Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gašper Spagnolo

Lokalizacija brezpilotnih letalnikov

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Luka Čehovin Zajc
Somentor: asist. Matej Dobrevski

Ljubljana, 2023

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Gašper Spagnolo
Naslov: Lokalizacija brezpilotnih letalnikov
Vrsta naloge: Diplomska naloga na univerzitetnem programu prve stopnje Računalništvo in informatika
Mentor: doc. dr. Luka Cehovin Zajc
Somentor: asist. Matej Dobrevski

Opis:
V zadnjem času postaja uporaba brezpilotnih letalnikov vse bolj razširjena in se uporablja v različnih področjih, kot so agrikultura, kartiranje, vojaške operacije in še mnogo drugih. Kljub njihovi vsestranskosti pa se poraja ključno vprašanje: kako se droni obnašajo, ko izgubijo stik z GPS sistemom? Diplomska naloga se osredotoča na to tematiko in predlaga metodo za lokalizacijo brezpilotnih letalnikov ob izgubi GPS signala.

Title: UAV localization

Description:
In recent times, the use of unmanned aerial vehicles (UAVs) has become increasingly prevalent, finding applications in various fields such as agriculture, mapping, military operations, and many others. Despite their versatility, a critical question arises: how do drones behave when they lose connection to the GPS system? This thesis focuses on this issue and proposes a method for localizing UAVs in the event of a GPS signal loss.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
UAV	unmanned aerial vehicle	brezpilotni letalnik
DNN	deep neural network	globoka nevronska mreza
CNN	convolutional neural network	konvolucijska nevronska mreza
RDS	relative distance score	ocena relativne razdalje
MSE	mean squared error	srednja kvadratna napaka
FPI	finding point in an image	iskanje tocke v sliki
WAMF	Weight-Adaptive Multi Feature fusion	Uteženo združevanje več značilk
PCPVT	Pyramid Vision Transformer with Conditional Positional encodings	Piramidni vizualni transformer s pogojnimi pozicijskimi kodiranji

Povzetek

Naslov: Lokalizacija brezpilotnih letalnikov

Avtor: Gašper Spagnolo

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja LaTeX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek. Dober povzetek vključuje: (1) kratek opis obravnavanega problema, (2) kratek opis vašega pristopa za reševanje tega problema in (3) (najbolj uspešen) rezultat ali prispevek diplomske naloge.

Ključne besede: Lokalizacija UAV, geo-lokalizacija, globoko učenje, transformer.

Abstract

Title: UAV localization

Author: Gašper Spagnolo

This sample document presents an approach to typesetting your BSc thesis using LaTeX. A proper abstract should contain around 100 words which makes this one way too short.

Keywords: UAV localization, geo-localization, deep learning, transformer.

Poglavje 1

Uvod

Brezpilotni letalniki so postali nepogrešljivo orodje v številnih sektorjih, od vojaških operacij do kmetijskega nadzora. Kljub njihovi široki uporabi pa se soočajo z izzivi pri avtonomni navigaciji, še posebej v okoljih, kjer je satelitski signal omejen ali nezanesljiv. V idealnih razmerah brezpilotni letalniki za svojo navigacijo uporabljajo GPS signale. Vendar pa lahko te signale motijo naravne in človeške ovire, kot so visoke stavbe, gorske formacije ali celo elektronske motnje. Izguba GPS signala lahko postane kritična, še posebej v tistih trenutkih, ko je natančna lokacija letalnika ključna za njegovo nalogo. Zato je iskanje alternativne metode za lokalizacijo brezpilotnih letalnikov postalo nujno.

