SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

POSLIJEDIPLOMSKI DOKTORSKI STUDIJ ELEKTROTEHNIKE I INFORMACIJSKE TEHNOLOGIJE

KVALIFIKACIJSKI ISPIT

DETEKCIJA LJUDI NA ZRAČNIM SLIKAMA UPOTREBOM KONVOLUCIJSKIH NEURONSKIH MREŽA

Mirela Kundid Vasić

Split, listopad 2017.

SADRŽAJ

1. Uvod 1
2. Primjena konvencionalnih metoda u detekciji ljudi na zračnim slikama 3
2.1. Pristupi detekcije uz pomoć konvencionalnih metoda strojnog učenja 3
2.1.1. SIFT (Scale-Invariant Feature Transform) 4
2.1.2. SURF (Speeded-Up Robust Features) 4
2.1.3. BRIEF (Binary Robust Independent Elementary Features) 5
2.1.4. HOG (Histogram of Oriented Gradients) 5
2.1.5. SVM (Support Vector Machines) 6
2.1.6. Logistička regresija (Logistic regression) 7
2.1.7. Algoritam nasumične šume (engl. Random forests algorithm) 8
2.1.8. Problemi primjene konvencionalnih metoda strojnog učenja 8
3. Neuronske mreže 10
3.1. Perceptron 11
3.2. Sigmoid neuron 12
3.3. Propagacija unazad 16
4. Konvolucijske neuronske mreže 18
4.1. Arhitekture konvolucijskih neuronskih mreža 20
4.1.1. AlexNet 21
4.1.2. ZF Net 23
4.1.3. VGG Net 24
4.1.4. GoogLeNet 25
4.1.5. Microsoft ResNet 26
4.1.6. R-CNN 26
4.1.7. You Only Look Once 27

4.1.8. DetectNet 28
4.2. Primjene metoda dubokih neuronskih mreža 30
4.3. Preporuke za korištenje učenja prenošenjem 32
4.4. Primjena DetectNet arhitekture u detekciji na zračnim slikama 35
4.4.1. Format podataka DetectNet arhitekture 35
4.5. Primjene konvolucijskih neuronskih mreža u detekciji ljudi na zračnim slikama 38
5. Zaključak 42
6. Literatura 44
PRILOG – TABLICA SLIKA 48
PRILOG – ARHITEKTURA KONVOLUCIJSKE NEURONSKE MREŽE 49

1. UVOD

Prepoznavanje ljudi na slikama iz zraka postao je važan dio razvoja sustava za spašavanje i nadzor pomoću samostalnih bespilotnih letjelica. Ovaj pristup sve više dolazi do izražaja u raznim civilnim i vojnim aplikacijama zahvaljujući mogućnosti obrade i pregleda velikih terena. Misija potrage i spašavanja unesrećenih ljudi na nepristupačnim područjima može se poboljšati upotrebom samostalnih bespilotnih letjelica koje mogu brzo reagirati i trenutno prikupiti podatke, a omogućavaju i pretragu većih područja jednom snimkom. Ključni izazov je robustan sustav koji može detektirati statičnu ili dinamičnu osobu na slici koja se može nalaziti u različitim sredinama, od izoliranih područja na selima, do kaotičnih gradskih područja. Sustav također mora biti neovisan o vremenskim i atmosferskim prilikama, kao i o samom lokalitetu.

Klasične metode za detekciju ljudi razvijene su za rad sa slikama snimljenim s tla zemlje, gdje se kamera uobičajeno nalazi u paralelnoj ravnini s tlom i gdje traženi objekt zauzima veliki dio piksela slike i vidljiv je vizualni izgled ljudskog tijela, što samim time olakšava detekciju. Slike snimljene iz zraka su jako velike (snimaju veliki dio područja), a traženi objekt (čovjek) na njima se prostire na malom području, tj. zauzima jako malo piksela slike i nema vidljive dijelove tijela, što uvelike otežava razlikovanje objekta od pozadine i čini klasične metode detekcije ljudi neprimjenjivim u ovom slučaju. Dodatni problem je to što je kamera montirana na pokretnu zračnu platformu pa se pomicanjem platforme veličina i vidljiva razlikovna obilježja osobe mogu još više smanjiti. Također, time se uvodi veliki broj mogućih orijentacija u kojima se osoba može pojaviti na slici. Zbog tih nedostataka, konvencionalne metode primijenjene za detekciju ljudi u zračnim slikama rezultiraju s puno lažnih detekcija i prilično su spore.

U zadnjih nekoliko godina, primjena neuronskih mreža i dubokog učenja (*engl. Deep Learning*) u području detekcije na slikama rezultira pojavljivanjem metoda koje se kontinuirano razvijaju i postaju sve robusnije i preciznije. Budući da se neuronske mreže često kosrite u slučajevima gdje nisu poznata pravila prema kojima se može dovesti u vezu ulazni i izlazni skup podataka iz željenog sustava, u ovom radu razmotrit će se mogućnost primjene konvolucijskih neuronskih mreža (*engl. Convolutional Neural Networks - CNN*) u području detekcije ljudi na slikama snimljenim iz zraka, uz pomoć bespilotnih letjelica.

Struktura rada je postavljena na sljedeći način: u drugom poglavlju razmatra se problematika detekcije ljudi na zračnim slikama dobivenih uz pomoć bespilotnih letjelica (UAV). U tom poglavlju opisani su konvencionalni pristupi koji se temelje na metodama strojnog učenja. U trećem poglavlju ukratko je postavljen formalizam i temelji neuronskih mreža. Četvrto poglavlje daje prikaz konvolucijskih neuronskih mreža i njihovih primjena, postojećih arhitektura i rezultata primjene. Također je prikazana upotreba DetectNet programskog okvira za potrebe detekcije ljudi u operacijama potrage i spašavanja ljudi (*engl. Search and Rescue - SAR*).

2. PRIMJENA KONVENCIONALNIH METODA U DETEKCIJI LJUDI NA ZRAČNIM SLIKAMA

Zadnjih godina bespilotne zračne letjelice (*engl. Unmanned areal vehicels - UAVs*) se široko primjenjuju u nizu područja djelovanja, prvenstveno zbog nekoliko prednosti kao što je mobilnost, jednostavnost implementacije, stabilnost, mala težina te ugrađena mogućnost obrađivanja slika dobivenih s kamere. U nizu mogućih primjena koriste se u video nadzoru, za smanjenje rizika i povećanje sigurnosti na radnom mjestu, pregledima vjetroturbina, dalekovoda, cjevovovoda, željeznica, tornjeva i sl., u poljoprivrednim okruženjima kao što je detekcija bolesti u usjevima ili obrana od šumskih požara, za mjerenje zračenja te brojnim drugim primjenama. Neke od primjena UAV letjelica su i detekcija i praćenje objekata na termalnim slikama oceana [1], nadziranje prirodnih nepogoda u cilju smanjenja prirodne ili čovjekom uzrokovane katastrofe, kao i za detekcija zemljotresom uzrokovane štete na kućama [2]. Jedna od primjena bespilotnih zračnih letjelica je u operacijama potrage i spašavanja. Ključni faktor koji utječe na mogućnost preživljavanja žrtve je činjenica koliko brzo se ta osoba može pronaći. U ovom slučaju moguće je primijeniti metode detekcije ljudi na zračnim slikama, dobivenih primjenom bespilotnih letjelica. „Ručna“ pretraga velikog broja zračnih slika je jako zahtjevan proces pa je nužna automatizacija ovog procesa započela već 1970ih godina. Iako je postignut značajan uspjeh u zadnjih 40 godina, postoji samo nekoliko polu-automatskih sustava za određenu upotrebu i samo nekoliko pristupa za detekciju ljudi na zračnim slikama [3]. Uz nedavnu pojavu slika koje imaju rezoluciju čak i do 100px po metru kvadratnom, slikane uz pomoć ručno upravljanih bespilotnih letjelica, javlja se i niz metoda koje omogućavaju automatsku obradu takvih slika. Upravo ovakve bespilotne letjelice bi se mogle iskoristiti u svrhu lakšeg pronalaženja žrtava na područjima niskog raslinja u području Sredozemlja, kao što je južna Hercegovina i Dalmacija. U potpunosti automatiziran sustav za klasifikaciju terena je kompleksan problem koji konstantno podiže granice strojnog učenja i računalnog vida.

2.1. Pristupi detekcije uz pomoć konvencionalnih metoda strojnog učenja

Detekcija ljudi na zračnim slikama predstavlja jedan od zahtjevnijih zadataka strojnog učenja danas. Porastom mogućnosti i dostupnosti bespilotnih letjelica (UAV) javile su se i mogućnosti primjene istih u raznim područjima. Posebno su se pokazale korisnim u poljoprivredi te video nadziranju, no povećanjem dostupnosti i pojavom kamera visoke rezolucije javljaju se i pristupi

temeljeni na konvencionalnim načinima detekcije, prepoznavanja i lokalizacije objekata na slikama. Prethodnih par desetljeća deskriptori značajki su bili primarna tehnika za rješavanje niza problema, a isti su često kombinirani s tradicionalnim algoritmima nadziranog strojnog učenja kao što su SVM metoda *(engl. Support Vector Machines*) i KNN (*engl. K Nearest Neighbour*s) metoda. Općenito, ovaj proces može se opisati kao pronalazak značajki koje pomažu pri klasifikaciji objekata. Prvi korak se sastoji od pronalaska i izvlačenja vektora značajki koje smanjuju dimenzionalnost slike te na kojima se primjenjuju navedeni algoritmi strojnog učenja.

Proces pronalaska i izvlačenja značajki koje najbolje rješavaju određeni problem sastoji se od ručnog definiranja značajki kao što su rubovi, krajevi, boja i druge značajke, te se može reći da je ovakav pristup više vođen uz pomoć čovjeka (*engl. human driven*). Općenito, značajke se mogu podijeliti na lokalne i globalne značajke. Neke od lokalnih značajki su: SIFT (Scale-Invariant Feature Transform) [4], SURF (Speeded-Up Robust Features) [5], BRIEF (Binary Robust Independent Elementary Features) [6], HOG (Histogram of Oriented Gradients) [7] i druge.

2.1.1. SIFT (Scale-Invariant Feature Transform)

SIFT algoritam predstavljen je već 1999. godine [4], a suočava se s problemom skaliranja određenih značajki. Promatranjem određenog ruba ili kraja nekog objekta na slici, lako je primijetiti da on može izgledati potpuno drugačije ako se slika rotira ili uvećava nekoliko puta. SIFT desktiptor je invarijantan na skaliranje te se pokazao jako koristan za uparivanje slika za detekciju objekata u realnim uvjetima. SIFT deskriptor sastoji se od metode za detekciju područja od interesa (*engl. Points Of Interest POI*). U orginalnoj formulaciji, SIFT se koristio za detekciju POI iz sivih slika gdje su se koristile statistike o lokalnim usmjerenjima intenziteta slike. Osnovni smisao je bio uparivanje odgovarajućih POI na različitim slikama. U svrhu detekcije na UAV slikama, SIFT značajke koriste se na zadatcima kao što su detekcija automobila [8] i detekcija poljoprivrednih plodova [9]. Detekcija se realizira korištenjem tri različite vrste grupiranja (klasteriranja) s prostornom segmentacijom i raznim morfološkim operacijama na slikama.

2.1.2. SURF (Speeded-Up Robust Features)

Problem na koji se nailazi upotrebom SIFT algoritma je činjenica da je dosta spor, koristi razliku Gauss-ijana za normalizaciju skale. 2006 predstavljena je ubrzana verzija SIFT algoritma - SURF [5]. U svrhu postizanja invarijantnosti na rotaciju, SURF algoritam najprije računa Haarove valiće u x i y smjeru, a nakon toga se radi brza aproksimacija razlika Gauss-ijana koristeći zamućivanje,

što ubrzava proces detekcije značajki. SURF značajke se u UAV slikama često primjenjuju u spajanju više slika u jednu veliku sliku (*engl. stitching*) [10].

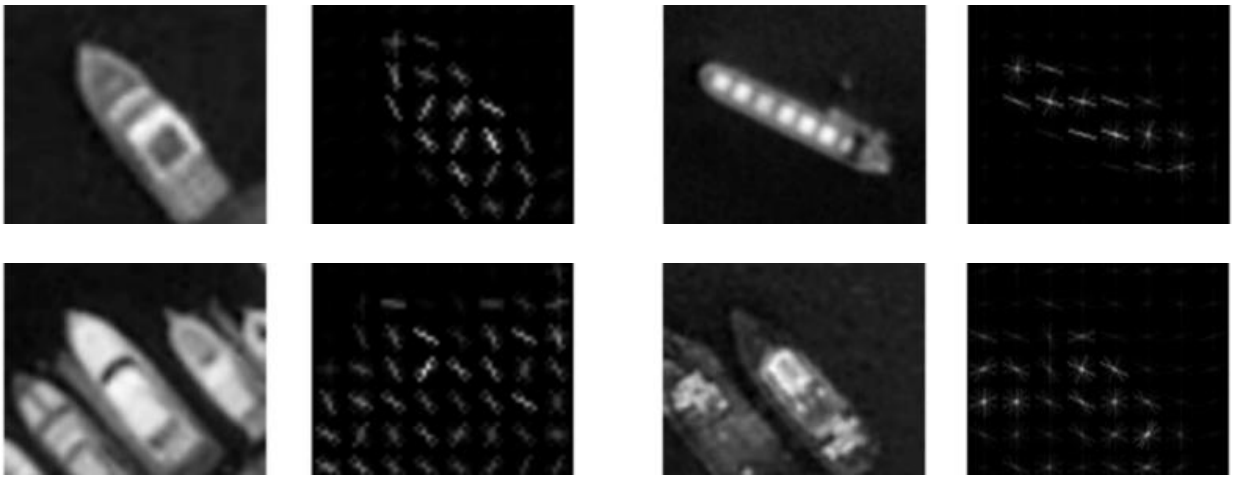
2.1.3. BRIEF (Binary Robust Independent Elementary Features)

U praksi SIFT koristi 128-dimenzionalni vektor za deskriptore značajki, dok SURF koristi 64, a tradicionalno ovi vektori se smanjuju koristeći metode za smanjivanje dimenzionalnosti kao što su PCA (*engl. Principal Component Analisys*) [11] ili ICA (*engl. Independent Component Analysis*) [12]. BRIEF izbjegava izračunavanje značajki deskriptora i umjesto toga koristi proceduru za odabir *n* blokova piksela te izračunava metriku intenziteta piksela. Ovih *n* blokova piksela koriste se za predstavljanje slike, što se u praksi pokazalo brže nego izračun deskriptora značajki te pronalazak onih koje su reprezentativne.

2.1.4. HOG (Histogram of Oriented Gradients)

Histogram of Oriented Gradients (HOG) je jedna od osnovnih metoda za detekciju i lokalizaciju objekata, čiji se uspjeh može mjeriti s uspjehom dubokih neuronskih mreža. Za UAV slike gdje su objekti koji nas zanimaju jako mali, možda je jedini način detekcije i raspoznavanja takvih objekata upravo uz pomoć kreiranja modela koji će naučiti prepoznati siluetu određenog objekta. HOG značajke „hvataju“ takve informacije. HOG značajke se mogu dobiti uz pomoć običnog CPU-a, te su uz to jednostavnije i brže. Informacije o gradijentu se računaju u histogramu orijentacija koji služi kao vektor koji se koristi u algoritmima strojnog učenja kao što je Logistička regresija (*engl. Logistic Regresion*), Slučajna šuma (*engl. Random Forest*) (predstavlja mnoštvo stabala odlučivanja) i SVM. Na slici 2.1 prikazan je par satelitskih snimki broda koje jasno iskazuju orijentacije kontura broda.

Na HOG značajkama primjenjuje se PCA analiza kako bi se pronašao reprezentativan skup značajki te smanjila dimenzionalnost, što uvelike smanjiva veličinu podataka za treniranje te povećava brzinu treniranja i testiranja, a s druge strane smanjuje preciznost za manje od 1%, što je neznatna razlika.



