Kvantifikacija stabilnosti lokalizacije objekta pravokutnim okvirom u problemu detekcije objekta

Torbarina, Roko

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Roko Torbarina

KVANTIFIKACIJA STABILNOSTI LOKALIZACIJE OBJEKTA PRAVOKUTNIM OKVIROM U PROBLEMU DETEKCIJE OBJEKTA

Diplomski rad

Voditelj rada:
prof. dr. sc. Luka Grubišić

Zagreb, rujan, 2023.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvala
Petri i Vedranu na teorijskoj pomoći
Domagoju na +1
Roditeljima na svemu

Sadržaj

Uvod

Detekcija objekata je fundamentalni zadatak u području strojnoga vida. S primjenama u automobilskoj, medicinskoj, vojnoj, i brojnim drugim industrijama, sposobnost računalnih sustava da prepoznaju i lociraju objekte iznimno je važno.

U automobilskoj industriji, detekcija objekata ključna je za razvoj autonomnih vozila. Sposobnost prepoznavanja pješaka, vozila, biciklista i drugih objekata na cesti omogućuje vozilima da donose informirane odluke i reagiraju na okolinu.

Detekcija objekta sastoji se od dva ključna zadatka: klasifikacije i lokalizacije. Dok se klasifikacija bavi određivanjem vrste objekta koji se nalazi na slici, lokalizacija precizno određuje granice okvira koji obuhvaća objekt. Ovaj okvir, poznat kao pravokutni ili granični okvir, pomaže detekcijskim modelima da ne samo prepoznaju prisutnost objekta, već i da ga točno pozicioniraju u prostoru slike.

Najbolji rezultati detekcije objekata postižu se neuronskim mrežama. U radu su najprije objašnjene neuroneske mreže i njihove najbitnije komponente. Zatim je u drugom poglavlju raspisan kratak uvod u problem detekcije objekta. U trećem poglavlju je opisana važnost kvantifikacije ne samo pravokutnog okvira objekta, nego i njegove stabilnosti. U posljednjem poglavlju opisana je i testirana jedna od metoda kvantifikacije stabilnosti pravokutnog okvira objekta. Cilj rada je unaprijediti razumijevanje i kvalitetu detekcije objekta, s fokusom na njegovu stabilnost u različitim scenarijima. Uz ovaj rad, objavit ćemo i kodove kojima se mogu reporoducirati izabrani ogledni primjeri. Kod će biti dostupan kroz repozitorij `https://github.com/082T/KvantifikacijaStabilnostiOgledniPrimjer`.

Poglavlje 1

Uvod u neuronske mreže

Umjetne neuronske mreže ili neuronske mreže (eng. *Artificial Neural Networks* - ANN) su složeni matematički modeli koji se koriste za rješavanje raznih problema u strojnom učenju. Inspirirani ljudskim živčanim sustavom, ovi modeli revolucionalizirali su računalnu obradu podataka i donošenje inteligentnih odluka. Simulirajući ponašanje međusobno povezanih neurona, neuronske mreže pružaju rješenja za brojne probleme u svakodnevnom životu. Neke od njihovih najčešćih primjena su u autonomnoj vožnji, medicinskoj dijagnostici, obradi prirodnog jezika, personalizaciji preporuka i oglasa, optimizaciji potrošnje sredstava, financijskoj alazi i mnogim drugima.

Prvi koncept neuronskih mreža pojavljuje se 1943. godine, objavom članka "A Logical Calculus of the ideas Imminent in Nervous Activity" neurofiziologa Warren McCulloch i matematičara Water Pitts [13]. Petnaest godina kasnije, 1958. godine, psiholog Frank Rosenblatt je izumio prvu umjetnu neuronsku mrežu, zvanu Perceptron, koja je bila kreirana s namjerom uporabe u prepoznavanju uzoraka na slikama poput linija i krivulja [17].

Unatoč lošijim rezultatima u početcima razvoja neuronskih mreža, zahvaljujući novijim doprinosima u tom području, neuronske mreže su postale zanačajan faktor u napretku strojnog učenja i umjetne inteligencije. Ključni trenutak nastupio je s razvojem višeslojnih perceptrona ili višeslojnih neuronskih mreža, što je omogućilo učinkovito učenje složenih reprezentacija podataka.

Još jedan od faktora koji čini neuronske mreže vodećim pokretačen razvoja strojnog učenja je količina sakupljenih podataka koji se mogu primijeniti za treniranje modela neuronskih mreža.

1.1 Glavne komponente neuronskih mreža

Neuroni i slojevi neuronske mreže

Neuroni su osnovne gradivne jedinice neuronskih mreža. Njihovo ponašanje je inspirirano načinom na koji rade neuroni u ljudskom mozgu. Najčešće su poredani po slojevima te su svojim poveznicama povezani sa neuronima idućega sloja. U neuronskim mrežama postoje tri vrste slojeva:

Ulazni sloj - Sloj koji dobiva izvorne podatke ili značajke koje se trebaju procesuirati kroz neuronsku mrežu.

Skriveni slojevi - Slojevi na kojima se odvija obrada podataka. Neuroni u ovim slojevima primjenjuju težine na svoje ulaze, zbrajaju težinske ulaze i rezultat procesuiraju kroz aktivacijsku funkciju.

Izlazni sloj - Sloj koji proizvodi konačne rezultate računanja neuronske mreže. Također vrši izračune kao i skriveni slojevi.

Opća neuronska jedinica može se opisati sljedećom formulom:

$$a = f\left(\sum_i w_i x_i + b\right), \tag{1.1}$$

gdje su x_i ulazi u neuron, w_i njihove težine, b pomak, f nelinearna aktivacijska funkcija te a vrijednost neuronske aktivacije. Više o težinama, pomaku i aktivacijskim funkcijama u nastavku poglavlja.

Težina i pomak

Svaki ulaz koji pristupa neuronu unutar mreže množi se s karakterističnim skalarom poznatim kao težina. Tokom procesa treniranja, osnovna svrha neuronske mreže je optimizirati ove težine kako bi izlaz mreže što bolje odgovarao željenim rezultatima. Kroz iterativni proces prilagodbe težina, neuronska mreža postupno konvergira prema boljem modeliranju stvarnih podataka i rješavanju specifičnih zadataka.

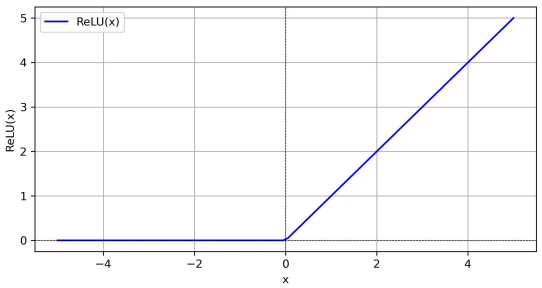
Uz težine, svaki neuron u neuronskoj mreži ima dodatni parametar poznat kao pomak (eng. *bias*). Dodavanje pomaka omogućava pomak aktivacijske funkcije ulijevo ili udesno, što pruža mreži dodatnu fleksibilnost i prilagodljivost. Pomak je iznimno koristan u situacijama gdje se želi naglasiti ili smanjiti utjecaj određenih aspekata ulaznih podataka, primjerice u problemu klasifikacije gdje se ispravnim podešavanjem pomaka mreža može naučiti pridodati više pažnje manjinskoj klasi te time ostvariti bolja uspješnost.

Aktivacijske funkcije

Aktivacijske funkcije omogućuju složenu i nelinearnu obradu podataka. Primjenjuju se na izlazne vrijednosti pojedinog neurona kako bi se odredilo treba li taj neuron biti aktiviran ili ne na temelju rezultata težinske sume njegovih ulaznih vrijednosti. Bez aktivacijskih funkcija, neuronske mreže bi se svodile samo na jednostavne linearno-parametarske modele s ograničenim sposobnostima obrade podataka. Dodatno, prisutnost aktivacijskih funkcija omogućava efikasno dodavanje skrivenih slojeva, što je ključno za rješavanje složenih problema i postizanje dubokog učenja.

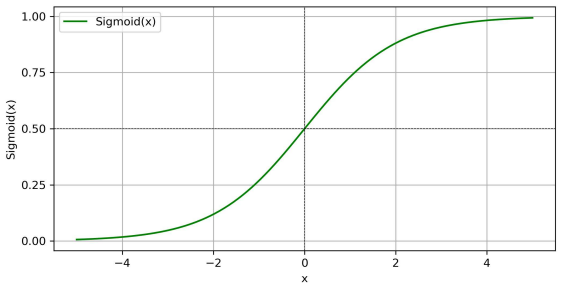
Neke od najpopularnih aktivacijskih funkcija su:

ReLU (eng. *Rectified Linear Unit*): $ReLU(x) = max(0, x)$



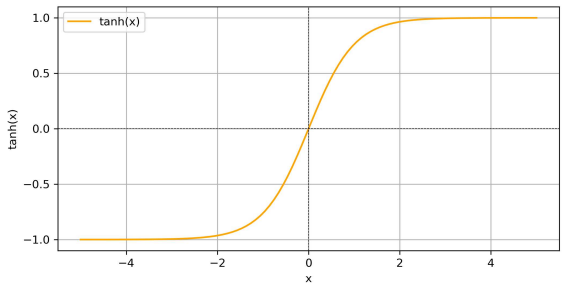
Slika 1.1: Aktivacijska funkcija ReLU

Sigmoida: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$



Slika 1.2: Aktivacijska funkcija sigmoida

Tangens hiperbolni: $tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} + 1}{e^{2x} - 1}$



Slika 1.3: Aktivacijska funkcija tangens hiperbolni

Funkcija gubitka i trening neuronske mreže

Funkcija gubitka je funkcija koja uspoređuje stvarne vrijednosti ulaza s predviđenim vrijednostima dobivenim prolaskom ulaza kroz neuronsku mrežu. Odabir funkcije gubitka uvelike ovisi o problemu koji se pokušava riješiti. a dva najčešća problema su: problem klasifikacije i problem regresije.

Cilj problema klasifikacije je predviđanje kojoj klasi zadani ulaz pridapa. Budući da su moguće klase uglavnom unaprijed poznate, neuronska mreža se gradi tako da na izlazu ima broj čvorova jednak broju klasa. Najčešće korištena funkcija gubitka u klasifikacijskim problemima je Gubitak unakrsne entropije. Njome se mjeri uspješnost modela čije su vrijednosti pripadanja pojedinoj klasi između 0 i 1, stoga se najčešće koristi sa *softmax* aktivacijskom funkcijom na izlaznom sloju koja skalira vrijednosti u traženi interval. Gubitak unakrsne entropije definiran je na idući način:

$$CE = -\frac{1}{m} \sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i) \tag{1.2}$$

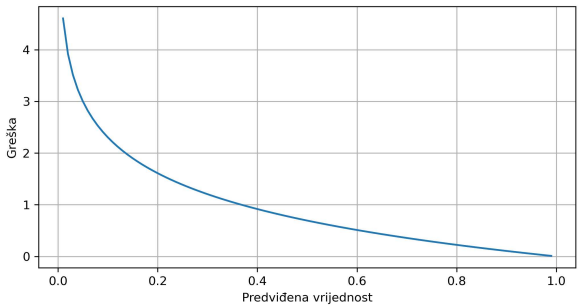
gdje je m broj klasa, $y_i \in \{0, 1\}$ stvarna vrijednost i-te klase, $\hat{y}_i \in [0, 1]$ predviđena vrijednost i-te klase. Vrijednosti u slučaju binarne klasifikacije prikazani su slikom 1.4.

Rješavanje problema regresije je predviđanje neprekidne numeričke vrijedosti na temelju ulaznih podataka. U tu svrhu, broj izlaznih neurona postavlja se na dimenziju prostora u kojem se nalaze tražene vrijednosti. Primjerice, u zadatku predviđanja cijene nekretina, cilj je predvidjeti samo jednu vrijednost te je stoga dovoljan jedan izlazni neuron s ciljem da njegova vrijednost bude što točnija stvarnoj cijeni nekretnine.

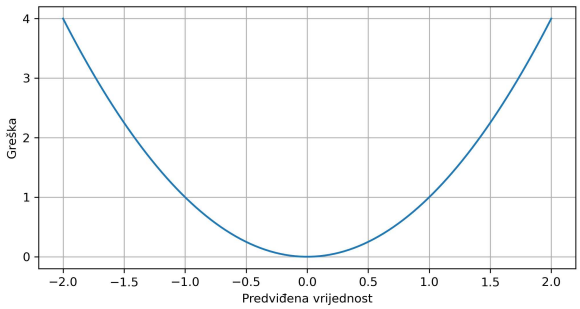
Najčešća funkcija gubitka korištena u problemima regresije je Srednja kvadratna greška (eng *Mean Squared Error*). Prikazana je slikom 1.5 te je definirana na idući način:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2, \tag{1.3}$$

gdje n predstavlja dimenziju izlaza, Y_i stvarnu vrijednost i-tog neurona a $\hat{Y}_i$ predviđenu vrijednost i-tog neurona.



Slika 1.4: Gubitak binarne unakrsne entropije za stvarnu vrijednost 1



Slika 1.5: Srednja kvadratna greška za stvarnu vrijednost 0

1.2 Učenje neuronske mreže

Učenje neuronske mreže je iterativni postupak podešavanja težina veza između neurona. Tijekom učenja, cilj kreiranog modela je minimizirati funkciju gubitka. U ovome poglavlju će se opisati pojmove i metode vezane za proces učenja.

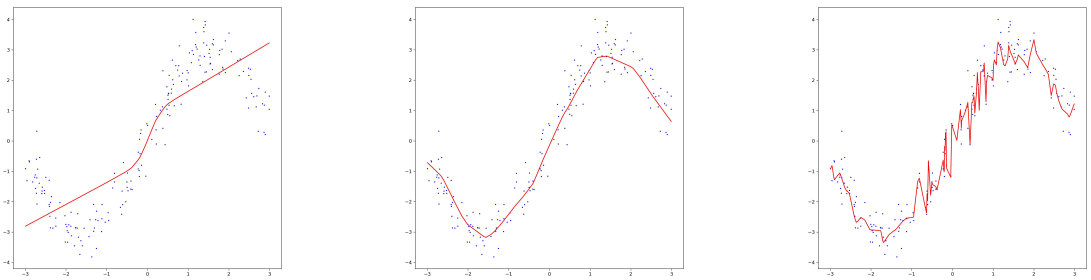
Skup za trening i skup za validaciju

Tijekom treninga neuronske mreže, bitno je razlikovati dva skupa podataka: podatke za trening i podatke za validaciju.

