim teorijski obrađenim u poglavlju 2.6 uz varijacije na korištene *backbone* mreže u odnosu na modele opisane u originalnim radovima.

Tablica 5. Arhitektura treniranih modela s pripadajućim brojem učenih parametara

Arhitektura detektora	***Backbone* mreža**	**Broj učenih parametara**	**Brzina (ms)**	**COCO mAP**
SSD	Inception V2	13 983 844	42	24
	MobileNet V2	4 643 788	31	22
RetinaNet	ResNet-50 + FPN	31 692 680	76	35
	MobileNet V1 + FPN	10 932 872	56	32
Faster RCNN	ResNet-50	43 302 572	58	28
	ResNet-101	62 294 700	106	32
	ResNet-101 (zadnji blok)	34 759 276	/	/

3.2.1 Tensorflow biblioteka

Tensorflow je računska biblioteka namijenjena izražavanju složenih matematičkih operacija nad višedimenzionalnim podacima (tenzorima) u obliku grafova toka podataka (engl. *dataflow graph*).Čvorovi grafa predstavljaju matematičke operacije, a bridovi povezuju čvorove i predstavljaju ulazne podatke / izlaze čvorova. U području dubokog učenja koriste se za implementaciju neuronskih mreža gdje se tok podataka između slojeva umjetnih neurona i odgovarajuće operacije od kojih se sastoje duboke arhitekture mogu izraziti kao grafovi toka podataka. Grafovi osim ulaznih i izlaznih podataka sadržavaju i druge varijable, poput parametara modela i međurezultata unaprijednog prolaza koji se čuvaju da bi se pri unatražnom prolasku (engl. *backward pass*) koristili za računanje gradijenta algoritmom propagacije unatrag (engl. *backpropagation*) .

Tensorflow je prilagođen programskom jeziku Python što omogućuje izgradnju modela dubokog učenja implementacijom sučelja visoke razine apstrakcije te daje modularnost i višestruku iskoristivost TF komponenti. Osim osnovnih funkcionalnosti TF nudi i druge važne alate za područje dubokog učenja kao što su vizualizacija modela i nadgledanje procesa učenja kroz *TensorBoard* grafičko sučelje, pohrana grafova modela spremnih za inferenciju, *checkpoint* funkcije i dr.

Object Detection API je dio *Tensorflow* ekosustava namijenjen jednostavnoj implementaciji modela za detekciju i klasifikaciju. Osim predtreniranih modela *Object Detection* API omogućuje izgradnju vlastitih modela, prilagodbu hiperparametara učenja, generiranje vizualizacija i odabir evaluacijskih metrika definiranjem *protobuf* konfiguracijske datoteke (Slika 21). *Protobuf* je *Googleov* format za serijalizaciju strukturiranih podataka, neovisan o platformi i jeziku te sličan XML-u.

```
model {
(... Definiranje arhitekture modela ...)
}

train_config : {
(... Postavke hiperparametara učenja ...)
}

train_input_reader: {
(... Učitavanje podataka za učenje ...)
}

eval_config: {
(... Specifikacija evaluacijskih metrika ...)
}

eval_input_reader: {
(... Učitavanje podataka za evaluaciju...)
}
```

Slika 21. Kostur konfiguracijske datoteke za definiranje TF *Object Detection* API modela

3.3 Treniranje modela

Tijekom procesa učenja na temelju povremene evaluacije na validacijskom skupu sačuvane su kontrolne točke (engl. *checkpoint*) sa najboljim parametrima modela. Rezultati izloženi u sljedećem poglavlju dobiveni su na temelju evaluacije modela na testnom skupu s parametrima kontrolne točke s najmanjim ukupnim gubitkom na validacijskom skupu.

Modeli su trenirani i evaluirani na radnoj stanici HP Z6 G4 specifikacija:

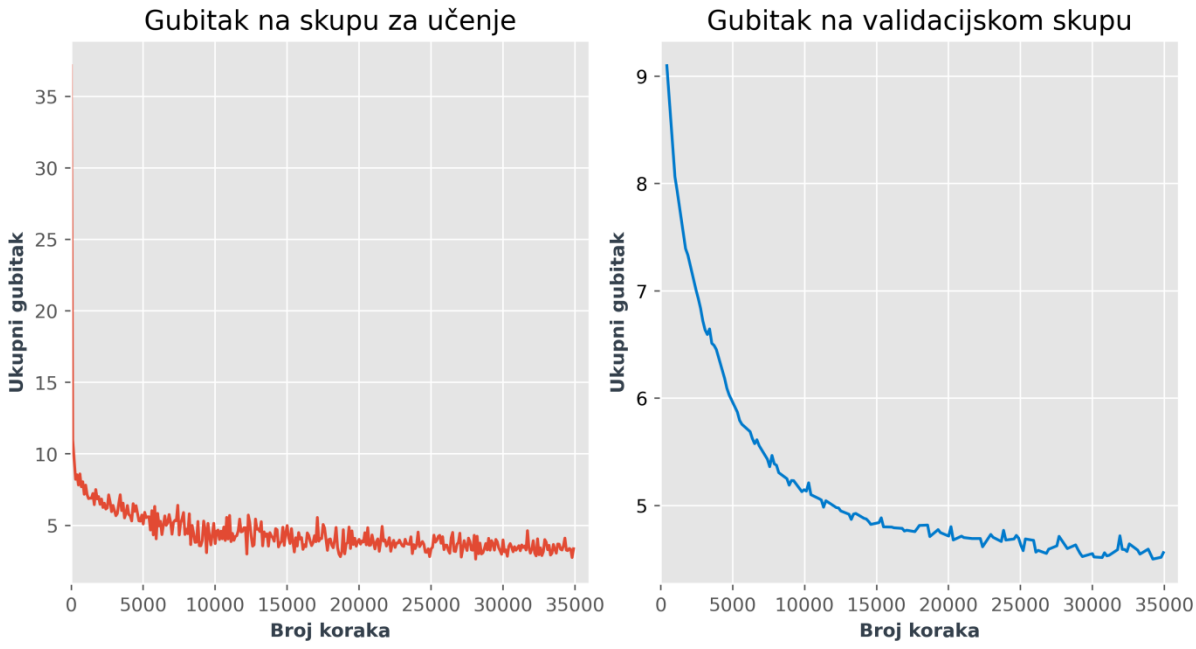
- Intel Xeon Silver 4110, 2.10 GHz
- 32.0 GB RAM
- 2 × NVIDIA Quadro P4000 8 GB DDR5