Slika 2.1 Prikaz zračne slike broda i značajki izvučenih u obliku HOG deskriptora

Nekim od ranih pristupa strojnom učenju smatra se pridruživanje oznake klase određenom pikselu na osnovu vektora značajki, gdje su značajke tipično vrijednosti na različitim spektralnim frekvencijama na mjestu određenog piksela. Automatska klasifikacija uz pomoć konvencionalnih metoda strojnog učenja je moguća i daje obećavajuće rezultate za ovakav problem, ali je zahtjevan zadatak i postoji niz različitih problema vezanih za implementaciju. Konvencionalne metode ekstrakcije značajki i metoda analize slike izvode se uz pomoć ručno odabranih značajki. Takve značajke su Histogram of Oriented Gradient (HOG)[13] koje su se uspješno pokazale u detekciji vozila na zračnim slikama [14], Scale Invariant Feature Transform (SIFT) [15] [16], kao i brojne druge značajke korištene u sličnim problemima detekcije, svaka sa svojim prednostima i nedostatcima. Nedavne su primjene strojnog učenja na slikama visoke rezolucije urodile velikim brojem pristupa jako velike preciznosti [17] [18]. Sve ove značajke se klasificiraju uz pomoć određenog klasifikatora kao što je SVM, Logistička regresija itd.

2.1.5. SVM (Support Vector Machines)

SVM (*engl. Support Vector Machines*)[19] spada u skup nadziranih metoda učenja (nadgledano učenje) koje se koriste za klasifikaciju i regresiju. Nakon početnog osposobljavanja generira se SVM model, nakon čega slijedi klasifikacija. Budući da je SVM binarni klasifikator, SVM klasifikacija radi na temelju podjele svih točaka u prostoru u dvije kategorije prema margini između vektora podrške. Algoritam traži najveću marginu između dvije kategorije, a ovaj postupak se naziva linearna klasifikacija.

Međutim, moguće je izvršiti klasifikaciju u nekoliko kategorija, što se postiže izvršavanjem tzv. trika jezgre (*engl. Kernel trick*) koji implicitno mapira ulaze u višedimenzionalnom prostoru. Trik izbjegava eksplicitno mapiranje, što je neophodno kako bi se dobio linearni algoritam učenja koji se obučava s nelinearnom funkcijom. SVM stvara granične odluke odvojene granice koja mora biti maksimalna, a na ulaznom skupu podataka možemo dobiti linearnu raspodjelu dvaju razreda. Računanje SVM klasifikatora iznosi minimiziranje izraza označenog formulom 2.1

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(w*x_i+b)\right)\right]+\Lambda\left|\left|w\right|\right|^2 \qquad (2.1)$$

U formuli 2.1. y_i su oznake za svaku značajku x_i, w je normalni vektor koji odvaja podatke u dvije ravnine, a b je margina između hiper ravnine i klasificiranih podataka. Parametar Λ označava razmjenu između povećanja veličine margina i osigurava da je x_i unutar prave ravnine. SVM je jedan od najčešće korištenih klasifikatora u području zračnih slika, a u kombinaciji s HOG značajkama u većini slučajeva pokazuje jako dobru preciznost. Npr. kod brojanja skupljača školjki na primorskom terenu [20] pokazao je preciznost od 80%, što ga čini boljim od drugih klasifikatora kao što je algoritam slučajne šume. SVM se pokazao kao jedan od osnovnih pristupa u detekciji objekata na zračnim slikama gdje je primijenjen u kombinaciji s HOG značajkama u detekciji ljudi [21], kao i u detekciji drugih objekata na zračnim slikama [22].

2.1.6. Logistička regresija (Logistic regression)

Logistička regresija (*engl. Logistic regression - LR*) [23] je regresijski model gdje je zavisna varijabla kategorična. Binarni logistički model koristi se za procjenu vjerojatnosti binarnog odgovora na temelju jedne ili više varijabli prediktora (značajke). LR mjeri odnos između kategoričke zavisne varijable i jedne ili više neovisnih varijabli pomoću logističke funkcije, a može se može promatrati kao poseban slučaj generaliziranog linearnog modela. Ovaj model se koristi u različitim poljima jer je vrlo lagana i daje dobre rezultate. Uvjetna distribucija može biti Bernoulli ili Gaussova, jer ishod događaja može biti binaran (varijabla zavisna može imati samo dvije vrijednosti). Matematički, LR je zadatak procjenjivanja logaritamskih vjerojatnosti određenog događaja, a procjenjuje višestruke linearne regresijske funkcije kako je prikazano u formuli 2.2.

$$y=\beta_0+\beta_1*x_{i2}+\beta_2*x_{i2}+\cdots+\beta_p*x_m \qquad (2.2)$$

U formuli 2.2 $\beta_0, \beta_1, \ldots, \beta_p$ su regresijski koeficijenti koji ukazuju na relativni učinak varijable *x*, koja predstavlja ulaz, na izlaznu varijablu y. LR se u području zračnih slika pokazao kao dobar klasifikator u području prepoznavanja brodova na satelitskim snimkama gdje je uz pomoć ovog algoritma postignuta preciznost od 79% [24].

2.1.7. Algoritam nasumične šume (engl. Random forests algorithm)

Algoritam nasumične šume (RFC) [25] je metoda učenja za klasifikaciju, regresiju i druge zadatke, a djeluje tako da gradi mnoštvo stabala odlučivanja. U slučaju klasifikacije, izlaz predstavlja pripadnost ili nepripadnost određenoj klasi (0 ili 1), a u zadatcima regresije izlaz je uglavnom realan broj u intervalu od 0 do 1 koji predstavlja srednje predviđanje pripadnosti klasi. Ova metoda postiže najbolje rezultate s većim brojem stabala, što predstavlja i glavnu prednost metode - što je veći broj stabala odlučivanja, to je bolji rezultat klasifikacije. S obzirom na skup za treniranje $X = [x_1, x_2, \ldots, x_n]$ s odgovarajućim oznakama $Y = [y_1, y_2, \ldots, y_n]$, odabire se slučajni podskup iz skupa za treniranje i uklapa stabla na te uzorke nakon čega se postupak ponavlja B puta (B predstavlja broj stabala). Nakon treniranja, uzorci *x′* koji nisu korišteni u treniranju, mogu se dobiti pronalaženjem srednjih vrijednosti predviđanja f'_b svih stabala kako je prikazano u formuli 2.3.

$$f' = \frac{1}{b}\sum_{b=1}^{B} f'_b(x') \quad (2.3)$$

U formuli 2.3 Drugi način je uzimanje većine glasova između stabala odluke. Primjena stabala odluke je također jedna od razmotrenih mogućnosti zajedno s HOG značajkama u klasifikaciji ljudi na zračnim slikama. Brojni su primjeri upotrebe ove metode u detekciji auta, cesta, mapiranja zračnih slika [26], [27], ali također u detekciji ljudi na zračnim slikama [20].

2.1.8. Problemi primjene konvencionalnih metoda strojnog učenja

Dosadašnji pristupi u prepoznavanju ljudi na zračnim slikama susreću se s jako velikim problemima [28]. Objekti koje je potrebno detektirati na slikama nemaju uniformiran oblik, jako su mali u odnosu na cijelu sliku (u prosjeku oko 50x50 piksela na slici 4000*3000 piksela, a u nekim slučajevima znaju biti i manji i zauzimati samo 20-ak piksela slike). Upravo zbog toga standardne procedure izvlačenja značajki te detekcije objekta na slici predstavljaju jako veliki izazov. Postoje pristupi koji koriste nenadzirane metode strojnog učenja [29] kako bi grupirali regije sličnog intenziteta kao i pristupi koji se temelje na pretpostavci detekcije ljudi na osnovu mape ispupčenosti (*engl. visual saliency map*) [3]. Također postoje pristupi koji se koriste u video

nadzoru za detekciju i praćenje objekata od interesa uz pomoć Lucas-Kanade algoritma [30] i adaptivnog mean-shift algoritma [31] [32]. Unatoč velikom trudu uloženom u klasifikaciju, postoji još jako puno mjesta za poboljšanje. Kako se rezolucija takvih snimki povećava, tako i raste mogućnost primjene novih metoda koji se bave obradom slike s tolikom razinom detalja i bogatim sadržajem. Tradicionalne metode u klasifikaciji i mapiranju slika dobivenih uz pomoć UAV koriste lokalne značajke kao što je SIFT, SURF, HOG i BREIF u kombinaciji s metodom vreće slikovnih riječi (*engl. Bag of Visual Words BOW*) kako bi izvršili prepoznavanje i klasifikaciju objekata. Većina algoritama detekcije objekata usvaja pristup pomičnog prozora (*engl. sliding window*) na kojemu se vrši ekstrakcija značajki te klasifikacija i detekcija kako bi pridružili određenu klasu određenoj regiji na slici [33]. Postoji mogućnost poboljšanja detekcije, pogotovo za metode koje još nisu isprobane na slikama za detekciju ljudi u operacijama spašavanja ljudi (*engl. search and rescue - SAR*). Preciznost postojećih sustava za detekciju objekata od interesa na slikama se definitivno može poboljšati raznim pristupima pa i kombinacijom postojećih. Primarni problem oko primjene konvencionalnih metoda strojnog učenja je u tome što je način na koji čovjek detektira takve objekte na slikama još uvijek nepoznanica, pa je stoga teško napraviti skup značajki koje će biti reprezentativne i robusne za prepoznavanje. Ljudi na slikama uglavnom nemaju neka karakteristična obilježja kao što je oblik, boja ili nešto drugo pa je zbog toga jako teško razlučiti što je objekt od interesa na slici, a što ne. Još jedan problem za ovakve metode je što ne postoji univerzalna baza slika na kojima bi se mogli istrenirati ovakvi klasifikatori, što ovaj zadatak čini jako zahtjevnim. Ovi problemi bi se mogli umanjiti primjenom dubokog učenja na način da se ovaj zadatak pokuša riješiti uz pomoć neuronskih mreža treniranim na velikim skupovima podataka te primjenom učenja prenošenjem (*engl. Transfer learning*) kako bi se razvio novi pristup u detekciji ljudi na zračnim snimkama te pokušalo poboljšati dosad postignute rezultate. U nastavku rada opisane su neuronske mreže i najbitnije arhitekture konvolucijskih neuronskih mreža koje bi se mogle iskoristiti na ovom konkretnom problemu.

3. NEURONSKE MREŽE

Pojam neuronske mreže [34] odnosi se na relativno jednostavan računalni model koji se bazira na neuronskoj strukturi mozga. Mozak uči iz iskustva i upravo je to hipoteza na kojoj se zasnivaju neuronske mreže. Za predodžbu koliko je ljudski mozak moćan, dovoljno je razmotriti samo jedan dio sustava koji ljudski mozak obrađuje, a to je vizualni sustav. U svakoj hemisferi ljudskog mozga nalazi se primarni vizualni korteks (V1) koji se sastoji od 140 milijuna neurona. Neuroni su povezani s drugim neuronima preko milijardu veza. V1 je samo jedan u nizu vizualnih korteksa kojih ima ukupno pet, te svaki od njih izvršava složenije zadatke. Brojni zadatci kao što su čitanje, pisanje, hodanje itd. zahtijevaju prepoznavanje elemenata okoline kao što su znakovi, objekti, prepreke, lica itd. Bilo koji od ovih zadataka nije jednostavan, no ljudi ga jednostavno izvršavaju i to rade nesvjesno pa u većini slučajeva i ne cijene činjenicu koliko velik problem vizualni sustav rješava.

Ovaj problem postaje očit prilikom pokušaja pisanja računalnog programa za bilo koji od navedenih zadataka. Nešto što je prilično jednostavno naizgled, postaje jako kompleksno, pa čak i nemoguće. Jednostavne heuristike koje pomažu u prepoznavanju postaju jako teške kada ih je potrebno pretvoriti u određeni algoritam, odnosno definirati skup pravila koji će omogućiti računalu prepoznavanje elemenata. Jedan od načina rješavanja ovog problema je simulacija rada ljudskog mozga korištenjem računalnih alata kao što je neuronska mreža (*engl. neural network – NN*). Neuronska mreža ovom problemu pristupa na drugi način, a temelji se na pretpostavci kako je za prepoznavanje određenog objekta najprije potrebno naučiti što najviše utječe na klasifikaciju tog objekta u određenu kategoriju.

Neuronska mreža je jedna od metoda nadziranog strojnog učenja u kojoj se koriste veliki skupovi označenih podataka, koji se inače zovu podatci za treniranje. Na osnovu podataka za treniranje stvara se model, te neuronska mreža uči kako riješiti određeni zadatak. Drugim riječima, dozvoljava se neuronskoj mreži da sama prepozna koja pravila se koriste za rješavanje određenog zadatka. Jedna jako bitna činjenica je da što je veći skup za treniranje, to je i veća vjerojatnost da će neuronska mreža naći traženo rješenje problema [35].

3.1. Perceptron

Neuronsku mrežu čine neuroni, a posebna vrsta neurona se zove perceptron [36]. Perceptron na osnovu niza ulaznih varijabli proizvodi jednu izlaznu varijablu tj. računa sumu svih ulaznih varijabli. Za sumiranje svih ulaza $x_1, x_2, \dots, x_n, x_i \in \{0,1\}$ koriste se i težinske vrijednosti $w_1, w_2, \dots, w_n, w_i \in \mathbb{R}$ koje definiraju važnost pojedinih ulaznih varijabli. Što je veća težinska vrijednost w_i veća je i važnost ulazne varijable x_i. Vrijednost izlazne varijable ovisi od toga je li $\sum_i w_i * x_i$ manja od granične vrijednosti t kao što je prikazano u jednadžbi 3.1.

$$izlaz = \begin{cases} 0 & ako \sum_i w_i * x_i \leq t \\ 1 & ako \sum_i w_i * x_i > t \end{cases} \quad (3.1)$$

Zbog nepogodnosti uvjeta $\sum_i w_i * x_i > t$, postoje značajne promjene koje bi mogle pojednostavniti pojam perceptrona. Prva promjena je navedeni produkt $\sum_i w_i * x_i$ napisati kao točka produkt (*engl. dot product*) dvaju vektora $\sum_i w_i * x_i \equiv w \cdot x$, a druga je da se granična vrijednost prebaci na drugu stranu izraza. Ova granična vrijednost poznata je kao sklonost (*engl. bias*) perceptrona $b \equiv -t$. Sada se iskaz može prikazati kao u jednadžbi 3.2.

$$izlaz = \begin{cases} 0 & ako\ w \cdot x + b \leq 0 \\ 1 & ako\ w \cdot x + b > 0 \end{cases} \quad (3.2)$$

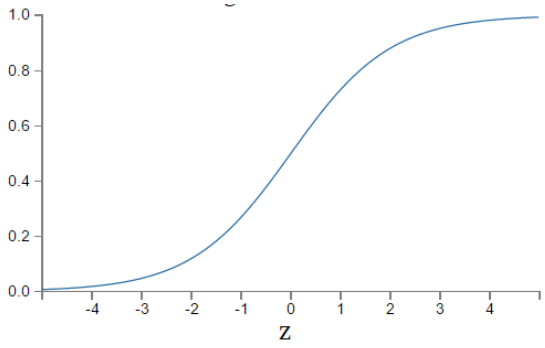
Sklonost perceptrona je mjera koja definira lakoću da neuronska mreža izbaci pozitivnu klasu. Drugim riječima, b je pristranost prema pozitivnoj klasi, dakle što je veća sklonost veća je vjerojatnost da neuronska mreža predvidi pozitivnu klasu. No, pitanje je kako neuronska mreža izbaci model koji služi za predikciju, a odgovor je u algoritmu za učenje. Ako postoji mreža perceptrona koju je potrebno naučiti riješiti određeni problem (npr. problem klasifikacije ručno napisanih znamenki) onda je potrebno naučiti težine i sklonosti kako bi se ulazni niz parametara pretvorio u odgovarajući, a pritom i ispravni izlazni niz parametara.