Podatci za trening su skup podataka namjenjen za treniranje neuronske mreže. Tokom treninga neuronske mreže, prolazi se više puta kroz skup podataka za trening. Jedan takav prolazak naziva se epoha. Na temelju rezultata prolaska podataka za trening kroz mrežu, podešavaju se parametri i minimizira funkcija gubitka između stvarnih i predviđenih vrijednosti.

Podatci za validaciju poseban su skup podataka izdvojen za evaluaciju modela. Ti podatci, kao i podatci za treniranje, prolaze kroz mrežu. Na temelju njih mreža ne podešava svoje parametre nego se zaključuje koliko je uspješna u generaliziranju na nove, nepoznate podatke. Također, promatrajući vrijednost funkcije gubitka na validacijskom setu kroz epohe treninga, detektira se pretreniranost (eng. *overfitting*) mreže. Prolazak kroz podatke za evaluaciju tipično se radi nakon svake epohe treninga.

Za model se kaže da je pretreniran ukoliko su njegovi parametri pretočno usklađeni s podatcima za trening. S previše prolazaka kroz podatke za trening, model počne učiti ”šum” u podatcima i fokusirati se na nerelevantne informacije. Tim postupkom, model postepeno gubi mogučnost generalizacije te počne gubiti točnost na validacijskom skupu. Na slici 1.6 prikazani su primjeri mreže kroz različite faze treninga.



Slika 1.6: Na lijevoj strani nalazi se primjer podtrenirane mreže, u sredini dobro istrenirane mreže te pretrenirane mreže na desnoj strani

Najpopularniji odabiri skupa za validaciju:

- **Validacija uz zadržavanje -** Validacija uz zadržavanje (eng. *Holdout*) je najjednostavnija metoda odabira validacijskog skupa gdje se skup podataka nasumično podijeli na skup za trening i skup za validaciju u odabranom omjeru. Taj omjer je najčešće 70:30 ili 80:20. Brzo se implementira i pogodna je za velike skupove podataka.
- **K-Struka unakrsna validacija -** K-Struka unakrsna validacija (eng. *K-fold Cross-validation*) vrši se tako da se početni skup podataka podijeli na K podskupova. Model se trenira K puta, svaki put koristeći različiti skup za validaciju. Budući da se svi početni podatci koriste i za trening i za validaciju, ova metoda može proizvesti bolje utrenirane modele. Zbog većih komputacijskih zahtjeva, K-Struka unakrsna validacija je najkorisnija kad je skup podataka relativno malen.
- **Izostavi-jednog unakrsna validacija -** Izostavi-jednog unakrsna validacija (eng. *Leave-one-out Cross-validation*) je poseban slučaj K-struke unakrsne validacije kad je K jednak ukupnom broju podataka. Iako ima najmanju pristranost zbog najvećeg broja podataka u skupu za trening, rijetko je korištena zbog komputacijske zahtjevnosti.

Gradijentni spust

Kako bi se uspješno minimizirala funkcija gubitka, potrebno je pronaći njezin lokalni minimum. Jedna od najpoznatijih i najčešće primjenjena metoda za pronalaženje približnog minimuma naziva se gradijentni spust.

Definicija 1.2.1. *Neka je $n \in \mathbb{N}$. Jakobijeva matrica funkcije $f \in \mathbb{R}^n$ s oznakom J_f je definirana kao:*

$$J_f = \left(\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_1} \right)$$

.

Vektor iz $\mathbb{R}^n$ s istim elementima naziva se gradijent funkcije f i ima oznaku ∇f.

Definicija 1.2.2. *Točke iz otvorenog intervala funkcije f u kojima je gradijent jednak nuli nazivamo stacionarnim točkama funkcije f.*

Definicija 1.2.3. *Točka $c \in \mathbb{R}^n$ je fiksna točka funkcije $f \in \mathbb{R}^n$ ukoliko je $f(c) = c$*

Definicija 1.2.4. *Gradijentni spust je metoda iteracije fiksne točke za preslikavanje* $\psi_\alpha := x - \alpha\nabla f(x)$ *takva da za* $k \in \mathbb{N}$*:*

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

.

Skalari α_k *se biraju tako da* $\|x_{k+1} - x_k\| \to 0$.

Gradijentni spust je iterativna tehnika koja se koristi za postupno približavanje minimumu funkcije gubitka. Osnovni koncept leži u iskorištavanju gradijenta funkcije, koji služi kao kompas ukazujući na najbrži uspon vrijednosti funkcije.

Krenuvši od neke početne točke, računa se gradijent funkcije gubitka u toj točki i pomiće se u suprotnom smjeru tog gradijenta te se tako smanjuje vrijednost funkcije gubitka. Taj korak ponavlja se iterativno, svaki put računajući novi gradijent u novoj točki i pomičući se u odgovarajućem smjeru.

Definicija 1.2.5. *Neka je* f *neprekidna i* $D(f) = Q$*. Kažemo da je* f *neprekidno diferencijabilna na* Q *ako postoje sve njene parcijalne derivacije i neprekidne su za svaki* $x \in Q$*.*

Lema 1.2.6. *O konvergenciji gradijentnog spusta za konveksne funkcije*

Neka je Q *konveksan skup te* f *neprekidno diferencijabilna na* Q*. Neka postoji* $x_* \in D(f)$ *takav da* $\nabla f(x_*) = 0$*. Neka postoji* $\gamma \in [0, 1)$ *takav da je za neki* $\alpha \neq 0$

$$\|\psi_\alpha(x) - \psi_\alpha(y)\| \leq \gamma\|x - y\|,$$

za sve $x, y \in D(f)$*. Tada niz* x_k*, generiran gradijentnim spustom uz* $\alpha_k = \alpha$ *konvergira u* x_* *za svaki* x_0*.*

Dokaz. $\|x_{k+1} - x_*\| = \|(x_k - \alpha\nabla f(x_k)) - (x_* - 0)\| = \|(x_k - \alpha\nabla f(x_k)) - (x_* - \alpha\delta f(x_*)\| = \|\psi_\alpha(x_k) - \psi_\alpha(x_*)\| \leq \gamma\|x_k - x_*\| \leq \gamma^{k+1}\|x_0 - x_*\|$

Puštanjem $k \to \infty$ dobivamo $\gamma^{k+1}\|x_0 - x_*\| \to 0$ [4] □

Propagacija unatrag

Propagacija unatrag je ključna tehnika treniranja neuronskih mreža koja omogućuje prilagodbu težina kako bi se minimizirao gubitak između predviđenih i stvarnih vrijednosti. Algoritam se sastoji od prolaska unaprijed, gdje se izračunavaju izlazi mreže, te prolaska unatrag, gdje se primjenom lančanog pravila računaju gradijenti gubitka. Ti gradijenti se zatim koriste za prilagodbu težina mreže koristeći optimizacijski algoritam poput gradijentnog spusta.

Primjer 1.2.7. *Primjer propagacije unatrag u mreži s 2 ulazna neurona, jednim neuronom u jedinom skrivenom sloju i jednim neuronom u izlaznom sloju:*

Neka je w_{ij}^l težina između i-tog neurona u sloju $l-1$ i j-tog neurona u sloju l. Neka je b_j^l pomak j-tog neurona u sloju l. Radi pojednostavljenja se uzimaju linearne aktivacijske funkcije $a = z$, gdje z predstavlja težinsku sumu ulaska u neuron. Neka je funkcija gubitka MSE.

Prvi korak je prolazak mrežom unaprijed, računajući vrijednosti za ulaz x_1 i x_2:

$$a_1^1 = z_1^1 = w_{11}^1 x_1 + w_{12}^1 x_2 + b_1^1,$$

$$a_1^2 = z_1^2 = w_{11}^2 a_1^1 + b_1^2,$$

gdje je vrijednost a_1^2 dobiveni izlaz iz neuronske mreže.

Zatim se u prolasku unatrag računa osjetljivost s_1^2 greške u izlaznom sloju:

$$s_1^2 = \partial L / \partial a_1^2 = 2(a_1^2 - y_{true})$$

Propagacija osjetljivosti u skriveni sloj je:

$$s_1^1 = w_{11}^2 s_1^2$$

Lančanim pravilom se računaju gradijenti za skriveni sloj:

$$\partial L / \partial w_{11}^2 = a_1^1 s_1^2$$

$$\partial L / \partial b_1^2 = s_1^2$$

Gradijenti za ulazni sloj su:

$$\partial L / \partial w_{11}^1 = \partial L / \partial a_1^2 \cdot \partial a_1^2 / \partial w_{11}^1 = x_1 s_1^1$$

$$\partial L / \partial w_{12}^1 = \partial L / \partial a_1^2 \cdot \partial a_1^2 / \partial w_{12}^1 = x_2 s_1^1$$

$$\partial L / \partial b_1^1 = \partial L / \partial a_1^2 \cdot \partial a_1^2 / \partial b_1^1 = s_1^1$$

Posljednji korak je ažurirati vrijednosti parametara (gradijentni spust):

$$w_{ij}^k = w_{ij}^k - \eta \cdot \partial L / \partial w_{1ij}^k,$$

$$b_i^j = b_i^j - \eta \cdot \partial L / \partial b_i^j,$$

gdje je η parametar dan za brzinu učenja (eng. *learning rate*). Težine i pomaci se višestrukim pozivima propagacije unatrag podešavaju kako bi se postigla optimizacija modela na skupu za treniranje.

Poglavlje 2

Problem detekcije objekata

2.1 Konvolucijske neuronske mreže

Slike u digitalnom formatu obično su reprezentirane kao rešetka piksela, gdje svaki piksel ima svoju boju i informaciju o njenom intenzitetu. U praksi, to bi značilo da je slika reprezentirana kao trodimenzionalni vektor oblika (w, h, c), gdje je h visina slike (broj redova u pikselima), w širina slike (broj stupaca) te c N-dimenzionalni vektor kanala boje. Ovisno o formatu u kojem je slika spremljena, c se najčešće sastoji od jednog ili tri kanala. U slučaju crno-bijele slike, koristi se samo jedan kanal koji označava intenzitet piksela. U slučaju slike u boji, najčešće se koristi RGB format u kojemu tri kanala reprezentiraju intenzitet crvene, zelene i plave boje na pikselu. Vrijednosti intenziteta obično su cijeli brojevi iz skupa $\{0, 1, \ldots, 255\}$.

Budući da tradicionalne neuronske mreže svoje ulazne podatke reprezentiraju u obliku N-dimenzionalnog vektora, nisu optimalne za opisivanje strukturiranih podataka poput slika gdje prostorna udaljenost u ulaznim podatcima igra veliku ulogu. Kako bi se riješio navedeni problem, razvijene su konvolucijske neuronske mreže (CNN).

Konvolucijske neuronske mreže posebna su arhitektura dubokih neuronskih mreža dizajnirana s ciljem efikasne obrade slika. U CNN-ima, ulaz prolazi kroz jedan ili više konvolucijskih slojeva koji primjenjuju postupak diskretne konvolucije nad ulaznim podatcima.

Konvolucija i konvolucijski slojevi

Definicija 2.1.1. *Neka su je F polje i $n, m \in \mathbb{N}$. Neka su $A, B \in M_{n,m}$ nad F. Frobeniusov (unutarnji) produkt matrica A i B, s oznakom $(A \circ B)_F$ definiramo kao:*

$$(A \circ B)_F = tr(B^T A)$$

Definicija 2.1.2. *Neka su $m, n, p, q \in \mathbb{N}$ takvi da je $p \leq m, q \leq n$. Neka je $A \in M_{m,n}$ i $B \in M_{p,q}$. Kažemo da je matrica $C = (A * B) \in M_{m-p+1,n-q+1}$ dobivena konvolucijom matrice A s filterom B ako vrijedi:*

$$c_{i,j} = (A_{i,j} \circ B)_F, \quad \forall\ 1 \leq i \leq m - p + 1, 1 \leq j \leq n - q + 1$$

$$gdje\ je\ A_{i,j} = \begin{bmatrix} a_{i,j} & \dots & a_{i,j+q} \\ \vdots & \ddots & \vdots \\ a_{i+p,j} & \dots & a_{i+p,j+q} \end{bmatrix}$$

Dobivena matrica imat će $m - p + 1$ redaka i $n - q + 1$ stupaca.

Primjer 2.1.3. *Konvolucija matrice 3x3 filterom uobičajeno korištenim za detekciju rubova na slici.*

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 2 & 4 & 2 \\ 7 & 1 & 2 & 3 & 4 \\ 3 & 3 & 3 & 4 & 3 \\ 1 & 2 & 1 & 3 & 5 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 9 & -5 & 5 \\ -12 & -1 & -2 \\ 3 & 2 & 4 \end{bmatrix}$$

Konvolucijski sloj mreže je sloj koji prima N-kanalnu matricu kao ulaz te na nju primjenjuje niz konvolucijskih filtera. Konvolucijski filter može se zamisliti kao okvir koji se pomiče po svim mogućim pozicijama na matrici. Broj i dimenzija konvolucijskih filtera unaprijed su zadani dok se elementi filtera uče na način poput prethodno opisanih težina. Kao broj filtera standardno se uzima potencija broja 2 poput 64, 128, 256 te odabir broja filtera ovisi o zahtjevima mreže.

U konvolucijskim slojevima, osim broja filtera, dimenzije i standardnog pomaka, pojavljuju se još dva podesiva hiperparametra. To su: nadopunjavanje i korak.

Nadopunjavanje (eng. *padding*) je operacija koja se izvršava nad ulaznom matricom prije primjene konvolucijskih filtera. Dodavanjem piksela oko rubova slike se izbjegava sma-

njenje dimenzionalnosti prilikom procesa konvolucije. Dodani pikseli najčešće se inicijaliziraju kao "prazni", tj. s vrijednosti 0.

Korak (eng. *stride*) je podesivi parametar kojim se može znatno smanjiti dimenzionalnost izlaza konvolucijskog sloja. Postavljanjem koraka na vrijednost $n \in \mathbb{N}$, definira se broj piksela slike koje konvolucijski filter preskače prije izvršavanja iduće konvolucije.

Sloj sažimanja

Sažimanje (eng. *pooling*) je tehnika koja podrazumijeva podjelu ulazne matrice na manji broj blokova koji su najčešće veličine 2x2. U svakome bloku uzima se samo jedna vrijednost, najčešće maksimalna vrijednost unutar pojedinog bloka. Te vrijednosti su iskorištene u kreiranju nove matrice. Slojevi sažimanja mogu biti inicijalizirani s proizvoljnim nadopunjavanjem i korakom. Tehnika sažimanja najčešće se koristi neposredno nakon konvolucijskog sloja.