S obzirom da korišteni modeli imaju velik broj hiperparametara koji se mogu podešavati, proces treniranja je sveden na ručno podešavanje parametara odabranih od strane autora i nadgledanje procesa učenja kroz *TensorBoard* grafičko sučelje. Nasumična pretraga prostora hiperparametara nije bila izvediva na vremenski efikasan način sa dostupnom opremom.
Za svaki od treniranih modela proces učenja je u sljedećim poglavljima opisan grafovima ukupnog gubitka na skupu za učenje i validacijskom skupu te popisom odabranih hiperparametara s kojima je model treniran. Napominjem da su zbog sažetosti popisani samo oni hiperparametri čija su izmjene imale utjecaj na uspješnost treniranja modela. Potpuna

konfiguracija svakog od modela potrebna za reprodukciju rezultata može se pronaći u konfiguracijskim datotekama unutar projektnog direktorija isporučenog s ovim radom.

SSD + Inception V2

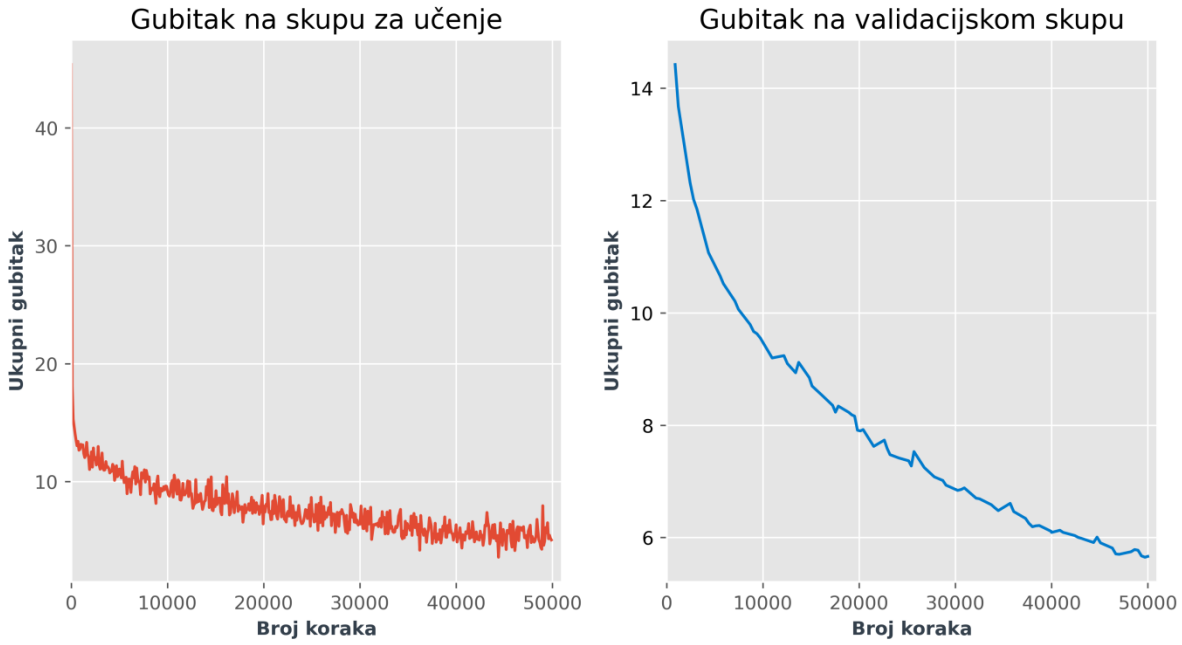
Postavke hiperparametara učenja	
Veličina ulaznih slika	300 × 300
Optimizator	RMS Prop
Bazna stopa učenja	0.0009
Opadanje stope učenja	/
Veličina mini-grupe	32
Broj koraka	35 000
L2 regularizacija	0.0004
Augmentacija	– *random_horizontal_flip* – *ssd_random_crop*



Slika 22. Gubitak za SSD + Inception V2 model

SSD + MobileNet V2

Postavke hiperparametara učenja	
Veličina ulaznih slika	300 × 300
Optimizator	RMS Prop
Bazna stopa učenja	0.0003
Opadanje stope učenja	/
Veličina mini-grupe	16
Broj koraka	50 000
L2 regularizacija	0.005
Augmentacija	– *random_horizontal_flip* – *ssd_random_crop*