3.2. Sigmoid neuron

Slično kao i perceptron, sigmoid neuron ima ulazni niz podataka, ali umjesto da izlaz bude 0 ili 1, on je realan broj, a dobiva se uz pomoć funkcije $\sigma(w \cdot x + b)$ gdje je σ sigmoid funkcija zapisana u kao u jednadžbi 3.3.

$$\sigma = \frac{1}{1+\exp(-\sum_j (w_j x_j - b)} \qquad (3.3)$$

Oblik ove funkcije (slika 3.1) je jako bitan, i predstavlja izglađenu verziju granične funkcije. Izglađenost funkcije znači da će mala promjena Δw_j u težinskim vrijednostima i mala promjena u sklonostima Δb uzrokovati i malu promjenu Δy u izlazu neurona. Δy je linearna funkcija promjene u težinskim vrijednostima i sklonostima, što znači da će i jako male promjene težinskih vrijednosti i sklonosti utjecati na izlaz.

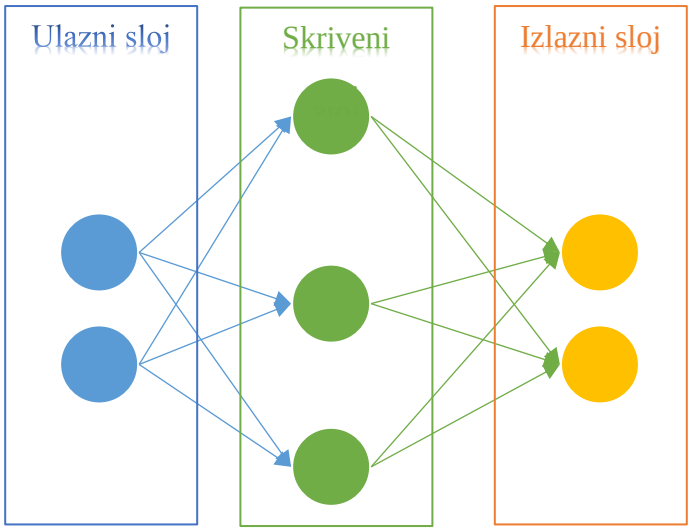


Slika 3.1 Sigmoid funkcija[1]

Općenito, neuronska mreža sastoji se od slojeva i to standardno najmanje tri sloja. Jedan sloj je ulazni sloj (*engl. input layer*), drugi sloj je skriveni sloj (*engl. hidden layer*) i treći sloj je izlazni sloj (*engl. output layer*) kao što je prikazano na slici 3.2. Broj čvorova u ulaznom sloju određen je dimenzijom ulaznih podataka, a slično tome broj slojeva u izlaznom sloju je jednak broju klasa koje neuronska mreža mora predvidjeti. Što više čvorova postoji u srednjem sloju, to se složenije funkcije mogu odrediti, ali visoka dimenzionalnost zahtjeva puno računalnih resursa u pogledu računanja i donošenja odluke. Također, ako postoji previše ulaznih parametara, neuronska mreža se može previše prilagoditi podatcima te davati krive rezultate u realnom svijetu (*engl. overfitting*).

[1] http://neuralnetworksanddeeplearning.com/chap1.html, 17.05.2017.

Veličinu skrivenog sloja određuju neke osnovne preporuke i smjernice, no uvijek ovisi o konkretnom problemu. Mreže sa dva ili više skrivenih slojeva nazivaju se duboke neuronske mreže *(engl. deep neural networks)*



Slika 3.2 Arhitektura obične neuronske mreže s jednim skrivenim slojem i sa dva ulaza i dva izlaza koja su potpuno povezana s oba sloja

Oblikovanje ulaznog i izlaznog sloja neuronske mreže je često prilično jednostavno. Za ulazni niz parametara u ulaznom sloju koji ima istu dimenzionalnost kao i broj parametara ulaznog vektora, neuronska mreža pridružuje jednu od više mogućnosti u izlaznom sloju, ovisno o broju klasa. Najčešće, najveći problem predstavlja oblikovanje skrivenih slojeva. Postoje određene heuristike koje pomažu odrediti broj skrivenih slojeva u odnosu na vrijeme koje je potrebno da se mreža istrenira. Neuronske mreže u kojima čvorovi svakog sljedećeg sloja primaju izlaz iz prethodnog sloja zovu se unaprijedne (*engl. feedfoward*) neuronske mreže. Ovakve mreže ne vraćaju izlaz nazad (ne postoje petlje). Mreže koje sadrže petlje zovu se rekurentne neuronske mreže (*engl. recurrent neural networks – RNN*). Ideja RNN je da se neuroni mogu izvršiti, a njihove posljedice se osjete samo određeno vrijeme, nakon čega se „utišaju“. Ta pojava izaziva veći efekt drugih neurona koji se također izvršavaju. Upravo zbog činjenice da je efekt svakog neurona kratkoročan, petlje u ovakvoj mreži ne predstavljaju problem.

Javlja se potreba za algoritmom koji pomaže naći težinske vrijednosti i sklonosti takve da izlaz ispravno aproksimira izlazni vektor $y(x)$ za svaki x. Mjeru koliko je dobro aproksimiran $y(x)$ definira tzv. „funkcija troška“ (*engl. cost function*) prikazana formulom 3.4.

$$C(w,b) = \frac{1}{2n}\sum_x \left|\left|y(x) - a\right|\right|^2 \qquad (3.4)$$

U formuli 3.4. parametri $w\ i\ b$ predstavljaju sve težinske vrijednosti i sklonosti, n je ukupni broj elementa u ulaznom sloju, a je vektor izlaza koji ovisi o ulazu, težinskim vektorima te o vrijednosti sklonosti. $C(w,b)$ je funkcija sume magnitude vektora koja je dobivena razlikom očekivane i dobivene vrijednosti. Ova funkcija se također zove tzv. „kvadratna funkcija troška“ (*engl. quadratic cost function*) ili „srednja kvadratna greška“ (*engl. mean square error – MSE*). Vrijednost funkcije je pozitivna, a kada vrijedi $C(w,b) \approx 0$ to znači da je $y(x)$ približno jednaka izlazu za sve trenažne ulazne podatke, stoga je cilj algoritma za treniranje minimizirati MSE, čime se zadatak pretvara u problem optimizacije. Drugim riječima, potrebno je pronaći težinske vrijednosti i sklonosti koje koštaju najmanje, a u tu svrhu može se iskoristiti algoritam gradijentnog spusta (*engl. gradient descent*). Najjednostavniji način određivanja minimuma funkcije je upotreba derivacija, ali se taj način komplicira s porastom broja varijabli. Budući da neuronske zahtijevaju jako puno varijabli, štoviše veći broj varijabli bi trebao označavati precizniji sustav, ovakav analitički pristup nije dobar. No, postoji jedna analogija koja zamišlja ovu funkciju kao udubinu te ukoliko se postavi imaginarna loptica, ona će se pod utjecajem gravitacije spustiti do dna ove udubine, samim time pronaći minimum ove funkcije, što je zapravo i cilj. Postoji mogućnost nasumičnog odabira početne točke i simuliranja gibanja (spuštanja) loptice, a takvu simulaciju moguće je analitički izračunati uz pomoć prve i druge derivacije $C(w,b)$ koje zapravo sadrže karakteristike oblika „udubine“.

Ako je C funkcija troška (*engl. cost function*) dvije varijable v_1 i v_2, gdje su Δv_1 i Δv_2 mali pomaci dvaju vektora, a u obliku parcijalnih derivacija oblik cijele funkcije može se zapisati kao u jednadžbi 3.5.

$$\Delta C \approx \frac{\partial \mathrm{C}}{\partial \mathrm{v}_1}\Delta v_1 + \frac{\partial \mathrm{C}}{\partial \mathrm{v}_2}\Delta v_2 \qquad (3.5)$$

Potrebno je izabrati Δv takve da je ΔC negativna kako bi se simuliralo padanje loptice prema dnu. Vektor svih promjena $\Delta v = (\Delta v_1, \Delta v_2)^T$ također gradijent od C, je vektor parcijalnih derivacija $\nabla \mathrm{C} = (\frac{\partial \mathrm{C}}{\partial \mathrm{v}_1}, \frac{\partial \mathrm{C}}{\partial \mathrm{v}_2})^T$ pa se ΔC može zapisati kao:

$$\Delta C \approx \nabla C \cdot \Delta v \qquad (3.6)$$

U jednadžbi 3.6 prikazano je zašto se $\nabla \mathrm{C}$ naziva gradijent vektorom jer povezuje promjene v sa promjenama funkcije C što je nešto što bi gradijent trebao raditi, no jako je bitno naglasiti kako izabrati Δv i ΔC. Npr. ako se uzme da je $\Delta v = -\eta \nabla \mathrm{C}$ tada će $\Delta C \approx \nabla C \cdot -\eta \nabla \mathrm{C}$ što garantira da će ΔC biti negativna. Vrijednost η je vrijednost koja se zove brzina učenja (*engl. learning rate*). Korištenjem navedene jednadžbe, vektor v se pomiče za određeni iznos Δv (jednadžba 3.7).

$$\mathrm{v} \rightarrow \mathrm{v}' = \mathrm{v} - \eta \nabla \mathrm{C} \quad (3.7)$$

Gradijent spuštanja radi na način da pokušava preračunati stupanj ∇C, a zatim ga usmjeriti u suprotnom pravcu. Ovakav pristup u izračunavanju stupnja spuštanja ima jedan glavni nedostatak, a to je činjenica da računanje druge parcijalne derivacije od C zahtjeva jako puno resursa. Način na koji se ova metoda može primijeniti na neuronsku mrežu, je primjena pravila stupnja silaska na težinske vrijednosti w_k i na b_l. Sada vektor stupnja ima dvije komponente $\frac{\partial \mathrm{C}}{\partial \mathrm{w_k}}$ i $\frac{\partial \mathrm{C}}{\partial \mathrm{b}_l}$ a ako se raspišu sve te komponente dobivaju se jednadžbe 3.8 i 3.9:

$$\mathrm{w_k} \rightarrow \mathrm{w_k'} = \mathrm{w_k} - \eta \frac{\partial \mathrm{C}}{\partial \mathrm{w_k}} \quad (3.8)$$

$$\mathrm{b_l} \rightarrow \mathrm{b_l'} = \mathrm{b_l} - \eta \frac{\partial \mathrm{C}}{\partial \mathrm{b_l}} \quad (3.9)$$

U praksi je za izračunavanje stupnja ∇C potrebno izračunati stupnjeve ∇C_x odvojeno za svaki trenažni podatak, a zatim naći prosjek. Nažalost, kada je broj podataka za treniranje jako velik, ovo može dugo potrajati, što znači da je učenje sporo. Poduzorkovanjem skupa za treniranje nasumično odabranim podatcima zvanim engl. *mini batch* moguće ke jako dobro aproksimirati ∇C_x, te s time ubrzati proces treniranja neuronske mreže. Ovaj pristup naziva se stohastički gradijentni spust (*engl. stohastic gradient descent*) [37]. Daljnji proces je treniranje neuronske mreže s mini-batch podatcima kao ulazom, a taj proces se ponavlja sve dok se ne iscrpe podatci na kojima se vrši treniranje, što se naziva i iscrpljene epohe (*engl. epoch*) treniranja. Kao što je prikazano u jednadžbama 3.8 i 3.9, stupanj je podijeljen s brojem podataka u ulaznom sloju, a ponekad se taj podatak može čak i izbaciti te sumirati po cijelom skupu podataka, što je posebno korisno u slučajevima kad je nepoznato koliko će podataka biti u ulaznom skupu za treniranje.

3.3. Propagacija unazad

Algoritam propagacije unazad originalno pojavio se tijekom 1970ih godina, no nije zaživio do 1986. kada je objavljen poznati članak [38] koji pokazuje kako propagacija unazad radi puno brže od prethodnih primjera za učenje, čime je omogućena primjena neuronskih mreža. Algoritam propagacije unazad omogućava jednostavan način računanja minimizacije funkcije troška. Ovaj algoritam propagira ulaz kroz mrežu od ulaznog do izlaznog sloja, određuje grešku i tu grešku propagira unazad sve do ulaznog sloja, nakon čega ju ugrađuje u formulu za učenje, a eksplicitno se može napisati u nizu koraka:

1. **Ulaz $\boldsymbol{x}$**: niz odgovarajućih aktivacija a^l za ulazni sloj
2. **Propagacija unaprijed**: za svaki $l = 2,3, \dots, L$ izračunava se $z^l = w^l a^{l-1} + b^l$
3. **Izlazna greška $\boldsymbol{\delta^L}$**: Izračun vektora $\delta^l = \nabla_{\mathrm{a}} \mathrm{C} \odot \sigma'(z^L)$, pri čemu je $\odot$ Hadamardov produkt dvaju matrica (radi se o množenju elemenata dvaju matrica i većina biblioteka pruža brze implementacije ovog produkta)
4. **Propagracija greške unazad**: za svaki $l = L-1, L-2, \dots, 2$ računamo $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. **Izlaz**: padanje funkcije troška dobivamo: $\frac{\partial \mathrm{C}}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ I $\frac{\partial \mathrm{C}}{\partial b^l_j} = \delta^l_j$, pri čemu je vidljivo da se padanje optimizira prema parametrima težinskih vrijednosti i sklonosti.

Ispitivanjem ovog algoritma jasno je vidljivo zašto se zove upravo propagacija unazad. Naime, vektor pogreške δ^l računa se unazad, počevši od zadnjeg sloja. Algoritam propagacije unazad računa padanje funkcije troška za jedan primjer podataka za treniranje $C = C_x$. U praksi često dolazi do pronalaska lokalnog umjesto globalnog minimuma. Zbog toga se propagacija unazad kombinira s algoritmom za učenje kao što je stohastički gradijent spusta (*engl stochastic gradient descent* - SGD) gdje se računa padanje više podataka za treniranje. Stoga se algoritam može zapisati kao:

1. **Ulazni skup podataka za treniranje**
2. **Za svaki podatak za treniranje:** Za skup odgovarajućih ulaznih aktivacija primjeni:
    - **Propagacija unaprijed**: za svaki $l = 2,3, \dots, L$ izračunava se $z^l = w^l a^{l-1} + b^l$
    - **Izlazna greška $\boldsymbol{\delta^L}$**: Izračun vektora $\delta^l = \nabla_{\mathrm{a}} \mathrm{C} \odot \sigma'(z^L)$

- **Propagacija greške unazad**: za svaki $l = L-1, L-2, \dots, 2$ računamo $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

3. **Stupanj padanja:** Za svaki $l = 2,3, \dots, L$ ažuriraju se težinske vrijednosti prema pravilu $w^l \to w^l - \frac{\eta}{m} \sum_x \delta^l (a^{x,l-1})^T$ i naravno sklonosti $b^l \to b^l - \frac{\eta}{m} \sum_x \delta^l$

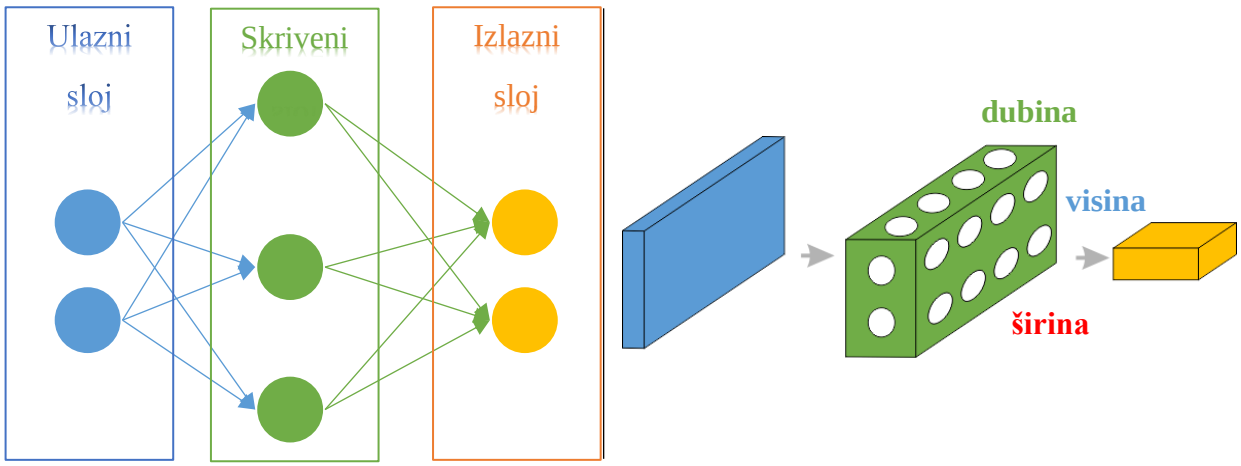
   Za razumijevanje ovog algoritma bitno je shvatiti kako izraz parcijalne derivacije $\frac{\partial C}{\partial w_{jk}^l}$ utječe na funkciju gubitka C u odnosu na težinske vrijednosti w (ili sklonosti b) u mreži. Zapravo, ovaj algoritam pokazuje kako i koliko težinske vrijednosti i sklonosti utječu na cjelokupnu mrežu.

