Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Ambrožič

Detekcija in klasifikacija objektov v vodnem okolju s pomočjo konvolucijskih nevronskih mrež

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matej Kristan

Ljubljana, 2018

Zahvaljujem se mentorju izr. prof. dr Mateju Kristanu za vso pomoč, nasvete, vloženi trud in strokovno usmerjanje, ki je vodilo v izdelavo tega dela.

Hvala moji družini za vso spodbudo, potrpežljivost in vso izkazano podporo v času mojega izobraževanja.

Nejc Ambrožič, 2018

Hard work pays off.

"Any fool can know. The point is to understand."

— Albert Einstein

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
NN	neural network	nevronska mreža
CNN	convolutional neural network	konvolucijska nevronska mreža
BN	batch normalization	paketna normalizacija
TP	true positive	pravilno pozitivni
FN	false negative	napačno negativni
AP	average precission	povprečna natančnost

Povzetek

Naslov: Detekcija in klasifikacija objektov v vodnem okolju s pomočjo konvolucijskih nevronskih mrež

Problematika, ki jo naslavljamo v magistrskem delu, je detekcija objektov s konvolucijskimi nevronskimi mrežami v vodnem okolju za detekcijo nevarnih ovir v realnem času. Zanesljiva in hitra detekcija ovir je osnovni problem avtonomne vožnje. Konvolucijske nevronske mreže so pogosto uporabljene v avtonomnih avtomobilih, v vodnem okolju pa še niso bile temeljito preizkušene. V magistrskem delu analiziramo dve izmed najnovejših konvolucijskih nevronskih mrež za detekcijo in klasifikacijo objektov: YOLO in BlitzNet. Predlagamo modificirano konvolucijsko nevronsko mrežo YoloW in novo podatkovno zbirko za detekcijo objektov v vodnem okolju WODD. Podatkovna zbirka vsebuje 19691 anotiranih ovir, ki se pojavijo v 12168 slikah. Predlagamo tudi prilagojen postopek učenja v podatkovni zbirki z negotovimi učnimi primeri, ki je primeren za učenje konvolucijskih nevronskih mrež na realnih podatkovnih zbirkah. V delu evalviramo delovanje predstavljenih konvolucijskih nevronskih mrež na predlagani podatkovni zbirki. S povečevanjem učne množice se natančnost vseh predstavljenih mrež izboljšuje. Po učenju na celotni testni množici podatkovne zbirke doseže mreža BlitzNet povprečno natančnost $89,68\%$, YOLO $96,78\%$, medtem ko naša mreža YoloW doseže $97,72\%$. Mreža YoloW deluje v realnem času in je sposobna detekcije ovir v povprečno $30,12$ slikah na sekundo.

Ključne besede

računalniški vid, strojno učenje, globoko učenje, konvolucijske nevronske mreže, detekcija objektov

Abstract

Title: Object detection and classification in aquatic environment using convolutional neural networks

We address the problem of real-time floating obstacle detection in aquatic environments with convolutional neural networks. Reliable and fast detection of obstacles is crucial for autonomous driving. Convolutional neural networks are often used in autonomous cars but have not yet been thoroughly tested in the aquatic environment. For this purpose, we analyze two of the latest convolutional neural networks for object detection and classification: YOLO and BlitzNet. We propose a modified convolutional neural network for obstacle detection YoloW and a new dataset for object detection in the aquatic environment. The dataset contains 19691 annotated obstacles appearing in 12168 images. We propose a customized learning process from uncertain training examples, which is suitable for training convolutional neural networks on real world datasets. We evaluate the performance of the presented convolutional neural networks on the proposed dataset. By increasing the number of training examples, the accuracy of all models is improved. After training on the entire training set of our dataset, BlitzNet achieves an average accuracy of 89.68%, YOLO 96.78%, while our model YoloW achieves an average accuracy of 97.72%. The proposed YoloW works in real-time and is capable of obstacle detection at 30.12 images per second on average.

Keywords

computer vision, machine learning, deep learning, convolutional neural networks, object detection

Poglavje 1

Uvod

V zadnjem obdobju je bil opazen velik napredek na področju avtonomnih vozil [1]. Praktično vsako avtomobilsko podjetje vlaga v razvoj različnih sistemov avtonomnme vožnje [2], ustanavljajo se zagonska podjetja [3, 4], ki se ukvarjajo z razvojem sistemov za avtonomno vožnjo in celo na fakultetah se vpeljujejo predmeti, katerih cilj je spoznati znanja potrebna za avtonomno vožnjo [5]. Najočitnejši primer podjetja, ki se zaveda pomembnosti avtonomne vožnje je Alphabet, ki je lastnik podjetja Google. Ta s svojim sistemom avtentikacije reCAPTCHA [6] pridobiva označene slike cestno prometnih znakov, kar s pridom izkorišča njihovo sestrsko podjetje Waymo, ki se ukvarja z razvojem avtonomnih vozil [7]. Rezultat napredka so rešitve, ki se razprostirajo od popolnoma avtonomne vožnje [8] do sistemov za pomoč pri vožnji, kot so pomoč pri vzdrževanju voznega pasu, avtomatsko sledenje prometu, zaznava ostalih vozil v voznikovem mrtvem kotu, samodejno parkiranje in preprečevanje naletov [9, 10]. Razviti so bili tudi že sistemi avtopilotov, ki so dosegli že skoraj popolno avtonomijo, kot so projekti podjetij Tesla, Waymo, Uber in drugi.

Kljub velikim investicijam, razvoju in napredku na področju avtonomnih avtomobilov so bila plovila deležna precej manj pozornosti. V zadnjih letih je bilo veliko avtonomnih plovil razvitih za obrambne [11] in okoljevarstvene namene, kjer so bila uporabljena za izvajanje meritev kakovosti vode [12].

Večinoma so bila avtonomna plovila uporabljena za podvodno delovanje in raziskovanje [13, 14]. Še manj pozornosti pa je bilo namenjene površinskim plovilom, kjer je zaznavanje okolice veliko težje in bolj raznoliko, saj je na vodni gladini veliko več aktivnosti in s tem povezanih več objektov, ki jih je potrebno prepoznavati v realnem času.

Človeška aktivnost se na morski gladini izraža na več načinov. Najočitnejši je seveda ladijski promet, kjer izziv predstavljajo raznolike oblike in velikosti ladij. Poleg ostalih plovil na gladini najdemo še različne boje oziroma druge oznake, ki so namenjene ureditvi morskega prometa; v bližini obal je potrebno biti pozoren na plavalce.

Za uspešno avtonomno navigacijo se morajo vozila in plovila zavedati svoje okolice in prepoznavati objekte v njej. Hkrati mora prepoznavanje okolice biti dovolj hitro in delovati v realnem času. Zaznava objektov v okolici je temeljni problem avtonomne navigacije, saj se vse odločitve, ki jih sistemi sprejmejo, zanašajo na pravilno zaznavo in identifikacijo objektov v okolici. Metode morajo tako biti dovolj robustne, delovati v realnem času in hkrati dovolj splošne za zaznavo vseh omenjenih objektov ter potencialno ostalih nepričakovanih objektov.

Detekcija objektov je eden klasičnih problemov računalniškega vida in praviloma velja za težak izziv. Z ostalimi področji računalniškega vida je sorodna, saj zahteva metodo, ki je invariantna na deformacijo objektov, spremembam v osvetlitvi in zorni točki. Detekcija objektov je poseben problem, ker vključuje iskanje in kategorizacijo regij slike. Za detekcijo objekta moramo imeti nekakšno idejo, kje bi objekt lahko bil in kako je slika segmentirana. To ustvari problem tipa "kokoš in jajce", kjer moramo poznati lokacijo objekta, da bi prepoznali njegovo obliko in razred. Na drugi strani potrebujemo lokacijo objekta, da poznamo njegovo obliko.

Problematika, ki jo naslavljamo v magistrski nalogi, je detekcija objektov s konvolucijskimi nevronskimi mrežami [15] v vodnem okolju za detekcijo nevarnih ovir. Te so pogosto uporabljene za avtonomne avtomobile, a v vodnem okolju še niso bile preizkušene.

1.1 Pregled sorodnih del

Razvoj konvolucijskih nevronskih mrež predstavlja osnovo za napredek na področju avtonomne vožnje. Konvolucijske nevronske mreže so v zadnjem obdobju ena izmed najpopularnejših metod pri problemih računalniškega vida. Njihova velika prednost je sposobnost delovanja neposredno s slikami, tako da raziskovalcem ni potrebno ročno generirati značilk. Prvotno so se raziskovalci večinoma osredotočili na probleme klasifikacije (AlexNet [16] leta 2012 in VGG [17] leta 2014), torej za dano sliko napovedati, kaj je na sliki. Kasneje so se začenjale pojavljati arhitekture nevronskih mrež, ki so naslovile tudi problem detekcije, torej kje na sliki je objekt, in ki jih v nadaljevanju predstavimo.

Avtorji Girshick idr. [18] so zaslužni za velik napredek na področju konvolucijskih nevronskih mrež za probleme detekcije, ko so leta 2014 predstavili metodo na regijah temelječe konvolucijske nevronske mreže (angl. *region-based Convolutional Neural Networks*, r-CNN). Ta je ob predstavitvi izboljšala do takrat najboljšo doseženo natančnost $40,9\%$ na podatkovni zbirki PASCAL VOC 2012 [19] in dosegla $53,3\%$ natančnost. Leto kasneje je avtor predlagal izboljšavo originalne metode in predstavil metodo hitre na regijah temelječe konvolucijske mreže (angl. *Fast r-CNN*) [20]. Ren idr. [21] so predstavili metodo hitrejše na regijah temelječe konvolucijske mreže (*angl. Faster r-CNN*), ki trenutno velja za najboljšo izvedbo konvolucijskih nevronskih mrež, ki temeljijo na ideji predlaganja regij. Redmon idr. [22] so leta 2016 predstavili metodo YOLO (angl. *You Only Look Once*), kjer za razliko od prej predstavljenih metod predstavijo problem detekcije kot regresijski problem in s tem dosežejo občutno pohitritev delovanja. Avtorji kasneje predstavijo model YOLO9000 [23], ki izboljša natančnost in hitrost delovanja originalnega modela. Leta 2016 so Liu idr. [24] predstavili metodo SSD (angl. *Single Shot MultiBox Detector*), ki sledi metodi YOLO in predlaga enotno konvolucijsko nevronsko mrežo za detekcijo.

V zadnjem obdobju je veliko avtorjev predstavilo nove arhitekture konvolucijskih nevronskih mrež za semantično segmentacijo slik. Med novejšimi in

najboljšimi so popolnoma konvolucijske nevronske mreže za semantično segmentacijo (*angl. Fully Convolutional Networks for Semantic Segmentation*) [25], U-net [26], ki je sicer bil uporabljen za segmentacijo biomedicinskih slik, Segnet [27] in najnovejši Blitznet [28], ki nadgradi metodo za detekcijo SSD in doda zmogljivosti izvajanja semantične segmentacije poleg detekcije ter hkrati deluje hitreje od vseh prej predstavljenih segmentacijskih metod. Kristan idr. [29] so problem semantične segmentacije v vodnem okolju reševali z metodo mešanice Gaussov.

1.2 Prispevki

V okviru dela so bile izvedene študije metod za detekcijo in klasifikacijo objektov v vodnem okolju. Predstavljeni in analizirani sta bili dve izmed najnovejših konvolucijskih nevronskih mrež za detekcijo objektov YOLO [23] in BlitzNet [28].

Glavni prispevek dela je nova metoda za detekcijo in klasifikacijo objektov v vodnem okolju, ki deluje v realnem času. Predlagana metoda uporablja prilagojeno konvolucijsko nevronsko mrežo na podlagi ene izmed predhodno predstavljenih in analiziranih arhitektur. Predstavljene prilagoditve izboljšajo natančnost in hitrost učenja.

Predstavimo modifikacijo metode učenja konvolucijskih nevronskih mrež, ki omogoča učenje mreže na učnih podatkih z negotovimi anotacijami. Definiramo področje pozornosti, ki definira okolico, v kateri se detekcije morajo pojaviti, medtem ko detekcije zunaj tega območja (na kopnem, nad obzorjem) ne vplivajo na postopek učenja.