Slojevi sažimanja, osim očitog smanjenja dimenzionalnosti koje ima izuzetno važnu ulogu u poboljšanju vremenskih performansi mreže, omogućuju i stvaranje snažnijih značajki iz ulaznih podataka. Tako dolazi do stvaranja značajki koje su manje osjetljive na šum u podatcima. Kroz proces uzimanja maksimalnih (ili često srednjih) vrijednosti unutar svakog bloka, model efikasno identificira najistaknutije karakteristike prisutne na slici.

Primjer 2.1.4. *Sažimanje matrice uzimanjem maksimalnih vrijednosti s filterom veličine 2x2 te korakom veličine 2x2.*

$$\begin{bmatrix} 1 & 2 & 5 & 4 \\ 2 & 3 & 1 & 4 \\ 0 & 0 & 2 & 1 \\ 1 & 3 & 1 & 2 \end{bmatrix} \xrightarrow{MaxPooling} \begin{bmatrix} 3 & 5 \\ 3 & 2 \end{bmatrix}$$

2.2 Uvod u detekciju objekta

Sa brojnim uporabama u medicinskoj, vojnoj, automobilskoj, sportskoj i brojnim drugim industrijama, detekcija objekta je jedna od najvećih grana strojnoga vida. Uobičajeno se sastoji oddvaju ključnih zadataka: klasifikacije i lokalizacije objekta.

Lokalizacija objekta

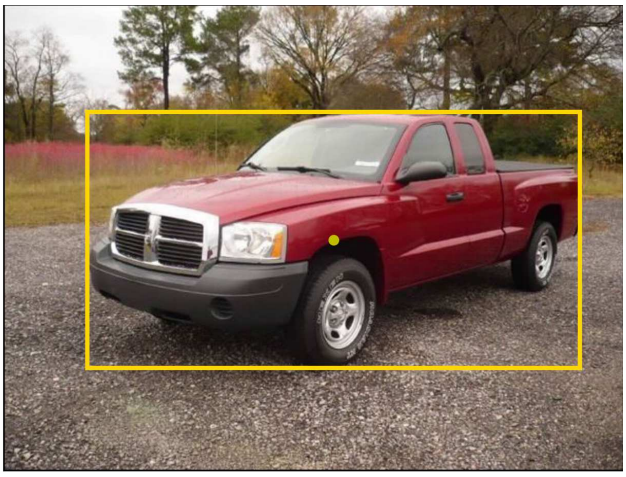
Zadatak lokalizacije objekta je pronaći sve bitne objekte na slici. Najbolji primjer primjene lokalizacije je u automobilskoj industriji gdje se nastoje detektirati te procjeniti lokacije pješaka i automobila u prometu. Pronađeni objekti najčešće se lokaliziraju unutar pravokutnih okvira (eng. *bounding box*) 2.1.

Pravokutni okvir najčešće je dvodimenzionalan, ali može biti i trodimenzionalan, ovisno o zadatku i podatcima koje imamo. Za lokalizaciju pojedinog objekta potrebna su četiri parametra: x, y koordinate (najčešće centar ili gornji-lijevi kut objekta), širina objekta i visina objekta.

Uz oznake w za širinu i h za visinu objekta te pretpostavku da x označava lokaciju u vodoravnoj osi, a y u okomitoj, granice okvira dobiju se na sljedeći način:

$$x_{min} = x_{centar} - w/2$$
$$x_{max} = x_{centar} + w/2$$
$$y_{min} = y_{centar} - h/2$$
$$y_{max} = y_{centar} + h/2$$

Za lokalizaciju objekta u trodimenzionalnom prostoru, potrebno je 9 parametara: x, y, z koji označavaju lokaciju središta objekta, w, h, l širinu, visinu i duljinu objekta, te *yaw*, *pitch*, *roll* orijentaciju objekta oko osi trodimenzionalnog prostora.



Slika 2.1: Auto lokaliziran pravokutnim okvirom

Detekcija objekta

Najveća problematika prilikom detekcije objekata je detektirati sve vidljive objekte sa slike. Kako bi se ostvarili što bolji rezultati razvijeni su modeli za detekciju objekta s ciljem da im izlaz bude lista objekata. Budući da se svaki objekt mora lokalizirati i klasificirati, za svaki element u listi potrebno je $4 + k$ izlaza, gdje je k jednak broju mogućih klasa.

Modeli za detekciju objekata dizajnirani su da detektiraju fiksnu količinu pravokutnih okvira, uglavnom znatno veću od stvarne. Budući da se na slikama nalazi varijabilan broj modelu bitnih objekata, potreban je način da se definira koji pravokutni okvir zaista sadrži traženi model, a koji ne. Mogući problem koji se može javiti je da se pojedini objekt detektira i lokalizira više puta. Dodatno je potrebno odbaciti i moguće suvišle duplikate.

Budući da je u klasifikacijskom sloju uglavnom korištena aktivacijska funkcija *softmax*, vrijednosti izlaznih neurona za klasifikaciju mogu se interpretirati kao postotak kojim je model uvjeren da je detektirani objekt član pojedine klase.

Primjer 2.2.1. *Neka je idućim uređenim sedmorkama prikazan izlaz modela kreiranog da detektira objekte koji mogu pripadati jednoj od tri klase:*

$$(0.34, 0.22, 0.13, 0.07, 0.01, 0.98, 0.03)$$

$$(0.88, 0.67, 0.11, 0.22, 0.08, 0.00, 0.01)$$

Model je prvi pravokutni okvir detektirao s $x_{centar} = 0.34$*,* $y_{centar} = 0.22$*,* $w = 0.13$*,* $h = 0.07$*. Za prvi detektirani objekt dao je* 0.01 *sigurnost da pripada prvoj,* 0.98 *da pripada drugoj te* 0.03 *da pripada trećoj klasi. Navedene rezultate pokazuju da je model dosta siguran da detektirani objekt pripada drugoj klasi.*

Za drugi detektirani okvir, model je naveo relativno malene razine sigurnosti za svaku od mogućih klasa. Za taj okvir se može reći da ne sadrži nijedan objekt od važnosti u sebi.

Kao što je u primjeru vidljivo, model za svaki okvir izbaci rezultate uvjerenosti pripadanja pojedinoj klasi, stoga je potrebno definirati granicu iznad koje se objekti deklariraju kao uspješno detektirani. Kako model može više puta detektirati isti objekt, dobiva se više presijecajućih okvira s velikom sigurnošću pripadanja istoj klasi. Dvostruki okviri filtriraju se pomoću presijek kroz unija i *Non-Max Suppression* (skraćeno NMS) algoritama.

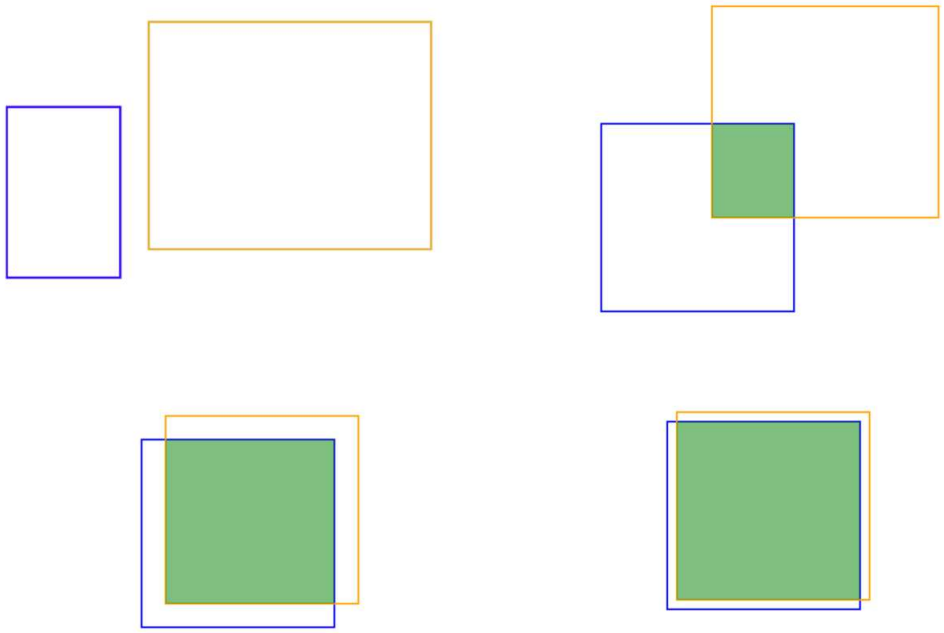
Presijek kroz unija

Presijek kroz unija (eng. *Intersection Over Union* - IoU) je mjera koja opisuje podudaranje dvaju objekata. Podudaranje se izračunava kao omjer površine presijeka dva pravokutna okvira i površine njihove unije.

Definicija 2.2.2. *Neka su A i B skupovi takvi da* $P(A \cup B) \neq 0$*. Tada definiramo:*

$$IoU(A, B) := \frac{P(A \cap B)}{P(A \cup B)} \tag{2.1}$$

IoU mjera često se koristi u NMS algoritmu kako bi se utvrdilo koji okviri pripadaju istome objektu. Prag za preklapanje se unaprijed definira ovisno o problemu i rezultatima na validacijskom skupu. Različite vrijednosti IoU prikazane su slikom 2.2.



Slika 2.2: Različite IoU vrijednosti, redom slijeva na desno, gore prema dolje: 0, 0.1, 0.6, 0.9

Non Max Suppression

NMS metoda je metoda kojom se filtriraju duplikati iz detekcija objekata. Detekcije se najprije sortiraju silazno po maksimalnoj pripadnosti nekoj klasi. Zatim se iterativno uzimaju

s početka liste i stavljaju u konačan popis detekcija. Nakon uzimanja svake pojedine detekcije se za sve ostale detekcije u listi računa vrijednost IoU podudaranja sa zadnje uzetom detekcijom, te se izbacuju iz liste ukoliko je IoU veći od zadanog praga.

Algorithm 1 Non-Maximum Suppression algoritam

```
B_final ← ∅                                   ▷ B_final je izlazna lista detekcija
for b_i ∈ B_initial do                        ▷ B_initial je ulazna lista detekcija
    remove_list ← ∅
    for b_j ∈ B_initial do
        if IoU(b_i, b_j) > threshold then
            remove_list ← remove_list ∪ {b_j}
        end if
    end for
    B_initial ← B_initial \ remove_list
    B_final ← B_final ∪ {b_i}
end for
```

2.3 Mjere uspješnosti

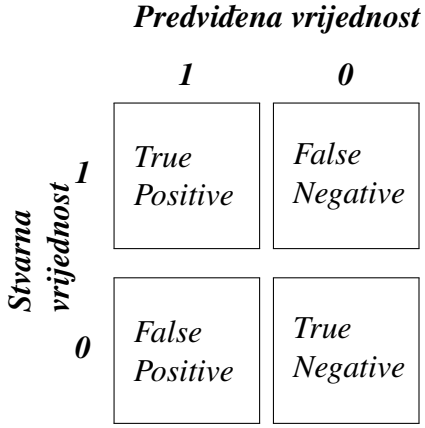
Mjere uspješnosti imaju ključnu ulogu u analizi performansi modela budući da omogućuju bolje rezumijevanje prednosti i nedostataka modela. Kroz kvantitivne mjere, dobivaju se jasne vrijednosti koje pomažu prilikom objektivne procijene koliko dobro model funkcionira u određenom kontekstu. U nastavku poglavlja bit će navedene najbitnije mjere uspješnosti u zadatku detekcije objekta.

Konfuzijska matrica

Konfuzijska matrica je najpopularnija mjera u problemima klasifikacije. Ova matrica pruža strukturiranu vizualizaciju stvarnih i predviđenih klasa te pomaže u analizi preciznosti i pouzdanosti modela. Konfuzijska matrica može se koristiti za probleme binarne klasifikacije, ali i za višeklasne probleme.

U slučaju binarne klasifikacije, konfuzijska matrica je veličine 2x2. Ukoliko podatke klasificiramo na pozitivnu i negativnu klasu, kao npr. u slučaju klasifikacije tumora na zločudni (1) i dobročudni (0), konfuzijska matrica će izgledati na idući način.

Primjer 2.3.1. *Konfuzijska matrica u slućaju binarne klasifikacije s mogućim klasama 1 i 0.*



- **True Positives (TP) -** Broj stvarno pozitivnih primjera, točno predviđenih od strane modela.
- **False Positives (FP) -** Broj stvarno negativnih primjera, koji su netočno predviđeni od strane modela kao pozitivni.
- **True Negatives (TN) -** Broj stvarno negativnih primjera, koji su točno predviđeni od strane modela kao negativni.
- **False Negatives (FN) -** Broj stvarno pozitivnih primjera, koji su netočno predviđeni od strane modela kao negativni.

U slučaju lokalizacije objekta, TP, FP, FN rezultati definiraju se kao broj točno/netočno lokaliziranih objekata. Za objekt se kaže da je točno lokaliziran ukoliko je njegovo IoU preklapanje sa stvarnim objektom veće od unaprijed definirane granice. Tada se TP, FP, FN rezultati definiraju na slijedeći način:

- **True Positives (TP) -** Broj objekata koji su točno klasificirani i lokalizirani, odnosno model je ispravno prepoznao prisutnost objekta i njegovu poziciju.
- **False Positives (FP) -** Broj objekata koji su pogrešno lokalizirani, odnosno model je prepoznao objekt na lokaciji na kojoj se ne nalazi ili ga je lokalizirao sa slabom točnosti.

- **False Negatives (FN) -** Broj stvarnih objekata koji nisu precizno lokalizirani. Drugim riječima, model nije uspio ispravno lokalizirati ove objekte.

Definicija 2.3.2. *Točnost (eng. Accuracy) modela definira se kao*

$$Acc = \frac{TP + TN}{TP + FP + TN + FN}$$

Točnost modela daje postotak točno klasificiranih primjera. Iako je vrlo česta mjera, nije uvijek najbolje za korištenje. Npr. u primjeru klasifikacije tumora, pozitivni primjeri su puno važniji od negativnih.

Definicija 2.3.3. *Preciznost (eng. Precision) modela definira se kao*

$$P = \frac{TP}{TP + FP}$$

Preciznost označava udio točno klasificiranih pozitivnih primjera u svima koji su klasificirani kao pozitivni.

Definicija 2.3.4. *Osjetljivost (eng. Recall) modela definira se kao*

$$R = \frac{TP}{TP + FN}$$

Osjetljivost označava udio točno klasificiranih pozitivnih primjera u svim stvarnim pozitivnim primjerima.