4. KONVOLUCIJSKE NEURONSKE MREŽE

Konvolucijske neuronske mreže (engl. Convolutional Neural Networks) [39] su po strukturi slične neuronskim mrežama spomenutim u prošlom poglavlju, a koriste najmanje jedan konvolucijski sloj umjesto potpuno povezanog sloja. Cijela mreža zapravo predstavlja proces gdje je u problemu vizualnog prepoznavanja ulaz uglavnom slika odnosno niz vrijednosti piksela na slici, dok su na drugoj strani (izlaz) vrijednosti pripadanja određenoj klasi. CNN također ima funkciju gubitka (engl. *loss function*) na zadnjem sloju te sve što je navedeno vezano uz neuronske mreže vrijedi i za konvolucijske neuronske mreže. ConvNet arhitektura daje eksplicitnu pretpostavku da su ulazni parametri slike, koje nam omogućavaju da enkodiramo određena svojstva u arhitekturu.

Kao što je naglašeno, neuronske mreže primaju neki ulaz te taj ulaz transformiraju kroz niz skrivenih slojeva na zadnji sloj. Osnovni problem s neuronskim mrežama je da se ne skaliraju dobro na cijele slike (velike slike) jer za cijelu sliku treba jako velik broj težinskih vrijednosti. Potpuno spojeni slojevi neuronskih mreža kod ovakvih problema s velikim brojem ulaznih podataka bi vrlo lako doveo do prevelike prilagodbe ulaznim podatcima za treniranje (engl. *overfitting*). Da bi bili precizniji, neuron je u prvom skrivenom sloju spojen na samo jedan dio ulazne slike, npr. na regiju 5x5 piksela. Ova regija u ulaznoj slici se zove lokalno osjetljivo polje (*engl. local receptive field*).

Za razliku od obične neuronske mreže, slojevi CNN-a imaju 3 dimenzije: visinu, širinu i dubinu, kako je prikazano na slici 4.1. Neuroni su u određenom sloju spojeni s malom regijom slojeva prije njih, umjesto da je svaki sloj spojen sa svim ostalim slojevima. Ne samo da su neuroni trodimenzionalni, nego i izlazni vektor ima tri dimenzije.



Slika 4.1 Arhitektura konvolucijske neuronske mreže (CNN)

CNN predstavlja niz slojeva i svaki taj sloj pretvara volumen vrijednosti preko diferencijalbilne funkcije. Postoje 3 osnovna tipa slojeva CNN-a: konvolucijski sloj (convolutional layer), sloj udruživanja (pooling layer) i potpuno povezani sloj (fully connected layer). Tipična CNN arhitektura uključivala bi sljedeće elemente: INPUT – CONV – RELU – POOL – FC. Pri čemu su:

- INPUT – ulazna slika koja se sastoji od niza piksela određene visine i širine te broja kanala (obično 3).
- CONV – sloj će izračunati izlazne neurone koji su spojeni na lokalne regije u ulaznom nizu podataka. Ovaj sloj može se sastojati od $[x * y * 12]$ ako se koristi 12 različitih filtera.
- RELU – sloj će primijeniti aktivacijsku funkciju kao što je $\max(0, x)$ gdje se granične vrijednosti stavljaju na 0. Ovaj sloj ne mijenja volumen.
- POOL – sloj će izvršiti uzorkovanje prema prostornim dimenzijama smanjujući volumen npr. na $[\frac{x}{2} * \frac{y}{2} * 12]$.
- FC – u potpunosti spojen sloj izračunava rezultate svih klasa, koji rezultira volumenom veličine $[1 * 1 * 10]$ ukoliko mreža ima 10 klasa, u ovom slučaju svi neuroni su spojeni sa svim neuronima u prethodnom sloju.

Na ovaj način CNN transformiraju originalnu sliku sloj po sloj od piksela do klase kojoj taj piksel pripada. Neki od navedenih slojeva ne samo da primjenjuju određene aktivacijske funkcije već i očekuju određene parametre (težinske vrijednosti i sklonosti) kao što su CONV i FC, dok drugi slojevi kao što su RELU i POOL primjenjuju fiksnu funkciju. Parametri CONV i FC slojeva se

prilagođavaju uz pomoć metode spuštanja stupnja da bi rezultati klasa bili konzistentni sa oznakama u skupu podataka za treniranje u svakoj slici.

Prilikom primjene konvolucijskog sloja postoje tri parametra koji utječu na veličinu izlaznog volumena, a to su dubina (*engl. depth*), korak (*engl. stride*) i popunjavanje nulama (engl. *zero padding*). Dubina označava broj filtera koje želimo primijeniti. Različiti neuroni kroz dimenziju dubine mogu se aktivirati ukoliko se pojavi različito orijentiran rub ili blok određenih boja. Korak određuje micanje filtera po ulaznoj slici - ako je korak 1 onda se konvolucijski prozor pomiče za jedan piksel kroz određenu sliku. Ponekad je potrebno sliku i popuniti nulama, a jako dobro svojstvo popunjavanja nulama je mogućnost kontroliranja veličine izlaznog volumena. Veličina izlaznog volumena može se izračunati uz pomoć ulaznog volumena W, veličine polja receptora F, s korakom koji je primijenjen S i količine primjenjenog popunjavanja nulama P. Formula za izračunavanje broja neurona koji „odgovaraju“ je formula 4.1.

$$\frac{W-F+2P}{S}+1 \quad (4.1)$$

Svaki neuron u skrivenom sloju ima sklonosti i težinske vrijednosti koje su spojeni s njegovim lokalnim osjetljivim poljem, a sve te vrijednosti su jednake za sve neurone u skrivenom sloju, što zapravo znači da svi neuroni u prvom skrivenom sloju detektiraju jednake značajke samo na različitim lokacijama. Uz pretpostavku da određene težinske vrijednosti i sklonosti određuju okomiti rub na slici, isto svojstvo može biti korisno i na vodoravnom rubu na toj slici. Drugim riječima, konvolucijske neuronske mreže su jako otporne na translaciju. Upravo iz ovih razloga mapa dobivena iz ulaznog sloja često se naziva mapom značajki (engl. *feature map*). Struktura mreže, naravno, nema samo jednu mapu značajki već više njih.

4.1. Arhitekture konvolucijskih neuronskih mreža

Konvolucijske neuronske mreže (*engl. convolutional neural networks - CNN*) i duboko učenje (*engl. deep learning*) ostvarile su veliki uspon u području strojnog učenja kroz zadnjih nekoliko godina. Stohastični stupanj spuštanja kroz propagaciju unazad bio je efikasan model za treniranje konvolucijskih neuronskih mreža još početkom 90-tih [39] no nisu se više pojavljivale zbog uspona SVM-a. Godine 2012. ponovo je probuđen interes za CNN [40] pokazujući jako dobre rezultate na *ImageNet Large Scale Visual Recognition Challenge (ILSVRC).* Njihov uspjeh je rezultat treniranja

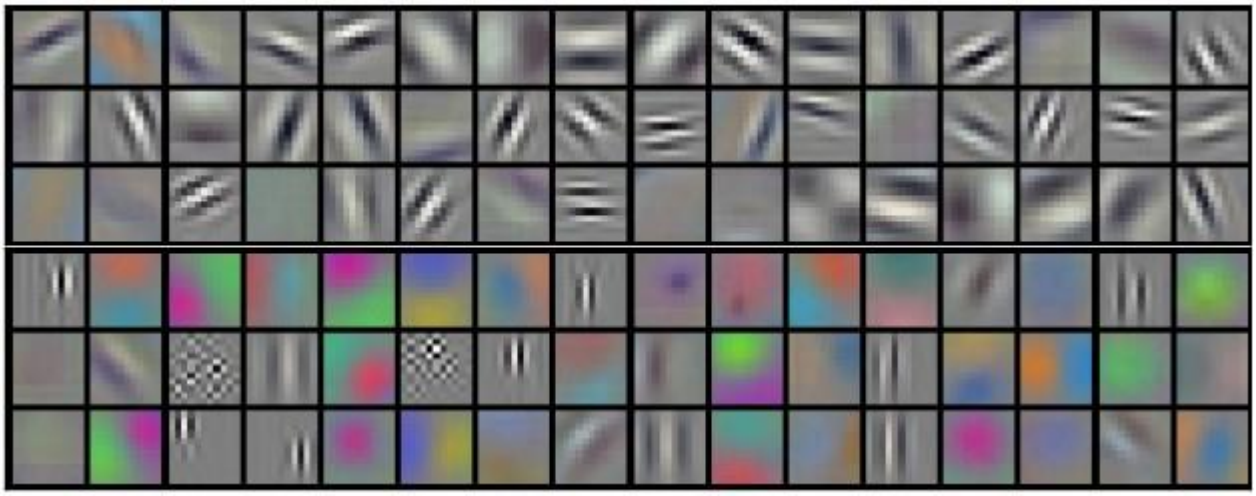
neuronske mreže na 1.2 milijuna označenih slika uz par modifikacija s konvolucijskom neuronskom mrežom zasnovanoj na ranijoj arhitekturi [39] uz par nadogradnji.

Različiti modeli kao što su GoogLeNet [41], ResNet [42], AlexNet [40], LeNet [39], DetectNet [43] te Caffe [44] programski okvir, postigle su zadivljujuće rezultate na raznim zadatcima strojnog učenja. Ovaj novi pokret u strojnom učenju daje obećavajuće rezultate i u raznim zadatcima računalnog vida. CNN je česta arhitektura neuronske mreže koja simulira način na koji mozak uči i razvija hijerarhijske strukture reprezentacije značajki. Ove naučene značajke potrebne su za uspješno razlikovanje određenih slika i njihovo svrstavanje u određenu klasu. CNN u većini zadataka prepoznavanja i identifikacije nadmašuje standardne metode s ručno napravljenim značajkama. Osnovni nedostatak dubokih neuronskih mreža je upravo u tome što zahtijevaju jako veliki skup podataka za treniranje te nisu robusne na skaliranje i rotaciju. No, postojeći modeli se mogu iskoristiti koristeći metodu prenošenja učenja iz treniranih modela. U sljedećim podpoglavljima su prikazane osnovne arhitekture koje su izazvale revoluciju u području računalnog vida.

4.1.1. AlexNet

AlexNet je arhitektura koja je pokrenula sve, te se smatra da je rad [40] jedan od najutjecajnijih radova u ovom području. Ova arhitektura je pobijedila na ImageNet 2012 izazovu. 2012. godina je označena kao prva godina gdje je korištena CNN kako bi se postigla stopa pogreške od 15.4%, dok je sljedeći najbolji rezultat imao stopu od 26.2%, što je bilo ogromno poboljšanje. Arhitektura je prihvaćala slike dimenzija $[227 * 227 * 3]$ na prvom konvolucijskom sloju broj receptora je $F = 11$, s pomakom $S = 4$ i bez popunjavanja nulama P=0. Budući da je ulazna slika $W = 277$, tada konvolucijski sloj ima veličinu $[55 * 55 * 96]$ (budući da je pomak 4 od slike 277*277 dobije se mapa značajki 55*55, a takvih je različitih mapa bilo 96). Svaki od navedenih neurona je povezan s određenom regijom veličine $[11 * 11 * 3]$ na ulaznoj slici. Već je vidljivo da je to jako veliki broj parametara $((11 * 11 * 3 + 1) * 55 * 55 * 96 = 105{,}705{,}600)$ te je njihovo računanje jako zahtjevno, zbog čega se uvodi shema dubine dijela koji definira iste parametre težinskih vrijednosti i sklonosti za sve neurone, stoga će za blok $11 * 11 * 3$ imati ukupno $34{,}944$ parametra. Primjer filtera veličine $11 * 11 * 3$ te svaki od njih se dijeli sa svih $55 * 55$ neurona u jednom djelu dubine. Vidljivo je da je dijeljenje parametara u određenim slučajevima i razumljivo, npr. u slučaju

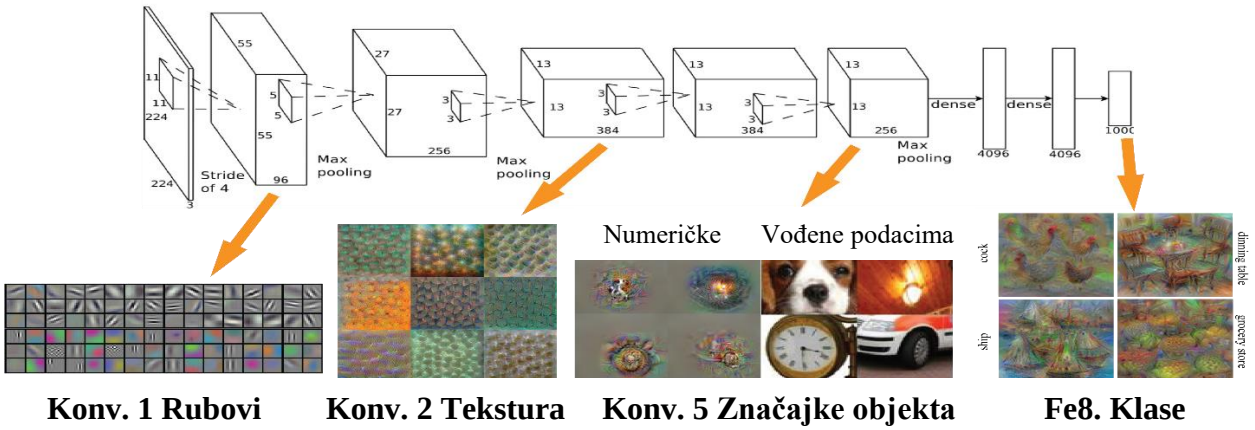
detekcije okomitog ruba na određenom dijelu slike, da se može prepoznati i na drugom dijelu slike. Na slici 4.2 prikazano je 96 filtera koje je naučila AlexNet neuronska mreža.



Slika 4.2 Primjer filtera naučenih na AlexNet arhitekturi [2]

Danas je ova arhitektura jedna od jednostavnijih, a sastoji se od 5 konvolucijskih slojeva: sloj maksimalnog udruživanja, sloj odbacivanja (*engl. dropout*) te 3 u potpunosti povezana sloja. Mreža se koristi za klasifikaciju 1000 različitih kategorija. AlexNet je treniran na ImageNet [45] koja se sastoji od 15 milijuna označenih slika s ukupno preko 22000 kategorija. Korištena je ReLU funkcija za funkciju nelinearnosti. Neke od značajnijih stvari su tehnike poboljšanja konvolucijske neuronske mreže primjenom translacija, vodoravnim refleksijama na slikama i sl. Model je treniran koristeći stohastički gradijentni spust s određenim vrijednostima za inerciju (*engl. momentum*) i raspadanje (*engl. decay*). Mreža je trenirana na Nvidia GTX 580 GPU nekoliko dana. Tehnike poboljšanja koje su korištene, kao i slojevi odbacivanja, ključni su elementi i danas se koriste u brojnim arhitekturama. Na slici 4.3. prikazana je AlexNet arhutektura zajedno sa svim izlazima iz konvolucijskih slojeva.