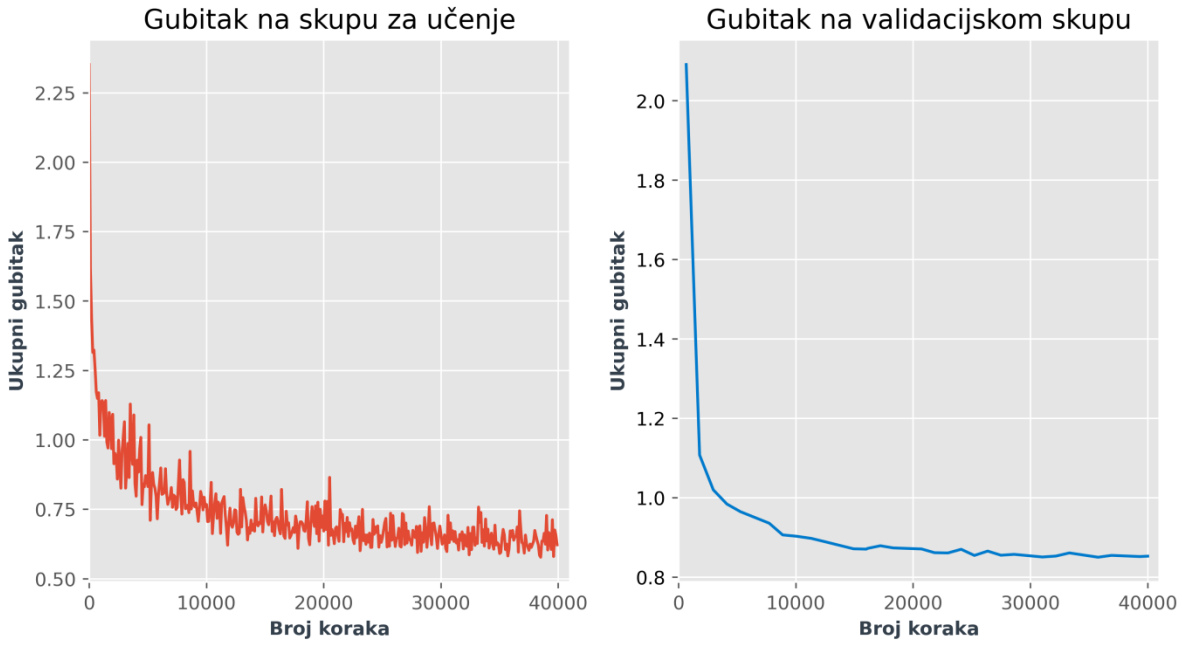


Slika 23. Gubitak za SSD + MobileNet V2 model

Učenje ovog modela prekinuto je na 50000 koraka dok je validacijski gubitak još padao pa postoji mogućnost da se dužim treniranjem može postići viša preciznost.

RetinaNet + MobileNet V1

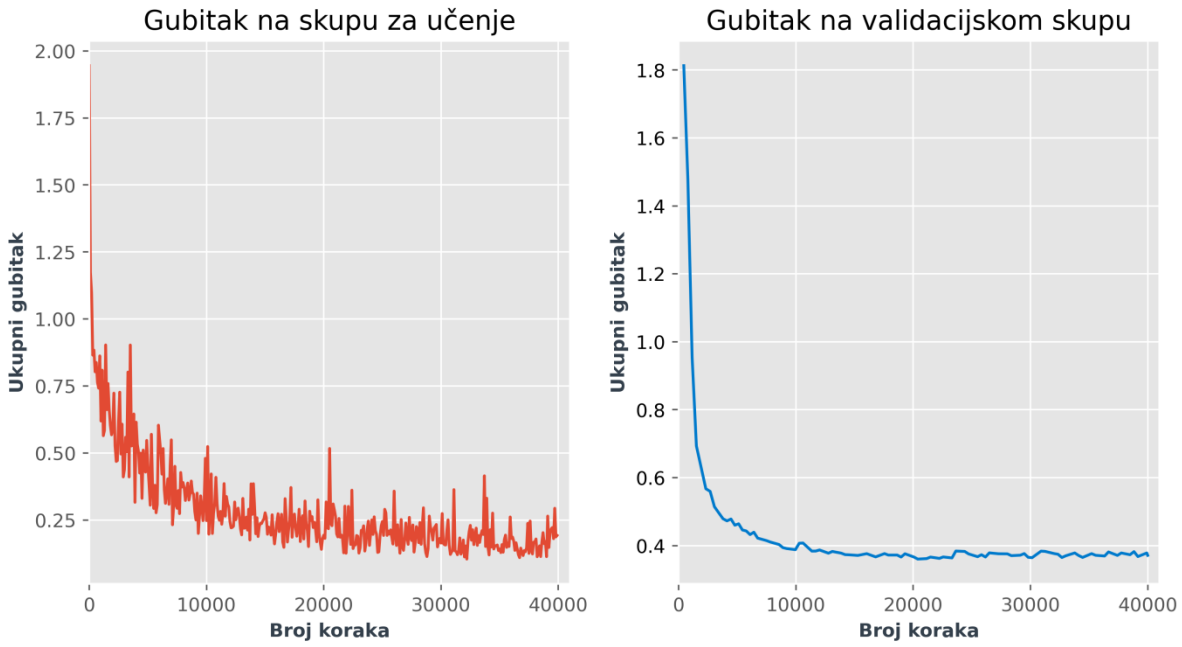
Postavke hiperparametara učenja	
Veličina ulaznih slika	320 × 320
Optimizator	SGD + moment
Bazna stopa učenja	0.001
***Warmup* stopa učenja**	0.0005 (1000 koraka)
Opadanje stope učenja	Kosinus (40 000 koraka)
Veličina mini-grupe	16
Broj koraka	40 000
L2 regularizacija	0.00004
Augmentacija	– *random_horizontal_flip* – *random_crop_image*



Slika 24. Gubitak za RetinaNet + MobileNet V1 model

RetinaNet + ResNet-50

Postavke hiperparametara učenja	
Veličina ulaznih slika	640 × 640
Optimizator	Adam
Bazna stopa učenja	0.00001
Opadanje stope učenja	/
Veličina mini-grupe	8
Broj koraka	40 000
L2 regularizacija	0.000004
Augmentacija	– *random_horizontal_flip* – *random_crop_image*