Definicija 2.3.5. *F-mjeru (F-score) modela definiramo kao*

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{P + R}$$

Posebno, F_1 rezultat je harmonijska sredina između preciznosti i osjetljivosti. Pogodan je za korištenje u slučajevima kad broj objekata po klasama nije balansiran u skupu podataka.

Srednja prosječna preciznost

Prosječna preciznost (eng. *Average Precision* - AP) je mjera korištena za kvantifikaciju točnosti modela kroz mnoge razine praga pouzdanosti detekcije pojedine klase. Detekcije se za određenu klasu sortiraju silazno po razini pripadnosti toj klasi. Za svaku od tih detekcija zna se je li TP ili FP ovisan o njenom IoU preklapanju sa stvarnim pravokutnim

okvirima. Prolazivši po sortiranoj listi detekcija, u svakom koraku izračunavaju se preciznost i osjetljivost modela na temelju dosad obrađenih elemenata u listi te se kreira krivulja preciznosti u ovisnosti o osjetljivosti modela ($p(r)$).

Tada se prosječna preciznost računa kao:

$$AP = \int_0^1 p(r)dr \tag{2.2}$$

Srednja prosječna preciznost (mAP) u modela s N klasa računa se kao aritmetička sredina prosječne preciznosti svih klasa:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.3}$$

Česta oznaka u literaturi $mAP@[.5:.95]$ označava aritmetičku sredinu mAP rezultata modela s obzirom na različite vrijednosti IoU praga korištenog za izračun TP, FP i FN mjera: (0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95) [7].

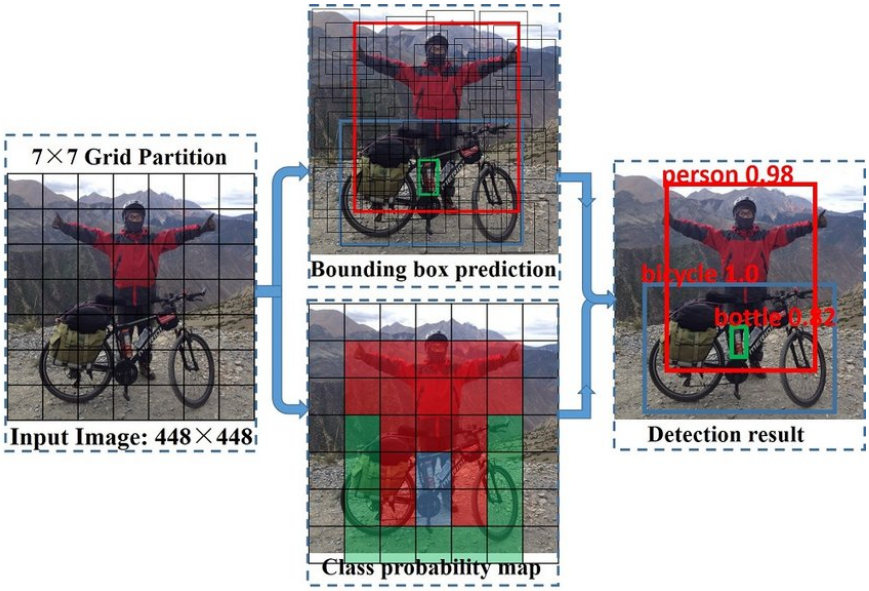
2.4 Jednokoračni detektori objekata

Jednokoračni (eng. *one-stage*) detektori objekta su modeli koji vrše detekciju objekta u jednom koraku. Ovi modeli su često korišteni u scenarijima gdje je brzina izuzeno bitna, poput autonomnih vozila i analize videa u stvarnom vremenu.

You Only Look Once

Jednokoračni model *You Only Look Once* (skraćeno YOLO) je 2016. godine revolucionalizirao detekciju objekata u stvarnom vremenu zahvaljujućim svojim performansama, odnosno brzini i preciznosti [15]. S vremenom su se razvile novije verzije YOLO detektora, među kojima su izričito popularne YOLOv3 i YOLOv7.

Glavna ideja iza YOLO detekcije objekata leži u podjeli početne slike na mrežu ćelija dimenzija NxN. Svaka od tih ćelija istovremeno provodi detekciju fiksnog broja pravokutnih okvira koji okružuju objekte, te odredbu kojoj klasi objekti pripadaju. Ovakav pristup omogućuje brzo otkrivanje objekata i njihovu pripadnost različitim klasama u jednom prolasku kroz mrežu. Kako bi se dobile konačne detekcije, primjenjuje se standardni NMS algoritam. Primjer postupka detekcije YOLO modelom dan je na slici 2.3.

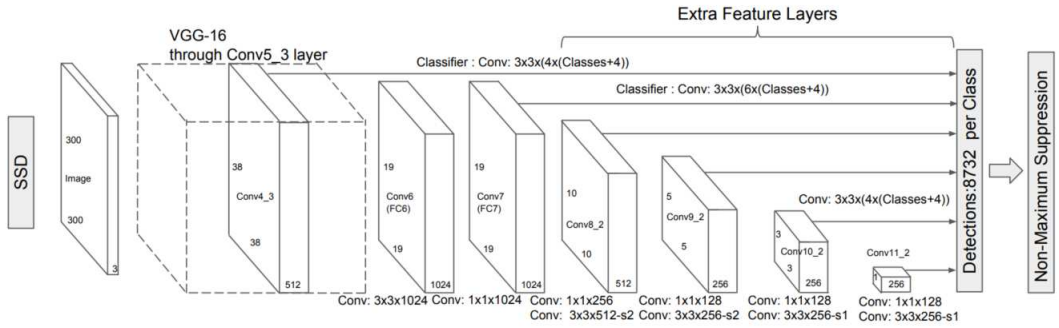


Slika 2.3: Primjer detekcije YOLO modelom

Single Shot Detector

Single Shot Detector (SSD) pripada jednokoračnim detektorima objekata [12]. Sastoji se od dva glavna djela: kralježnice (eng. *Backbone*) i glave (eng. *Head*). Kralježnica modela tipično se sastoji od mnogih konvolucijskih slojeva i slojeva sažimanja te služi za ekstrakciju značajki iz ulazne slike. Najčešće arhitekture kralježnica su ResNet, VGG i MobileNet. Originalna arhitektura SSD-a nalazi se na slici 2.4.

Ono što čini SSD-ove posebnima je njihova SSD glava. SSD glava sastoji se od više konvolucijskih slojeva. Tijekom prolaska ulazni podataka kroz konvolucijske slojeve, osim smanjenja dimenzija, generiraju se i predikcije objekata. Tim je postupkom modelu omogućena bolja detekcija objekata različitih veličina i proporcija. SSD koristi ”usidrene okvire” (eng. *Anchor boxes*) za detekciju objekata. Ti okviri su fiksne vrijednosti prethodno definirane veličine i omjera. Nakon što ulazne vrijednosti prođu kroz čitavu mrežu te se na izlazu dobije mnoštvo detekcija različitih veličina, koristi se NMS algoritam za konačno generiranje predikcija.



Slika 2.4: Arhitektura SSD mreže

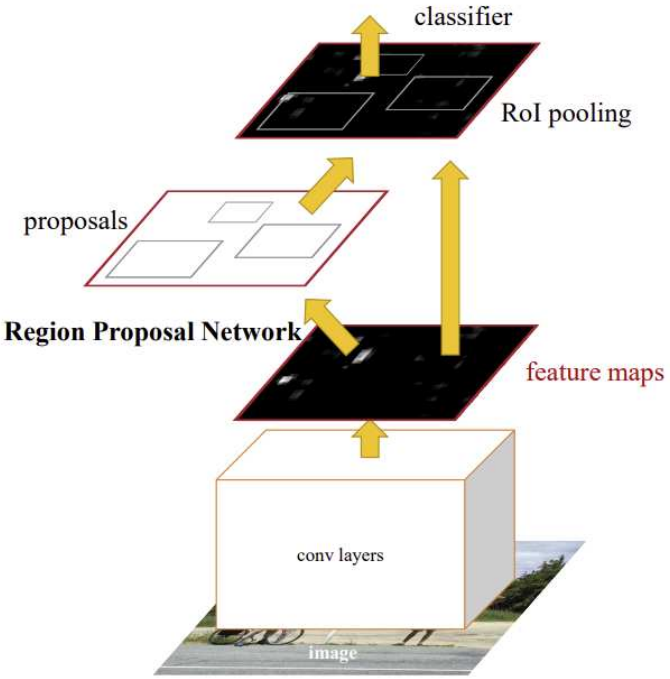
2.5 Dvokoračni detektori objekata

Za razliku od jednokoračnih detektora, dvokoračni (eng. *two-stage*) detektori koriste dvije faze u detekciji objekata kako bi ostvarili veću preciznost i bolju sposobnost detekcije objekata različitih veličina. Ovi detektori se sastoje od dva ključna koraka: generiranje regija značajki (*Region proposal Network* - RPN) te korak klasifikacije i lokalizacije objekta.

U prvom koraku detekcije objekata dvokoračnim detektorom, generiraju se potencijalne regije koje sadrže objekte. Nakon početnih konvolucijskih slojeva, mreža za predlaganje regija, odnosno RPN, prolazi kroz mapu značajki kako bi predvidjela potencijalne regije s objektima. Za svaku regiju, RPN predviđa vjerojatnost da regija sadrži objekt te korekciju za veličinu i položaj regije. Regije zatim prolaze kroz NMS algoritam te se koriste za slijedeći korak detekcije - preciznija klasifikacija i lokalizacija objekata.

Faster R-CNN

Faster R-CNN je jedan od najpoznatijih i najuspješnijih dvokoračnih detektora. Predstavljen je 2015. godine u radu "*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*" s kojim je postao osnovica za daljnji razvoj u području detekcije objekata [16]. Nakon detektiranja regija značajki, Faster R-CNN primjenjuje metodu sažimanja regije interesa (eng. *Region of Interest Pooling* - RoI Pooling) kako bi skalirao značajke na fiksnu veličinu (slika 2.5). *RoI Pooling* podijeli svaku regiju u rešetku fiksnih dimenzija (npr. $7x7$) te sažme vrijednosti slično kao u postpku opisanom u slojevima sažimanja. Nakon skaliranja, slijedi proces klasifikacije i lokalizacija objekta unutar regije.



Slika 2.5: Arhitektura Faster R-CNN

Poglavlje 3

Kvantifikacija stabilnosti lokalizacije objekta

3.1 Stabilnost lokalizacije objekta

Kad mreža detektira objekt, detektira se njegov pravokutni okvir te vjerojatnost pripadanja pojedinim klasama. Veća sigurnost pripadanja objekta određenoj klasi može se protumačiti kao veća stabilnost klasifikacije tog objekta. No, nakon što je objekt uspješno klasificiran, ne postoji kvantifikacija stabilnosti njegove lokalizacije.

Kvantifikacija stabilnosti lokalizacije objekta može biti korisna u mnogim situacijama. Primjerice kad je objekt djelomično prekriven drugim objektima ili kad izlazi iz ruba okvira slike, pouzdanost njegove lokalizacije značajno se smanjuje. U takvim slučajevima, poznavajući razinu stabilnosti lokalizacije, može se bolje procijeniti koliko se vjeruje točnosti te detekcije.

Dodatno, predviđanje stabilnosti lokalizacije postaje ključno u situacijama s nepovoljnim uvjetima poput nepreglednog osvjetljenja ili prisutnosti loših vremenskih uvjeta kao što su kiša ili snijeg. U takvim slučajevima, preciznost detekcije može značajno varirati, a kvantificiranje stabilnosti pomaže u boljem razumijevanju točnosti detekcije te može pridonijeti pouzdanijem sustavu za praćenje objekata.

Predviđanje stabilnosti također je korisno u situacijama gdje objekti imaju promijenjive parametre, kao što su veličina i orijentacija. Objekti koji imaju varijacije u svojim karakteristikama mogu dovesti do povećane nesigurnosti u lokalizaciji. Kvantifikacija stabilnosti omogućuje modelima da uzmu u obzir ovu nesigurnost i bolje procijene granice svoje pouzdanosti.

U problemu regresije pravokutnog okvira se kreiranje modela s kvantifikacijom nesigur-

nosti najčešće ostvaruje predviđanjem aritmetičke sredine μ_i te varijance σ_i^2, za svaki $i \in \{x_{min}, x_{max}, y_{min}, y_{max}\}$

3.2 Aleatorička i epistemička nesigurnost

Izvore nesigurnosti u neuronskim mrežama može se podijeliti na dvije glavne grane: aleatoričku nesigurnost i epistemičku nesigurnost.

Aleatorička nesigurnost, također poznata kao statistička nesigurnost ili nasumična greška, predstavlja svojstvenu varijabilnost i slučajnost u samim podatcima. Proizlazi iz nepredvidivih faktora koji utječu na promjenjivost podataka bez obzira na to koji model se koristi. U kontekstu lokalizacije objekta, u aleatoričku nesigurnost pripada nesigurnost dobivena okluzijom objekta, šumom u ulaznim podatcima te ostalim nepredvidivim okolnostima.

Epistemička nesigurnost, također poznata kao sistematička, se odnosi na nesigurnost uzrokovanu manjkom znanja. Za razliku od aleatoričke, epistemička nesigurnost se može smanjiti povećanjem skupa podataka za trening te korištenjem složenijih modela. U aktivnom učenju, epistemička nesigurnost može se pokazati korisnom prilikom odabira modela i podataka za trening [8].

3.3 Bayesovske neuronske mreže

U standardnim dubokim neuronskim mrežama, parametri poput težine i pomaka su egzaktne numeričke vrijednosti koje se optimiziraju tijekom treninga kako bi se minimizirala pogreška modela. U Bayesovskim neuronski mrežama (skraćeno BNN-ima), za razliku od standardnih neuronskih mreža, koristi se probabilistički pristup učenju modela. U takvom pristupu se parametri promatraju kao slučajne varijable s odgovarajućim vjerojatnosnim distribucijama. Takva probabilistička raspodjela omogućava BNN-ima kvantifikaciju nesigurnosti ne samo u predikcijama, nego i samim parametrima modela [14].

Definicija 3.3.1. *Neka je (Ω, F, P) vjerojatnosni prostor te $B \in F$ događaj takav da je $P(B) > 0$. Uvjetnu vjerojatnost događaja A uz dano B definiramo formulom:*

$$P(A|B) := \frac{P(A \cap B)}{P(B)}$$

Teorem 3.3.2. *Neka su A i B događaji takvi da $P(B) \neq 0$. Tada, Bayesov teorem, u matematičkom zapisu, glasi:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

U Bayesovskim neuronskim mrežama prvobitno se postavlja pretpostavka o vjerojatnosnoj ovisnosti između izlaza mreže $f^w(x)$ i stvarnih vrijednosti y: $p(y_i \mid f^w(x_i))$. U slučaju regresije, najčešće se koristi pretpostavka $p(y_i \mid f^w(x_i)) \sim N(f^w(x_i), \sigma^2)$.