[2] http://cs231n.github.io/convolutional-networks/

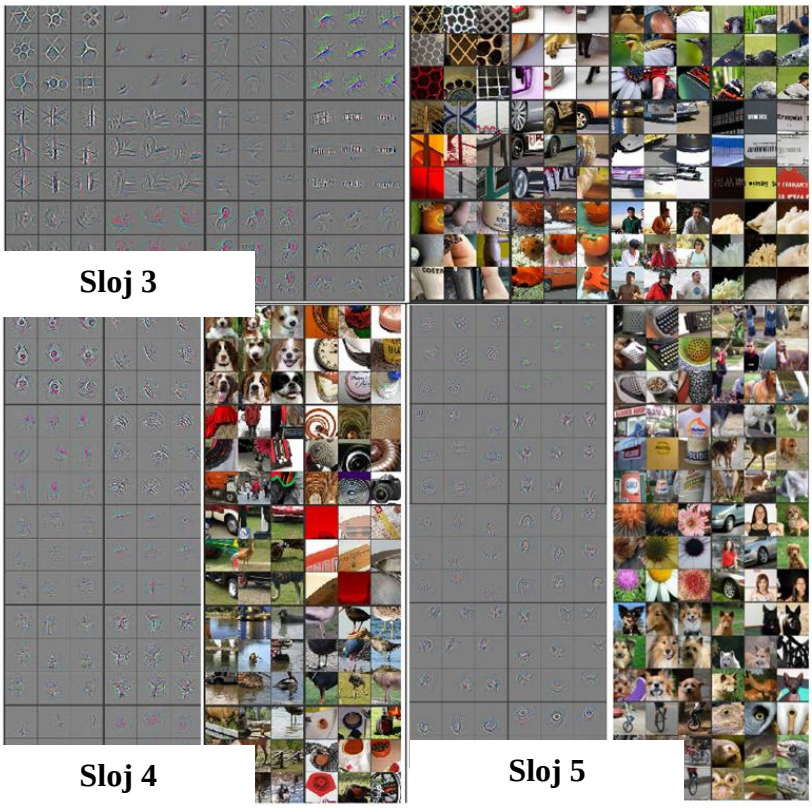


Slika 4.3 Prikaz alexNet Arhitekture i izlaza iz pojedinih konvolucijskih slojeva[3]

4.1.2. ZF Net

Pobjednik na ILSVRC 2013 natjecanju je upravo ZF Net [46] koji je postigao stopu pogreške od 11.2%. Ovaj model je poboljšao nedostatke AlexNet arhitekture, ali je i razvijen niz ključnih ideja koje poboljšavaju performanse mreže. U radu se navodi kako je s povećanjem dostupnih resursa i povećanjem snage obrade podataka uz pomoć GPU moguće razviti robusnije modele. U odnosu na AlexNet, ZF Net nema velikih razlika, osim u pogledu performansi. Dok je AlexNet treniran na 15 milijuna slika, ZF Net je treniran na 1.3 milijuna slika. Umjesto regije veličine 11*11 korištena je regija 7*7 i smanjena je veličina pomaka. Ovo omogućava da se zadrže orginalne informacije iz ulaznog volumena. Filter korišten u AlexNet arhitekturi preskakao je dosta bitnih informacija. ZF Net je uveo dekonvolucijsku mrežu (*engl. DeconvNet*) koja pomaže u istraživanju aktivacija i njihovih „veza“ s ulaznim podatcima, tj. DeconvNet pomaže mapiranje značajki prema pikselima. Niži slojevi CNN-a su uvijek značajke niže razine koje detektiraju jednostavne rubove ili boje, dok se višim slojevima detektiraju više kružni elementi, a na najvišim slojevima, kao što je prikazano na slici 4.4, vide se značajke viših razina kao što su lica psa ili ptice.

[3] http://www.cc.gatech.edu/~hays/compvision/proj6/



Slika 4.4 Deconvnet prikaz konvolucijskih slojeva na ZF Net arhitekturi[4]

ZF Net je sa svojim pristupom pokazao detaljnije kako radi CNN uz DeconvNet, no pokazao je i kako dodatno poboljšati pristup u primjeni CNN u zadatcima prepoznavanja i detekcije na slikama. Mreža je trenirana na Nvidia GTX 580 GPU dvanaest dana.

4.1.3. VGG Net

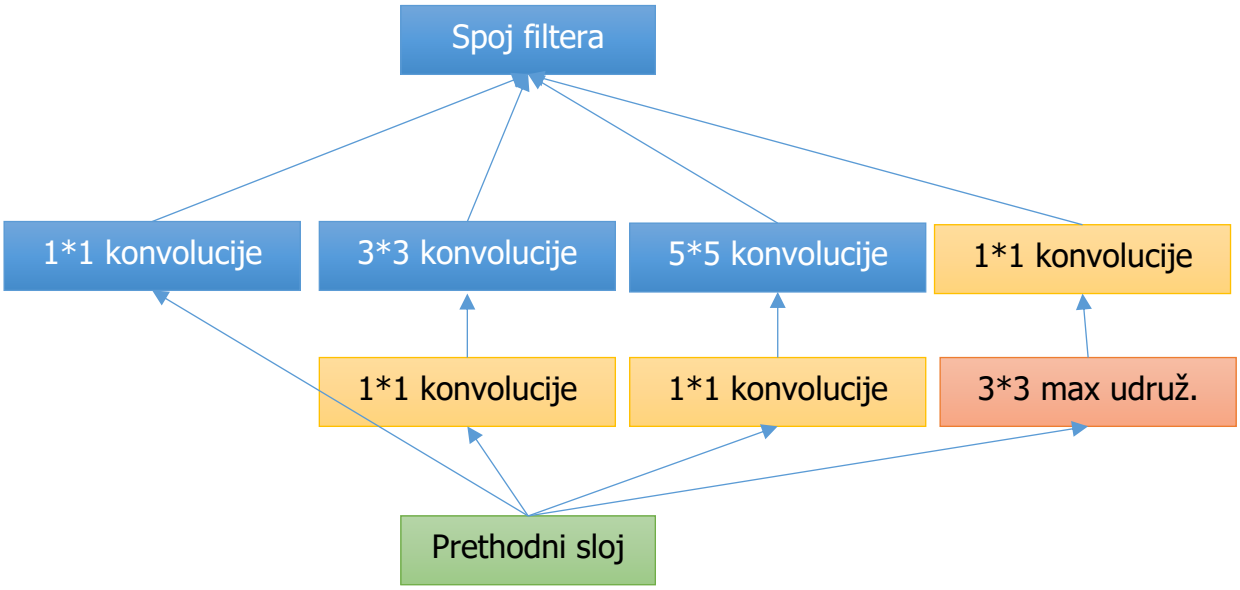
VGG Net model [47] je postigao stopu pogreške od 7.3%. Zasnovan je na pretpostavkama jednostavnosti i dubine. Mreža se sastojala od 19 slojeva i striktno je koristila 3*3 filtere s pomakom i popunjavanjem od 1 s 2*2 max slojem udruživanja s pomakom od 2. Ovi filteri se razlikuju od prethodnih po tome jer se koriste kombinacije dvaju 3*3 slojeva s osjetilnim poljem od 5*5 što simulira veće filtere dok zadržava prednosti manjih filtera. Broj filtera se uduplava nakon svakog max sloja udruživanja. Korištena je ReLU funkcija nakon svakog konvolucijskog sloja i treniranje sa stohastičnim gradijentnim spustom. Ovaj pristup smanjuje prostorne dimenzije,

[4]https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

ali povećava dubinu neuronske mreže. Ova arhitektura je trenirana na 4 Nvidia Titan Black GPU tri tjedna.

4.1.4. GoogLeNet

Ideja jednostavnosti neuronske mreže na arhitekturi GoogLeNet [41] ne vrijedi. Ova arhitektura se sastoji od 22 sloja te s postignutom stopom pogreške od 6.7% predstavlja jednu od prvih pet arhitektura na svijetu. Koristi pristup koji nije zahtijevao da se konvolucijski slojevi i slojevi udruživanja slažu jedni za drugim u sekvencijalnom redoslijedu. Ovakav pristup zahtjeva jako velike resurse. GoogLeNet za razliku od ostalih arhitektura sadrži dijelove koji se izvode paralelno. Prikaz paralelnog dijela GoogLeNet arhitekture prikazan je na slici 4.5.



Slika 4.5 Prikaz paralelnog dijela GoogLeNet arhitekture

„Prethodni sloj“ na slici je ulaz a „Spoj filtera“ je izlaz modela. U suštini na prethodnim arhitekturama u svakom koraku modela potrebno je donijeti odluku hoće li se koristiti sloj udruživanja ili konvolucije. Početni modul (*engl. inception module*) prikazan na slici omogućava izvođenje više operacija paralelno. Modul se sastoji od „mreže unutar sloja mreže“, srednje velikih konvolucijskih filtera, jako velikih konvolucijskih filtera i operatora udruživanja. Ovaj pristup „izvlači“ jako malene detalje u slici, a pored toga na svakom konvolucijskom sloju se nalazi ReLU koji pomaže u poboljšanju nelinearnosti mreže. GoogleNet arhitektura također se može koristiti za

detekciju, a osnovne su joj korištenje 12 puta manje parametara nego AlexNet, te višestruko korištenje izrezane iste slike tijekom treniranja i testiranja.

4.1.5. Microsoft ResNet

ResNet [42] je arhitektura od 152 sloja koja je trenutno oborila sve rekorde u klasifikaciji, detekciji i lokalizaciji. Pored rekorda u broju slojeva, ResNet ima stopu pogreške od 3.6%, dok ljudi ovisno o njihovim vještinama i stručnosti postižu rezultate od 5% do 10% pogreške. Ideja ResNet-a je u tome da računa rezultat $F(x)$ nakon što ulaz *x* prođe kroz niz ReLU-CONV slojeva, nakon čega taj rezultat dodaje na početnu sliku $H(x) = F(x) + x$. Zapravo, računa se razlika promjene ulaza i primijenjenih filtera kako bi se dobila nova reprezentacija na kojoj se dalje vrši procesiranje. Trenutno je to najbolja arhitektura te je jako upitno što se može postići na sljedećim natjecanjima, uz pretpostavku da slaganje sve više slojeva CNN-a neće dovesti do velikih skokova u performansama. ResNet kao takva je zahtjevna arhitektura koju je potrebno trenirati na 8 GPU dva ili tri tjedna.

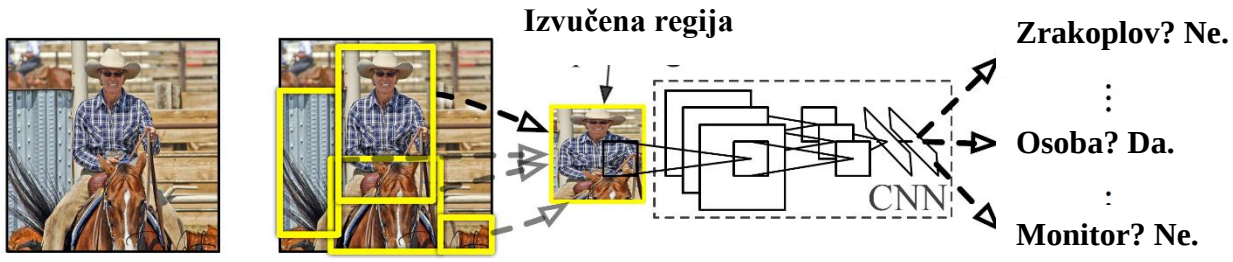
Tablica 4.1 Prikaz i usporedba arhitektura i postignuta preciznost na pojedinim zadatcima klasifikacije slika

Naziv arhitekture	**Broj i vrsta uređaja na kojem je trenirana**	**Postignuta preciznost**	**Vrijeme treniranja**
AlexNet	2 GTX-580 GPU	15.4%	5-6 dana
ZF Net	GTX 580 GPU	11.2%	12 dana
VGG Net	4 GPU Titan Black	7.3%	2-3 tjedna
GoogLeNet	„nekoliko GPU-a“	6.7%	1 tjedan
ResNet	8 GPU	3.6%	2-3 tjedna

4.1.6. R-CNN

Regionalni CNN (R-CNN) arhitektura prvi put je predstavljena 2014. godine [48], nakon čega su u 2015. godini objavljena dva nova članka koja pokazuju značajno poboljšanje u brzini izvršavanja detekcije, Fast R-CNN [49] i Faster R-CNN [50]. Osnovna ideja ove arhitekture je iskoristiti duboku neuronsku mrežu koja je već trenirana za klasifikaciju slika i modificirati ju za detekciju objekata [48]. Za razliku od klasifikacije na slikama, detekcija zahtjeva lokalizaciju objekta. U tu svrhu, potrebno je napraviti detektor s pomičnim prozorom [51][52] koji za vrijeme testiranja

generira oko 2000 nezavisnih regija te izvlači vektor značajki fiksne duljine iz svake regije, neovisno o obliku regije. Cijeli proces prikazan je na slici 4.6.



Slika 4.6 Arhitektura R-CNN konvolucijske neuronske mreže[5]

Klasifikacija je izvršena uz pomoć SVM klasifikatora, zbog čega je prethodno potrebno izračunati CNN značajke, a u tu svrhu je korišteno omatanje slike za računanje vektora značajki fiksne duljine. Ovaj pristup se razlikuje od drugih pristupa koji koriste metodu pomaknutog prozora kao što je OverFeat [53] i omogućava izvlačenje regija koristeći metode prijedloga regija. Na izvučenim regijama primjenjuju se klasifikatori kako bi se ostvarila detekcija. Nakon klasifikacije izvršava se ponovno procesiranje kako bi se refinirali rezultati, uklonili duplikati itd.

4.1.7. You Only Look Once

Spomenuti pristup je jako teško optimizirati zbog složenosti, stoga postoje pristupi kao što je Yolo (*engl. You Only Look Once*) [54] koji formuliraju problem na način da se pikseli na slici pretvaraju u regije i pripadajuće klase. Najveća prednost ovog pristupa je u njegovoj brzini, dok se ostale prednosti očituju u tome da ova arhitektura promatra sliku kao cjelinu prilikom treniranja i testiranja te na taj način izvlači kontekstualne informacije o klasama i njihovom pojavljivanju. Za razliku od R-CNN gdje se najprije predviđaju regije pa se na njima radi klasifikacija, u YOLO pristupu se znatno smanjuje broj pogrešno pozitivnih regija (*engl. false positive*) jer je na većim područjima lakše odrediti kontekst i sami zadatak se generalizira. Arhitektura YOLO inspirirana je GoogLeNet modelom za klasifikaciju i ima 24 konvolucijska sloja te 2 potpuno povezana sloja, a njezin osnovni nedostatak je u tome što je za detekciju malih objekata jako loša te ne pokazuje obećavajuće rezultate.