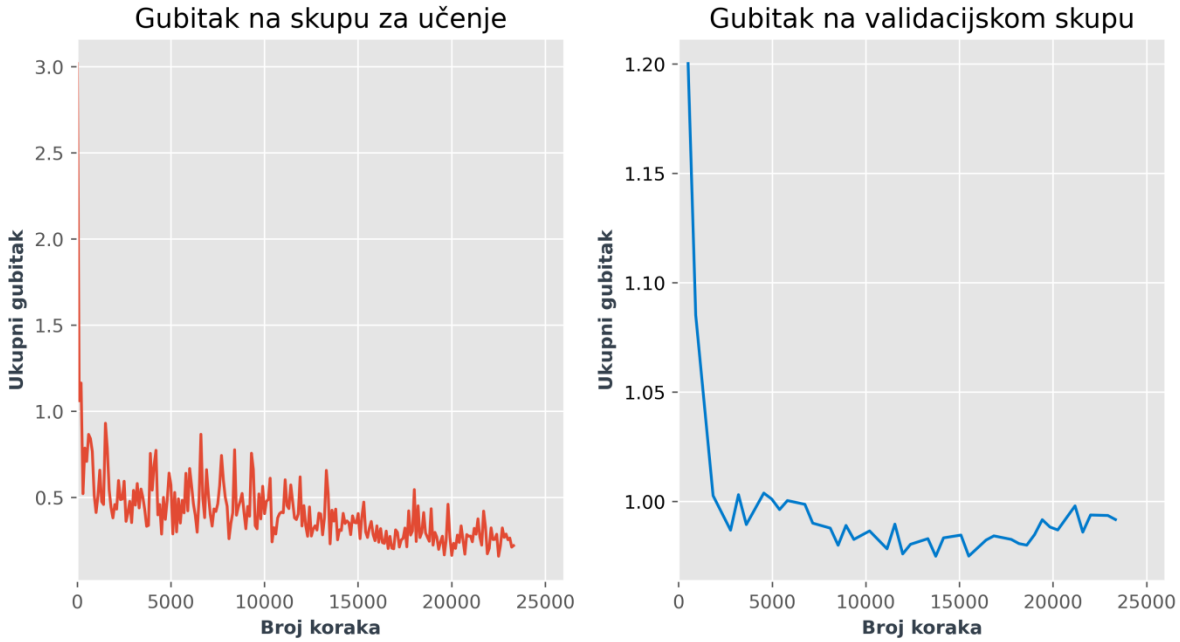


Slika 25. Gubitak za RetinaNet + ResNet-50

Treniranje oba RetinaNet modela pokazalo se izrazito stabilnim i robusnim. Uz praćenje ukupnog validacijskog gubitka praćena je i preciznost na COCO metrikama. Sve metrike su pokazivale poželjan brz rast u ranijim stadijma i zasićenje i stabilizaciju nakon 25000 koraka bez većih fluktuacija.

Faster RCNN + ResNet 50

Postavke hiperparametara učenja	
Veličina ulaznih slika	480 × 480
Optimizator	SGD + moment
Bazna stopa učenja	0.0005
***Warmup* stopa učenja**	0.00001 (1000 koraka)
Opadanje stope učenja	Kosinus (40 000 koraka)
Veličina mini-grupe	8
Broj koraka	23 000
L2 regularizacija I. faze	0.0003
L2 regularizacija II. faze	0.0007
Augmentacija	*-random_horizontal_flip*

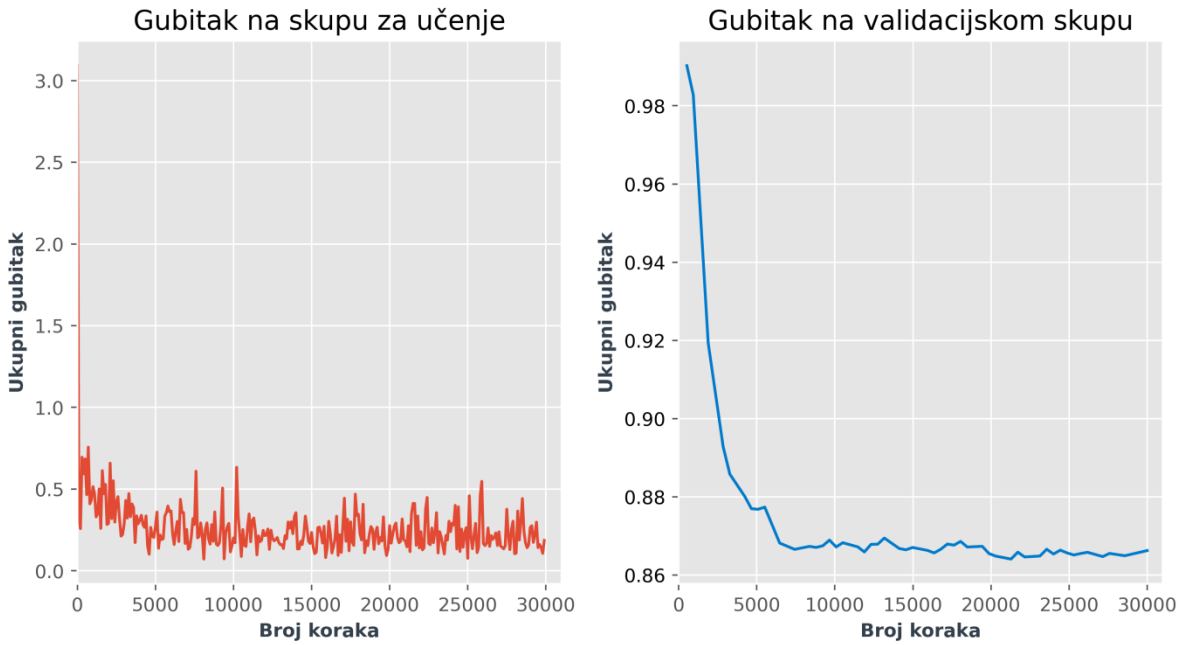


Slika 26. Gubitak za Faster RCNN + ResNet 50 model

Tijekom treniranja ovog modela uočene su česte fluktuacije na pojedinim članovima gubitka više zadataka (gubitak klasifikacije i regresije za RPN i Fast RCNN detektor). Posljedica je nestabilnost u ukupnom gubitku validacije vidljiva na grafu.