Umjesto pravih vrijednosti, cilj BNN-a je pronalazak distribucija unutar kojih se težine w mreže nalaze, odnosno aposteriorna distribucija: $p(w \mid D)$, gdje D predstavlja skup podataka za treniranje. Takva distribucija pronalazi se pomoću Bayesovog teorema:

$$p(w \mid D) = \frac{p(D \mid w)p(w)}{p(D)} \tag{3.1}$$

- $p(D \mid w)$ predstavlja vjerodostojnost i računa se:

$$p(D \mid w) = \prod_{i=1}^{N} p(y_i \mid f^w(x_i)) \tag{3.2}$$

- $p(w)$ je apriorna vjerojatnost. Predstavlja prethodno znanje o parametrima i odabire se ručno.
- $p(D)$ je marginalna vjerodostojnost i može se izračunati kao:

$$p(D) = \int p(D, w)p(w)dw \tag{3.3}$$

Zbog kompleksnosti neuronskih mreža, direktno računanje integrala marginalne vjerodostojnosti često nije moguće. Stoga, postoje drugi pristupi računanja aposteriori distribucije poput varijacijske inferencije (skraćeno VI) i Markovljevog lanca Monte Carlo (skraćeno MCMC).

Varijacijska inferencija

Cilj Varijacijske inferencije je pronalazak distribucije Q nad w ovisne o parametrima θ tako da $Q_\theta(w) \approx p(w \mid D)$. Ta aproksimacija se postiže minimizacijom *Kullback-Leibler* divergencije između distribucija[2].

Definicija 3.3.3. *Kullback-Leibler divergencija, s oznakom D_{KL}, između vjerojatnosnih distribucija P i Q definira se kao:*

$$D_{KL}(p \parallel q) = \mathop{\mathbb{E}}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

Minimizacija D_{KL} svodi se na pronalazak vrijednosti:

$$\arg\min_{\theta} D_{KL}\left[q_\theta(w) \parallel p(w \mid D)\right]$$

Kako D_{KL} ovisi o posterioru koji nije poznat, KL divergencija ne može direktno biti optimizirana te se stoga raspisuje na slijedeći način:

$$D_{KL}\left[q_\theta(w) \parallel p(w \mid D)\right] = \mathop{\mathbb{E}}_{q_{\theta(w)}}\left[\log \frac{q_\theta(w)}{p(w \mid D)}\right] =$$

$$= \mathop{\mathbb{E}}_{q_{\theta(w)}}\left[q_\theta(w)\right] - \mathop{\mathbb{E}}_{q_{\theta(w)}}\left[\log p(w, D)\right] + \log p(D) =$$

$$= - \mathop{\mathbb{E}}_{q_{\theta(w)}}\left[\log p(D \mid w)\right] + D_{KL}(q_\theta(w) \mid p(w))$$

Uz pretpostavku da su podatci nezavisni i jednako distribuirani, vrijedi:

$$\log p(D, w) = \sum_{i=1}^{N} \log p(y \mid x, w). \tag{3.4}$$

Dobiven rezultat naziva se negirani *Evidence Lower Bound* (skraćeno ELBO) te se pronalazak aposteriori distribucije svodi na njegovu minimizaciju. Ta minimizacija provodi se putem stohastičkog gradijentnog spusta (eng. *Stochastic gradient descent*), gdje se parametri ažuriraju kroz nasumično odabrani podskup podataka za trening kroz svaku iteraciju.

3.4 Ansambli

Korištenje ansambla u kontekstu detekcije objekata i lokalizacije okvira znači da se umjesto oslanjanja na jedan model za donošenje odluka, koristi više modela i kombiniraju njihovi rezultati. Ova tehnika često se primjenjuje kako bi se postigla veća točnost, te da bi se kvantificirala pouzdanost prilikom detekcije objekata. Kako bi se izbjeglo oslanjanje na jedan model koji sadrži vlastite detekcije i stabilnosti, ansambli omogućuju iskorištavanje raznolikosti među različitim modelima.

Razvijeno je više metoda za formiranje ansambla modela gdje svaka sadrži različite internalne karakteristike. Jedna od tih metoda je *bagging*, u kojoj se modeli treniraju na različitim podskupovima skupa za trening. Ovakav pristup rezultira raznolikim modelima unutar jednog ansambla. Sljedeći najčešće korišten pristup naziva se *boosting*, gdje se modeli treniraju sekvencijalno, pri čemu svaki novi model nastoji ispraviti greške prethodnog

modela. Dodatni faktori koji mogu utjecati na raznolikost i stabilnost modela su različite arhitekture, hiperparametri i ulazni podatci.

Kombiniranje rezultata ansambla moguće je postići različitim načinima. Najjednostavniji način je pristup glasanja većine, gdje se za svaki objekt odabere klasa koja je najčešće predviđena. U problemu stabilnosti lokalizacije, to bi značilo uzeti aritmetičku sredinu i varijancu svih predikcija za određeni parametar pravokutnog okvira. Popularan pristup je i ponderirano glasanje, gdje se izlazu svakog modela stavlja težina ovisno o njegovom uspjehu na podatcima za validaciju.

Iako ansambli mogu dovesti do poboljšanja rezultata i smislene kvantifikacije nesigurnosti, glavni razlog rijetkog korištenja ansambla u detekciji objekata su vremenska ograničenja. Izrada ansabla često zahtjeva izradu više različitih arhitektura modela za treniranje. Sam trening ansambla zahtjeva N puta više računalnih resura od treninga pojedine neuronske mreže, gdje je N broj modela u ansamblu. No, najveći problem je taj da pri generiranju pojedine predikcije, pojedina slika mora proći kroz N modela u ansablu, što čini ansable neupotrebivim u situacijama gdje je brzina generiranja predikcija od velike važnosti.

3.5 Monte Carlo isključivanje

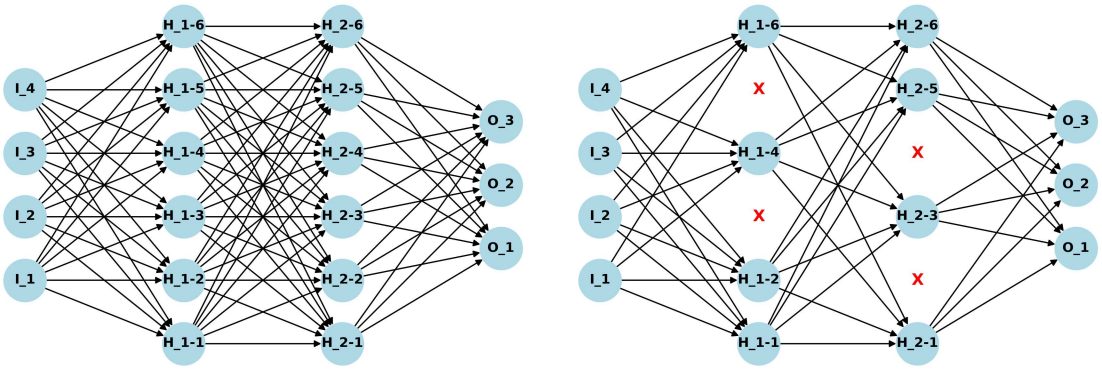
Isključivanje

Isključivanje (engl. dropout) predstavlja tehniku regularizacije koja se primjenjuje tijekom treninga kako bi se spriječilo pretreniranje i unaprijedila generalizacija modela. U postupku treninga, dok podatci prolaze kroz mrežu, neuroni se između slojeva gdje je primijenjeno isključivanje nasumično isključuju s unaprijed definiranom vjerojatnošću. Ta vjerojatnost se najčešće nalazi u vrijednostima $[0.2, 0.5]$. Prikaz mreže nakon isključivanja nalazi se na slici 3.1.

Ovaj proces osigurava da se pojedini neuroni ne oslanjaju isključivo na specifične uzorke iz skupa za treniranje, već potiče mrežu da koristi raznolike kombinacije neurona za donošenje odluka. Time se poboljšava sposobnost modela da se nosi s različitim situacijama i smanjuje rizik od pretreniranja, gdje bi model previše prilagodio sebe treniranim podacima te izgubio sposobnost općenitog zaključivanja na nepoznatim podacima.

Monte Carlo isključivanje

Iako se isključivanje inače koristi samo tijekom treninga mreže, može se koristiti i tijekom inferencije. Korištenje nasumičnog isključivanja tijekom inferencije dovodi do nedeterminizma u detekcijama, odnosno do različitih rezultata pri svakom prolasku istog ulaza kroz



Slika 3.1: Primjer mreže prije i poslije isključivanja

mrežu. Metodu generiranja niza predikcija korištenjem isključivanja prilikom višestrukog prolaska ulaza kroz mrežu nazivamo Monte Carlo isključivanje (eng. *Monte Carlo Dropout*).

Slično kao i kod ansambala, glavna ideja Monte Carlo isključivanja je generirati nasumične predikcije i interpretirati ih kao uzorke iz neke vjerojatnosne distribucije. Takva interpretacija rezultata naziva se Bayesovska interpretacija.

Iznimna prednost Monte Carlo isključivanja jest u tome da za njegovo korištenje nije potrebno trenirati više odvojenih modela, što predstavlja značajno pojednostavljenje u odnosu na metodu ansambala. Osim toga, budući da se kroz mrežu prolazi samo jednom, sve do prvog sloja s isključivanjem, proces inferencije primjenom Monte Carlo isključivanja brži je u usporedbi s ansamblima. Iz tih razloga, Monte Carlo isključivanje se pokazalo puno jeftinijom i privlačnijom opcijom kod kvantifikacije stabilnosti detekcija.

3.6 Mjere uspješnosti

Za razliku od evaluacije uspješnosti standardne detekcije objekta, gdje su dane stvarne vrijednosti klase i pravokutnog okvira svakog objekta na slici, stvarne vrijednosti nesigurnosti lokalizacije objekta nisu dane. Takav nedostatak stvara potrebu za posebnim mjerama za ocjenjivanje stabilnosti koje će usporediti predviđenu distribuciju granica pravokutnih okvira sa stvarnim granicama.

Očekivana kalibracijska greška

Očekivana kalibracijska greška (eng. *expected calibration error*) mjeri koliko dobro se vjerojatnosti klasifikacije objekta slažu sa stvarnim vjerojatnostima. Računa se na način da se dobivene maksimalne vjerojatnosti podijele u N jednako velikih intervala B_i između 0 i 1 gdje je svaki veličine $1/N$. Tada, za svaku vrijednost $i \in 1, \ldots, N$ računa se empirijska vjerojatnost pripadanja uzorka intervalu B_i; $\frac{|B_i|}{N}$ [5].

Za svaki B_i računa se:

$$conf(B_i) := \frac{1}{|B_i|} \sum_{t \in B_i} p_t, \tag{3.5}$$

gdje su p_t predviđene vjerojatnosti.

Nakon toga, računa se:

$$acc(B_i) := \frac{1}{|B_i|} \sum_{t \in B_i} \mathbb{1}_{y_i = \hat{y}_i}, \tag{3.6}$$

gdje je $\mathbb{1}$ karakteristična funkcija jednaka 1 ukoliko se predviđena vrijednost $\hat{y}_i$ podudara sa stvarnom vrijednosti y_i.

Nakon što su dobivene vrijednosti $conf$ i acc, može se iskoristiti formula:

$$ECE = \sum_{i=1}^{M} \frac{|B_i|}{N} |acc(B_i) - conf(B_i)| \tag{3.7}$$

Ovisnosti vrijednosti acc i $conf$ daju uvid u kalibraciju klasifikacije modela. Model je dobro kalibriran ako se acc i $conf$ podudaraju.

RMV i RMSE

Neka su granice pravokutnih okvira objekta standardno detektirane svaka s dva parametra; μ i σ^2. Tada se kaže da je model dobro kalibriran ukoliko vrijedi:

$$\forall \sigma : \mathbb{E}\left[(\mu(x) - y)^2 | \sigma(x)^2 = \sigma^2\right] = \sigma^2 \tag{3.8}$$

Drugim riječima, ukoliko je očekivana greška modela, mjerena pomoću prosječne kvadratne greške, jednaka predviđenoj varijanci σ^2 [9].

Budući da se vrijednosti σ^2 poprimaju iz skupa $\mathbb{R}$, one se grupiraju u N intervala B_i, uz pretpostavku da N dijeli broj uzoraka T. Tada su vrijednosti σ^2 ravnomjerno poredane u intervale u uzlaznom poretku tako da $B_i = \left\{(j-1)\frac{T}{N} + 1, \ldots, j\frac{T}{N}\right\}$.

Nakon toga se računaju korijen srednje varijance:

$$RMV(j) = \sqrt{\frac{1}{|B_j|}\sum_{t \in B_j} \sigma_t^2}, \tag{3.9}$$

te empirijski korijen srednjih kvadrata:

$$RMSE(j) = \sqrt{\frac{1}{|B_j|}\sum_{t \in B_j} (y_t - \hat{y}_t)^2} \tag{3.10}$$

Vizualiziranjem $RMSE$ kao funkciju u ovisnosti o RMV, dobiva se uvid u kalibraciju i samouvjerenost modela.

Uzimanjem prosječne kalibracijske greške u svakom intervalu i normalizirajući je s intervalovom srednjom vrijednošću, dobiva se mjera analogna ECE iz klasifikacije:

$$ENCE = \frac{1}{N}\sum_{j=1}^{N} \frac{|RMV(j) - RMSE(j)|}{RMV(j)} \tag{3.11}$$

Poglavlje 4

Gaussova negativna logaritamska vjerodostojnost kao metoda procjene stabilnosti lokalizacije objekta

U ovome poglavlju, opisat će se *KITTI* skup podataka. Zatim će se dati intuicija iza izgradnju modela sa sposobnošću kvantifikacije stabilnosti pravokutnog okvira. Opisat će se postupak treniranja tog modela te analizirati rezultati na prethodno navedenom skupu podataka.

4.1 Priprema podataka

Za treniranje i evaluaciju modela, korišten je *KITTI Object Detection dataset* [3]. Prvi put predstavljen 2012. godine, danas je jedan od najpopularnijih skupova podataka za detekciju objekta. Dijeli se na dva skupa podataka: podatke za treniranje koji čine 7481 slika i podatke za testiranje koji čine 7518 slika. Budući da stvarne vrijednosti podataka za testiranje nisu javno dostupne, fokus ovoga rada bit će na podatcima za trening.