[5]https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf

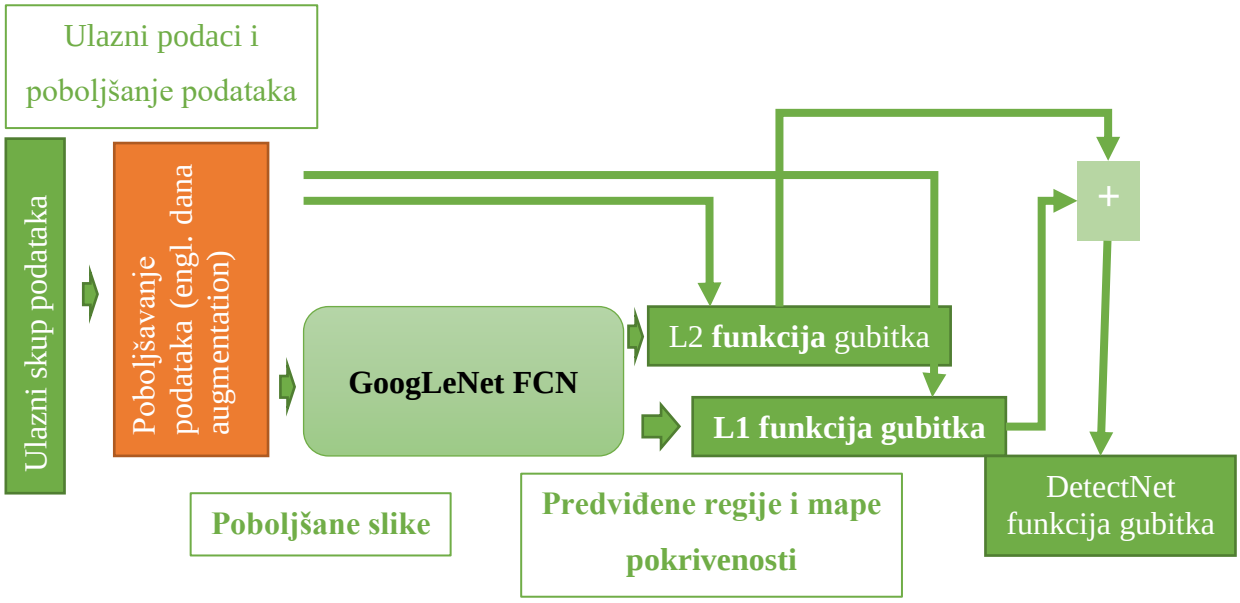
4.1.8. DetectNet

Nadogradnja na detekciju je programski okvir od Nvidia tvrtke koja se zove DetectNet [43], a rješava ključan problem po tome što uvodi trodimenzionalnu oznaku što omogućava obradu slika različitih veličina, s različitim brojem objekata. Način na koji ovaj programski okvir radi je da sliku dijeli na više dijelova gdje se za svaki blok dodjeljuje oznaka klase i koordinate vrha regije objekta kojeg želimo detektirati. Također, uključuju se i drugi parametri kao što je skrivenost objekta te ukoliko se u istom bloku nalazi više objekata, neuronska mreža u obzir uzima veći objekt. DetectNet predviđa je li objekt prisutan i gdje su rubovi objekta u odnosu na sredinu bloka.

Arhitektura DetectNet se sastoji od pet dijelova, pri čemu su jako bitna prva tri koraka:

1. Podatkovni sloj uzima slike za treniranje i njihove oznake, a transformacijski sloj primjenjuje „poboljšanje“ podataka (*engl. data augmentation*).
2. U potpunosti povezani konvolucijski sloj (FCN) izvršava ekstrakciju značajki te predikciju klasa i regiju unutar bloka.
3. Funkcija gubitka (*engl. loss function*) istovremeno mjeri grešku u dva zadatka predviđajući pokrivenost objekta bloka u kojem se nalazi.
4. Funkcija grupiranja proizvodi završni niz regija tjekom evaluacije.
5. Na kraju se izračunava srednja vrijednost prosječne preciznosti (*engl mean Average Precision mAP*) kako bi se izmjerila preciznost modela preko podataka za validaciju.

Proces treniranja DetectNet arhitekture je prikazan na slici 4.7. DetectNet omogućava i definiciju razmaka između blokova na način da se definira korak konvolucijske neuronske mreže u pikselima, a može se definirati i veličina dijela slike. Kada su postavljeni ovi parametri, svaki put kada se doda slika u DetectNet tijekom treniranja uzima se nasumično podslika zadanih dimenzija kao ulaz. Ovaj princip može biti koristan u slučajevima kada su slike jako velike, a objekti od interesa jako mali.



Slika 4.7 Treniranje DetectNet

Poboljšanje podataka je ključan korak za uspješno treniranje visoko osjetljivih i preciznih detektora objekta. Postoje parametri koji se mogu prilagođavati skupu podataka za treniranje, kao što je broj okretanja i pomaka piksela koji se primjenjuju na ulaznom skupu za treniranje. Prednost ovakvog pristupa je u tome što neuronska mreža nikad ne vidi istu sliku više puta pa je više otpornija na prilagodbu podatcima. Struktura podmreže FCN ima istu strukturu kao i GoogLeNet bez ulaznih slojeva, zadnjeg POOLING sloja i izlaznog sloja. FCN je konvolucijska neuronska mreža koja nema potpuno povezanih slojeva i s time omogućava ulazne slike raznih dimenzija i efektivno omogućava i primjenjuje CNN na principu pomičnog prozora. Izlaz je matrica realnih vrijednosti koja se može staviti preko ulazne slike. DetectNet koristi linearnu kombinaciju dviju različitih funkcija gubitka za optimizaciju. Prva je funkcija gubitka mape pokrivenosti koja predstavlja kvadratnu sumu razlike između istinite vrijednosti i predviđenog objekta kroz sve blokove u skupu podataka za treniranje.

$$\frac{1}{2N}\sum_{i=1}^{N} |pokrivenost_i^t - pokrivenost_i^p|^2 \quad (4.2)$$

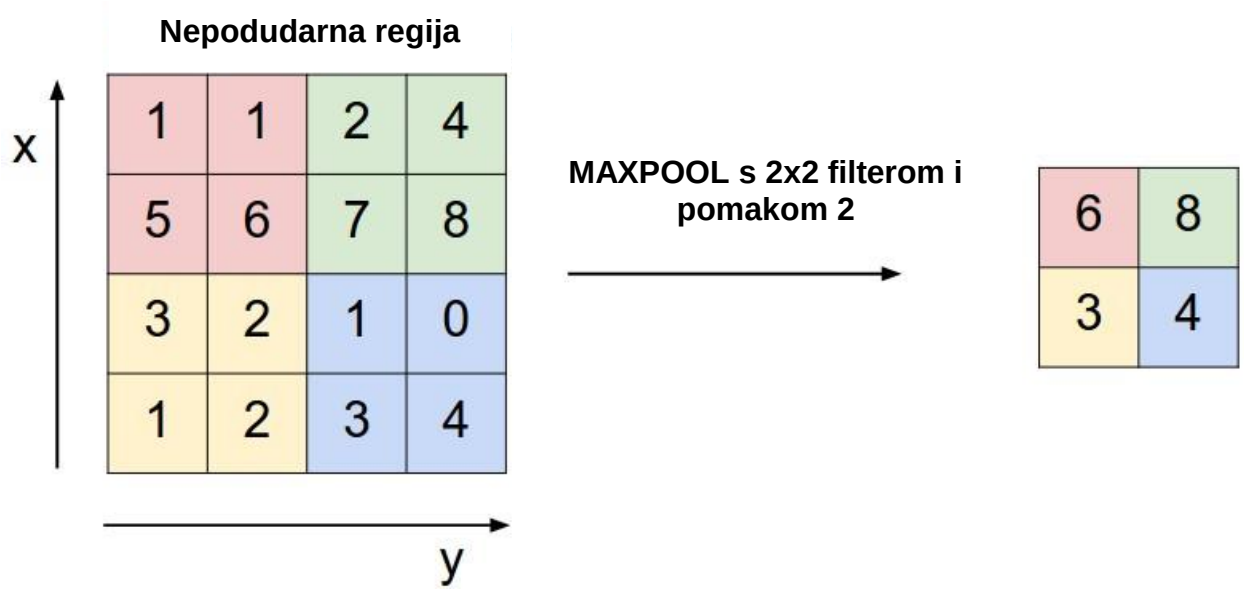
Druga funkcija gubitka je srednja apsolutna razlika predviđenih rubova regije i trenažnih rubova regije u svakom bloku.

$$\frac{1}{2N}\sum_{i=1}^{N}[\left|x_1^t - x_1^p\right| + \left|y_1^t - y_1^p\right| + \left|x_2^t - x_2^p\right| + \left|y_2^t - y_2^p\right|] \quad (4.3)$$

Caffe [44] programski okvir minimizira težinsku sumu ovih vrijednosti. Posljednji slojevi DetectNet arhitekture se koriste za spajanje i filtriranje skupova regija koje su generirane za svaki blok.

4.2. Primjene metoda dubokih neuronskih mreža

Kao što je vidljivo iz prethodnih poglavlja CNN su uvele revoluciju u području računalnog vida. Već pojavom prve arhitekture pojavili su se zapanjujući rezultati koje je bilo gotovo nemoguće postići tradicionalnim metodama strojnog učenja. CNN postižu bolje rezultate u brojnim zadatcima računalnog vida [41], [55]. Arhitektura jedne CNN započinje s konvolucijskim slojem koji je u većini slučajeva 3D volumen neurona, a on se sastoji od skupa detektora značajki koji koriste konvolucijsku masku koja se konstantno pomiče preko ulaznog sloja kako bi izvukla značajke koje su otporne na šum i translaciju. Ponavljanjem ovog postupka preko ulaznog sloja dozvoljava se dijeljenje težinskih vrijednosti koji smanjuju broj parametara koje je potrebno naučiti, što zauzvrat ima rezultat bržeg izvođenja i mogućnosti treniranja velikih i „moćnih“ neuronskih mreža. Nakon konvolucijskog sloja izvodi se nekoliko operacija kao što je uzorkovanje (*engl. pooling*), čime se smanjuje prostorna dimenzija volumena, a samim time i broj parametara za sljedeći sloj. Najčešće korištena tehnika je korištenje funkcije maksimuma na nepodudarne podregije inicijalne reprezentacije kao što je prikazano na slici 4.8. gdje je na 4x4 regiji korišten 2x2 blok s pomakom od 2, a u svakoj regiji uzima se piksel koji ima najveći intenzitet.



Slika 4.8 Prikaz primjene MAXPOOOL operacije nad nepodudarnom regijom

U cilju povećanja raspršenosti podataka, nakon svakog konvolucijskog sloja, primjenjuju se aktivacijske funkcije (ReLU). Ovaj sloj se također bavi i problemima poput problema nestajućeg padanja (*engl. vanishing gradient*) u propagaciji greške unazad. Skup konvolucijskih i poduzorkovanih slojeva na kraju se spajaju u potpuno povezani sloj s klasifikacijskim slojem, a u ovom sloju mreža je potpuno povezana sa svim neuronima gdje se koristi neka aktivacijska funkcija kao što je softmax. Treniranje dubokih neuronskih mreža odvija se s ogromnim skupom podataka. Jedan od razumnih načina da se ovaj problem riješi je korištenje učenja prenošenjem (*engl. transfer learning*) što u nekim slučajevima predstavlja korištenje težinskih vrijednosti modela koji je predtreniran na velikom skupu podataka, te korištenje istih kako bi se inicijalizirao vlastiti model. Ovakav pristup u brojnim radovima je pokazao dobre rezultate [56]–[58]. Korištenje dubokih višeslojnih neuronskih mreža tradicionalno zahtjeva jako veliku količinu podataka kako bi se omogućio olakšani pristup kompleksom procesu izvlačenja značajki te procesu reprezentacije i klasifikacije. Kad je riječ o zračnim slikama, ograničena baza podataka za treniranje mreže za detekciju objekata od interesa, predstavlja jako veliki problem. Za nadvladavanje ovog problema najčešće i najučinkovitije je korištenje tehnike učenja prenošenjem kako bi se iskoristila predtrenirana CNN mreža koja je trenirana na generaliziranim zadacima klasifikacije za koje postoje veliki skupovi podataka kao i modeli koji su trenirani nekoliko tjedana na bazama od nekoliko milijuna slika na GPU procesorima jako visokih performansi.

4.3. Preporuke za korištenje učenja prenošenjem

Učenje prenošenjem ili induktivno prenošenje je metoda koja se bazira na ideji pohranjivanja stečenih znanja prilikom rješavanja jednog problema i primjene istih u rješavanju drugog, srodnog problema. Ideja se prvi put pojavljuje 1993. kada je L. Y. Pratt predstavio algoritam prijenosa temeljenog na diskriminabilnosti (engl. discriminability-based transfer -DBT) [59]. Neuronske mreže obično se treniraju od nule, oslanjajući se samo na vlastite trening podatke. Međutim, kako se sve više i više mreža trenira za različite zadatke, postaje razumno tražiti metode koje izbjegavaju treniranje od nule na način da se grade na rezultatima prethodno osposobljenih mreža. Ova ideja temelji se na činjenici da ni ljudi ne primaju pojedine zadatke za učenje izolirane jedne od drugih, nego su svi zadatci sekvencijalni. Intuitivno, puno je lakše naučiti sekvencu međusobno povezanih zadataka, nego naučiti svaki od tih zadataka odvojeno, neovisno jedan od drugog. Primjerice, ljudski vizualni sustav može lakše naučiti prepoznati breskvu, ako već zna prepoznati jabuku i naranču. Također, čovjek će lakše naučiti francuski jezik ako već zna engleski i latinski. Isto se događa i s neuronskim mrežama. Jedan od primjera je mreža za prepoznavanje govora koja je trenirana samo na američkim govornicima engleskog jezika koja može biti iskorištena za ubrzavanje učenja mreže Britanskih govornika, jer se radi o podraspodjela iste raspodjele (u oba slučaja radi se o engleskom jeziku, ali različiti su naglasci). Postavlja se pitanje kako već trenirana neuronska mreža može biti korištena za treniranje neke nove mreže, tj. suština ovog pristupa, odnosno prenošenje (*engl. transfer*). Postoji nekoliko različitih scenarija kod korištenja učenja prenošenjem.

Prvi se odnosi na korištenje CNN-a kao fiksnog ekstraktora značajki gdje je CNN predtrenirana na ImageNet bazi bez zadnjeg potpuno povezanog sloja, a gdje se ostatak CNN koristi kao ekstraktor značajki na novom skupu podataka. U AlexNet arhitekturi ovaj sloj bi imao 4096-dimenzionalni vektor za svaku sliku koja sadrži aktivacije skrivenog sloja. Ove značajke se zovu CNN kodovi, a kada se izvrši ekstrakcija 4096 dimenzionalnih kodova za sve slike, pokreće se linearni klasifikator za novi skup podataka.

Drugi pristup je podešavanje parametara CNN-a gdje se vrši ponovno treniranje klasifikatora preko CNN-a na novim podatcima te podešavanje parametara predtrenirane mreže nastavljajući proces propagacije pogreške unazad. Moguće je podesiti sve slojeve CNN-a ili ostaviti neke slojeve fiksnima, a podesiti parametre viših nivoa mreže. Ovaj pristup je motiviran primjedbom da su ranije

značajke CNN-a sadržavale generičke značajke koje mogu biti korisne za mnoge zadatke, a CNN postaje osjetilniji na detalje klasa koji se nalaze u originalnom skupu podataka. U primjeru ImageNet-a koji sadrži jako puno vrsta pasa, logično je da će neuronska mreža biti osjetilnija na detekciju različitih vrsta pasa. Dva su osnovna faktora u odluci kada koristiti učenje prenošenjem. Prvi faktor se odnosi na činjenicu je li nova baza na kojoj se želi napraviti treniranje i klasifikacija velika ili mala, dok je drugi faktor činjenica da li je velika sličnost podataka na kojima se želi izvršiti treniranje. Sukladno tome, postoje 4 različita scenarija:

1. **Novi skup podataka je malen i sličan originalnom skupu podataka** – budući da je skup podataka malen nema smisla podešavati parametre CNN zbog problema s prevelikom prilagodbom podatcima. Stoga je u ovim slučajevima najbolje koristiti težinske vrijednosti početnog modela.
2. **Novi skup podataka je velik i sličan originalnom skupu podataka** – budući da se skup podataka samo proširuje, u ovakvim slučajevima najbolje je podešavati parametre postojeće mreže.
3. **Novi skup podataka je malen i različit od originalnog skupa** – budući da je skup podataka jako malen najbolje je trenirati linearni klasifikator. Kako je skup podataka različit, nije dobro trenirati klasifikator na osnovu mreže s viših slojeva, već je bolje trenirati SVM klasifikator iz aktivacija koje su se javile nazad u mreži.
4. **Novi skup podataka je velik i različit od originalnog skupa podataka** – u ovakvim slučajevima potrebno je treniranje neuronske mreže iz početka, no u praksi je dobro inicijalizirati težinske vrijednosti predtreniranog modela. To je način dodatnog optimiziranja parametara kroz cijelu mrežu.

Moguće je koristiti konvolucijske slojeve iz predtrenirane neuronske mreže. Zbog dijeljenja parametara moguće je iste primijeniti na slikama različitih dimenzija. Također, često se koriste malene stope učenja za CNN kojoj se podešavaju parametri, zbog pretpostavke da su podatci već dobri, samo ih treba još malo doraditi [60].