Tradicionalne metode prepoznavanja slik se v kontekstu lokalizacije brezpilotnih letalnikov zdijo kot obetavna alternativa. Vendar pa se ob njihovi uporabi pojavi cela paleta izzivov. Prvič, potrebujemo ogromno slikovno bazo, ki vključuje kompresirane satelitske slike območij, nad katerimi letalnik leti. Velikost in obseg te baze lahko povzročita precejšnje računske in pomnilniške zahteve, kar lahko oteži njeno integracijo v realnočasovnih sistemih, kot so brezpilotni letalniki. Drugič, vsaka posodobitev ali sprememba v osnovni nevronski mreži, ki se uporablja za prepoznavanje slik, zahteva ponovno obdelavo celotne slikovne baze. To ne le da je časovno potratno, ampak tudi poveča stroške, saj morajo vse slike ponovno potekati skozi po-

stopek predprocesiranja in razpoznavanja. Tretjič, ko brezpilotni letalnik zajame sliko za primerjavo, mora ta slika biti primerjana z vsako sliko v bazi, da se ugotovi najboljše ujemanje. V praksi to pomeni, da če imamo bazo sestavljeno iz milijonov slik, bo vsaka nova poizvedovalna slika potrebovala milijone primerjav, kar je zelo časovno potratno in računsko intenzivno.

V luči omejitev tradicionalnih metod prepoznavanja slik so raziskovalci razvili inovativen pristop, imenovan FPI (Finding Point with Image) [4]. Ta pristop se močno razlikuje od običajnih metod v smislu strukture in delovanja. Osnova metode FPI je, da vzame dva vhodna podatka: sliko, posneto z dronom, in pripadajočo satelitsko sliko. V kontekstu te satelitske slike je mesto, kjer je bila dronska slika posneta.

Za obdelavo vsake slike se uporablja posebna nevronska mreža, kjer vsaka mreža obdeluje svoj nabor podatkov brez deljenja uteži z drugo. Ko sta obe sliki obdelani in njihove značilke ekstrahirane, se med njima izvede operacija korelacije. Ta korelacija se izraža v obliki toplotne karte, ki podaja podrobnosti o ujemanju med dronsko in satelitsko sliko. Najvišja vrednost na toplotni karti natančno označuje mesto, kjer je dron posnel svojo sliko na večji satelitski sliki. Ta informacija se nato neposredno prevede v natančno lokalizacijo drona na satelitski sliki.

Kot pogosto v znanosti, iz ene inovacije nastanejo druge. Nadgradnja metode FPI, znana kot WAMF-FPI, je dodatno izboljšala natančnost in učinkovitost lokalizacije dronov [4]. Ta metoda si je sposodila koncepte iz sledenja objektov in jih uporabila za lokalizacijo, kljub težavam, ki jih povzročajo razlike med slikami UAV in satelitskimi slikami. Z uporabo dveh različnih uteži za izvleček značilnosti UAV in satelitskih slik, WAMF-FPI omogoča natančnejše in bolj zanesljivo ujemanje slik. Dodatno izboljšanje je bilo doseženo z uporabo WAMF modula in Hanning loss-a, ki sta povečala učinkovitost modela.

WAMF-FPI je evolucija osnovne metode FPI in prinaša številne izboljšave pri procesiranju slik. Ključna prednost WAMF-FPI je njegova napredna piramidna struktura ekstrakcije značilk, ki omogoča bolj natančno in raznoliko

analizo vhodnih podatkov. Z uporabo te piramidne strukture se značilke ekstrahirajo na več različnih ravneh, nato pa se skalirajo in medsebojno primerjajo, kar pridobi bolj robusten in natančen sklop informacij. Poleg tega WAMF-FPI optimizira kompresijske zmogljivosti, kar pripomore k hitrejšemu in učinkovitejšemu procesiranju podatkov. Medtem ko je v osnovni FPI metodi končna velikost značilk bila stisnjena na 16-krat manjšo od izvorne satelitske slike, v WAMF-FPI ta kompresijski faktor znaša samo 4-krat manjšo velikost. To omogoča WAMF-FPI-ju, da ohrani več informacij ter pridobi boljšo lokalizacijsko natančnost ob hkratnem zmanjšanju računske obremenitve.