U radu je KITTI skup podataka podijeljen na slijedeći način:

- 70% podataka za trening - 5236 slika
- 15% podataka za validaciju - 1122 slike
- 15% podataka za testiranje - 1123 slika

KITTI slike su originalno dimenzija 375x1242 piksela. Za model korišten u ovome radu je širina slika smanjena na 1024 piksela. Zatim su slike nadopunjene ekvivalentno s gornje i

donje strane crnim pikselima kako bi njihove konačne dimenzije bile 1024x1024. Primjer slike iz KITTI skupa podataka dan je slikom 4.1.

Uz svaku sliku u KITTI skupu podataka, dolaze podatci o objektima koji se na njoj nalaze. Informacije koje su dostupne su:

- **Klasa objekta:** Svaki objekt svrstan je u jednu od sljedećih klasa: ”Car”, ”Van”, ”Truck”, ”Pedestrian”, ”Cyclist”, ”Tram”, ”Person_sitting”, ”Misc” ili ”DontCare”
- **2D pravokutni okvir:** Pravokutni okvir predstavljen s četiri broja, svaki između 0 i 1241 (0 i 1024 skalirano) u redosljedu x_{min}, y_{min}, x_{max}, y_{max}
- **Odsijecanje:** Vrijednost između 0 i 1. Označava koliki postotak auta je odsječen iz slike.
- **Okluzija:** Prirodni broj iz skupa $\{0, 1, 2, 3\}$. Nula označava da je objekt u potpunosti vidljiv, jedinica označava da je djelomično prekriven, dvojka označava da je uglavnom prekriven. Trojka označava da vrijednost okluzije nije poznata.
- **Alpha:** Obzervacijski kut objekta, između $-\pi$ i π.
- **3D pravokutni okvir:** Trodimenzionalni podatci o lokaciji i veličini objekta u metrima. Također je dana rotacija objekta oko y-osi (kut između $-\pi$ i π).

Slika 4.1: Primjer slike iz KITTI skupa podataka

U nastavku rada će fokus biti na 2D pravokutnom okviru objekta pa će problem klasifikacije biti pojednostavljen na dvije klase:

- klasa ”Vehicle” u koju se svrstavaju svi objekti izvornih klasa ”Car”, ”Van”, ”Truck” i ”Tram”,
- klasa ”VRU” (*Vulnerable Road Users*) u koju se svrstavaju svi objekti izvornih klasa ”Pedestrian”, ”Person_sitting” i ”Cyclist”.

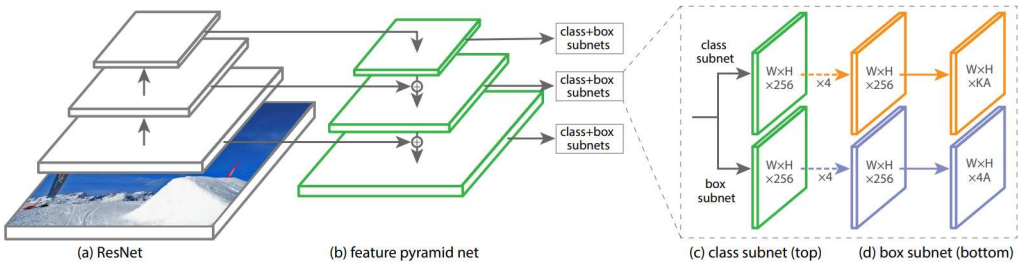
4.2 Odabir modela

Model koji je korišten za detekciju je *SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)* s *TensorFlow2 Object Detection model zoo*-a [1]. U toj kolekciji nalaze se mnogi modeli za detekciju objekta, poput *CenterNet*, *EfficientDet*, *SSD*, *Faster-RCNN*. Modeli su predtrenirani na *Common Object in Context* (skraćeno COCO) skupu podataka. COCO skup podataka sastoji se od 338 000 slika na kojima se nalaze objekti podijeljeni u 80 klasa [11].

RetinaNet50 koristi ResNet-50 kao kralježnicu koja je sastavljena od 50 slojeva [10]. ResNet-50 sastoji se od 48 konvolucijskih slojeva te dva sloja sažimanja. Izlaz se šalje dalje u *Feature Pyramid Network* (FPN), koji je ključna komponenta arhitekture RetinaNet (slika 4.2).

FPN pruža sposobnost detekcije objekata različitih veličina tako što usklađuje i spaja značajke iz različitih slojeva kralježnice. Konkretno, FPN koristi tehniku uspinjanja (eng. *upsampling*) kako bi povećao rezoluciju nižih razina značajki i potom kombinira poboljšane značajke s onima dobivenih na višim razinama (s većom rezolucijom). Tako se kreira piramida značajki koja obuhvaća objekte različitih veličina.

Nakon što se informacije obrade kroz FPN, izlazi se šalju u dvije odvojene konvolucijske mreže. Jedna mreža je odgovorna za problem lokalizacije, a druga za problem klasifikacije objekata. Kombinacijom izlaza tih dvaju mreža, model kreira detekcije objekata na slici.



Slika 4.2: Arhitektura mreže RetinaNet

4.3 Funkcija gubitka Gaussove negativne logaritamske vjerodostojnosti

Svaka od prethodno navedenih metoda kvantifikacije stabilnosti pravokutnog okvira ima svoje nedostatke. Bayesovske neuronske mreže zahtijevaju potpunu promjenu arhitekture mreže tako da fokus bude na probabilističkim detekcijama. S druge strane, ansambli i Monte Carlo isključivanje zahtjevaju višestruke prolaske kroz mrežu, što ih čini neupotrebivim u situacijama gdje nam je brzina izvršavanja od velike važnosti. Stoga se u eksperimentima testira metoda koja bi zahtjevala minimalne promjene u arhitekturi te samo jedan prolazak kroz mrežu.

Za preciznu lokalizaciju objekata na usidrenim okvirima dobivenim kroz razne slojeve FPN-a, zaslužan je *box_subnet*. *Box_subnet* se sastoji od četiri $3x3$ konvolucijska sloja, svaki s 256 filtera. Nakon toga dolazi izlazni sloj koji se sastoji od $4 \cdot A$ filtera po lokaciji, gdje je $A = 6$ broj usidrenih okvira. U sljedećim eksperimentima mreža se modificira tako da umjesto originalne četiri koordinate, izbacuje osam parametara za lokalizaciju objekta: μ_i i σ_i, $\forall i \in \{x_{min}, x_{max}, y_{min}, y_{max}\}$, preciznije; svaka koordinata pravokutnog okvira modelirat će se kao Gaussova distribucija s parametrima μ i σ^2. To se postiže modifikacijom završnog sloja *box_subnet*-a tako da primjenjuje $8 \cdot A$ filtera, gdje će novi izlaz za svaki usidreni okvir predstavljati navedene parametre.

Dodatno je potrebno definirati funkciju gubitka koja će u obzir uzimati oba parametra modela. Ta funkcija gubitka naziva se gubitak Gaussove negativne logaritamske vjerodostojnosti (eng. *Gaussian negative Log-Likelihood loss*) i izvodi se iz Gaussove distribucije.

Definicija 4.3.1. *Gaussova normalna distribucija varijable y, s parametrima μ i σ definira se kao:*

$$N(y; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right)$$

Parametri μ i σ nazivaju se aritmetička sredina i standardna devijacija.

Traženi cilj je pronalazak parametara μ i σ. To se postiže maksimiranjem vjerojatnosti od y, uz parametre modela μ i σ. Ta metoda se naziva estimacija najveće vjerodostojnosti te se može zapisati kao:

$$\mu, \sigma = \underset{\mu,\sigma}{\operatorname{argmax}} N(y|\mu, \sigma)$$

Budući da je logaritam strogo rastuća funkcija, problem maksimiziranja funkcije N jednak je problemu maksimiziranja funkcije $\log N$. Radi lakšeg računanja te kako bi se izbijegli premali brojevi, maksimizira se logaritam normalne distribucije. Uzimanjem prirodnog logaritma te korištenjem identiteta $log(ab) = log(a) + log(b)$, dobiva se:

$$\log N(y;\mu,\sigma) = \log\left[\frac{1}{\sigma\sqrt{2\pi}}\right] - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

Moguće je odstraniti logaritam s desne strane jednakosti na slijedeći način:

$$\log\left[\frac{1}{\sigma\sqrt{2\pi}}\right] = \log\frac{1}{\sigma} + \log\frac{1}{\sqrt{2\pi}} = \log\frac{1}{\sigma} + C = -\log\sigma + C$$

Budući da su aditivne konstante suvisle u funkciji gubitka, zanemaruje se $+C$ te se dobiva:

$$\log N(y;\mu,\sigma) = -\log\sigma - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

Problem maksimizacije funkcije $\log N$ jednak je problemu minimizacije funkcije $-\log N$. Kako je konvencija u neuronskim mrežama riješiti problem minimizacije, dobiva se sljedeća funkcija gubitka za jedan uzorak y:

$$-\log N(y;\mu,\sigma) = \frac{1}{2}\left[\log\sigma^2 + \frac{(y-\mu)^2}{\sigma^2}\right]$$

Kako bi dobili konačnu funkciju gubitka za lokalizaciju objekata, uzima se aritmetička sredina grešaka svih uzoraka:

$$\mathcal{L}_{NLL} = \mathop{\mathbb{E}}_{X,Y}\left[\log\sigma(X)^2 + \frac{(Y-\mu(X))^2}{\sigma(X)^2}\right], \tag{4.1}$$

gdje su μ i σ^2 funkcije ovisne o ulaznim podatcima X, sa stvarnim vrijednostima Y.

Pri implementaciji $\mathcal{L}_{NLL}$ dodaje se parametar ϵ kako bi se izbjegle numeričke greške pri izračunu logaritma i nazivnika u razlomku. Uz pomoć TensorFlow aplikacijskog sučelja za detekciju objekata, $\mathcal{L}_{NLL}$ je moguće implementirati na slijedeći način:

```
import tensorflow as tf
from object_detection.core.losses import Loss

class GaussianNLLLoss(Loss):
    def _compute_loss(self,
                      pred_mean,
                      pred_log_var,
                      tgt_mean,
                      eps):
        var = tf.exp(pred_log_var)

        log_loss = tf.math.log(tf.math.maximum(var, eps))
        frac_loss = (pred_mean - tgt_mean) ** 2 / tf.math.maximum(var,
                                                  eps)

        return 0.5 * tf.reduce_mean(log_loss + frac_loss)
```

Code Listing 4.1: Gaussian Negative Log-Likelihood gubitak

4.4 Treniranje mreže

Gradijenti $\mathcal{L}_{NLL}$ su:

$$\partial \mathcal{L}_{NLL}/\partial \mu = \mathop{\mathbb{E}}_{X,Y}\left[\frac{Y-\mu(X)}{\sigma^2(X)}\right] \tag{4.2}$$

$$\partial \mathcal{L}_{NLL}/\partial \sigma^2 = \mathop{\mathbb{E}}_{X,Y}\left[\frac{1}{2\sigma^2(X)} - \frac{Y-\mu(X)}{2\sigma^4(X)}\right] \tag{4.3}$$

Treniranje isključivo s $\mathcal{L}_{NLL}$ pokazalo se neuspješnim. Iako se iz gradijenta funkcije može vidjeti da se μ kreće u smjeru stvarnih vrijednosti Y, parametri modela konvergiraju u neoptimalna rješenja. Primjer toga vidljiv je na slici 4.3.

U [18] ustanovljena su dva glavna razloga takvog ponašanja:

- Početna spljoštenost prostora značajki može stvoriti nedovoljno kompleksno, ali lokalno stabilno prilagođavanje aritmetičke sredine.
- $\mathcal{L}_{NLL}$ smanjuje vrijednosti gradijenta loše predviđenih granica zahvaljujući dobro predviđenim granicama. Takvo ponašanje postaje sve jače izraženo tijekom treninga.

Slika 4.3: Neoptimalna konvergencija $\mathcal{L}_{NLL}$

U istome članku, predložena je $\beta-NLL$ metoda gdje se $\mathcal{L}_{NLL}$ množi sa $stop\_gradient(\sigma^{2\beta})$, gdje je $stop\_gradient$ funkcija koja sprječava propagaciju gradijenata unatrag, tretirajući svoj unos kao konstantu pri računanju gradijenata. Na taj se način odabirom parametra $\beta \in [0, 1]$ dobiva interpolacija gubitka između NLL i MSE.

U konačnici ispada da ni ta metoda nije uspješna te se pojavljuju isti problemi kao i kod $\mathcal{L}_{NLL}$ te su problemi izraženi i kod visokih vrijednosti $\beta = 0.9$.

Tok treninga

Originalna funkcija gubitka lokalizacije korištena pri treniranju težina cijelog modela na COCO skupu podataka naziva se *WeightedL2LocalizationLoss* te ima formulu:

$$\mathcal{L}_{WeightedL2} = \sum_{i=1}^{N} w_i \parallel y_i - x_i \parallel^2, \tag{4.4}$$

gdje je vrijednost w_i jednaka 1 za regije koje sadrže stvarni pravokutni okvir, a 0 ukoliko nije riječ o stvarnom pravokutnom okviru.

Originalna funkcija gubitka klasifikacije naziva se *Focal Loss* te je po prvi put predstavljena u istome članku kao i *RetinaNet* arhitektura [10]. Može se interpretirati kao dinamička greška unakrsne entropije gdje se dodavanjem dodatnog parametra za skaliranje smanjuje učenje na bolje klasificiranim a povećava učenje na loše klasificiranim primjerima. Računa se na slijedeći način:

$$\mathcal{L}_{Focal} = -\sum_{i=1}^{N}(i - p_i)^{\gamma}\log(p_i), \tag{4.5}$$

gdje je γ parametar postavljen ručno (najčešće vrijednosti su 0.5, 1, 2).

Treniraju se samo konvolucijske glave modela (jedna za klasifikaciju i dvije za lokalizaciju) dok se ostatak mreže zamrzne.