Predlagamo novo podatkovno zbirko za razvoj metod detekcije objektov v vodnem okolju WODD (angl. *Water Object Detection Dataset*). Podatkovna zbirka je ločena na učni in testni del. Učni del je sestavljen iz že objavljenih podatkovnih zbirk MODD (angl. Marine Object Detection Dataset) [30] ter SMD (angl. *Singapore Maritime Dataset*). Dodane so bile sekvence in slike vzete s spleta. Slednje so bile ročno anotirane. Testni del množice sestavlja

podatkovna zbirka MODD2 (angl. *Marine Object Detection Dataset 2*) [31]. Že obstoječe anotacije podatkovnih zbirk MODD [30] in SMD so bile pretvorjene v format, ki ga za detekcijo objektov uporabljajo nekatere od najbolj znanih in pogostokrat uporabljenih podatkovnih zbirk, kot so PASCAL VOC 2007 in VOC 2012 [32, 19] ter Microsoft COCO [33]. Večina metod predstavljenih v Poglavju 1.1 za učenje in testiranje uporablja eno ali več omenjenih podatkovnih zbirk. Pretvorba anotacij omogoča uporabo podatkovne zbirke MODD, MODD2 in SMD z omenjenimi modeli. Hkrati bo raziskovalcem, ki uporabljajo bolj znane podatkovne zbirke, ponudila dodatno podatkovno zbirko za učenje in evalvacijo svojih modelov.

1.3 Struktura naloge

Magistrsko delo je sestavljeno iz šestih poglavij. V Poglavju 2 predstavimo konvolucijske nevronske mreže in njihove gradnike. V Poglavju 3 predstavimo dve arhitekturi konvolucijskih nevronskih mrež, ki smo jih v našem delu analizirali. V Poglavju 4 predstavimo glavna prispevka našega dela: našo konvolucijsko nevronsko mrežo in novo podatkovno zbirko za detekcijo ovir v vodnem okolju. V Poglavju 5 predstavimo kvantitativno in kvalitativno analizo delovanja izbranih konvolucijskih nevronskih mrež. Sklepne ugotovitve predstavimo v Poglavju 6.

Poglavje 2

Konvolucijske nevronske mreže

Konvolucijske nevronske mreže (angl. *Convolutional Neural Network*, CNN) so prvič uspešno uporabili LeCun in drugi leta 1989, ko so s pomočjo vzvratnega prehoda (angl. *backpropagation*) naučili mrežo prepoznave ročno napisanih poštnih številk [34]. Danes imajo konvolucijske nevronske mreže pomemben vpliv na področju računalniškega vida. Za svoje delovanje potrebujejo veliko število učnih podatkov. Zaradi vedno večjega števila učnih podatkov in vedno bolj kompleksnih arhitektur nevronskih mrež sta pomemben dejavnik postala čas in strošek učenja. Zato se je razvilo veliko ogrodij za implementacijo nevronskih mrež, ki optimizirajo delovanje na grafičnih karticah, ki so zaradi svoje arhitekture najboljša strojna oprema za učenje nevronskih mrež.

V Poglavju 2.1 opišemo nevronske mreže in v Poglavju 2.1 algoritem vzvratnega razširjanja napake, ki je algoritem, ki nevronskim mrežam omogoča učenje. Nato v Poglavju 2.3 predstavimo posebnosti konvolucijskih nevronskih mrež in vse uporabljene računske bloke, ki jih uporabljamo v našem delu.

2.1 Nevronske mreže

Nevronska mreža (angl. *Neural Network*, NN) definiramo kot funkcijo $g(\cdot)$, ki slika poljubne vhodne podatke $\boldsymbol{x}$ v izhod $\boldsymbol{y}$. Funkcija $g(\cdot) = f_L \circ ... \circ f_1$ je kompozitna in sestavljena iz več preprostejših, katere imenujemo nivoji ali bloki. Če definiramo $\boldsymbol{x_0}$ kot vhod mreže, se vsak vmesni izhod $\boldsymbol{x_1, x_2, ..., x_L}$ izračuna kot rezultat funkcije $f_l(\cdot)$ izhoda prejšnega nivoja $\boldsymbol{x_{l-1}}$ in naučenega parametra uteži $\boldsymbol{w_l}$,

$$\boldsymbol{x_i} = f_l(\boldsymbol{x_{l-1}}, \boldsymbol{w_l}). \tag{2.1}$$

2.2 Vzvratno razširjanje napake

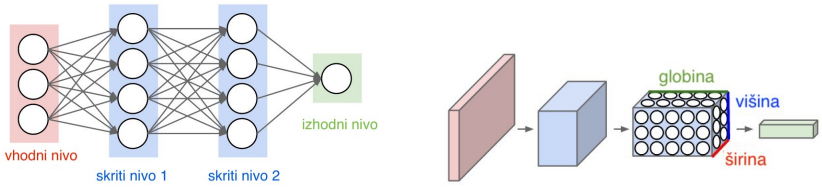
Vzvratno razširjanje napake je metoda za izračun odvodov uteži nevronske mreže, ki je predstavljena kot kompozitna funkcija $g(\cdot)$ v odvisnosti od funkcijskih vhodov in je praktična implementacija verižnega pravila (angl. *chain rule*) [35]. Nevronska mreža se uči z izbiro uteži vseh nevronov, tako da se mreža nauči preslikati ciljne izhode iz znanih vhodov. Proces učenja nevronske mreže z vzvratnim razširjanjem napake je predstavljen kot optimizacijski problem, kjer želimo minimizirati vrednost kriterijske funkcije (pogosto poimenovane tudi cost/error function), ki za dane vhode izračuna odstopanje napovedi mreže od resničnih podatkov (angl. *ground truth*). Analitično je te uteži praktično nemogoče izračunati v globokih nevronskih mrežah. Vzvratno razširjanje napake predstavi preprosto in efektivno rešitev za izračun želenih uteži iterativno. Standardna rešitev uporablja gradientni spust (angl. *gradient descent*) kot optimizacijsko metodo. Gradientni spust ne zagotavlja globalnega minimuma napake, ampak z nastavitvijo pravilnih parametrov deluje dobro v praksi. Veliko raziskovalcev je namreč prepričanih, da so lokalni minimumi zadovoljiva rešitev pri optimizaciji nevronskih mrež, saj so njihove vrednosti približno podobne globalnemu minimumu zaradi kompleksnosti mrež [36].

Pri inicializaciji nevronske mreže se uteži nastavijo na določene vrednosti.

V teoriji se lahko uteži inicializira naključno, kar se je pokazalo, da v praksi vodi v nestabilno učenje (problem izginjajočih gradientov [37]) in počasnejšo konvergenco mreže. Boljša rešitev je inicializacija uteži z vzorčenjem različnih porazdelitev [38, 39, 40]. Algoritem v prvi fazi vhodno matriko pošlje naprej po nevronski mreži. Izhod nevronske mreže se primerja z želenim izhodom z uporabo kriterijske funkcije. To razliko pogosto poimenujemo vrednost napake nevronske mreže. Odvod kriterijske funkcije napake se nato razširja po mreži nazaj. Ker je vsak računski blok nevronskih mrež odvedljiv, se na vsakem nivoju izračuna lokalni gradient z uporabo verižnega pravila odvajanja. V zadnjem koraku se uteži vsakega nevrona posodobijo glede na svoj gradient, od katerega se odšteje delež gradienta napake. Ta delež se imenuje stopnja učenja (angl. *learning rate*), ki je lahko dinamičen ali statičen. Ko so vse uteži posodobljene postopek ponovimo z drugim vhodom, dokler vrednosti uteži ne konvergirajo.

2.3 Konvolucijske nevronske mreže

Konvolucijske nevronske mreže so dobile ime po enem od njihovih glavnih gradnikov – računskih blokov konvolucije. Konvolucijske nevronske mreže so pogosto uporabljene za računalniški vid, saj imajo podatke v slikah prostorsko strukturo, kar dobro zajame konvolucijski računski blok. Vizualni vhod konvolucijske nevronske mreže $\boldsymbol{x_i}$ ima praviloma 3 dimenzije: višino H, širino W in število kanalov C. Slikovni vhodi v mrežo imajo načeloma 3 kanale C: rdečega, zelenega in modrega (angl. *red, green, blue*, RGB). Višina in širina pa se interpretirata kot prostorski dimenziji. Slika 2.1 prikazuje primerjavo arhitekture preproste nevronske mreže in konvolucijsko nevronsko mrežo, kjer so prikazani večdimenzionalni podatki.



(a) NN **(b)** CNN

Slika 2.1: Primerjava preproste 3-slojne nevronske mreže (a) in preproste konvolucijske nevronske mreže (b), kjer je vidno, kako so nevroni razporejeni v treh dimenzijah. Slika povzeta po [41].

2.4 Računski bloki

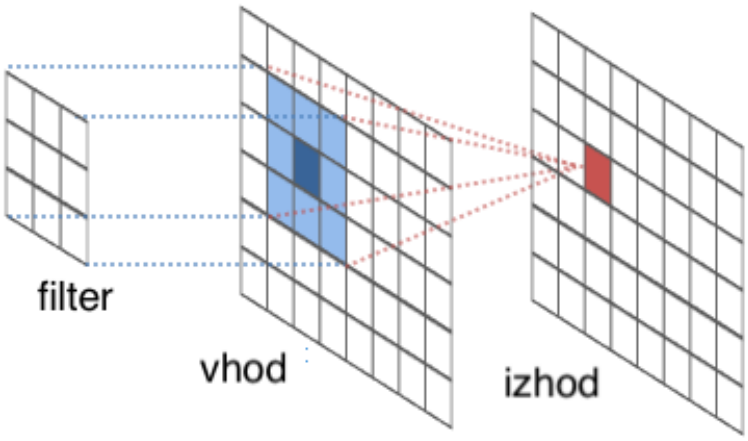
V tem podpoglavju bomo opisali standardne gradnike nevronskih mrež - računske bloke, ki so uporabljeni v delu: konvolucija, aktivacijske funkcije, združevalni nivo, paketno normalizacijo in funkcijo Softmax.

2.4.1 Konvolucija

Konvolucjski nivo izračuna konvolucijo vhoda $\boldsymbol{x} \in \mathbb{R}^{H \times W \times D}$ z več filtri $\boldsymbol{f} \in \mathbb{R}^{H' \times W' \times D \times D''}$ in določeno pristranskostjo b (angl. *bias*). Rezultat konvolucije $\boldsymbol{y} \in \mathbb{R}^{H'' \times W'' \times D''}$ je vhod v naslednje nivoje konvolucijske nevronske mreže.

Vsak filter $\boldsymbol{f}$ je matrika števil, ki jih imenujemo uteži in so naučeni razpoznavati določene značilke v vhodu $\boldsymbol{x}$. Konvolucija je operacija med vhodom $\boldsymbol{x}$ in filtrom $\boldsymbol{f}$. Vsak filter premikamo po celotnem vhodu, pri čemer mora biti celoten filter znotraj vhoda $\boldsymbol{x}$, kar vodi v zmanjšanje prostorskih dimenzij izhoda. Če želimo ohraniti prostorske dimenzije izhoda, je potrebno vhod obrobiti (angl. *padding*). Praviloma ima dodana obroba same vrednosti 0. Izhodna vrednost operacije konvolucije je utežena vsota vhodnih vrednosti, uteži pa predstavljajo vrednosti filtra, kar prikazuje Slika 2.2. Rezultat konvolucije fitrov $\boldsymbol{f}$ z vhodom $\boldsymbol{x}$ in pristranskostjo b se lahko izrazi kot

$$y_{i''j''d''} = \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^{D} (f_{i'j'd} \times x_{i''+i'-1, j''+j'-1, d''}) + b_{d''}. \tag{2.2}$$



Slika 2.2: Prikaz delovanja filtra na vhodu $\boldsymbol{x}$.