Zaradi pomanjkanja dostopnih podatkovnih zbirk smo se odločili, da bomo ustvarili svojo. To smo storili s pomočjo Google Earth Studio. Naša zbirka vključuje 11 večjih evropskih mest z raznoliko strukturo. Cilj izdelave te zbirke je bil zagotoviti raznolike podatke, ki bi lahko služili kot robustna osnova za testiranje in validacijo naše implementacije WAMF-FPI. Zato smo se odločili, da bomo v tej diplomski nalogi sami implementirali WAMF-FPI, kakor je opisano v izvornem članku, in preverili njegovo delovanje. Implementirali smo vse, kakor je v članku opisano, z namenom dobiti objektivno sliko o učinkovitosti in natančnosti metode. V tej diplomski nalogi bomo podrobno raziskali te tehnike, njihove prednosti in pomanjkljivosti ter potencialne aplikacije in izboljšave za prihodnost. Ocenjevali bomo njihovo učinkovitost in natančnost, s poudarkom na njihovi uporabi v realnih scenarijih lokalizacije brezpilotnih letalnikov. Naš cilj je ponuditi temeljito analizo metode WAMF-FPI in njenih aplikacij, da bi olajšali nadaljnji razvoj in uporabo v industriji brezpilotnih letalnikov.

1.1 Osnonvi pojmi in terminologija

1.1.1 Brezpilotni letalnik

Brezpilotni letalniki, znani tudi kot droni, so daljinsko vodena ali avtonomna zračna plovila, opremljena s senzorji in GPS tehnologijo. Uporabljajo

se v različnih sektorjih, od vojaških do rekreativnih, in so postali ključni za zbiranje podatkov v realnem času, terenske raziskave in opravljanje potencialno nevarnih nalog. Njihova prilagodljivost omogoča uporabo v številnih aplikacijah, kot so inspekcije, filmska produkcija in dostava. Kljub hitremu razvoju in rasti trga pa tehnologija prinaša izzive, povezane z zasebnostjo, varnostjo in zakonodajo, kar spodbuja nadaljnje raziskave na tem področju.

1.1.2 Satelit

Sateliti so tehnološke naprave, ki krožijo okoli Zemlje ali drugih nebesnih teles. Uporabljajo se za različne namene, kot so komunikacija, vremensko opazovanje, navigacija in znanstvene raziskave. Komunikacijski sateliti omogočajo globalno povezovanje in prenos podatkov, medtem ko vremenski sateliti pomagajo pri napovedovanju vremena. Sistem GPS, ki temelji na navigacijskih satelitih, je postal ključen za določanje lokacije. Znanstveni sateliti prispevajo k razumevanju vesolja in Zemlje.

1.1.3 Geolokalizacija

Geolokalizacija je proces ugotavljanja geografske lokacije naprav, kot so mobilni telefoni ali vozila. Ključna je za GPS in navigacijske sisteme, omogoča ciljno usmerjanje oglasov, sledenje vozil, iskanje izgubljenih naprav in deljenje lokacij v socialnih omrežjih. Uporablja se tudi v znanstvenih raziskavah, kot je spremljanje selitve živali. Vendar natančnost varira glede na tehnologijo in okolje, prav tako pa se pojavljajo izzivi v zvezi z zasebnostjo in varnostjo.

1.1.4 Toplotna karta

Toplotna karta vizualno predstavlja podatke z barvami, ki odražajo vrednosti v matriki. Uporabljena je za vizualizacijo razporeditve spremenljivk v dvodimenzionalnem prostoru in je koristna v znanstvenih ter poslovnih aplikacijah, kot so statistika, biologija in geografija. Omogoča hitro razumevanje

kompleksnih podatkov, saj barvni prehodi razkrivajo vzorce in trende.