Faster RCNN + ResNet 101

Postavke hiperparametara učenja	
Veličina ulaznih slika	300 × 300
Optimizator	Adam
Bazna stopa učenja	0.00001
Opadanje stope učenja	Ručno (3000: 1e-6, 10000: 1e-7)
Veličina mini-grupe	8
Broj koraka	30 000
Augmentacija	*-random_horizontal_flip*



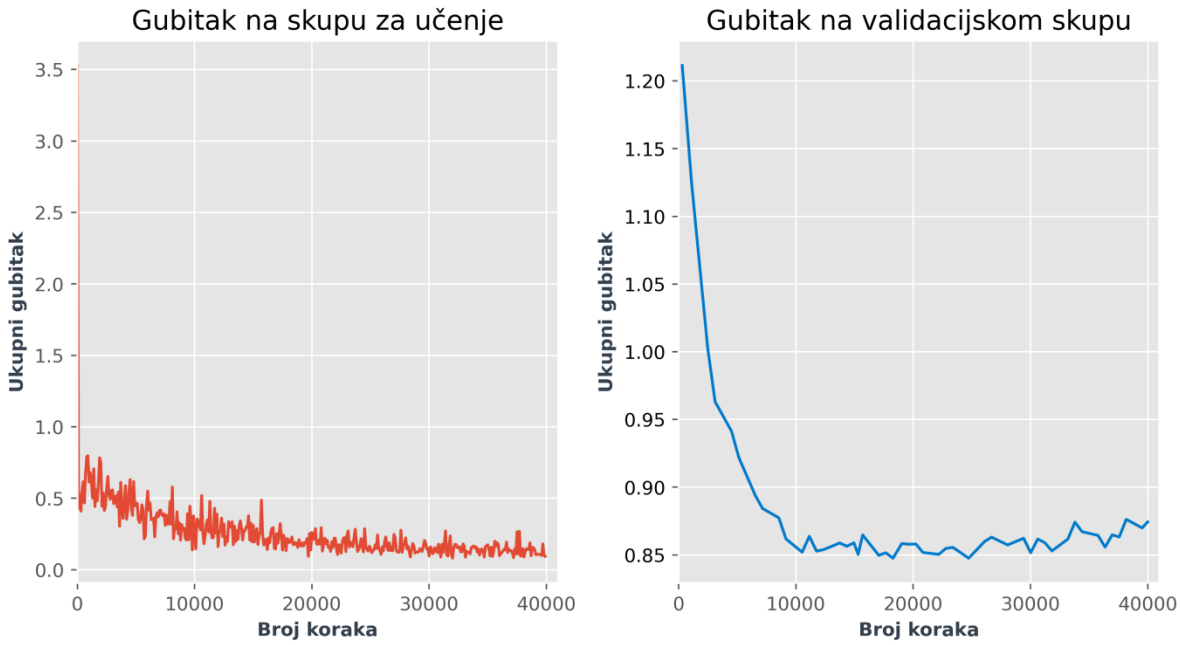
Slika 27. Gubitak za Faster RCNN + ResNet 101

Ovaj model ne pokazuje značajniji napredak nakon 5000 koraka na trening skupu ni na validacijskom skupu. Vrijednost trening gubitka je značajno manja od gubitka validacije što nas navodi na zaključak da je model „zapamtio“ podatke za učenje i da ima prevelik kapacitet s obzirom na količinu podataka.

Faster RCNN + ResNet 101

(treniran samo zadnji blok)

Postavke hiperparametara učenja	
Veličina ulaznih slika	300 × 300
Optimizator	Adam
Bazna stopa učenja	0.00001
Opadanje stope učenja	/
Veličina mini-grupe	8
Broj koraka	40 000
L2 regularizacija	0.001
Augmentacija	*-random_horizontal_flip*
„Zamrznuta“ baza	*conv1, conv2_x, conv3_x, conv4_x*



Slika 28. Gubitak za Faster RCNN + ResNet 101 kojem je treniran samo zadnji blok

Budući da je gornji Faster RCNN model pokazivao znakove preučenosti primjenjeno je prijenosno učenje kako bi se smanjio broj učivih parametara. Zamrznute su sve značajke *backbone* ResNet-101 mreže dobivene predtreniranjem na COCO skupu, osim zadnjeg rezidualnog bloka. Ponašanje modela nije se značajno promjenilo ovom modifikacijom.

3.4 Rezultati

U ovom poglavlju sažeti su rezultati evaluacije na testnom skupu. Vizualizacija izlaza najboljeg modela prikazana je na slici 29. U tablici 6 ispisani su rezultati svih modela prema COCO evaluacijskim metrikama. Slike 30, 31 i 32 prikazuju rezultate po kategorijama za sve modele s obzirom na AP_{50} metriku. Grafovi su grupirani s obzirom na arhitekture detektora.