2.4.2 Aktivacijske funkcije

Aktivacijske funkcije so tisti računski blok, ki v model vpeljuje nelinearnost. Brez aktivacijskih funkcij bi se tako mreže obnašale kot preprost percepetron. Obstaja več takih funkcij, kot so linearna, sigmoidna (v preteklosti najbolj uporabljena), hiperbolični tangens in ReLU (ang. *Rectified Linear Unit*). Slednja se je v praksi za konvolucijske nevronske mreže pokazala za najbolj uporabno. V zadnjih letih so se pojavile še variacije ReLU aktivacijske funkcije, kot so uhajajoča ReLU (angl. *Leaky ReLU*) [42], exponentna linearna enota (angl. *Exponential Linear Unit*, ELU) [43] in pomanjšana eksponentna linearna enota (angl. *Scaled exponential Linear Unit*, SELU) [44]. Izmed naštetih so v našem delu uporabljene (Slika 2.3):
linearna aktivacijska funkcija

$$a(x) = x, \tag{2.3}$$

sigmoidna aktivacijska funkcija

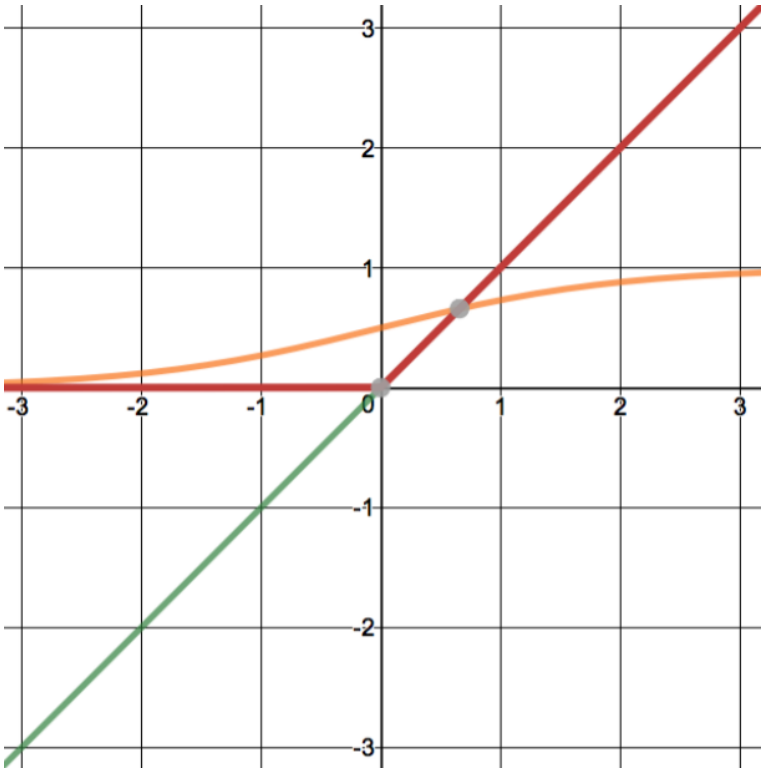
$$a(x) = \frac{1}{1 + e^{-x}}, \tag{2.4}$$

aktivacijska funkcija ReLU

$$a(x) = max(0, x), \tag{2.5}$$

in Leaky ReLU s parametrom $\alpha = 0.01$,

$$a(x) = max(\alpha x, x). \tag{2.6}$$

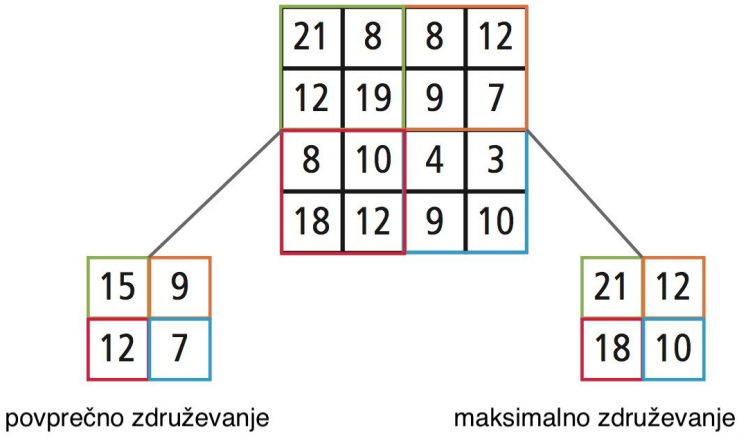


Slika 2.3: Vizualizacija aktivacijskih funkcij: ReLU v rdeči, sigmoidna funkcija v oranžni in linearna v zeleni, ki se v pozitivnem delu x osi prekriva z ReLU.

2.4.3 Združevalni nivo

Združevalni nivoji (angl. *Pooling Layers*) so pogosto postavljeni med zaporedne konvolucijske nivoje. Namenjeni so postopnemu zmanjšanju prostorske dimenzije vhodov v naslednje nivoje in zmanjšanju števila parametrov ter potrebnih računskih operacij. Hkrati služijo zmanjševanju lokacijske odvisnosti med značilkami [45] in preprečujejo preveliko prilagajanje učnim podatkom

(angl. *overfitting*). V praksi sta se uveljavila dva tipa združevalnih blokov: povprečno združevanje (angl. *Average Pooling*) in maksimalno združevanje (angl. *Max Pooling*). Združevalni blok, z velikostjo okna $w \times h$. Poenostavljen primer primerjave delovanja povprečnega in maksimalnega združevanja prikazuje Slika 2.4, kjer so vidne razlike med obema pristopoma.



Slika 2.4: Prikaz delovanja povprečnega in maksimalnega združevanja z velikostjo okna 2×2.

2.4.4 Paketna normalizacija

Paketna normalizacija (angl. *Batch Normalization*) je tehnika, ki izboljšuje natančnost in stabilnost učenja nevronskih mrež. Gre za poseben računski blok, saj izvaja operacije nad celotnimi izhodi predhodnih linearnih nivojev in pred nelinearnimi računskimi bloki. Ker se normalizacija izvaja na vsakem nivoju, se osredotočimo na aktivacijo $\boldsymbol{x}^{(k)}$ in za jasnost izpustimo (k). Vsak paket $\boldsymbol{\beta}$ velikosti M ima m aktivacij

$$\boldsymbol{\beta} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}\}\,. \tag{2.7}$$

Če označimo normalizirane vrednosti paketa kot $x'_{1..M}$ in njihove linearne transformacije kot $y_{1..M}$, potem definiramo transformacijo paketne normali-

zacije kot

$$BN_{\gamma,\beta} : x_{1..m} \rightarrow y_{1...m}. \tag{2.8}$$

Za vsak paket se izračuna njegovo povprečje

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x_i} \tag{2.9}$$

in varianca

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1)}^{m}(x_i - \mu_\beta)^2. \tag{2.10}$$

Aktivacijska mapa se normalizira kot opisano v (2.11). Za potrebe numerične stabilnosti je v imenovalec izračuna dodan ϵ:

$$\boldsymbol{x'} = \frac{\boldsymbol{x_i} - \mu_\sigma}{\sqrt{\sigma_\beta^2 + \epsilon}}. \tag{2.11}$$

Rezultat transformacije paketne normalizacije (BN) je y_i, ki je podan kot

$$\boldsymbol{y_i} = \gamma \boldsymbol{x_i'} + \boldsymbol{\beta} \equiv BN_{\gamma,\beta}(\boldsymbol{x_i}). \tag{2.12}$$

Tako imajo vhodi v nelinearne računske bloke vedno enotni standardni odklon in povprečje 0. V praksi se je pokazalo da paketna normalizacija vodi k hitrejši konvergenci modela in hkrati odstrani potrebo po drugih oblikah regularizacije [46].

2.4.5 Funkcija Softmax

Funkcija Softmax je generalizacija logistične funkcije, ki preslika poljuben K dimenzionalen vektor $\mathbf{z}$ v K dimenzionalen vektor $\sigma(\boldsymbol{z})$ realnih števil, kjer je vrednost vsakega elementa v intervalu $(0, 1]$, medtem ko je vsota vseh vrednosti enaka 1. Funkcija je definirana kot

$$\sigma : \mathbb{R}^K \rightarrow \left\{ \sigma \in \mathbb{R}^K | \sigma_i > 0, \sum_{i=1}^{K} \sigma_i = 1 \right\}, \tag{2.13}$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad j = 1, \ldots, K. \tag{2.14}$$

Poglavje 3

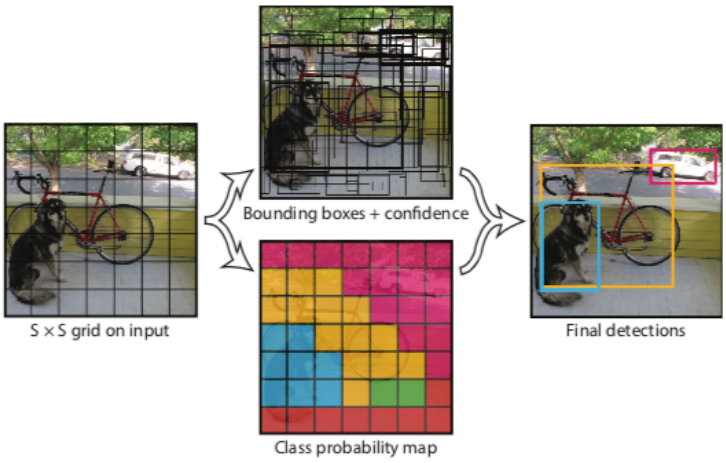
Detekcija objektov s CNN

Pri izboru konvolucijskih nevronskih mrež (CNN), uporabljenih v delu sta bila glavna kriterija natančnost in hitrost delovanja. Na podlagi pregleda sorodnih del in preliminarnih testov smo izbrali YOLO [47] in BlitzNet [28]. Arhitekturo YOLO na kratko predstavimo v Poglavju 3.1, BlitzNet pa v Poglavju 3.2. Obe arhitekturi sta bili uporabljeni in analizirani za detekcijo ter klasifikacijo objektov.

3.1 Arhitektura YOLO

Arhitekturo YOLO (angl. *You Only Look Once*) so leta 2016 predstavili Redmon idr. [47]. Posodobljeno arhitekturo sta leto kasneje predstavila Redmon in Farhadi [48]. Posebnost arhitekture YOLO, po kateri se razlikuje od ostalih detektorjev objektov je, da uporabi eno konvolucijsko nevronsko mrežo za klasifikacijo in lokalizacijo objektov, pri čemer uporabi omejevalno polje (angl. *bounding box*). YOLO za učenje uporablja cele slike in neposredno optimizira delovanje detekcije. Opisan enotni pristop ima več prednosti. Prva prednost s stališča avtonomne vožnje je, da mreža deluje zelo hitro. Ker je detekcija objektov predstavljena kot regresijski problem, ni potrebe po kompleksnem sistemu pred in po-procesiranja. Druga prednost je, da model pri napovedih upošteva celotno sliko. Za razliko od tehnik premikajočih se

oken (angl. *sliding windows*) in predlogov regij (angl. *region proposal*) model vidi celotno sliko med treniranjem in med testiranjem ter se tako implicitno nauči konteksta, v katerem se določeni razredi objektov pojavljajo in kako izgledajo. Fast R-CNN [20] pogosto dele ozadja napačno napove kot objekt. Zaradi dostopa do globalnih informacij o sliki YOLO takih napak naredi veliko manj. Prav tako se model nauči zelo dobre generalizacijske predstavitve objektov.



Slika 3.1: Sistem modelira detekcijo kot regresijski problem. Sliko razdeli na mrežo velikosti $S \times S$ in za vsako celico mreže napove omejevalno polje B (angl. *bounding box*), verjetnosti teh polj in razred C ter njegovo verjetnost. Te predikcije so zakodirane kot matrike dimenzij $S \times S \times (B \times 5 + C)$. Slika vzeta iz [47].

Mreža razdeli vhodno sliko na mrežo velikosti $S \times S$. Če center objekta pade v določeno celico mreže, je ta specifična celica zadolžena za detekcijo objekta. Vsaka celica napove B različnih omejevalnih polj in njihove verjetnosti. Verjetnosti je možno interpretirati kot zaupanje modela, torej ali dano omejevalno polje vsebuje objekt in kako natančno je omejevalno polje postavljeno nad objekt.

YOLO uporablja edinstveno in razmeroma enostavno konvolucijsko ne-

vronsko mrežo, ki je predstavljena v Tabeli 3.1. Vsebuje 19 konvolucijskih in 5 združevalnih nivojev. Veliko sistemov za detekcijo uporablja VGG16 [17] kot izhodiščni model, ki iz slike izlušči značilke, a je izredno kompleksen. Konvolucijski nivoji VGG16 zahtevajo za en prehod slike skozi mrežo $30,69$ milijard operacij s plavajočo vejico za sliko dimenzij 224×224, medtem ko mreža YOLO zahteva le $8,52$ milijard operacij. Na zadnjem nivoju arhitekture YOLO je uporabljena sigmoidna aktivacijska funkcija; ostalim konvolucijskim plastem sledi računski blok *Leaky ReLU*, ki je opisan v Poglavju 2.4.2.