Za model detekcije ljudi gdje je jedan od najvećih problema malena baza podataka te gdje baza uključuje zračne slike, a većina CNN arhitektura je trenirana na panoramskim slikama, preporuka je da se koriste niži slojevi za ekstrakciju značajki na osnovu kojih se trenira SVM klasifikator. Intuicija iza ovoga je da se mreža u nižim slojevima još uvijek nije prilagodila podatcima za

treniranje nego samo globalnim značajkama slike kao što su linije i rubovi koje neuronska mreža može prepoznati.

CNN su uspješno primijenjene na slikama detekcije ljudi u slučajevima prirodnih katastrofa kao što su lavine. Pristup je koristio učenje prenošenjem kako bi izvukao deskriptore na osnovu kojih je treniran SVM klasifikator kako bi prepoznao ljude na slici, a rad pokazuje rezultate točnosti od 65,71% do 97.59% u kasnijim eksperimentima [61].

4.4. Primjena DetectNet arhitekture u detekciji na zračnim slikama

Kako bi omogućili znanstvenicima brz i jednostavan pristup u testiranju CNN-a na raznim zadatcima, NVIDIA tvrtka je u sklopu svojih istraživanja razvila sustav DIGITS koji u novoj verziji uključuje i jednostavan programski okvir za detekciju - DetectNet. Na slici 4.9. prikazani su rezultati detekcije vozila na zračnim slikama.

Slika 4.9 Detekcija vozila na zračnim slikama upotrebom DetectNet arhitekture[6]

DetectNet je uključen u DIGITS 4 definiciji i treniran je uz pomoć Caffe programskog okvira. U daljnjem teksu opisana je DetectNet arhitektura i načini korištenja u procesu detekcije objekata na zračnim slikama.

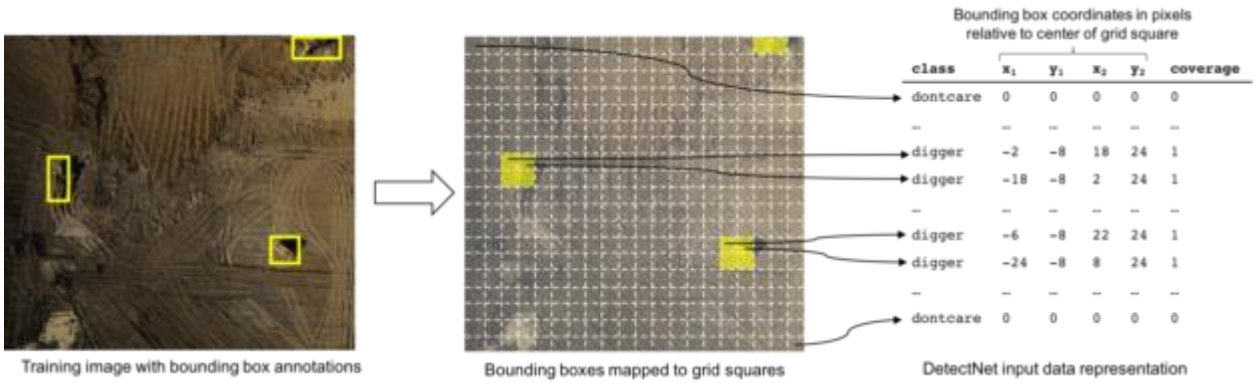
4.4.1. Format podataka DetectNet arhitekture

Skup podataka za treniranje su jednostavno slike ili dijelovi slika koji sadrže jedan objekt označen prema klasi kojoj pripadaju. Budući da detekcija objekata zahtjeva jako puno podataka, skup za

[6] DetectNet: Deep Neural Network for Object Detection in DIGITS
https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/

treniranje u DetectNet arhitekturi su uglavnom slike na kojima su označeni objekti gdje se, osim oznake pripadnosti klasi, na svakom objektu definiraju i koordinate kvadrata na slici koji obuhvaća taj objekt. Budući da broj objekata na slici može varirati od slike do slike, naivni izbor formata označavanja s varijabilnom dužinom i dimenzionalnošću može rezultirati definiranjem funkcije gubitka jako teškim zadatkom.

DetectNet ovaj problem rješava uvođenjem fiksnih 3D oznaka koji omogućava ovoj arhitekturi obradu slika s različitim brojem objekata na slici. DIGITS [62] preko slike stavlja kvadratnu mrežu s razmakom malo manjim od najmanjeg objekta koji se detektira. Svaki kvadratić mreže je označen s dva ključna dijela informacije, a to je klasa objekta i koordinate objekta, koje su relativne u odnosu na sredinu kvadrata. U slučaju gdje objekt nije dostupan, koristi se posebna „dontcare“ oznaka tako da podatci održavaju fiksnu veličinu. Vrijednost prekrivanja (0 ili 1) pokazuje je li objekt prisutan u kvadratiću ili ne. U slučaju gdje se više objekata nalazi u istom kvadratiću, uzima se onaj objekt koji zauzima najveću površinu. Odabir ovih vrijednosti za zračne slike je nasumičan, a puno je značajniji za panoramske slika gdje niža *y* vrijednost objekta znači da je objekt bliži kameri. Format reprezentacije podataka je prikazan na slici 4.10.



Slika 4.10 DetectNet format podataka[7]

Zadatak treniranja za DetectNet je predviđanje reprezentacije podataka za odrezanu sliku, tj. za svaki kvadratić slike DetectNet predviđa je li objekt prisutan i gdje su rubovi u odnosu na sredinu kvadratića mreže.

[7] DetectNet: Deep Neural Network for Object Detection in DIGITS
https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/

Arhitektura DetectNet okruženja sastoji se od više dijelova koji su definirani u Caffe modelu. Tijekom treniranja tri su osnovna koraka koja su jako bitna, a ostala dva se koriste prilikom validacije.

1. Podatkovni slojevi uzimaju podatke za treniranje i oznake te sloj transformacije primjenjuje operacije augmentacije podataka u kojemu se izvršavaju razne transformacije nad objektima kako bi bilo moguće objekt nad kojim se trenira prepoznati u različitim pozicijama (rotiran, translatiran, skaliran i sl.).
2. Potpuna konvolucijska mreža (FCN) obavlja ekstrakciju značajki i predikciju klasa objekata i lokaciju objekta unutar kvadrata.
3. Funkcija gubitka mjeri grešku dvaju zadataka – predikcije i iscrtavanja kvadratića oko detektiranog objekta.
4. Funkcija grupiranja proizvodi završni skup označenih kvadrata
5. Pojednostavljena verzija srednje preciznosti (*engl. mean Average Precision mAP*) računa se u svrhu mjerenja performansi modela.

Također se mogu definirati i parametri u kojoj se definiraju veličine pod područja slike, gdje se svaki put kada se mreža inicijalizira uzima nasumično pod slike definiranih dimenzija. Ovakav pristup može biti jako koristan u slučaju kada imamo velike slike na kojima se nalaze mali objekti.

```
detectnet_augmentation_param {
    crop_prob: 1.0
    shift_x: 32
    shift_y: 32
    scale_prob: 0.4
    scale_min: 0.8
    scale_max: 1.2
    flip_prob: 0.5
    rotation_prob: 0.0
    max_rotate_degree: 5.0
    hue_rotation_prob: 0.8
    hue_rotation: 30.0
    desaturation_prob: 0.8
    desaturation_max: 0.8
  }
```

U programskom odsječku je prikazan jedan od najbitnijih djelova ove arhitekture, a to je prilagodba podataka. Ovaj proces je jako bitan prilikom treniranja i ovdje se definira doseg do kojeg želimo izvršavati transformacije nad podatcima. Transformacije, kao što su rotacija translacija i zrcaljenje,

primjenjuju se višestruko na slikama. Prednost ovakvog pristupa je da mreža nikad ne vidi istu sliku dva puta pa je robusnija i otpornija na prilagodbu podatcima.

Već spomenuta FCN mreža može prihvatiti slike različitih dimenzija i efektivno primijeniti CNN na principu pomičnog prozora. Definiranje baze podataka izvršava se u KITTI [63] formatu, što je maksimalno olakšava proces treniranja u svrhu definiranja vlastitih zadataka detekcije te izgradnje vlastite baze podataka.

Treniranje DetectNet na skupu podataka od 307 slika s 24 slike za vrednovanje (sve slike dimenzija 1536x1024 piksela) traje 63 minute na Titan X u DIGITS 4 sa NVIDIA Caffe i cuDNN RC 5.1. Za vrednovanje je korišteno Python Layers sučelje za vrednovanje preciznosti te je implementiran način provjere ispravnosti detektiranih objekata prema [64].

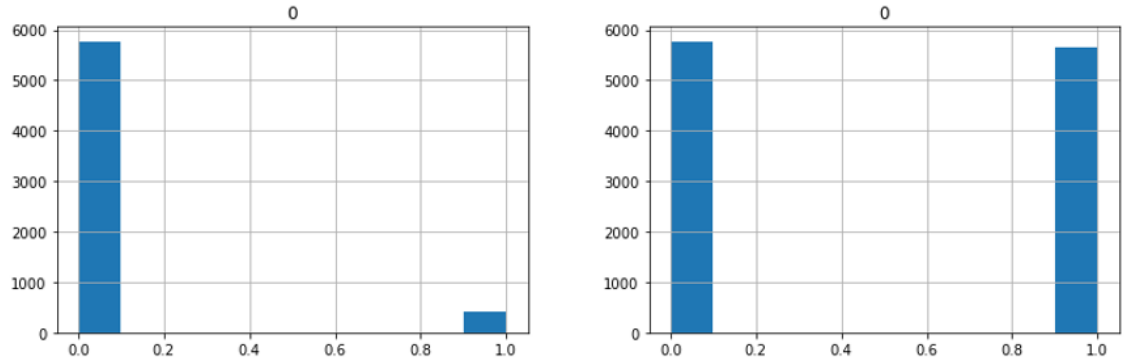
4.5. Primjene konvolucijskih neuronskih mreža u detekciji ljudi na zračnim slikama

Detekcija objekata na slici u suštini se svodi se na klasifikaciju blokova dobivenih razbijanjem slika na dijelove. Proces primjene konvolucijskih neuronskih mreža na slikama koje nisu panoramske uvodi jako velika ograničenja u prepoznavanje objekata na slici. Slike koje su slikane iz zraka se bitno razlikuju od panoramskih slika (panoramske slike su zbog blizine kamere u trenutku slikanja puno detaljnije i jasnije), što onemogućava primjenu predtrenirane neuronske mreže. Za treniranje kvalitetne neuronske mreže potrebno je prvenstveno napraviti jako veliku bazu slika, gdje se javlja problem jer izgradnja baze za treniranje prepoznavanja ljudi na slikama u akcijama portage i spašavanja zahtjeva simuliranje ovih akcija s velikim brojem ljudi u raznim situacijama. Drugi pristup rješavanja ovog problema je iskorištavanje predtrenirane neuronske mreže, no i ovaj pristup ima svoja ograničenja. Budući da su sve neuronske mreže trenirane na panoramskim slikama, koje su slikane iz blizine, samim time objekti na slici su veliki. Na slikama koje su slikane iz zraka objekti na slici mogu biti jako mali, što znači da nema puno detalja i značajki više razine na koje je neuronska mreža naučila. Rješenje ovog problema je treniranje neuronske mreže, ali uz pomoć težinskih vrijednosti iz nižih slojeva predtreniranih modela kao što su VGGNet, GoogLeNet i dr. Stoga je potrebno izgraditi relativno malu bazu, npr. od 1000 pozitivnih i 1000 negativnih primjeraka na kojima se mreža može trenirati. Pojedine slike dimenzija 4000x3000 dijele se na blokove 50x50, što je na slici ovih dimenzija otprilike veličina čovjeka koji leži. Budući da se u postojećoj bazi na slikama nalazi jako malo osoba koji su sudjelovali u izgradnji baze, gotovo je nemoguće izvući dovoljan broj pozitivnih primjera, dok

negativnih primjera ima jako puno. Ovom problemu se također može pristupiti na više načina. Jedan od načina je da se u slučaju kad se pojavljuje nerazmjeran broj pozitivnih i negativnih primjera često primjenjuje metoda pridruživanja težinskih vrijednosti prema formuli 4.4.

$$w_c = \log_2(\mu * \frac{t}{x_c}) \qquad (4.4)$$

Parametar w_c predstavlja vrijednost težine određene klase, t je ukupan broj svih primjeraka, a x_c je broj primjeraka u trenutnoj klasi. μ je proizvoljni parametar koji je eksperimentalnim putem postavljen na vrijednost 0.25. Ovaj pristup u izjednačavanju klasa pokazao je relativno mala poboljšanja upravo zbog nerazmjernosti klasa što prilikom treniranja vodi prema tome da se neuronska mreža pretjerano prilagođava negativnoj klasi. Kako bi se ovaj problem riješio, primijenjena je tehnika pred procesiranja koja vrši razne transformacije pozitivnih i negativnih slika, kao što je translacija, rotacija i skaliranje, a u svrhu povećanja blokova za treniranje. Budući da ni ovaj pristup nije pokazao značajna poboljšanja i da se sustav se opet pretjerano prilagođavao negativnoj klasi, jedino rješenje je povećavanje baze ili korištenje već naučenih modela. Učenjem prenošenjem moguće je uzeti model koji je osposobljen za veliki skup podataka i treniran za neki posve drugačiji zadatak, s istim ulazom, ali različitim izlazom. Nakon toga se pronalaze slojevi čije se značajke mogu iskoristiti. Izlaz tog sloja može se koristiti kao ulazna značajka za treniranje mnogo manje mreže koja zahtjeva manji broj parametara. Ova metoda korištena je za treniranje modela za prepoznavanje ljudi na zračnim slikama. Budući da su ulazni podatci za treniranje između predtreniranog modela i baze zračnih slika jako različiti, cilj je bio preuzeti težinske vrijednosti iz nižih slojeva te ih iskoristiti za daljnje treniranje vlastite neuronske mreže. Ovaj pristup je omogućio da se više raspoznaje pozitivna klasa u odnosu na negativnu, no ni to nije pomoglo u pronalaženju razlika između pozitivne u negativne klase (slika 4.11). Na kraju, bilo je potrebno duplicirati pozitivnu klasu kako bi se izjednačio broj pozitivnih i negativnih primjeraka, što je postignuto uz pomoć translacije, rotacije i skaliranja.



Slika 4.11 Odnos pozitivnih i negativnih blokova na slici prije i nakon izjednačavanja

U niže slojeve neuronske mreže prenešene su težinske vrijednosti iz nižih slojeva VGGNet mreže, preciznije iz slojeva 1, 2, 3 i 4. Mreža se sastoji od ukupno 5 slojeva pri čemu prvi sloj prima blokove veličine 50x50. Prvi konvolucijski sloj sastoji se od 5 različitih slojeva s pomakom 3x3 prozora i RELU (*engl. Rectified Linear Unit*) aktivacijom koja svaku vrijednost manju od nule postavlja nan nulu. Slijedi maxpool sloj veličine 2x2 s pomakom 2x2 piksela, a nakon toga niz konvolucijskih i maxpool slojeva. Cijela arhitektura neuronske mreže može se pronaći u prilogu. Pojedini slojevi su „obogćeni“ s vrijednostima nižih slojeva VGG mreže. U svrhu detekcije, cijela slika od 12MP podijeljena je na blokove 50x50 nakon čega je primijenjena trenirana neuronska mreža. Mreža je istrenirana na NVIDIA Titan X grafičkoj kartici, a process treniranja trajao je jedan dan. Za izgradnju modela korištena je programska podrška Keras[8] u programskom jeziku Python[9] i TensorFlow GPU[10] verzija kao pozadina za Keras. Nakon toga istrenirani model je iskorišten za prepoznavanje objekata na slici , a rezultat testiranja detekcije prikazan je na slici 4.12.