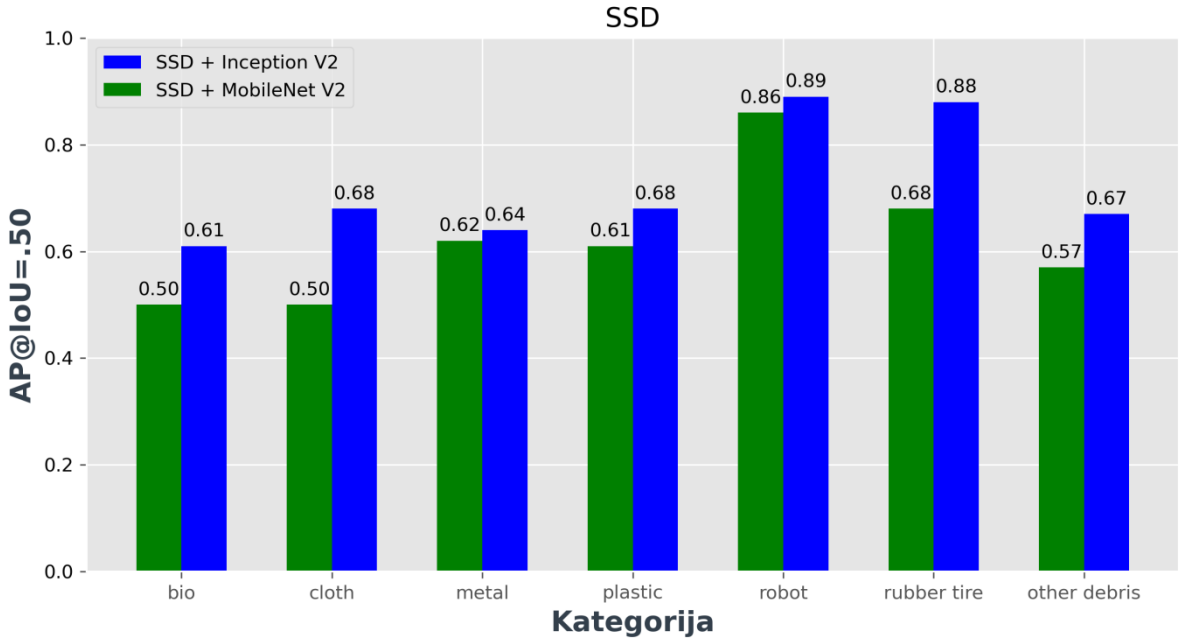
Slika 29. 4 primjera testnog skupa evaluirana na RetinaNet + ResNet-50 modelu

COCO metrike

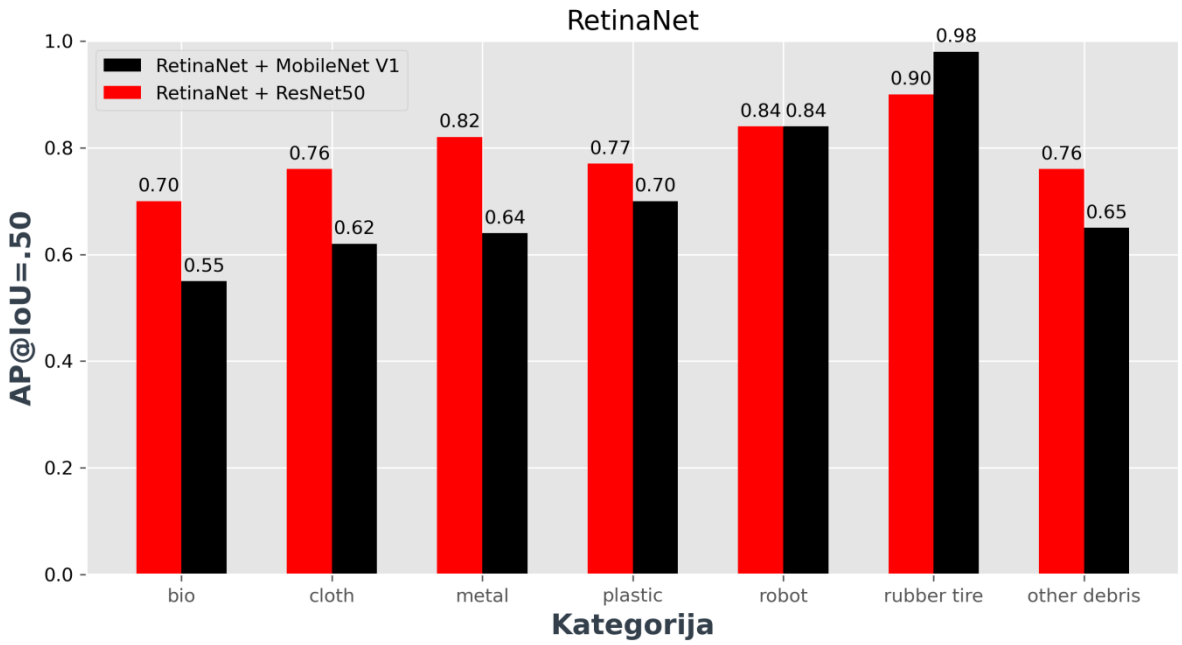
Tablica 6. Rezultat evaluacije modela na testnom skupu

Model	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L
SSD + Inception V2	0.47	0.72	0.52	0.39	0.42	**0.59**
SSD + MobileNet V2	0.39	0.62	0.42	0.25	0.33	0.54
RetinaNet + MobileNet V1	0.48	0.71	0.56	0.29	0.49	0.57
RetinaNet + ResNet-50	**0.54**	**0.79**	**0.64**	**0.49**	**0.54**	0.57
Faster RCNN + ResNet-101	0.37	0.57	0.44	0.14	0.36	0.46
Faster RCNN + ResNet-101 (samo zadnji blok)	0.36	0.59	0.40	0.21	0.34	0.45
Faster RCNN + Resnet-50	0.42	0.66	0.47	0.31	0.41	0.50