Zadnji nivo mreže YOLO napove verjetnosti posameznih razredov in koordinate omejevalnega polja. Predikcija omejevalnega polja in njene komponente so vizualizirane na Sliki 3.2. Izhod mreže je 5 predikcij za vsako celico mreže. Vsak izhod je sestavljen iz 5 koordinat omejevalnega polja: t_x, t_y, t_w, t_h in t_o. Center celice mreže je od levega zgornjega roba slike oddaljen za (c_x, c_y) in ima omejevalno polje predhodno širino p_w in višino p_h. Predikcija omejevalnega polja b, ki je sestavljeno iz

$$b_x = \sigma(t_x) + c_x, \tag{3.1}$$

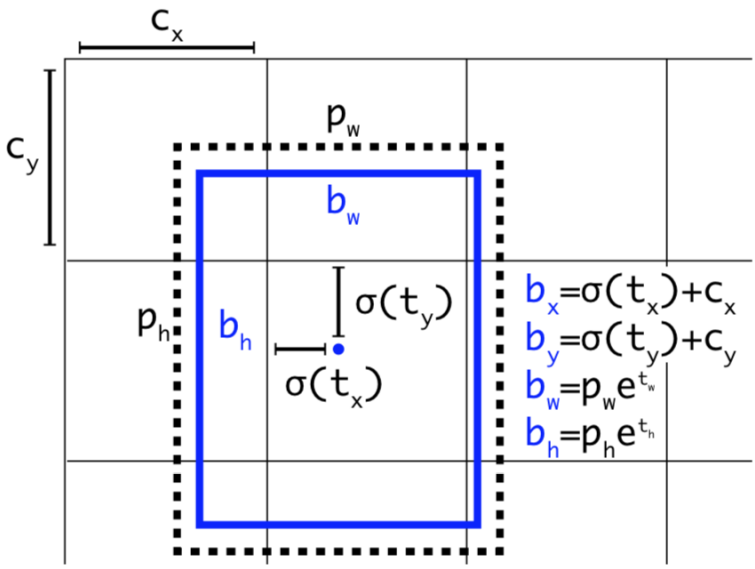
$$b_y = \sigma(t_y) + c_y, \tag{3.2}$$

$$b_w = p_w e^{t_w}, \tag{3.3}$$

$$b_h = p_h e^{t_h}. \tag{3.4}$$

računski blok	**število filtrov**	**velikost/korak**	**dimenzije izhodov**
konvolucija	32	3 x 3	224 x 224
maksimalno združevanje		2 x 2/2	112 x 112
konvolucija	64	3 x 3	112 x 112
maksimalno združevanje		2 x 2/2	56 x 56
konvolucija	128	3 x 3	56 x 56
konvolucija	64	1 x 1	56 x 56
konvolucija	128	3 x 3	56 x 56
maksimalno združevanje		2 x 2/2	28 x 28
konvolucija	256	3 x 3	28 x 28
konvolucija	128	1 x 1	28 x 28
konvolucija	256	3 x 3	28 x 28
maksimalno združevanje		2 x 2/2	14 x 14
konvolucija	512	3 x 3	14 x 14
konvolucija	256	1 x 1	14 x 14
konvolucija	512	3 x 3	14 x 14
konvolucija	256	1 x 1	14 x 14
konvolucija	512	3 x 3	14 x 14
maksimalno združevanje		2 x 2/2	7 x 7
konvolucija	1024	3 x 3	7 x 7
konvolucija	512	1 x 1	7 x 7
konvolucija	1024	3 x 3	7 x 7
konvolucija	512	1 x 1	7 x 7
konvolucija	1024	3 x 3	7 x 7
konvolucija	1000	1 x 1	7 x 7
povprečno združevanje		globalno	1000
softmax			

Tabela 3.1: Arhitektura YOLO [48].



Slika 3.2: Predikcija mreže YOLO.

Verjetnost napovedanega objekta $P(objekt)$ lahko izrazimo s pomočjo mere IoU (angl. *Intersection over Union)* med predikcijo omejevalnega polja $\boldsymbol{b}$ in anotacijo objekta

$$P(objekt) * IOU(b, objekt) = \sigma(t_o). \tag{3.5}$$

Širino w in višino h anotacij se normalizira s širino in višino slike, tako da vrednosti padejo na interval med 0 in 1. Parametra x in y lokaliziramo glede na razdaljo iz centra celice, kar pomeni, da sta absolutni vrednosti preslikani na interval med 0 in 1 glede na njuno razdaljo od centra celice, v kateri je bilo omejevalno polje postavljeno. Predikcije širine $\hat{w}$, višine $\hat{h}$, koordinate $\hat{x}$ in $\hat{y}$ sledijo istim normalizacijskim pravilom. Med treniranjem mreža optimizira

večdelno kriterijsko funkcijo

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\
&+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2\right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S^2} 1_i^{obj},
\end{aligned} \tag{3.6}$$

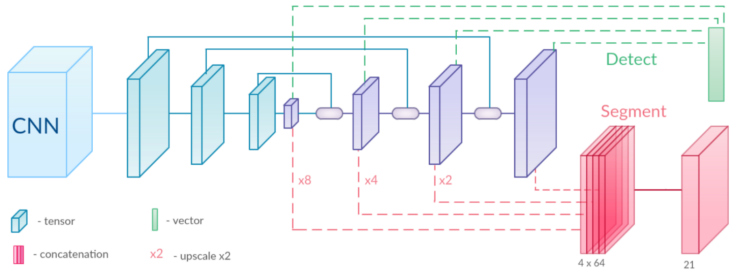
kjer 1_i^{obj} označuje, če je objekt prisoten v i-ti celici in 1_{ij}^{obj} označuje, da je j-ti prediktor omejevalnih polj odgovoren za predikcijo objekta v i-ti celici. Razred objekta označimo s C_i, napovedan razred pa z $\hat{C}_i$. Tako definirana kriterijska funkcija kaznuje klasifikacijske napake le, če je objekt prisoten v celici in kaznuje zgolj napake v lokalizaciji za odgovornega prediktorja.

3.2 Arhitektura BlitzNet

BlitzNet so leta 2017 predstavili Dvornik idr. [28]. Tako kot YOLO opisan v Poglavju 3.1 sodi Blitznet med sisteme, ki opravljajo detekcijo objektov v realnem času. Posebnost mreže BlitzNet je sposobnost opravljanja detekcije in semantične segmentacije sočasno, a hkrati neodvisno enega od drugega. V našem delu smo se osredotočili na del mreže, ki izvaja detekcijo. Arhitektura mreže je prikazana na Sliki 3.3.

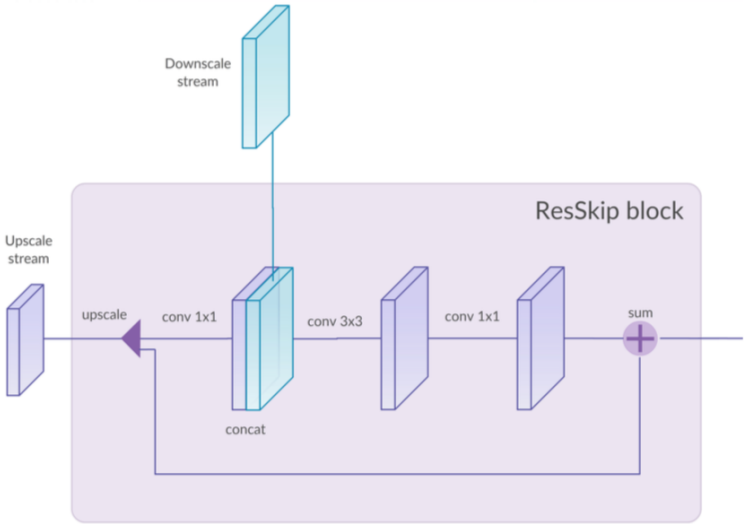
Za razliko od metode YOLO, kjer je celotna arhitektura sestavljene iz ene konvolucijske mreže, BlitzNet uporabi ResNet50 [49] kot ekstraktor visokodimenzionalnih značilk. Pridobljene visokodimenzionalne značilke so nato iterativno zmanjšane. Te se uporabijo za iskanje omejitvenih polj na večih skalah, kar sledi pristopu *Single Shot Detectorju* (SSD) [50]. Pridobljene regije so nato povečane z dekonvolucijskimi nivoji. Na vsakem nivoju zmanjševanja in večanja vhoda mreža izvaja detekcijo objektov v različnih skalah. Predikcije

generira skupek konvolucijskih nivojev: en za detekcijo, en za segmentacijo v enem samem prehodu slike v mreži.



Slika 3.3: Arhitektura BlitzNet. Slika vzeta iz [28].

Arhitektura je zasnovana s preprostimi bloki, imenovanimi *ResSkip*, ki idejno izvirajo iz arhitekture ResNet [49] in uporabijo preskočne povezave. Ti bloki omogočajo združevanje nivojev, kot je prikazano na Sliki 3.4.



Slika 3.4: ResSkip blok, ki omogoča združevanja več nivojev in hkrati pohitri ter poenostavi učenje. Slika vzeta iz [28].

Najprej so vhodne značilke prevzorčene na ustrezno velikost preskočne povezave z uporabo bilinearne interpolacije. Nato so tako prevzorčene in

združene značilke poslane skozi računski blok, ki ga sestavljajo trije konvolucijski nivoji 1×1, 3×3 in 1×1 konvolucija. Izhod bloka *ResSkip* je izračunan kot vsota izhoda omenjenega računskega bloka in prevzorčenih vhodnih značilk. Kriterijska funkcija, ki predstavlja napako mreže, je utežena vsota lokacijske napake L_{loc} in napake zaupanje L_{conf}

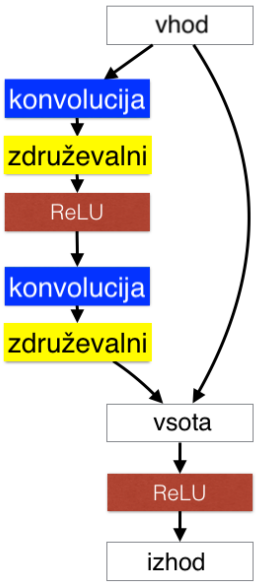
$$L(x, c, l, g) = \frac{1}{N} \left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g) \right), \tag{3.7}$$

kjer je N število privzetih omejevalnih polj. Za lokalizacijo omejevalnega polja in verjetnosti določenega razreda arhitektura uporablja aktivacije vseh predhodnih nivojev (brez ResNet50, ki deluje zgolj kot ekstraktor značilk).

Predikcije omejevalnih polj so ob generiranju predikcij filtrirane z uporabo algoritma *non-maxima supression* (NMS) [51]. Izvajanje NMS na velikem številu predlaganih omejevalnih polj je lahko dolgotrajen postopek. Avtorji zato predlagajo svoj postopek za procesiranje izhodov mreže: za vsak razred izberejo 400 predikcij omejevalnih polj z najvišjo verjetnostjo. Nad izbranimi omejevalnimi polji NMS izlušči zgolj 50 predikcij. Končne predikcije za dano sliko tako predstavlja zgornjih 200 omejevalnih polj z najvišjimi rezultati po aplikaciji NMS.

3.2.1 Mreža ResNet50

Globoke nevronske mreže so omogočile velik napredek na področju računalniškega vida in v času razvoja postajale vedno bolj globoke oziroma večplastne. S povečano kompleksnostjo postaja tudi učenje vedno težje. Ta problem so s predstavitvijo arhitekture ResNet (*Residual Network*) 2015 naslovili Kaiming He idr. [52]. Osnovni blok arhitekture ResNet je vizualiziran na Sliki 3.5. Mreža se ne uči visokodimenzionalnih značilk neposredno, ampak se nauči ostanke (angl. *residual*). Ostanek $\boldsymbol{r}$ je preprosto lahko definiran kot razlika izhoda $\boldsymbol{x_i}$ in vhoda $\boldsymbol{x_i}$ v nivo mreže i: $\boldsymbol{r_i} = \boldsymbol{x_{i-1}} - \boldsymbol{x_i}$. Za učinkovit izračun ostanka uporablja ResNet preskočne povezave (angl. *skip connections*).



Slika 3.5: Prikaz enega bloka preskočne povezave, ki jih uporabljajo arhitekture ResNet.

Mrež, ki uporabljajo arhitekturo ResNet, je več. Razlikujejo se po globini - številu nivojev. ResNet50, ki ga uporablja BlitzNet, sestavlja 50 nivojev, kar je razvidno že iz njegovega imena.

Poglavje 4

Detekcija objektov v vodnem okolju

V tem poglavju podrobno opišemo glavne prispevke našega dela. V Poglavju 4.1 opišemo modifikacije referenčne arhitekture YOLO: YoloW (angl. *You only look once at Water*), nato v Poglavju 4.2 predlagamo in opišemo novo podatkovno zbirko WODD (angl. *Water Object Detection Dataset*) za probleme detekcije v vodnem okolju. V podpoglavju 4.2.1 opišemo postopek učenja konvolucijskih nevronskih mrež na podatkovnih zbirkah z nezanesljivimi anotacijami.

4.1 Prilagoditev arhitekture YOLO: YoloW

Originalni model YOLO omogoča klasifikacijo velikega števila razredov, ki so vsebovani v podatkovnih zbirkah Pascal VOC 2007 in 2017 [32, 19] ter COCO [33]. Za hitro in robustno detekcijo kakršnihkoli nevarnih ovir nima smisla uporabljati mreže, ki lahko detektira tako veliko število razredov objektov. Prav tako v vodnih okoljih ne moremo pričakovati, da bodo ovire ravno tistih kategorij, ki so bile vsebovane v učni množici originalne mreže YOLO. Naš klasifikacijski problem smo tako reducirali na dve kategoriji: **voda**, ki je zelo kompaktna kategorija, in **ovira**, ki je zelo raznolika. V kategorijo ovir so

vključeni vsi objekti v vodi, zato mora mreža biti sposobna visoke stopnje generalizacije.