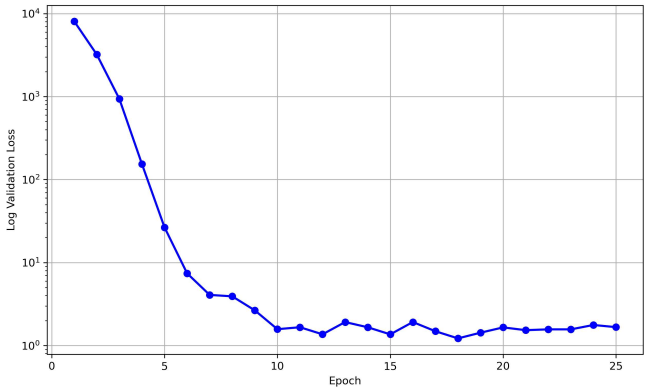
Glave za lokalizaciju objekata prethodno su predtrenirane na tri epohe s funkcijom gubitka $\mathcal{L}_{WeightedL2}$. Zatim je model istreniran na 5 epoha s lokalizacijskim gubitkom $\mathcal{L}_{NLL}$ sa *stop_gradient* funkcijom na parametru μ kako bi se izbjegli prethodno navedeni problemi. Konačno, model je treniran sa smanjenim *learning_rate* parametrom kombinirajući $\mathcal{L}_{WeightedL2}$ i $\mathcal{L}_{NLL}$ u omjeru 90:10. Kako bi se izbjegli ekplodirajući gradijenti za male vrijednosti σ^2, korišteno je rezanje gradijenata izvan vrijednosti intervala $[-0.1, 0.1]$. Kao gubitak lokalizacije na skupu za validaciju koristi se isključivo $\mathcal{L}_{NLL}$. Klasifikacija se trenira istovremeno kao lokalizacija s originalnom funkcijom gubitka *Focal Loss*. Kao optimizator uzima se *Adam*.

Kao najbolji model uzima se *checkpoint* na kojem je validacijska greška $\mathcal{L}_{NLL} + \mathcal{L}_{Focal}$ najmanja te je trening prekinut nakon 7 epoha bez poboljšanja modela. Greška lokalizacije na validacijskom skupu podataka tijekom treninga prikazana je slikom 4.4.

Ogledni primjer korištenog koda nalazi se na:

`https://github.com/082T/KvantifikacijaStabilnostiOgledniPrimjer`

Ovim kodom se može reproducirati eksperiment čije rezultate smo prikazali na slici 4.3. Ovaj primjer pokazuje način na kojega se u paketu tensorflow može implementirati optimizacija parametara neuroske mreže uz korištenje negativne Gaussove logaritamske funkcije izglednosti.



Slika 4.4: Prosječna vrijednost $\mathcal{L}_{NLL}$ na validacijskom skupu podataka

4.5 Rezultati

Kvaliteta detekcija

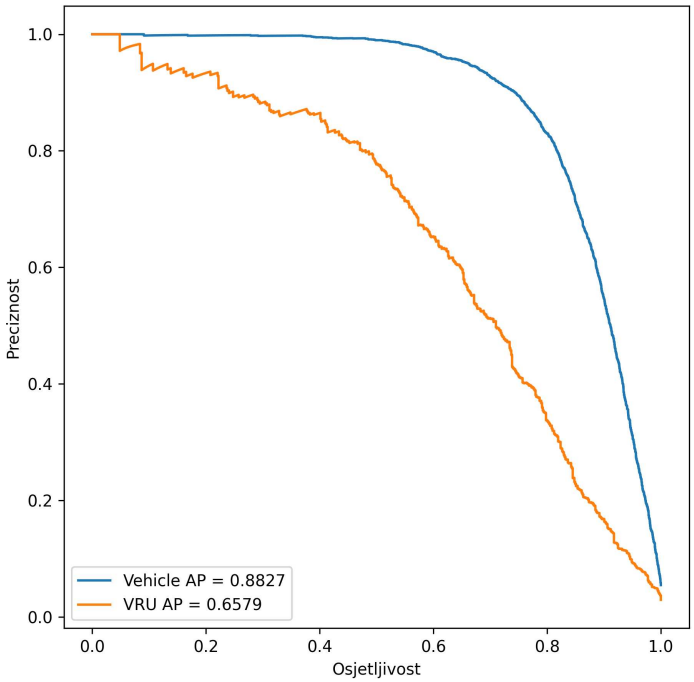
Uporabom spremljenog modela s najmanjom $\mathcal{L}_{NLL}$ greškom na validacijskom skupu podataka te primjenom NMS algoritma za $IoU > 0.6$, dobiveni su slijedeći rezultati na skupu za testiranje:

IoU granica	AP za klasu ”Vehicle”	AP za klasu ”VRU”	mAP
0.50	0.8827	0.6579	0.7703
0.75	0.8215	0.4780	0.6497
0.90	0.3011	0.1963	0.2487
.5:.95 (21)	0.6879	0.4308	0.5594

Na slici 4.5 dane su krivulje preciznosti u ovisnosti o osjetljivosti modela za IoU granicu 0.5.

Originalan model istreniran isključivo s $\mathcal{L}_{WeightedL2}$ postiže vrijednost $mAP@[.5 : .95] = 0.5612$ (+ 0.0018) što pokazuje zanemariv gubitak u sposobnosti detektiranja objekata pomoću eksperimentalnog modela.
U nastavku, detektirani objekti su oni kojima je razina pripadanja pojedinoj klasi veća ili jednaka 0.4. Uspješno detektirani objekti su oni objekti koji su detektirani i kojima je IoU preklapanje sa stvarnim objektom veće ili jednako 0.5. Za takve detekcije dobivaju se sljedeći rezultati:

$\#\mathbf{TP}_{Vehicle} = 3978$
$\#\mathbf{FP}_{Vehicle} = 1140$
$\#\mathbf{FN}_{Vehicle} = 1086$
$\#\mathbf{TP}_{VRU} = 433$
$\#\mathbf{FP}_{VRU} = 252$
$\#\mathbf{FN}_{VRU} = 434$



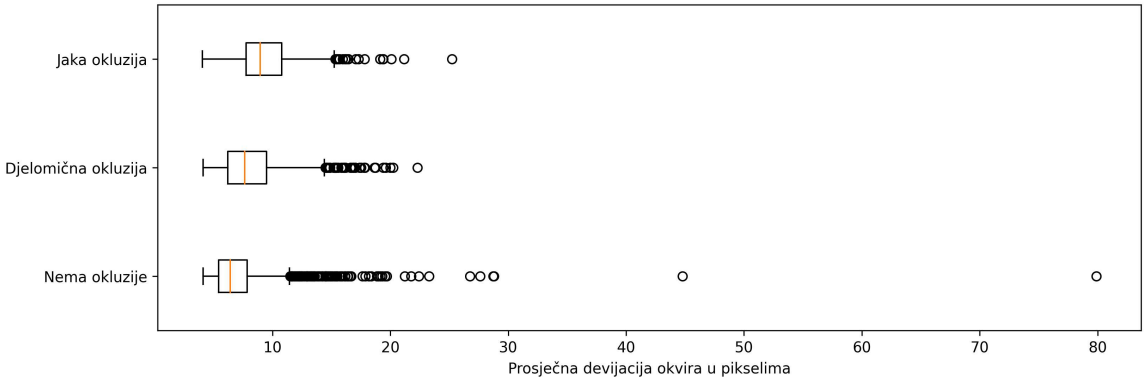
Slika 4.5: Krivulja preciznosti u ovisnosti o osjetljivosti za IoU granicu 0.50

Detekcija okluzije

Kako je opisano u poglavlju 4.1, stvarne vrijednosti okluzija objekata označene su s ”nema okluzije”, ”djelomična okluzija”, ”velika okluzija”, ”nepoznata okluzija”. Budući da je cilj da model sam nauči postoji li okluzija ili ne, te vrijednosti nisu korištene tijekom treninga modela, ali su proučavane na skupu za testiranje.

U skupu podataka navedeno je samo postoji li okluzija ili ne, a ne s koje je strane vozilo prekriveno, stoga se uspoređuje prosječna standardna devijacija okvira ($(\sigma_{x_{min}} + \sigma_{x_{max}} + \sigma_{y_{min}} + \sigma_{y_{max}})/4$) s navedenim stvarnim vrijednostima.

Za klasu *Vehicle* sveukupno je uspješno detektirano 3978 objekata, od kojih 2080 pripada klasi "Nema okluzije", 1129 pripada klasi "Djelomična okluzija", 575 pripada klasi "Jaka okluzija" te za 194 objekta okluzija nije poznata.

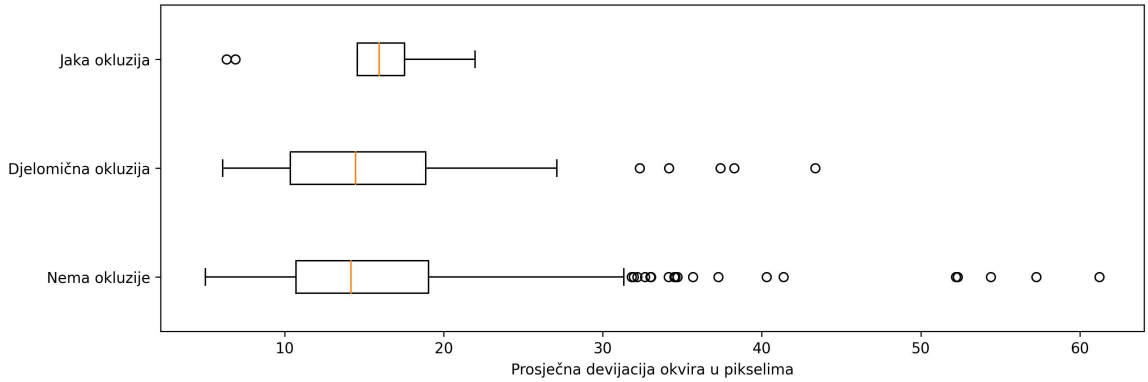


Slika 4.6: Boxplotovi prosječne devijacije u ovisnosti o stvarnoj okluziji za klasu *Vehicle*

Iz rezultata prikazanih slikom 4.6 može se zaključiti da model uspješno određuje okluziju na klasi *Vehicle*. S vrijednosti od 8.98, medijan prosječne standardne devijacije okvira detektiranih vozila s jakom standardnom devijacijom je veći od 75. centila prosječne devijacije detektiranih vozila bez okluzije.

Stvarna vrijednost	Minimum	25. centil	Medijan	75. centil	Maksimum
Nema okluzije	4.12	5.45	6.42	7.85	79.89
Srednja okluzija	4.11	6.22	7.66	9.49	22.31
Jaka okluzija	4.06	7.76	8.98	10.78	25.23

Za klasu *VRU* sveukupno su detektirana 433 objekta, od kojih 325 pripada klasi "Nema okluzije", 80 pripada klasi "Djelomična okluzija", 9 pripada klasi "Jaka okluzija" te za 19 objekata okluzija nije poznata. Rezulati dobiveni na tim detekcijama prikazani su na slici 4.7.



Slika 4.7: Boxplotovi prosječna devijacije u ovisnosti o stvarnoj okluziji za klasu *VRU*

Stvarna vrijednost	Minimum	25. centil	Medijan	75. centil	Maksimum
Nema okluzije	5.00	10.71	14.15	19.03	61.23
Srednja okluzija	6.10	10.34	14.44	18.86	43.35
Jaka okluzija	6.34	14.56	15.94	17.52	21.96

Zbog niskog broja detektiranih objekata klase *VRU* sa srednjom ili jakom okluzijom, testirane su i vrijednosti standardne devijacije za sve točno lokalizirane objekte (koji nisu nužno zadovoljili prag pripadanja klasi *VRU* ali im je IoU preklapanje sa stvarnim okvirom > 0.5). Tako se dobivaju vrijednosti medijana srednje standardne devijacije redom: 15.07, 15.76, 17.65. Takvih okvira ima 450 za nepostojeću, 168 za srednju te 47 za jaku okluziju.

Osim slabije ovisnosti stabilnosti pravokutnog okvira o okluziji, dobivene standardne devijacije klase *VRU* veće su u odnosu na standardne devijacije klase *Vehicle*. Glavni uzrok slabijih rezultata je znatno manji broj objekata klase *VRU* u podatcima za trening (4512 objekata klase *VRU* naspram 22612 objekata klase *Vehicle*). Potencijalni uzrok manje stabilnosti je česta pojava gusto raspoređenih pješaka u podatcima za trening. Također, budući da se pješak i biciklist svrstavaju u jednu klasu, veća nesigurnost može biti uzrokovana pogrešnim detekcijama biciklista izbacujući granice bicikla iz lokalizacije.

Brzina modela

Brzina eksperimentalnog i originalnog modela uspoređuje se na grafičkoj kartici Nvidia GeForce RTX 2060. Modeli su testirani jedan za drugim s 10 prolazaka kroz mrežu, 10

puta u petlji. Dobiveni rezultati su:

- $\mu = 168ms, \sigma = 13.25$ za originalni model
- $\mu = 169ms, \sigma = 9.52$ za eksperimentalni model

Iz dobivenih rezultata može se zaključiti da je vremenski gubitak dodavanjem nove glave za kvantifikaciju stabilnosti lokalizacije objekata zanemarivo usporio model.

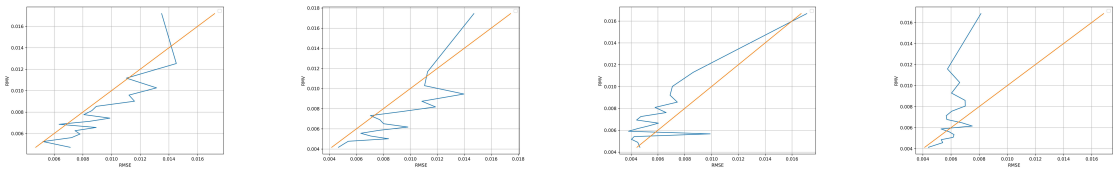
Kalibracija lokalizacije okvira

Kalibracija regresije modela govori koliko je točno σ usklađen sa stvarnom greškom. Za računanje kalibracije koriste se $RMSE$ i RMV kako je opisano u 3.6.

Tijekom treninga modela, kako bi se izbjegle preniske vrijednosti varijance σ^2, funkcija greške je podešena da izbacuje varijancu u stvarnim pikselima, dok je μ skaliran između 0 i 1. Stoga, za izračun kalibracije, varijanca se pretvara u skalirane koordinate:

$$\sigma^2_{new} = \left(\sqrt{\sigma^2_{old}}/1024 \right)^2$$

Rezultati kalibracije za klasu *Vehicle* prikazani su na slici 4.8 dok su rezultati kalibracije za klasu *VRU* prikazani slikom 4.9.