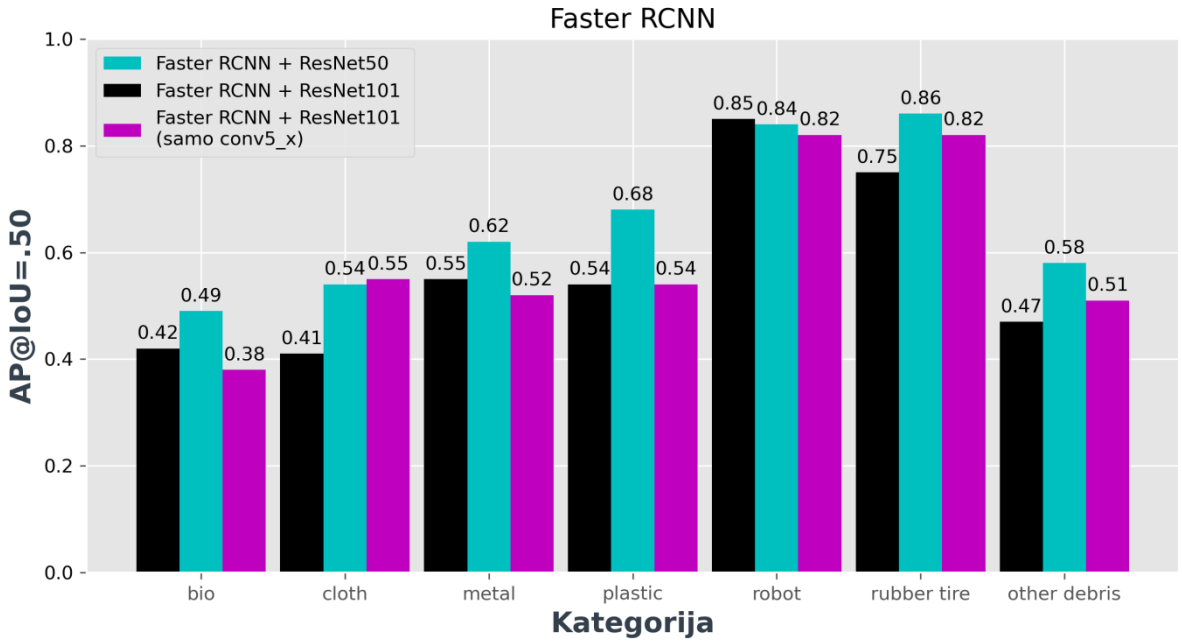
Preciznost po kategorijama



Slika 30.: Preciznost po kategorijama za modele sa SSD arhitekturom



Slika 31.: Preciznost po kategorijama za modele sa RetinaNet arhitekturom



Slika 32.: Preciznost po kategorijama za Faster RCNN modele

4 ZAKLJUČAK

Cilj ovog rada bio je ispitati primjenjivost dubokih konvolucijskih modela na zadatku detekcije i klasifikacije morskog otpada iz podvodnih fotografija. Ovaj proces rezultirao je stvaranjem vlastitog podatkovnog skupa namijenjenog zadatku koji je uz prilagdobe već postojećih materijala iskorišten da bi se, prema znanju autora, dobio najveći do sad korišteni podatkovni skup za ovu primjenu.

Treniranjem modela 3 popularne arhitekture za detekciju, uz varijacije *backbone* mreža korištenih za izvlačenje svojstava, ispitane su potrebe za kapacitetom modela s obzirom na količinu podataka kao i mogućnost iskoristivosti ovih modela za inferenciju u realnom vremenu. Iako je pri evaluaciji RetinaNet model s kompleksnijom *backbone* mrežom ResNet-50 i najvećom rezolucijom ulaza postigao najbolje rezultate, vidimo da model s istom arhitekturom detektora, duplo manjim ulazom i puno manjom MobileNet V1 mrežom postiže približnu preciznost pri evaluaciji. Također vidljivo je da s obzirom na veliku razliku mAP_S metrike u korist modela s većim ulazom, samo FPN nije dovoljan za značajno poboljšanje predikcije malih objekata. Faster RCNN modeli trenirani sa većom *backbone* mrežom ResNet-101 pokazali su tendenciju preučavanju (značajnije razlike između gubitka na skupu za učenje i gubitka validacije) što sugerira da je kapacitet ove mreže prevelik s obzirom na dostupnu količinu podatka. Pokušaj regularizacije „zamrzavanjem“ parametara donja 3 rezidualna bloka nije značajno poboljšao model. Kapacitet manjeg Faster RCNN + ResNet-50 modela pokazao je bolje karakteristike od ResNet-101 modela, ali nedovoljno da bi se opravdao *overhead* po pitanju brzine s obzirom da su ga nadmašili brži i manje kompleksni jednofazni modeli poput SSD + Inception V2 i RetinaNet + MobileNet V1 modela.

Neki od ograničavajućih faktora pri treniranju ovih modela bili su količina dostupne memorije GPU i vrijeme potrebno za učenje modela s jednim skupom hiperparametara. Zbog nedostatka memorije korištene su relativno male mini-grupe (8, 16, 32) što je djelovalo kao regularizirajući faktor, ali je u nekim slučajevima značajno usporavalo konvergenciju. S obzirom da modeli za detekciju imaju ogroman broj hiperparametara koji se mogu podešavati, traženje optimalnih hiperparametara nasumičnom pretragom nije bilo vremenski izvedivo. Zbog toga proces treniranja modela se sastojao isključivo od ručnog podešavanje manjeg broja hiperparametara koji tipično imaju najveći utjecaj (stopa učenja, postavke optimizatora, L2 regularizacija, dodavanje augmentacije).

Jednofazni modeli s manjim *backbone* mrežama poput MobileNeta i nižom rezolucijom ulaza mogu izvoditi inferenciju s visokim brojem slika u sekundi, na relativno jeftinom sklopovlju kao što je *Jetson Nano* i *Raspberry Pi*. S obzirom da su rezultati dani evaluacijom upravo ovakvih modela postigli obećavajuće preciznosti, u bližoj budućnosti otvara se mogućnost integracije za primjene poput skupljanja otpada autonomnim podvodnim vozilima.

Najveće ograničenje za daljnje pomake u ovom području je količina dostupnih podataka u odnosu na varijabilnost uvjeta kako bi se mogla postići dovoljna razina generalizacije u realnom svijetu. S obzirom da je prikupljanje takvih podataka pod vodom zahtjevan zadatak koji zahtjeva posebnu opremu interesantan budući smjer istraživanja je iskorištavanje dubokih generativnih metoda za augmentaciju podataka i poboljšanje njihove kvalitete.

LITERATURA

[1] S. Lippiatt, S. Opfer i C. Arthur, »Marine Debris Monitoring and Assessment,« NOAA Technical Memorandum NOS-OR&R-46, 2013.