Potrebne modifikacije mreže se odražajo v zadnjih dveh nivojih arhitekture YOLO. Na zadnjem (softmax) nivoju mreže spremenimo število razredov za detekcijo na 1. Kot opisano v Poglavju 3.1 pri arhitekturi YOLO, zadnji konvolucijski nivo generira omejevalna polja. Sprememba števila napovedanih razredov zmanjša tudi število filtrov, ki so odgovorni za generiranje teh predikcij. Število potrebnih filtrov (*filters*) na zadnjem konvolucijskem nivoju za C razredov se izračuna po podani enačbi:

$$filters = num * (C + 5), \tag{4.1}$$

kjer je *num* parameter zadnjega (softmax) nivoja mreže in predstavlja število omejevalnih polj, ki jih napove zadnji nivo. V primeru enega razreda $C = 1$ to pomeni, da je na zadnjem konvolucijskem nivoju potrebnih 30 filtrov. Prilagojena arhitektura YoloW z opisanimi spremembami je dostopna na [53]. Za pridobitev končnih omejevalnih polj v času generiranja predikcij izhod mreže normaliziramo z algoritmom *non-maxina supression* [51].

4.2 Podatkovna zbirka WODD

Nevronske mreže potrebujejo za dobro delovanje čim večje število učnih primerov, saj vsebujejo ogromno število parametrov. V literaturi še ne obstaja ena sama referenčna podatkovna zbirka, ki bi vsebovala dovolj učnih primerov za učenje konvolucijskih nevronskih mrež za probleme detekcije in klasifikacije objektov v vodnem okolju ter hkrati vsebovala zadostno število testnih primerov za evalvacijo delovanja. Predlagana podatkovna zbirka WODD je sestavljena iz več delov:

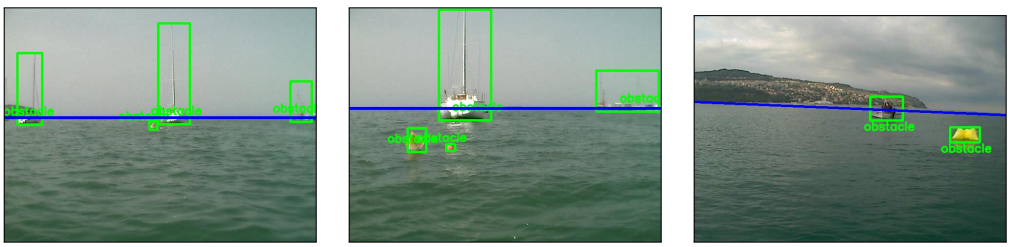
- $WODD_{\text{MODD1}}$: javno dostopna podatkovna zbirka MODD [29]. Vsebuje 4454 anotiranih primerov slik z resolucijo 640×480.
- $WODD_{\text{MODD2}}$: podatkovna zbirka MODD2 predstavljena v [31], ki vsebuje 5156 anotiranih slik z resolucijo 1278×958. To podmnožico

primerov v našem delu uporabljamo kot testno množico, saj gre za najbolj aktualno in obširno od objavljenih podatkovnih zbirk za detekcijo v vodnem okolju. Vsebuje več raznolikih ovir od ostalih in je zato najbolj primerna zbirka za evalvacijo metod za detekcijo objektov.

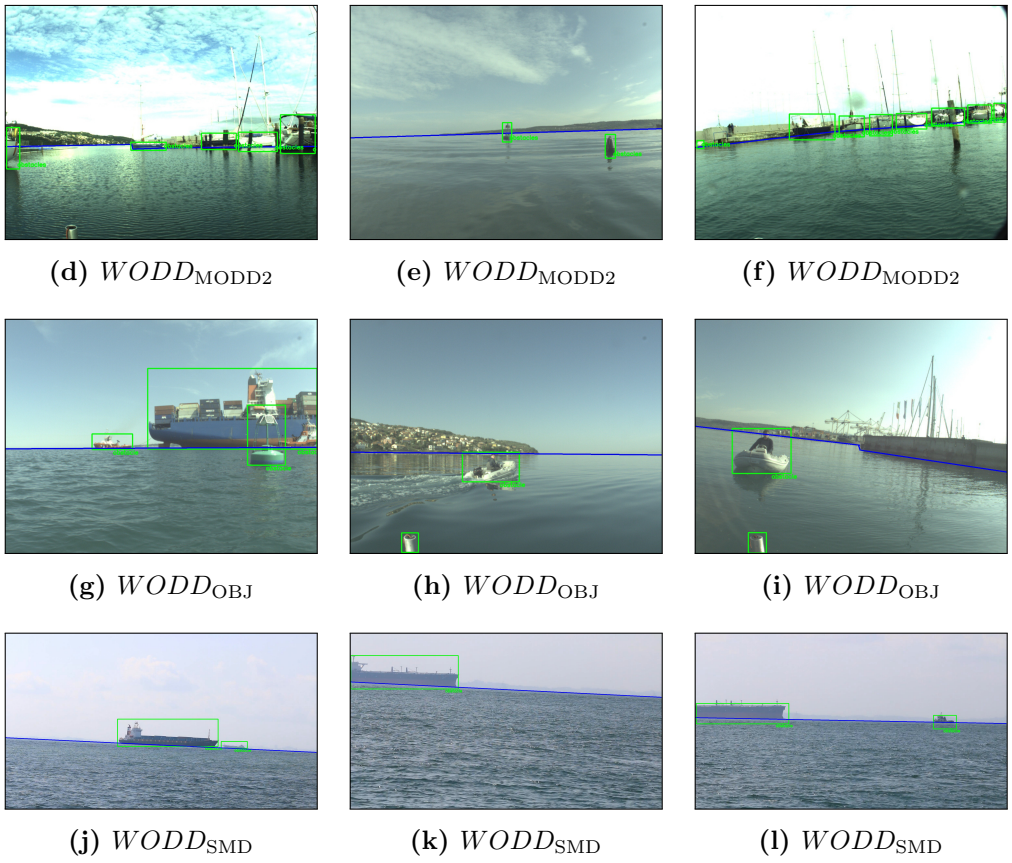
- $WODD_{\text{OBJ}}$: neobjavljena zbirka 162 slik, posnetih v času treh plovb z enakim sistemom kot v MODD2 [31]. Tem sekvencam smo dodali 38 slik ovir v vodi, vzetih s spleta, tako da množica skupno vsebuje 200 anotiranih slik. Zbirka je zanimiva, saj so izbrane slike, kjer so ovire jasno vidne in so zelo raznolike. Anotiranje za to zbirko smo opravili ročno in s tem zagotovili, da so anotirani vsi potrebni objekti. Prav tako je bil ročno anotiran horizont oziroma rob morja.

- $WODD_{\text{SMD}}$: javno dostopna podatkovna zbirka SMD, opisana v [54]. Vsebuje 2396 anotiranih slik z resolucijo 1920×1080, ki vsebujejo anotirane objekte in oznake horizonta.

Podatkovne zbirke smo ročno pregledali in zagotovili, da so v vseh zbirkah, ki smo jih vključili, objekti anotirani konsistentno. Podatkovna zbirka WODD vsebuje 12168 slik, od tega je 7050 slik v učni in 5156 v tesni množici. Celotna podatkovna zbirka vsebuje 19691 anotiranih ovir, od tega učna množica vsebuje 8364, testna množica pa 11327 anotiranih ovir. Slika 4.0 prikazuje primere iz celotne podatkovne zbirke WODD. Distribucijo velikosti objektov v učni in testni množici prikazuje Slika 4.1.



(a) $WODD_{\text{MODD1}}$ **(b)** $WODD_{\text{MODD1}}$ **(c)** $WODD_{\text{MODD1}}$

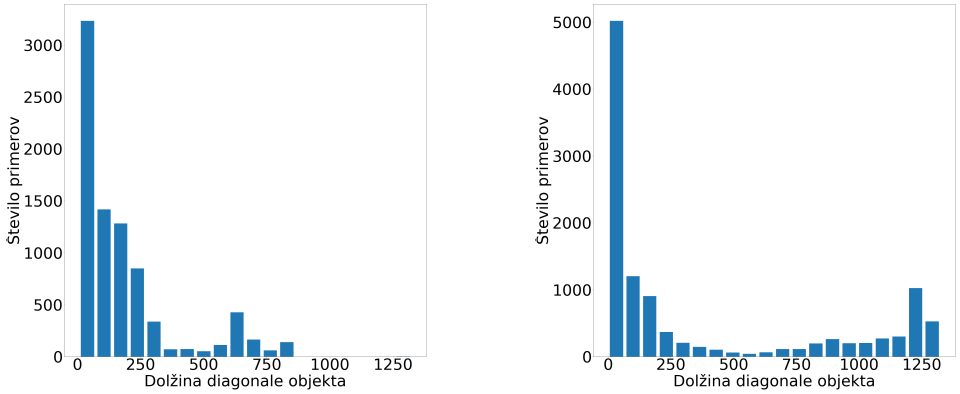


(d) $WODD_{\text{MODD2}}$ **(e)** $WODD_{\text{MODD2}}$ **(f)** $WODD_{\text{MODD2}}$

(g) $WODD_{\text{OBJ}}$ **(h)** $WODD_{\text{OBJ}}$ **(i)** $WODD_{\text{OBJ}}$

(j) $WODD_{\text{SMD}}$ **(k)** $WODD_{\text{SMD}}$ **(l)** $WODD_{\text{SMD}}$

Slika 4.0: Primeri iz vseh podmnožic podatkovne zbirke WODD z zeleno označenimi objekti. Rob morja (oziroma obzorje) je označeno z modro.



(a) Učna množica podatkovne zbirke WODD.

(b) Testna množica podatkovne zbirke WODD.

Slika 4.1: Distribucija velikosti objektov v učni in testni množici podatkovne zbirke WODD. Velikost je izražena kot dolžina diagonale omejevalnega polja v pikslih.

Zaradi različnih resolucij slik in pogojev, v katerih so nastale, je podatkovna zbirka zelo raznolika in predstavlja dobro ogrodje za testiranje robustnostih metod detekcije. Anotacije vsebujejo podatke o lokaciji objektov in horizonta, ki predstavlja vodno gladino.

Anotacije podatkovnih zbirk, ki smo jih vključili v podatkovno zbirko WODD niso bile konsistentne, saj je vsaka uporabljala svoj format. Definirali smo nov format anotacij in vse anotacije podatkovnih zbirk pretvorili v izbran format. Anotacije podatkovne zbirke WODD sledijo standardnemu formatu XML, ki ga uporabljajo ostale znane podatkovne zbirke, kot so Pascal VOC [32, 19] in Microsoft COCO [33]. Vsaka anotacija vsebuje naslednje podatke:

- *filename*: Ime datoteke, za uparjevanje anotacijskih XML datotek in pripadajočih slik.
- *folder*: Ime direktorija, kjer se originalna anotacija pojavi.
- *size*: Vsebuje podatke o višini (angl. *height*), širini (anlg. *width*) v

pikslih in številu barvnih kanalov slike oziroma globini (angl. *depth*).

- *object*: V sliki se lahko pojavi 0 ali več objektov. Vsak je predstavljen z omejevalnim poljem *bndbox*, ki vsebuje lokacijske koordinate objekta $xmin, ymin, xmax, ymax$ ter oznako oziroma labelo *name*.