[8] https://keras.io/ – Keras: The Python Deep Learning library
[9] https://python.org/ – Python programming language
[10] https://www.tensorflow.org/ - An open source software library for Machine intelligence

Slika 4.12 Rezultat testiranja detekcije

5. ZAKLJUČAK

Detekcija ljudi na zračnim slikama je jako složen zadatak. Moglo bi se čak i reći da je to jedan od najsloženijih zadataka računalnog vida. Ne samo zato što postoje praktična ograničenja za razvijanje robusnog sustava, nego i zato što ovaj zadatak nije lagan ni za ljude pa nije realno očekivati da nešto što je teško čak i ljudima, bude automatizirano. Zahvaljujući naprecima tehnologije, prvenstveno u razvoju robotike, a posebno bespilotnih zračnih letjelica (UAV) koje sa svojim tehničkim specifikacijama danas dosežu rezoluciju slike i do 100px/m^2, razvijaju se sve robusniji sustavi koji predstavljaju velik napredak u ovom polju. Velike rezolucije slika omogućavaju i razvoj sofisticiranijih metoda za raspoznavanje i detekciju uz pomoć strojnog učenja. Klasični pristupi zadatcima prepoznavanja, klasifikacije i detekcije zahtijevaju ručno definirane značajke, na koje se nakon smanjenje dimenzionalnosti i pronalaska svojstvenih vektora, primjenjuju određeni klasifikatori kako bi se izvršila implementacija i vrednovanje spomenutih zadataka. Prema dostupnoj literaturi, najčešće korištena konvencionalna metoda za klasifikaciju je korištenje HOG značajki u kombinaciji sa SVM klasifikatorom. No, jako je malo pristupa konkretnom problemu detekcije ljudi u operacijama potrage i spašavanja, što ostavlja jako puno prostora za poboljšanje. Konvencionalne metode strojnog učenja u ovakvim situacijama, gdje su objekti na slikama jako mali (objekt na slici rezolucije 4000x3000px može zauzimati prostor veličine tek 50x50px) nailaze na svojevrsna ograničenja. Također, ne postoji univerzalno pravilo po kojemu bi bilo moguće definirati značajke jer još nije u potpunosti jasno na osnovu čega uopće čovjek prepoznaje takve objekte na slici.

Plan daljnjeg rada je upotreba konvolucijskih neuronskih mreža u navedenom problemu, jer svojim uspjehom u raznim zadatcima pokazuju obećavajuće rezultate. Kao što je vidljivo iz prethodnih poglavlja, CNN su uvele revoluciju u području računalnog vida. Već pojavom prve arhitekture pojavili su se zapanjujući rezultati koje je bilo gotovo nemoguće postići tradicionalnim metodama strojnog učenja. Najnoviji rezultati na natjecanjima pokazuju čak i veću preciznost od one koju na istim natjecanjima postiže čovjek (čovjek postiže stopu pogreške od 5% do 10%, a sustav koji je pobijedio na ILSVRC2016 natjecanju je ResNet sa zapanjujućim rezultatom stope pogreške od 3.6%), što pokazuje da se konvolucijske neuronske mreže mogu iskoristiti i u ovom području. Štoviše, konvolucijske neuronske mreže danas se sve više i više kombiniraju sa zračnim i satelitskim snimkama i ostvaruju zadivljujuće rezultate. Dokaz te činjenice je i programski okvir

za detekciju, DetectNet. Osnovni problem u ovom pristupu strojnog učenja je nedostatak baze podataka, stoga je prvenstveni cilj za nastavak istraživanja izrada sustava za označavanje slika te izvoz istih u KITTI format, a nakon toga testiranje i vrednovanje detekcije na vlastitoj označenoj bazi podataka. Također, nužno je dodatno istraživanje mogućnosti primjene konvolucijskih neuronskih mreža na bazi podataka s različitim slikama te bolji pristup lokalizaciji i segmentaciji objekata na slici. Prijedlog je korištenje blokova 250x250 piksela nad kojima bi se upotrebom različitih algoritama za segmentaciju i primjenom neuronske mreže istrenirane na sličnim slikama ostvarila bolja preciznost u prepoznavanju ljudi na zračnim slikama.

6. LITERATURA

[1] F. S. Leira, T. A. Johansen, and T. I. Fossen, "Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera," in *2015 IEEE Aerospace Conference*, 2015, pp. 1–10.

[2] Shaodan Li *et al.*, "Unsupervised Detection of Earthquake-Triggered Roof-Holes From UAV Images Using Joint Color and Shape Features," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 9, pp. 1823–1827, 2015.

[3] S. Gotovac, V. Papic, and Z. Marusic, "Analysis of saliency object detection algorithms for search and rescue operations," in *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2016, pp. 1–6.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," *Proc. Seventh IEEE Int. Conf. Comput. Vis.*, vol. 2, 1999.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," Springer, Berlin, Heidelberg, 2010, pp. 778–792.

[7] B. Triggs and N. Dalal, "Histograms of Oriented Gradients for Human Detection," in *CVPR 2005. IEEE Computer Society Conference*, 2005.

[8] T. Moranduzzo and F. Melgani, "A SIFT-SVM method for detecting cars in UAV images," in *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2012, pp. 6868–6871.

[9] J. Senthilnath, A. Dokania, M. Kandukuri, R. K.N., G. Anand, and S. N. Omkar, "Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV," *Biosyst. Eng.*, vol. 146, pp. 16–32, 2016.

[10] C. Vincenzo, "Image Stitching for UAV remote sensing application," 2011.

[11] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, vol. 2, no. 6.

[12] J. V. Stone and J. V., *Independent component analysis : a tutorial introduction*. MIT Press, 2004.

[13] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 886–893.

[14] S. Tuermer, J. Leitloff, P. Reinartz, and U. Stilla, "Automatic vehicle detection in aerial image sequences of urban areas using 3d hog features", 2010

[15] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[16] W. Shao, W. Yang, G. Liu, and J. Liu, “Car Detection from High - Resolution Aerial Imagery Using Multiple Features,” pp. 4379–4382, 2012.

[17] S. Kluckner and H. Bischof, “Semantic classification by covariance descriptors within a randomized forest,” in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 665–672.

[18] V. Mnih and G. E. Hinton, “Learning to Detect Roads in High-Resolution Aerial Images,” Springer, Berlin, Heidelberg, 2010, pp. 210–223.

[19] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[20] M. Laroze, L. Courtrai, and S. Lefèvre, “Human Detection from Aerial Imagery for Automatic Counting of Shellfish gatherers.”

[21] O. Oreifej, R. Mehran, and M. Shah, “Human Identity Recognition in Aerial Images.”

[22] Xinghui Dong, Junyu Dong, and Liang Qu, “Enteromorpha detection in aerial images using support vector machines,” in *2009 IEEE Youth Conference on Information, Computing and Telecommunication*, 2009, pp. 299–302.

[23] P. (Peter) McCullagh and J. A. Nelder, *Generalized linear models*. Chapman and Hall, 1989.

[24] Adam Van Etten, “Histogram of Oriented Gradients (HOG) Boat Heading Classification,” 2016. [Online]. Available: https://medium.com/the-downlinq/histogram-of-oriented-gradients-hog-heading-classification-a92d1cf5b3cc, Pristupljeno: 10.05.2017.

[25] L. Breiman and Leo, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[26] P. Dollar, Z. Zhuowen Tu, and S. Belongie, “Supervised Learning of Edges and Object Boundaries,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR’06)*, 2006, vol. 2, pp. 1964–1971.

[27] J. Porway, K. Wang, B. Yao, and Song Chun Zhu, “A hierarchical and contextual model for aerial image understanding,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[28] J. Music, T. Marasovic, V. Papic, I. Orovic, and S. Stankovic, “Performance of Compressive Sensing Image Reconstruction for Search and Rescue,” *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 11, pp. 1739–1743, Nov. 2016.

[29] H. Turić, V. Papić, and H. Dujmić, “A procedure for detection of humans from long distance images,” in *Institute of Electrical and Electronics Engineers. Region 8. Institute of Electrical and Electronics Engineers. Croatia Section. European Association for Speech, Signal and Image Processing.*, 2008, p. 654.

[30] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. Morgan Kaufmann Publishers Inc., pp. 674–679, 1981.

[31] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[32] S. Kamate and N. Yilmazer, “Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles,” *Procedia Comput. Sci.*, vol. 61, pp. 436–441, 2015.

[33] T. Moranduzzo, F. Melgani, M. L. Mekhalfi, Y. Bazi, and N. Alajlan, “Multiclass Coarse Analysis for UAV Imagery,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 12, pp. 6394–6406, 2015.

[34] S. S. Haykin and Simon, *Neural networks : a comprehensive foundation*. Prentice Hall, 1999.

[35] M. A. Nielsen, “Neural Networks and Deep Learning.” Determination Press, 2015.

[36] Franck Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para - F. Rosenblatt - Google Knjige*. 460-461: Cornell Aeronautical Laboratory, 1957.

[37] D. P. Bertsekas and D. Bertsekas, *Nonlinear programming*, 2nd ed. Athena Scientific, 1999.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[39] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” pp. 1097–1105, 2012.

[41] C. Szegedy *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[43] Andrew Tao, Jon Barker, and Sriya Sarathy, “DetectNet: Deep Neural Network for Object Detection in DIGITS,” 2016., S Interneta, https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/, Pristupljeno: 02.05.2017

[44] Y. Jia *et al.*, “Caffe: Convolutional Architecture for Fast Feature Embedding,” in *Proceedings of the ACM International Conference on Multimedia - MM ’14*, 2014, pp. 675–678.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” 2009.

[46] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” 2013.

[47] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *IEEE Conference on Computer Vision and*

Pattern Recognition, 2014, pp. 580–587.

[49] R. Girshick, "Fast R-CNN," 2015.

[50] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015.

[51] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, 1998.

[52] R. Vaillant, "Original approach for the localisation of objects in images," *IEE Proc. - Vision, Image, Signal Process.*, vol. 141, no. 4, p. 245, 1994.

[53] P. Sermanet *et al.*, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *HTTP://ARXIV.ORG/ABS/1312.6229*, 2013.

[54] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[55] S. Zheng *et al.*, "Conditional Random Fields as Recurrent Neural Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1529–1537.

[56] A. Zeggada, F. Melgani, and Y. Bazi, "A Deep Learning Approach to UAV Image Multilabeling," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 694–698, 2017.

[57] R. F. Nogueira, R. de Alencar Lotufo, and R. Campos Machado, "Fingerprint Liveness Detection Using Convolutional Neural Networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1206–1213, 2016.

[58] N. Tajbakhsh *et al.*, "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[59] L. Y. Pratt, "Discriminability-Based Transfer between Neural Networks." pp. 204–211, 1993.

[60] "CS231n Convolutional Neural Networks for Visual Recognition." , S Interneta, http://cs231n.github.io/transfer-learning, Pristupljeno: 08.05.2017

[61] M. B. Bejiga, A. Zeggada, and F. Melgani, "Convolutional neural networks for near real-time object detection from UAV imagery in avalanche search and rescue operations," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 693–696.

[62] "NVIDIA DIGITS | NVIDIA Developer.", S Interneta, https://developer.nvidia.com/digits, Pristupljeno: 11.05.2017

[63] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[64] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing Error in Object Detectors."

PRILOG – TABLICA SLIKA

Slika 2.1 Prikaz zračne slike broda i značajki izvučenih u obliku HOG deskriptora *6*
Slika 3.1 Sigmoid funkcija *12*
Slika 3.2 Arhitektura obične neuronske mreže s jednim skrivenim slojem i sa dva ulaza i dva izlaza koja su potpuno povezana s oba sloja *13*
Slika 4.1 Arhitektura konvolucijske neuronske mreže (CNN) *19*
Slika 4.2 Primjer filtera naučenih na AlexNet arhitekturi *22*
Slika 4.3 Prikaz alexNet Arhitekture i izlaza iz pojedinih konvolucijskih slojeva *23*
Slika 4.4 Deconvnet prikaz konvolucijskih slojeva na ZF Net arhitekturi *24*
Slika 4.5 Prikaz paralelnog dijela GoogLeNet arhitekture *25*
Slika 4.6 Arhitektura R-CNN konvolucijske neuronske mreže *27*
Slika 4.7 Treniranje DetectNet *29*
Slika 4.8 Prikaz primjene MAXPOOOL operacije nad nepodudarnom regijom *31*
Slika 4.9 Detekcija vozila na zračnim slikama upotrebom DetectNet arhitekture *35*
Slika 4.10 DetectNet format podataka *36*
Slika 4.11 Odnos pozitivnih i negativnih blokova na slici prije i nakon izjednačavanja *40*
Slika 4.12 Rezultat testiranja detekcije *41*

PRILOG – ARHITEKTURA KONVOLUCIJSKE NEURONSKE MREŽE

Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 50, 50, 64)	1792
block1_conv2 (Conv2D)	(None, 50, 50, 64)	36928
conv2d_19 (Conv2D)	(None, 50, 50, 64)	36928
conv2d_20 (Conv2D)	(None, 50, 50, 64)	36928
conv2d_21 (Conv2D)	(None, 50, 50, 64)	36928
block1_pool (MaxPooling2D)	(None, 25, 25, 64)	0
block2_conv1 (Conv2D)	(None, 25, 25, 128)	73856
block2_conv2 (Conv2D)	(None, 25, 25, 128)	147584
conv2d_22 (Conv2D)	(None, 25, 25, 128)	147584
conv2d_23 (Conv2D)	(None, 25, 25, 128)	147584
block2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
block3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
block3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv3 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv4 (Conv2D)	(None, 12, 12, 256)	590080
block3_pool (MaxPooling2D)	(None, 6, 6, 256)	0
block4_conv1 (Conv2D)	(None, 6, 6, 512)	1180160
block4_conv2 (Conv2D)	(None, 6, 6, 512)	2359808
block4_conv3 (Conv2D)	(None, 6, 6, 512)	2359808
block4_conv4 (Conv2D)	(None, 6, 6, 512)	2359808
conv2d_24 (Conv2D)	(None, 6, 6, 512)	2359808
conv2d_25 (Conv2D)	(None, 6, 6, 512)	2359808
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 512)	0
dense_5 (Dense)	(None, 3, 3, 1000)	513000
dropout_3 (Dropout)	(None, 3, 3, 1000)	0

```
_________________________________________________________________
flatten_3 (Flatten)          (None, 9000)              0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 9001
=================================================================
Total params: 16,232,721
Trainable params: 16,232,721
Non-trainable params: 0
_________________________________________________________________
block1_conv1
block1_conv2
block1_pool
block2_conv1
block2_conv2
block2_pool
block3_conv1
block3_conv2
block3_conv3
block3_conv4
block3_pool
block4_conv1
block4_conv2
block4_conv3
block4_conv4
Epoch 1/10
10272/10272 [==============================] - 4006s - loss: 0.5528 - acc:
0.7223 - val_loss: 0.2385 - val_acc: 0.9375
Epoch 2/10
10272/10272 [==============================] - 4054s - loss: 0.2597 - acc:
0.9069 - val_loss: 0.0597 - val_acc: 1.0000
Epoch 3/10
10272/10272 [==============================] - 3844s - loss: 0.2146 - acc:
0.9294 - val_loss: 0.1052 - val_acc: 0.9688
Epoch 4/10
10272/10272 [==============================] - 3787s - loss: 0.2046 - acc:
0.9329 - val_loss: 0.1079 - val_acc: 0.9375
Epoch 5/10
10272/10272 [==============================] - 4658s - loss: 0.1974 - acc:
0.9374 - val_loss: 0.1167 - val_acc: 0.9375
Epoch 6/10
10272/10272 [==============================] - 3912s - loss: 0.1739 - acc:
0.9448 - val_loss: 0.0517 - val_acc: 0.9688
Epoch 7/10
10272/10272 [==============================] - 3997s - loss: 0.1709 - acc:
0.9447 - val_loss: 0.0435 - val_acc: 1.0000
Epoch 8/10
10272/10272 [==============================] - 4055s - loss: 0.1655 - acc:
0.9482 - val_loss: 0.1025 - val_acc: 0.9688
Epoch 9/10
10272/10272 [==============================] - 4067s - loss: 0.1662 - acc:
0.9472 - val_loss: 0.1820 - val_acc: 0.9375
Epoch 10/10
10272/10272 [==============================] - 3845s - loss: 0.1607 - acc:
0.9495 - val_loss: 0.0428 - val_acc: 1.0000
```