[2] Japan Agency for Marine-Earth Science and Technology, »JAMSTEC Deep-sea Debris Database,« [Mrežno]. Available: http://www.godac.jamstec.go.jp/catalog/dsdebris/e/. [Pokušaj pristupa 5 Lipanj 2020].

[3] M. Fulton, J. Hong i J. Sattar, »TrashCan: A Semantically-Segmented Dataset towards Visual Detection of Marine Debris,« 2020.

[4] L. F. Guilhoto, An Overview Of Artificial Neural Networks for, 2018.

[5] S. CS231n, »Mathematical model of artifical neuron,« [Mrežno]. Available: https://cs231n.github.io/neural-networks-1/. [Pokušaj pristupa 16 Rujan 2020].

[6] R. Duda, P. Hart i D. Stork, Pattern Classification, 2nd ur., Wiley, 2012.

[7] A. Krizhevsky, I. Sutskever i G. E. Hinton, »ImageNet Classification with Deep Convolutional Neural Network,« 2012.

[8] I. Goodfellow, Y. Bengio i A. Courville, Deep Learning, MIT Press, 2016.

[9] C. D. V. González, »Classification of Motor Imagery EEG Signals with CSP Filtering Through Neural Networks Models,« 2018.

[10] C. C. Aggarwal, »Introduction,« u *Neural Networks and Deep Learning*, Springer, 2018, pp. 1-4.

[11] G. Cybenko, »Approximation by Superpositions of a Sigmoidal Function,« 1989.

[12] M. Leshno, V. Lin, A. Pinkus i S. Schocken, »Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,« *Neural Networks,* svez. 6, br. 4, pp. 861-867, 1993.

[13] J. Krapac i S. Šegvić, »Konvolucijski modeli,« [Mrežno]. Available: http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf. [Pokušaj pristupa 06 Lipanj 2020].

[14] G. Weiguang Ding, »Github,« [Mrežno]. Available: https://github.com/gwding/draw_convnet.

[15] A. Amidi i S. Amidi, »CS 230 - Convolutional Neural Networks Cheatsheet,« [Mrežno]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#object-detection. [Pokušaj pristupa 16 Rujan 2020].

[16] A. Zhang, Z. C. Lipton, M. Li i A. J. Smola, Dive into Deep Learning, 0.8.0 ur., 2020.

[17] V. Dumoulin i F. Visin, »A guide to convolution arithmetic for deep learning,« 12 Siječanj 2018. [Mrežno]. Available: https://arxiv.org/pdf/1603.07285.pdf. [Pokušaj pristupa 20 Lipanj 2020].

[18] »CS231n Class Notes,« 2017. [Mrežno]. Available: https://cs231n.github.io/neural-networks-1/#actfun. [Pokušaj pristupa 6 Siječanj 2020].

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li i L. Fei-Fei, »ImageNet: A Large-Scale Hierarchical Image Database,« u *CVPR09*, 2009.

[20] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. G. J. Hays, P. Perona, D. Ramanan, C. Zitnick i P. Dollár., »Microsoft COCO: Common Objects in Context,« u *Computer Vision – ECCV 2014*, 2014.

[21] Y. J, C. J, B. Y i L. H, »How transferable are features in deep neural networks?«.

[22] C. Shorten i T. M. Khoshgoftaar, »A Survey on Image Data Augmentation,« *Journal of Big Data,* svez. 6, 2019.

[23] S. Šegvić i J. Krapac, »Duboke unaprijedne mreže,« [Mrežno]. Available: http://www.zemris.fer.hr/~ssegvic/du/du1feedforward.pdf. [Pokušaj pristupa 7 Lipanj 2020].

[24] M. Čupić, »Duboko učenje: Optimizacija parametara modela,« 5 Travanj 2019. [Mrežno]. Available: http://www.zemris.fer.hr/~ssegvic/du/du3optimization.pdf. [Pokušaj pristupa 7 Lipanj 2020].

[25] S. Ruder, »An overview of gradient descent optimization algorithms,« Insight Centre for Data Analytics, NUI Galway, Dublin.

[26] Dauphin, Pascanu, Gulcehre, Cho, Ganguli i Bengio, »Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,« 2014.

[27] G. Orr, »Momentum and Learning Rate Adaptation,« [Mrežno]. Available: https://www.willamette.edu/~gorr/classes/cs449/momrate.html. [Pokušaj pristupa 1 Rujan 2020].

[28] D. P. Kingma i J. L. Ba, »ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,« 2015.

[29] Wilson, Roelofs, Stern, Srebro i Recht, »The Marginal Value of Adaptive Gradient Methods in Machine Learning,« Long Beach, CA, 2018.

[30] A. Burkov, The Hundred-Page Machine Learning Book, 1 ur., Andriy Burkov, 2019.

[31] Y. Tong, W. Lu, Q.-q. Deng, C. Chen i Y. Shen, »Automated identification of retinopathy of prematurity by image-based deep learning,« 2020.

[32] S. Ren, K. He, R. Girshick i J. Sun, »Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,« u *Advances in Neural Information Processing Systems 28*, 2015.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu i A. C. Berg, »SSD: Single Shot MultiBox Detector,« 2018.

[34] Lin, Goyal, Girshick, He i Dollar, »Focal Loss for Dense Object Detection,« u *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] R. Padilla i E. A. da Silva, »A Survey on Performance Metrics for Object-Detection Algorithms,« 2020.

[36] M. Fulton, J. Hong, M. J. Islam i J. Sattar, »Robotic Detection of Marine Litter Using Deep Visual Detection Models,« 2019.

[37] Dive Against Debris ®, »Marine Debris Identification Guide,« Project AWARE Foundation, 2015.

PRILOZI

Popis tablica

Popis slika

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam diplomski rad izradio samostalno, isključivo znanjem stečenim na Odjelu za elektrotehniku i računarstvo, služeći se navedenim izvorima podataka i uz stručno vodstvo mentorice izv. prof. dr. sc. Ivana Palunko, kojoj se još jednom srdačno zahvaljujem.

Antun Đuraš