Primer anotacije iz podatkovne zbirke $WODD_{\text{MODD2}}$, ki vsebuje 3 ovire in horizont:

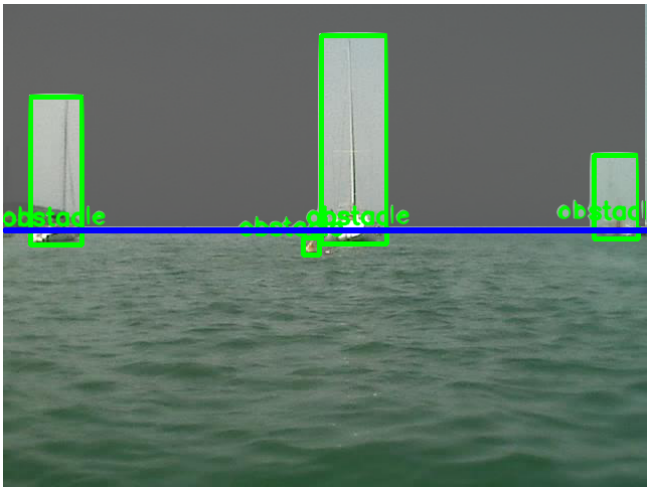
```
<annotation>
  <filename>00074552R</filename>
  <folder>modd2/kope67-00-00074432-00074612/frames</folder>
  <size>
    <height>958</height>
    <width>1278</width>
    <depth>3</depth>
  </size>
  <object>
    <bndbox>
      <xmax>1264</xmax>
      <xmin>29</xmin>
      <ymax>520</ymax>
      <ymin>544</ymin>
    </bndbox>
    <name>horizon</name>
  </object>
  <object>
    <bndbox>
      <xmax>93</xmax>
      <xmin>83</xmin>
      <ymax>561</ymax>
      <ymin>547</ymin>
    </bndbox>
```

```
    <name>obstacles</name>
  </object>
  <object>
    <bndbox>
      <xmax>1143</xmax>
      <xmin>1101</xmin>
      <ymax>563</ymax>
      <ymin>506</ymin>
    </bndbox>
    <name>obstacles</name>
  </object>
  <object>
    <bndbox>
      <xmax>1250</xmax>
      <xmin>1241</xmin>
      <ymax>534</ymax>
      <ymin>524</ymin>
    </bndbox>
    <name>obstacles</name>
  </object>
</annotation>
```

4.2.1 Ignoriranje negotovih območij za robustno učenje

Za pospešitev učenja in boljšo generalizacijo modela izkoriščamo prenos učenja (angl. *transfer learning*) konvolucijskih nevronskih mrež. Koncept prenosa učenja pomeni, da lahko uteži modela iste arhitekture, ki se je učil na drugačnih podakih, uporabimo kot osnovo za učenje [55, 56]. V praksi to pomeni hitrejšo konvergenco modela in s tem povezano hitrejše učenje, saj večino konvolucijskih slojev služi izluščevanju značilk iz slike, na podlagi katerih zadnji nivoji mreže opravljajo detekcijo in klasifikacijo. V vseh eksperimentih smo kot osnovo vzeli že prednaučeno mrežo.

Zaradi uporabe prenosa učenja, kjer je model videl veliko različnih objektov, je možno, da model zazna objekt nad gladino morja (na primer avtomobil na obali). Ker so objekti v sestavljeni podatkovni zbirki WODD anotirani zgolj v vodi, bi take detekcije kvarila postopek učenja mreže. Vsaka detekcija nad vodo bi veljala za napačno predikcijo, čeprav obstaja možnost, da je na sliki objekt res tam. Zaradi pomanjkanja anotacij nad gladino morja ni mogoče pravilno obravnavati takih predikcij. Načeloma bi ta problem lahko naslovili z ročnim anotiranjem vseh objektov na slikah. Poleg dejstva, da bi to bilo zelo zamudno, bi bilo tudi slabo pogojeno. Skupina objektov je na primer lahko s stališča ena same oznake (ovira) povsem sprejemljiva. Mreža torej detektira, da je tam ovira. Ali je sestavljena iz več manjših objektov ali enega večjega, ni tako zelo pomembno. S tega stališča bi bilo nesmiselno siliti mrežo, da detektira posamezne objekte, detekcijo celotne skupine skupaj pa obravnavati kot negativni primer. Problem rešimo z masko za nevtralizacijo napake, ki je prikazana na Sliki 4.2. Vse predikcije med učenjem, ki se zgodijo nad gladino morja in niso del anotiranega objekta, izničimo (pomnožimo z 0) in s tem preprečimo razširjanje nekonsistentnih informacij v mrežo v vzvratnem prehodu.



Slika 4.2: Maska, kjer se napake ne upoštevajo, označena s sivo; modra črta predstavlja obzorje; omejevalna polja ovir so v zeleni barvi.

Poglavje 5

Eksperimentalna evalvacija

V tem poglavju opišemo izvedene eksperimente in ovrednotimo rezultate. V Poglavju 5.1 opišemo parametre učenja in strojno opremo, ki je poganjala eksperimente. Nato v Poglavju 5.2 predstavimo kvantitativno analizo, kjer opišemo protokol analize, uporabljene mere delovanja in predstavimo ter analiziramo rezultate. V Poglavju 5.3 predstavimo kvalitativno analizo, kjer je prikazana grafična primerjava delovanja vseh testiranih mrež.

5.1 Podrobnosti implementacije

Učenje in evalvacija sta potekala na strežniku v laboratoriju Vicos FRI na grafični kartici NVIDIA GeForce Titan X (Pascal) z 12 GB spomina. Med treniranjem vseh arhitektur uporabljamo stopnjo učenja $1e-5$. Za potrebe paketne normalizacije uporabljamo paket, ki vsebuje 16 primerov zaradi omejitve velikosti pomnilnika na grafični kartici. Ostalih hiperparametrov referenčnih metod nismo spreminjali in so enaki kot v [28, 23].

Naše odprtokodne implementacije vseh mrež so na voljo: BlitzNet [57], medtem ko YOLO in YoloW uporabljata isti repozitorij, ki je dostopen na [53]. Vsa implementacija je bila opravljena s programskim jezikom Python v ogrodju TensorFlow [58].

5.2 Kvantitativna analiza

V našem delu smo z eksperimenti želeli odgovoriti na dve glavni vprašanji. Najprej nas je zanimala primerjava delovanja dveh različnih konvolucijskih nevronskih mrež za detekcijo: BlitzNet [28] in YOLO [23]. Nato smo želeli preveriti, kakšno je delovanje naše modificirane mreže: YoloW.

Delovanje modelov smo testirali na podatkovni zbirki WODD, ki je opisana v Poglavju 4.1. Primerjali smo delovanje treh mrež: referenčne prednaučene mreže BlitzNet [28] in YOLO [23] ter našo modificirano mrežo YoloW, opisano v Poglavju 4.2. Testiranje referenčno prednaučenih mrež nam omogoča oceniti izhodiščno delovanje in ovrednotiti delovanje mrež po treniranju na predlagani podatkovnih zbirki.

Referenčni metodi BlitzNet [28] in YOLO sta bili trenirani na podatkovnih zbirkah, ki vključujejo več kategorij, kot podatkovna zbirka WODD, predstavljena v Poglavju 4.2. Za učenje in testiranje na zadnjem nivoju omenjenih mrež združimo vse napovedane kategorijo v kategorijo, ki je vsebovana v podatkovni zbirki WODD: ovira. To nam omogoča učenje in primerjavo vseh predstavljenih mrež na enak način.

5.2.1 Protokol analize

Delovanje vseh mrež smo preverjali na podatkovni zbirki $WODD_{\text{MODD2}}$, medtem ko so ostali deli podatkovne zbirke bili uporabljeni v učni množici. Zaradi specifik anotacij podatkovne zbirke WODD (objekti, anotirani zgolj v vodi) smo definirali območje detekcij, v katerem ne upoštevamo napačnih predikcij. V podatkovni zbirki so objekti anotirani zgolj v vodi, zato med postopkom evalvacije izključimo predikcije, ki se sprožijo nad nivojem vode. Poleg tega premec ladje, ki je pogosto viden, ni anotiran kot objekt, zato z območja detekcij izvzamemo ozek pas tik pred plovilom. Primer maske območja detekcij je prikazan na Sliki 5.1.

Slika 5.1: Maska, ki označuje področje kjer se napake ne upoštevajo, označena s sivo.

5.2.2 Mere delovanja

Za določanje pravilnosti uporabljamo mero IoU (angl. *Intersection over Union)*, ki predstavlja razmerje med presekom in unije napovedanega omejevalnega polja $\boldsymbol{P}$ in resničnega omejevalnega polja $\boldsymbol{G}$

$$IoU(\boldsymbol{P}, \boldsymbol{G}) = \frac{\boldsymbol{P} \cap \boldsymbol{G}}{\boldsymbol{P} \cup \boldsymbol{G}}. \tag{5.1}$$

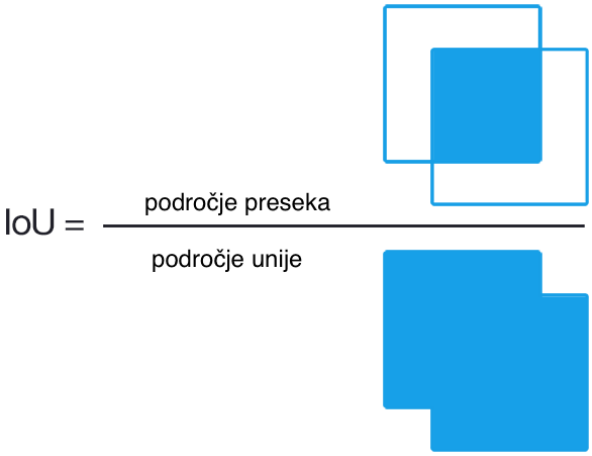
Grafični prikaz izračuna IoU je predstavljen na Sliki 5.2.

Za pozitivno predikcijo (angl. *True Positive*, TP) smo vedno upoštevali tako, ki ima z vsaj enim objektom istega razreda anotacije $IoU > 0,5$. Vsaka predikcija z $IoU < 0,5$ se je obravnavala kot napačna predikcija (angl. *False Positive*, FP). Vsak objekt, ki ga algoritem ni zaznal z nobeno predikcijo, obravnavamo kot zgrešen primer (angl. *False Negative*, FN). Natančnost (angl. *Precision*) je mera, ki nam pove, kolikšen delež predikcij je bil pravilen

$$Precision = \frac{TP}{TP + FP}. \tag{5.2}$$

Priklic (angl. *Recall*) je mera, ki nam pove, kolikšen delež objektov je algoritem detektiral

$$Recall = \frac{TP}{TP + FN}. \tag{5.3}$$



Slika 5.2: Grafični prikaz izračuna razmerja med področjem preseka in področjem unije dveh omejevalnih polj - mere *IoU*

Za končno vrednotenje delovanja algoritmov smo uporabili mero povprečne natančnosti (angl. *Average Precission*, AP), ki je bila prvič formalizirana v [59] in je uporabljena kot *de facto* mera za primerjavo algoritmov za detekcijo objektov. Ideja povprečne natančnosti je izračun površine področja pod grafom *natačnost-priklic*, kjer nadomestimo vrednosti priklica $p(r)$ z vrednostmi $p_{interp}(r) = \underset{\hat{r} \geq r}{max}\ p(\hat{r})$. Povprečna natančnost je izračunana kot povprečje maksimalnih natančnosti v teh enajstih točkah.

$$AP = \sum_{r \in \{0,0.1,0.2,..0.9,1.0\}} p_{interp}(\hat{r}) \tag{5.4}$$

5.2.3 Analiza delovanja

Vsako od mrež smo učili na več podmnožicah podatkovne zbirke WODD, ki smo jo razdelili na pet učnih podmnožic $\boldsymbol{W_{U1}}, \boldsymbol{W_{U1}}, \ldots, \boldsymbol{W_{U5}}$. Tabela 5.1 prikazuje sestavo učnih podmnožic in število vsebovanih učnih primerov.

	$WODD_{\text{SMD}}$	$WODD_{\text{MODD1}}$	$WODD_{\text{OBJ}}$	učni primeri
W_{U1}	✓			2396
W_{U2}		✓		4454
W_{U3}			✓	200
W_{U4}	✓	✓		6850
W_{U5}	✓	✓	✓	7050

Tabela 5.1: Prikaz učnih podmnožic uporabljenih v eksperimentih.

	Prednaučen	W_{U1}	W_{U2}	W_{U3}	W_{U4}	W_{U5}	FPS
Blitznet	79,49	81,91	84,29	85,30	87,07	89,68	16,54
YOLO	**82,47**	**85,34**	88,45	93,64	95,19	96,78	29,91
YoloW	/	85,31	**89,60**	**94,91**	**96,52**	**97,72**	**30,12**

Tabela 5.2: Primerjava povprečne natančnosti (AP) različnih mrež, izraženo v odstotkih, na testni množici $WODD_{\textbf{MODD2}}$ z različnimi učnimi množicami.

Namen eksperimentov je primerjava delovanja različnih mrež in preveriti, ali je natančnost korelirana s številom relevantnih učnih primerov. Tabela 5.2 prikazuje povprečno natančnost mrež na podatkovni zbirki testni množici podatkovne zbirke $WODD_{\textbf{MODD2}}$, kjer so bile mreže trenirane na različnih učnih podmnožicah.