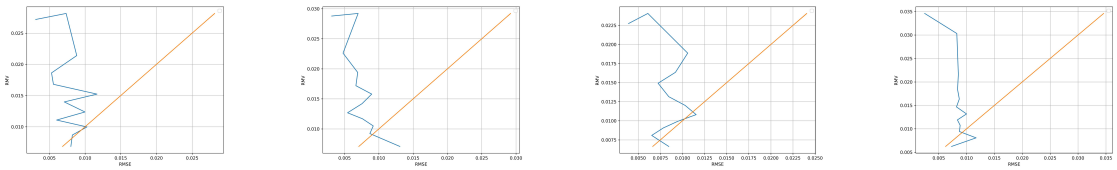


(a) (b) (c) (d)

Slika 4.8: RMSE kao funkcija od RMV granica pravokutnog okvira za klasu *Vehicle*. Redom: gornja, donja, lijeva, desna

Model je dobro kalibriran ukoliko prati liniju $RMSE = RMV$. Za klasu *Vehicle* može se primjetiti da su gornja i donja granica blago ispod krivulje što označava preveliku samouvjerenost modela. Lijeva i desna granica su obje iznad $RMSE = RMV$, što ukazuje na prenisku sigurnost modela u svoje vrijednosti μ što je posebno izraženo za desnu granicu okvira.

Na klasi *VRU*, model je puno lošije kalibriran. Razlozi toga opisani su u 4.5.



(a) (b) (c) (d)

Slika 4.9: RMSE kao funkcija od RMV granica pravokutnog okvira za klasu *VRU*. Redom: gornja, donja, lijeva, desna

Vizualni rezultati

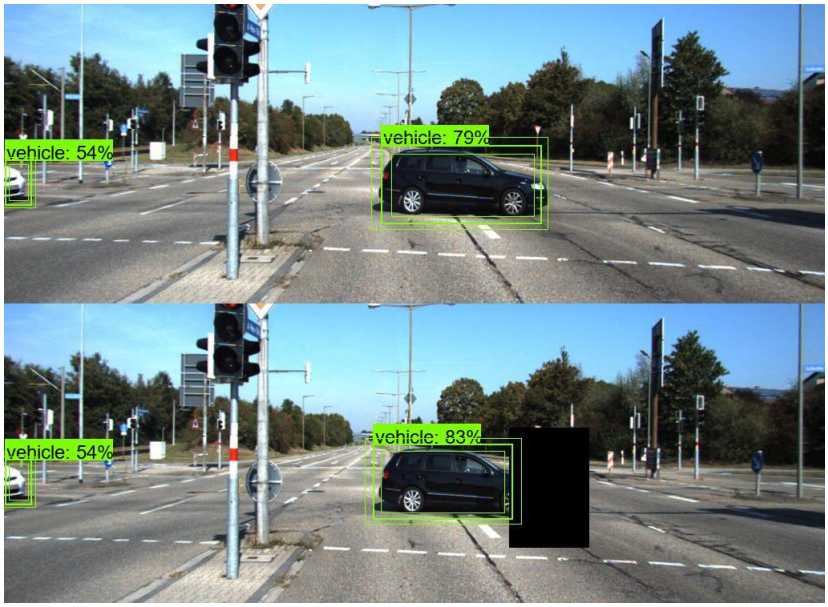
Svaki detektirani objekt na slici bit će vizualiziran s tri pravokutna okvira. Srednji okvir predstavlja vrijednosti μ za svaki parametar pravokutnog okvira. Vanjski okvir predstavlja okvir kreiran vrijednostima μ pomaknutim za pripadnu vrijednost σ prema vanjskoj strani objekta. Analogno vrijedi i za unutarnji okvir.

Kako ne postoje stvarne vrijednosti okluzije za svaku pojedinu granicu okvira objekta, ovisnost stabilnosti pojedine granice prikazat će se vizualno na umjetno kreiranom primjeru.

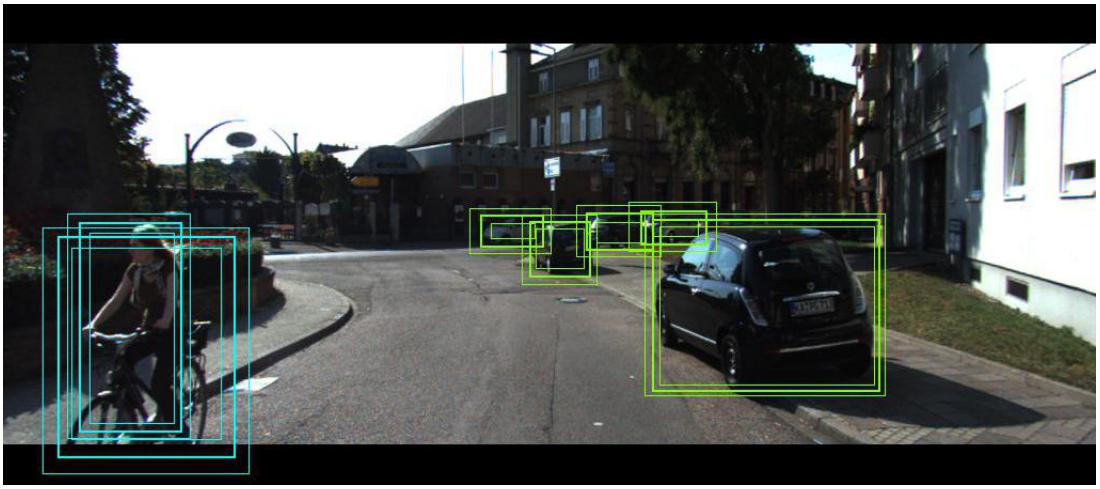
Na 4.10 prikazane su dvije slike jedna ispod druge. Na donjoj slici dodan je crni pravokutnik koji prekriva desnu stranu automobila. Taj automobil originalno ima vrijednosti standardne devijacije jednake 10.4, 12.0, 7.8 te 8.5 piksela redom za gornju, lijevu, donju i desnu stranu.

Nakon prekrivanja desne strane automobila crnim pravokutnikom, nove vrijednosti standardnih devijacija su redom: 7.6, 9.9, 7.2, 10.4 piksela. Model je uspješno prepoznao okluziju na desnoj strani auta te smanjio stabilnost detekcije na tome djelu. U navedenom primjeru također se može primjetiti da iako postoji korelacija između vrijednosti pripadanja klasi i okluzije, model je povećao sigurnost u pripadanje klasi ”Vehicle” okludiranog objekta.

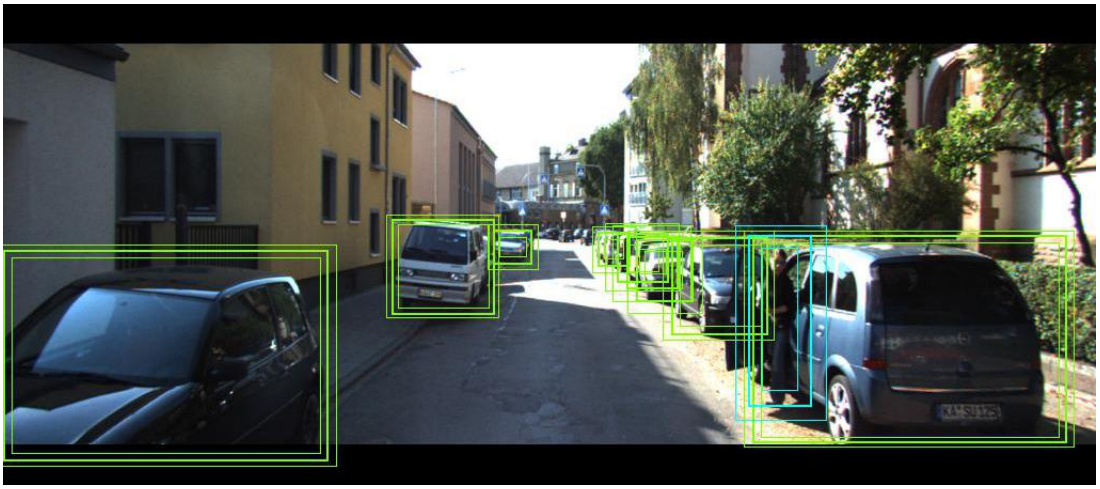
Slikama 4.11 i 4.12 prikazani su zanimljivi primjeri lokalizacija objekata. Radi bolje vidljivosti, maknute su vrijednosti pripadanja klasi.



Slika 4.10: Gornja slika predstavlja originalnu detekciju. Donja slika predstavlja detekciju nakon što je dio slike prekriven crnim pravokutnikom



Slika 4.11: Primjer dvostruke detekcije biciklista. Dok detekcija same osobe ima veću stabilnost pravokutnog okvira, detekcija bicikla izlazi van granica slike te joj je stabilnost okvira smanjena.



Slika 4.12: Auti parkirani s lijeve i desne strane ceste. Auti parkirani s lijeve strane u potpunosti su vidljivi te im je stabilnost okvira veća. Auti parkirani s desne strane gusto su raspoređeni i stabilnost okvira im je manja.

4.6 Zaključak

U ovom istraživanju uspješno je implementirana metoda kvantifikacije stabilnosti pravokutnog okvira objekta. Testirana je ovisnost devijacije okvira o okluziji te su dobiveni obečavajući rezultati. Testirana je kalibracija lokalizacije detekcija koja je ukazala da postoji prostor za poboljšanje modela, što može biti posljedica količine skupa podataka za treniranje.

Testirana je brzina modela te nije uočen značajniji gubitak brzine. Dodavanje glave za kvantifikaciju stabilnosti lokalizacije objekta na originalni model pokazalo se korisnim te nije oštetilo ostale mogućnosti modela.

4.7 Daljnje istraživanje

U svrhu daljnjeg istraživanja, može se testirati efikasnost $\mathcal{L}_{NLL}$ na drugim modelima poput Faster-RCNN. Također se može testirati *var voting* metoda kao zamjena za NMS [6]. Mogu se isprobati razne distribucije granica pravokutnog okvira, npr. poput Cauchyjeve. Također se može koristiti D_{KL} kao funkcija gubitka. Pritom se distribucije stvarnih vrijednosti mogu ručno postaviti (na distribuciju s niskim odskocima od μ) ili prethodno procjeniti Monte Carlo isključivanjem.

Bibliografija

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu i Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015, `https://www.tensorflow.org/`, Software available from tensorflow.org.

[2] David M. Blei, Alp Kucukelbir i Jon D. McAuliffe, *Variational Inference: A Review for Statisticians*, Journal of the American Statistical Association **112** (2017), br. 518, 859–877, `https://doi.org/10.1080%2F01621459.2017.1285773`.

[3] Andreas Geiger, Philip Lenz i Raquel Urtasun, *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*, Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[4] L. Grubišić, *Konveksno programiranje*, Predavanje iz kolegija: *Inverzni problemi i strojni vid*, PMF (2023), `https://www.pmf.unizg.hr/math/predmet/ipsv`.

[5] Chuan Guo, Geoff Pleiss, Yu Sun i Kilian Q. Weinberger, *On Calibration of Modern Neural Networks*, 2017.

[6] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides i Xiangyu Zhang, *Bounding Box Regression with Uncertainty for Accurate Object Detection*, 2019.

[7] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama i Kevin Murphy, *Speed/accuracy trade-offs for modern convolutional object detectors*, 2017.

[8] Alex Kendall i Yarin Gal, *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?*, 2017.

[9] Dan Levi, Liran Gispan, Niv Giladi i Ethan Fetaya, *Evaluating and Calibrating Uncertainty Prediction in Regression Tasks*, 2020.

[10] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He i Piotr Dollár, *Focal Loss for Dense Object Detection*, 2018.

[11] Tsung Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick i Piotr Dollár, *Microsoft COCO: Common Objects in Context*, 2015.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu i Alexander C. Berg, *SSD: Single Shot MultiBox Detector*, Computer Vision – ECCV 2016, Springer International Publishing, 2016, str. 21–37, https://doi.org/10.1007%2F978-3-319-46448-0_2.

[13] Pitts Walter McCulloch, Warren S., *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 1943.

[14] Duvenaud D. Bettencourt J. Qinghui Yu J., Creager E., *Bayesian Neural Networks*, (2023), https://www.cs.toronto.edu/~duvenaud/distill_bayes_net/public/.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick i Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, 2016.

[16] Shaoqing Ren, Kaiming He, Ross Girshick i Jian Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, 2016.

[17] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological Review, vol. 65, no. 6, p. 386-408, 1958.

[18] Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic i Georg Martius, *On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks*, 2022.

Sažetak

U teorijskom djelu rada, opisane su neuronske mreže i njihove glavne komponente. Postepenom izgradnjom i uvođenjem novih koncepata, dolazi se do konvolucijskih neuronskih mreža te njihove uporabe u problemu detekcije objekta. Predstavljena je kvantifikacija stabilnosti lokalizacije objekta pravokutnim okvirom te su navedene najpopularnije metode kvantifikacije stabilnosti poput Bayesovih neuronskih mreža i Monte Carlo isključivanja.

U praktičnom djelu rada, nakon predstavljanja skupa podataka, opisana je metoda korištenja Gaussove negativne logaritamske vjerodostojnosti kao funkcije gubitka za učenje kvantifikacije stabilnosti lokalizacije objekta. Opisana je *RetinaNet* arhitektura korištena za treniranje modela s navedenom funkcijom greške. Provedena su različita testiranja te su dobiveni obečavajući rezultati.

Summary

In the theoretical part of the thesis, neural networks and their main components are described. By gradually building and introducing new concepts, we arrive to convolutional neural networks and their use in object detection problem. Quantification of bounding box stability is introduced and the most popular methods of stability quantification, such as Bayesian neural networks and Monte Carlo dropout are discussed.

In the practical part of the paper, after the presentation of the data set, the method of using Gaussian negative log-likelihood as a loss function for learning the quantification of object localization stability is presented. The *RetinaNet* architecture used to train the model with the specified loss function is described. Various tests were carried out and promising results were obtained.

Životopis

Rođen sam 8. ožujka 1999. godine u Zagrebu. Osnovnu školu završavam 2013. godine te upisujem XVIII. gimnaziju. Nakon položene mature, 2017. godine upisujem preddiplomski sveučilišni studij Matematika na matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu na kojem 2021. godine upisujem diplomski studij računarstva i matematike.

Ljubav prema matematici imam još od vrtića gdje sam volio recitirati potencije broja 2. Jedno od najranijih sjećanja mi je kad me dida učio broj π pokazujući mi kružnicu od role papira. Tijekom studija, sve više me počelo zanimati računarstvo te se u slobodno vrijeme bavim kodiranjem svega i svačega.

Tijekom diplomskog studija zapošljavam se u automobilskoj diviziji firme *VisageTechnologies* gdje usavršavam svoje znanje iz strojnog učenja.