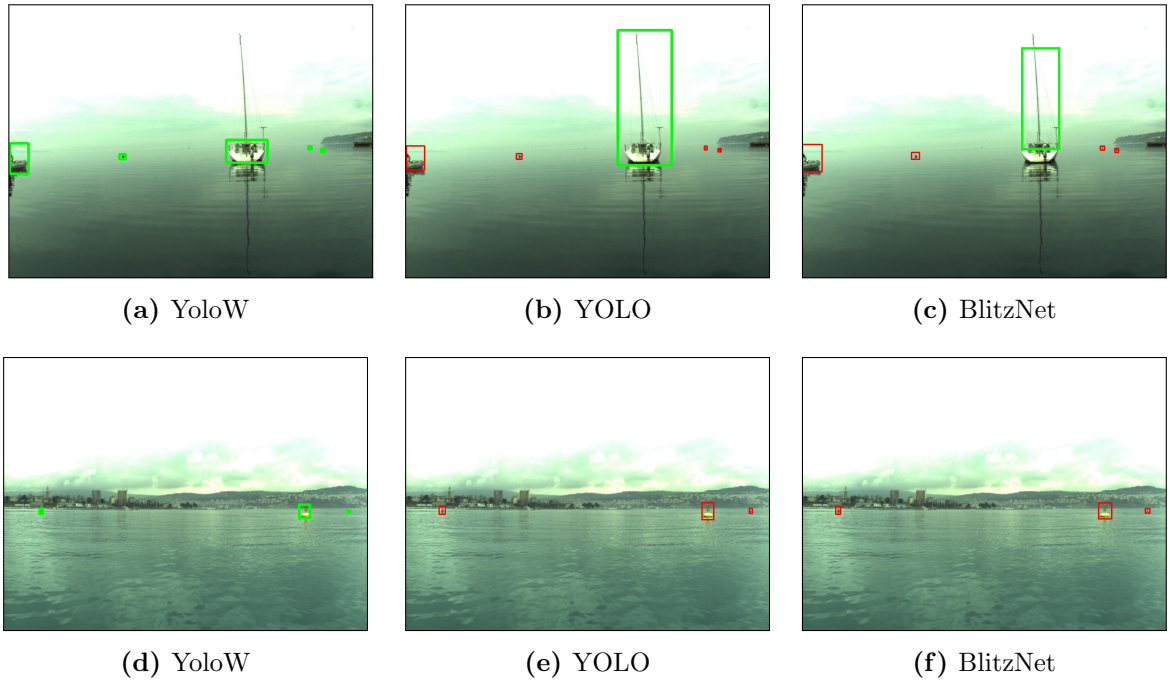
Primerjava prednaučenih mrež BlitzNet in Yolo (prvi stolpec Tabele 5.2) na testni množici $WODD_{\textbf{MODD2}}$ kaže na veliko boljšo natančnost mreže YOLO. Z dodajanjem relevantnih učnih primerov se natančnost obeh izboljšuje. Po učenju na celotni učni množici zbirke WODD doseže BlitzNet $89,54$ % natančnost, medtem ko YOLO doseže $96,78$ % natančnost. Primerjava referenčne mreže YOLO in naše mreže YoloW kaže na boljše delovanje referenčne metode pri manjšem številu učnih primerov. Sklepamo lahko, da se naša mreža ni učila dovolj časa oziroma je videla premalo relevantnih učnih primerov za boljše delovanje od referenčne mreže. Z učenjem na vedno večjih

učnih podmnožicah se natančnost naše mreže izboljšuje in preseže natančnost referenčne metode. Natančnost predlagane mreže YoloW, ki je bila učena na celotni učni množici je 97, 73 %. Razvidno je, da vsaka dodatna učna množica doprinese k natančnosti. Največji preskok je viden pri vključitvi podatkovne zbirke $WODD_{\mathbf{OBJ}}$, ki smo jo anotirali sami, saj so izvorne sekvence posnete z enakima kamerama kot v podatkovni zbirki $WODD_{\mathbf{MODD2}}$. Hkrati je v $WODD_{\mathbf{OBJ}}$ vključenih veliko raznolikih objektov.

Poleg natančnosti je za realnočasovne metode detekcije pomembna tudi hitrost delovanja. Hitrost delovanja merimo v številu obdelanih slik na sekundo (angl. *Frames Per Second*, FPS). Rezultati delovanja hitrosti so bili pridobljeni kot povprečna hitrost vsakega modela pri generiranju predikcij. Izmerjene hitrosti delovanja, predstavljene v Tabeli 5.2 (stolpec FPS) kažejo na veliko prednost arhitekture YOLO v primerjavi z BlitzNet. Primerjava med originalnim modelom YOLO in našo prilagojeno mrežo YoloW ne pokaže signifikatnih izboljšav v hitrosti delovanja kljub manjšemu številu potrebnih računskih operacij. To dejstvo je seveda razumljivo, saj se je število operacij zmanjšalo zgolj na zadnjem konvolucijskem nivoju, kar gledano globalno ne prinese prav veliko pohitritve. Hkrati je rezultat lahko odvisen tudi od predhodne temperature računalniškega sistema, na katerem so bili eksperimenti izvajani (ali se je sistem povsem ohladil od prejšnjega eksperimenta ipd.).

5.3 Kvalitativna analiza

V tem poglavju predstavimo primere delovanja obravnavanih konvolucijskih nevronskih mrež. Primerjava pokaže razlike v delovanju. V vsaki vrstici je prikazana ista slika in predikcije, ki so jih generirale mreže za dano sliko. Najprej vedno prikažemo delovanje naše mreže YoloW. Sledi referenčna mreža YOLO. Nazadnje predstavimo še delovanje referenčne metode BlitzNet. Vse mreže so bile trenirane na celotni učni množici podatkovne zbirke WODD. Na vseh prikazanih primerih delovanja so predikcije mrež označene z zeleno, ovire, ki jih mreže niso detektirale pa z rdečo barvo.

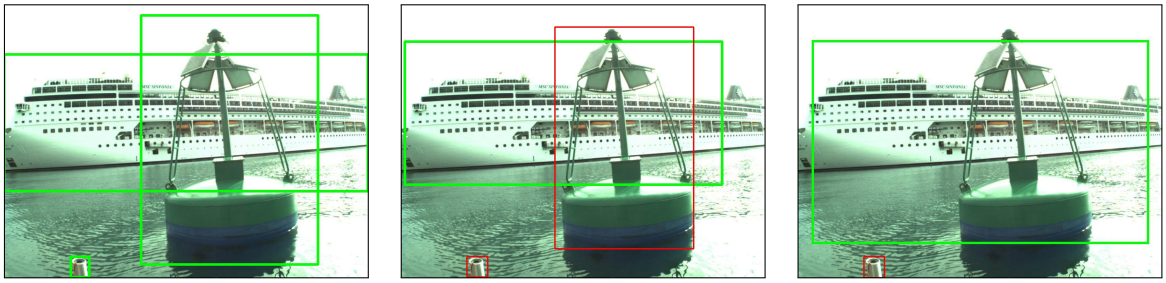


(a) YoloW **(b)** YOLO **(c)** BlitzNet

(d) YoloW **(e)** YOLO **(f)** BlitzNet

Slika 5.3: Kvalitativna analiza: primera 1 in 2.

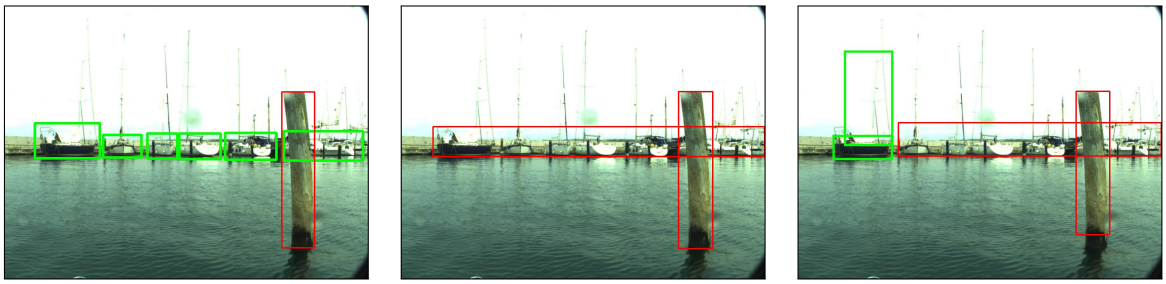
Prvi primer delovanja (Slike 5.3a, 5.3b, 5.3c) prikazuje izboljšano delovanje naše mreže YoloW za manjše objekte. Poleg boj v daljavi obe referenčni mreži ne uspeta detektirati čolna, ki je na levi strani. Podobno je vidno na drugem primeru (Slike 5.3d, 5.3e, 5.3f) , kjer YoloW uspešno zazna večjo rumeno bojo in dva manjši, medtem ko referenčni metodi za dano sliko ne detektirata nobenega objekta.



(a) YoloW **(b)** YOLO **(c)** BlitzNet

Slika 5.4: Kvalitativna analiza: primer 3.

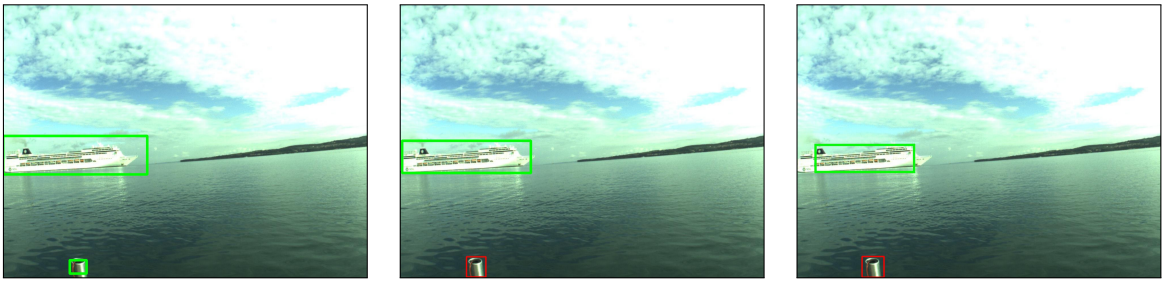
Tretji primer (Slike 5.4a, 5.4b, 5.4c) prikazuje primerjavo delovanja na večjih objektih. YoloW poleg premca detektira tako večjo ladjo v ozadju kot oviro v ospredju. YOLO zazna zgolj ladjo v ozadju, BlitzNet pa zazna obe oviri, a ju združi v eno detekcijo.



(a) YoloW **(b)** YOLO **(c)** BlitzNet

Slika 5.5: Kvalitativna analiza: primer 4.

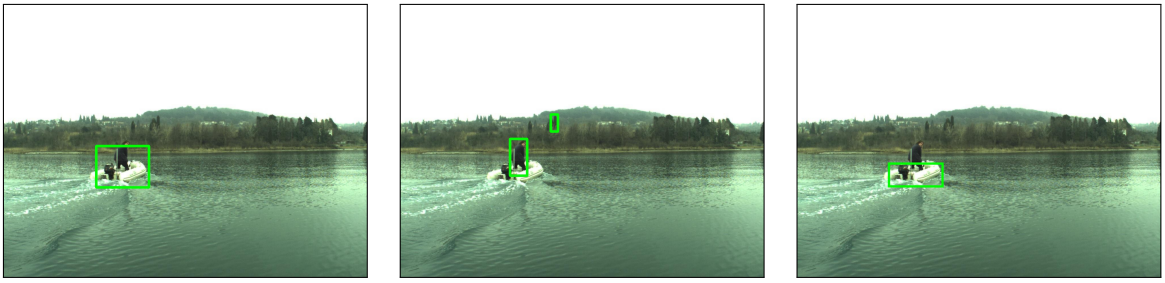
V četrtem primeru (Slike 5.5a, 5.5b, 5.5c) nobena mreža ne zazna lesenega droga v ospredju. Med mrežami so opazne velike razlike glede detektiranih ladij v ozadju. Primer je verjetno težak zaradi neobičajnega zornega kota, iz katerega so ladje vidne (viden le zadek). Poleg tega so postavljene pred pomol, kar še oteži detekcijo, saj ladje niso jasno ločene od vode kot običajno.



(a) YoloW **(b)** YOLO **(c)** BlitzNet

Slika 5.6: Kvalitativna analiza: primer 5.

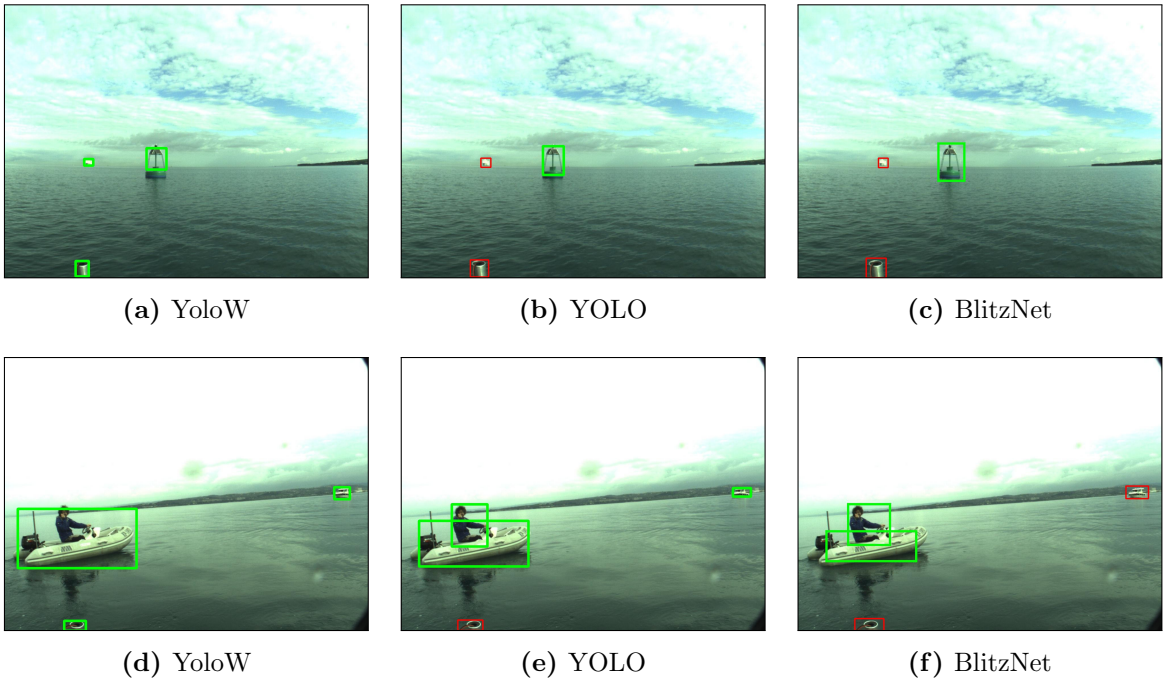
Primer pet (Slike 5.6a, 5.6b, 5.6c) prikazuje dobro delovanje vseh mrež na večjem objektu, ki se pojavi v daljavi.



(a) YoloW **(b)** YOLO **(c)** BlitzNet

Slika 5.7: Kvalitativna analiza: primer 6.

Podobno primer šest (Slike 5.7a, 5.7b, 5.7c) prikazuje zadovoljivo delovanje vseh mrež na objektu srednje velikosti, ko je na sredini slike. Pri YOLO je opazna še detekcija vrha enega od dreves, ki pa na izmerjeno natančnost zaradi postopka evalvacije opisanega v Poglavju 4.2.1 ne vpliva.



(a) YoloW **(b)** YOLO **(c)** BlitzNet

(d) YoloW **(e)** YOLO **(f)** BlitzNet

Slika 5.8: Kvalitativna analiza: primera 7 in 8.

Primer sedem (Slike 5.8a, 5.8b, 5.8c) vsebuje kombinacijo srednje velikega objekta na sredini slike in oddaljeno ladjo v ozadju. Naša mreža YoloW uspešno detektira vse objekte, medtem ko referenčni zaznata le objekt v osredju. Zadnji primer (Slike 5.8d, 5.8e, 5.8f) prikazuje podobno delovanje. Naša mreža YoloW zazna čoln v ospredju in ladjo v ozadju. Referenčni metodi obe detektirata čoln in osebo na njem. YOLO zazna tudi ladjo v ozadju, medtem ko je BlitzNet ne zazna.

Poglavje 6

Sklepne ugotovitve

V delu smo naslovili problem detekcije in klasifikacije v vodnem okolju. Problem smo reševali s konvolucijskimi nevronskimi mrežami, ki izvajajo detekcijo in klasifikacijo v realnem času.

V delu smo predstavili študijo in primerjavo dveh konvolucijskih nevronskih mrež za problema detekcije in klasifikacije objektov. Predlagali smo modifikacijo arhitekture YOLO, ki je prilagojena za detekcijo v vodnem okolju (YoloW). Opisali smo postopke učenja ter evalvacije v podatkovnih zbirkah z negotovimi anotacijami. Naša konvolucijska nevronska mreža YoloW deluje z visoko natančnostjo in hitrostjo.

Predlagamo novo podatkovno zbirko za problem detekcije v vodnem okolju (WODD), ki je sestavljena iz več že objavljenih podatkovnih zbirk in novo anotiranih sekvenc ter slik iz interneta. Vse zgoraj omenjene arhitekturo smo učili na različnih učnih podmnožicah predlagane podatkovne zbirke. Rezultati eksperimentov so pokazali, da prednaučen model YOLO na testni podatkovni zbirki WODD deluje bolje od BlitzNeta. Z dodatnim učenjem obeh mrež z vedno večjim številom učnih podmnožic podatkovne zbirke WODD smo pokazali izboljšave natančnosti obeh mrež.

Rezultati primerjave so bili glavni razlog, da smo za razvoj naše konvolucijske nevronske mreže kot osnovo vzeli YOLO. Arhitekturo smo modificirali za potrebe detekcije v vodnem okolju (število razredov), s čimer smo nepo-

sredno zmanjšali število operacij v konvolucijskih nivojih mreže. Čas učenja in čas generiranja predikcij sta se marginalno izboljšala, a večjih optimizacij glede hitrosti delovanja prilagoditev arhitekture mreže ni prinesla.

6.1 Nadaljnje delo

Nadaljnje raziskovanje bi lahko bilo usmerjeno v dodatne modifikacije arhitekture YOLO za še boljše in hitrejše delovanje. Problem arhitekture YOLO so predvsem majhni objekti, kar bi se dalo nasloviti s spremembami na konvolucijskih nivojih mreže. Tudi hitrost delovanja bi bilo mogoče optimizirati. Enostavnejša rešitev je uporaba preprostejše arhitekture, kar pomeni manjše število konvolucijskih nivojev, ki so glavni razlog za razmeroma počasno delovanje, saj se večino računskega časa porabi prav s konvolucijo. Manjša mreža bi tudi omilila zahteve za strojno opremo, potrebno za izvajanje detekcije v realnem času. Druga alternativa je uporaba predprocesiranja s segmentacijskimi maskami, ki bi pozornost mreže usmerjali v zanimive regije slike. S tem bi število predikcij, ki jih mora opraviti YOLO, znižali in potencialno dosegli hitrejše delovanje mreže brez kakršnekoli izgube natančnosti. Problem detekcije in klasifikacije objektov, predvsem v vodnem okolju, je zdaleč od rešenega problema in zagotovo bodo raziskave vodile v nove predstavljene arhitekture, ki bodo sposobne delovanja v realnem času. V prihodnosti bo tako mogoče testirati nove arhitekture, ki bodo imele višjo izhodiščno natančnost od predstavljenih. Te bi bilo vredno preučiti in jih potencialno modificirati za namene detekcije in klasifikacije objektov.

Literatura

[1] J. Walker, The self-driving car timeline – predictions from the top 11 global automakers (2018).
URL https://www.techemergence.com/self-driving-car-timeline-themselves-top-11-automakers/

[2] Autotech, 46 corporations working on autonomous vehicles (2018).
URL https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/

[3] T. Taylor, Top 8 self-driving startups to watch in 2018 and 2019 (2018).
URL http://techgenix.com/self-driving-startups

[4] L. Dixon, Self driving car startups to watch (2018).
URL https://angel.co/job-collections/top-self-driving-cars-startups

[5] L. Fridman, Mit 6.s094: Deep learning for self-driving cars (2018).
URL https://www.ceros.com/originals/recaptcha-waymo-future-of-self-driving-cars/

[6] Google, recaptcha: Easy on humans, hard on bots (2018).
URL https://www.google.com/recaptcha/intro/v3beta.html

[7] M. HEALY, Is recaptcha training robocars? (2017).
URL https://selfdrivingcars.mit.edu

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316.

[9] B. Mathur, H. Wang, R. Haendel, Vehicle lane position detection system, uS Patent 5,351,044 (Sep. 27 1994).
URL https://www.google.com/patents/US5351044

[10] T. Nishio, Vehicle crash predictive and evasive operation system by neural networks, uS Patent 5,541,590 (Jul. 30 1996).
URL https://www.google.com/patents/US5541590

[11] M. Caccia, M. Bibuli, R. Bono, G. Bruzzone, Basic navigation, guidance and control ofÂ anÂ unmanned surface vehicle, Autonomous Robots 25 (4) (2008) 349–365. doi:10.1007/s10514-008-9100-0.
URL https://doi.org/10.1007/s10514-008-9100-0

[12] M. Dunbabin, A. Grinham, J. Udy, An autonomous surface vehicle for water quality monitoring, in: Australasian Conference on Robotics and Automation (ACRA), 2009, pp. 2–4.

[13] M. Grasmueck, G. P. Eberli, D. A. Viggiano, T. Correa, G. Rathwell, J. Luo, Autonomous underwater vehicle (auv) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the straits of florida, Geophysical Research Letters 33 (23).

[14] B. Bingham, B. Foley, H. Singh, R. Camilli, K. Delaporta, R. Eustice, A. Mallios, D. Mindell, C. Roman, D. Sakellariou, Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle, Journal of Field Robotics 27 (6) (2010) 702–717.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.

[16] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[18] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Computer Vision and Pattern Recognition, 2014.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[20] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, 2015, pp. 91–99.

[22] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[23] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, arXiv preprint arXiv:1612.08242.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: European conference on computer vision, Springer, 2016, pp. 21–37.

[25] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[26] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.

[27] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, arXiv preprint arXiv:1511.00561.

[28] N. Dvornik, K. Shmelkov, J. Mairal, C. Schmid, BlitzNet: A real-time deep network for scene understanding, in: IEEE International Conference on Computer Vision (ICCV), 2017.

[29] M. Kristan, V. S. Kenk, S. Kovačič, J. Perš, Fast image-based obstacle detection from unmanned surface vehicles, IEEE transactions on cybernetics 46 (3) (2016) 641–654.

[30] M. Kristan, V. Sulic, S. Kovacic, J. Perš, Fast image-based obstacle detection from unmanned surface vehicles (2015).
URL `http://arxiv.org/pdf/1503.01918.pdf`

[31] B. Bovcon, R. Mandeljc, J. Pers, M. Kristan, Stereo obstacle detection for unmanned surface vehicles by imu-assisted semantic segmentation, Robotics and Autonomous Systems 104 (2018) 1–13.

[32] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: European conference on computer vision, Springer, 2014, pp. 740–755.

[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[35] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436.

[36] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, Y. LeCun, The loss surfaces of multilayer networks, in: Artificial Intelligence and Statistics, 2015, pp. 192–204.

[37] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.

[38] J. Y. Yam, T. W. Chow, A weight initialization method for improving training speed in feedforward neural network, Neurocomputing 30 (1-4) (2000) 219–232.

[39] L. Waghmare, N. N. Bidwai, P. Bhogle, Neural network weight initialization, in: Mechatronics and Automation, 2007. ICMA 2007. International Conference on, IEEE, 2007, pp. 679–681.

[40] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, in: Neural Networks, 1990., 1990 IJCNN International Joint Conference on, IEEE, 1990, pp. 21–26.

[41] A. Karpathy, Cs231n convolutional neural networks for visual recognition (2017).
URL `http://cs231n.github.io`

[42] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, Vol. 30, 2013, p. 3.

[43] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint arXiv:1511.07289.

[44] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 971–980.

[45] D. Scherer, A. Müller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in: Artificial Neural Networks–ICANN 2010, Springer, 2010, pp. 92–101.

[46] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167. arXiv:1502.03167.
URL http://arxiv.org/abs/1502.03167

[47] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[48] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, arXiv preprint.

[49] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[50] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: ECCV, 2016.

[51] A. Neubeck, L. Van Gool, Efficient non-maximum suppression, in: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, Vol. 3, IEEE, 2006, pp. 850–855.

[52] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CoRR abs/1512.03385. `arXiv:1512.03385`.
URL `http://arxiv.org/abs/1512.03385`

[53] N. Ambrožič, Darkflow (2018).
URL `https://github.com/nejcambrozic/darkflow`

[54] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, C. Quek, Video processing from electro-optical sensors for object detection and tracking in a maritime environment: a survey, IEEE Transactions on Intelligent Transportation Systems 18 (8) (2017) 1993–2016.

[55] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in neural information processing systems, 2014, pp. 3320–3328.

[56] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1717–1724.

[57] N. Ambrožič, Blitznet (2018).
URL `https://github.com/nejcambrozic/blitznet`

[58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale ma-

chine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL https://www.tensorflow.org/

[59] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International journal of computer vision 88 (2) (2010) 303–338.