Poglavje 2

Metodologija

V svetu računalniškega vida in strojnega učenja je razvoj učinkovitih metod za lokalizacijo specifičnih točk ali značilnosti na sliki ključnega pomena za številne aplikacije. V okviru pristopa WAMF-FPI je ta problematika še posebej izpostavljena, saj se osredotoča na natančno določanje točke na sliki. Da bi to dosegli, je potrebno uporabiti kombinacijo različnih arhitektur in tehnik, ki so se izkazale kot učinkovite v različnih scenarijih obdelave slik.

V tem poglavju bomo predstavili osnovne komponente, ki jih uporabljamo v našem modelu. Začeli bomo s konvolucijskimi nevronskimi mrežami, ki so temeljni gradnik večine modelov za obdelavo slik in nudijo močno orodje za izluščenje značilnosti iz vizualnih podatkov. Nadaljevali bomo s predstavitvijo transformerske arhitekture, ki je revolucionirala področje obdelave naravnega jezika in se v zadnjem času vedno bolj uporablja tudi v računalniškem vidu. Podrobneje se bomo osredotočili na zgradbo transformerja in njegove ključne komponente. V nadaljevanju bomo razpravljali o Vision Transformerju (ViT) in njegovi razširjeni verziji - Piramidnem Vision Transformerju (PVT). Posebno pozornost bomo posvetili tudi prilagojeni različici PVT, ki upošteva lokalne značilnosti, imenovani PCPVT. Zaključili bomo s siamskimi nevronskimi mrežami, ki predstavljajo ključno komponento pri primerjavi vzorcev. Te mreže so še posebej pomembne, ko želimo primerjati dva ali več podobnih vzorcev in ugotoviti, ali med njimi obstajajo razlike.

Z vključitvijo vseh teh komponent in tehnik v naš model WAMF-FPI želimo razviti robusten in natančen sistem za lokalizacijo točk na slikah. V nadaljevanju poglavja bomo vsako od teh komponent podrobno raziskali, da bi bolje razumeli njihove lastnosti in kako prispevajo k celotnemu modelu.

2.1 Konvolucijske nevronske mreže

Konvolucijske nevronske mreže (CNN – Convolutional Neural Networks) so metoda globokega učenja, specializirana za obdelavo vizualnih podatkov, zasnovana tako, da avtomatsko in adaptivno izvaja izvleček značilnosti iz slik.

2.1.1 Struktura in delovanje

Osnovna struktura CNN vključuje štiri glavne vrste plasti: konvolucijsko, aktivacijsko, združevalno (pooling) in polno povezano plast.

1. **Konvolucijska plast**: Vsak nevron v tej plasti je povezan le z majhnim območjem v prejšnji plasti, namesto da bi bil povezan z vsemi nevroni, kot je to v običajnih nevronskih mrežah. Ko se filter (ali jedro) premika preko slike, izvaja konvolucijsko operacijo:
$$(I * K)(x, y) = \sum_m \sum_n I(m, n) \cdot K(x - m, y - n) \tag{2.1}$$
2. **Aktivacijska funkcija**: Po konvolucijski operaciji se uporabi aktivacijska funkcija za vsak izhod. Najpogosteje se uporablja funkcija `ReLU`:
$$\text{ReLU}(x) = \max(0, x) \tag{2.2}$$
3. **Združevalna plast**: Po konvolucijski in aktivacijski operaciji sledi združevalna plast, ki zmanjšuje dimenzije slike z uporabo operacij, kot je "max pooling":
$$P(i, j) = \max_{m,n \in R} I(i + m, j + n) \tag{2.3}$$
4. **Polno povezane plasti**: Delujejo kot klasične plasti v običajnih nevronskih mrežah. Vsak nevron je povezan z vsemi izhodi prejšnje plasti.

2.1.2 Učenje in optimizacija

Glavni mehanizem učenja v CNN je nazajno širjenje napak (backpropagation). To je iterativna metoda, ki minimizira kriterijsko funkcijo, da bi ugotovili optimalne uteži mreže. Za mnoge naloge v obdelavi slik je kvadratna napaka (MSE) izbrana kot kriterijska funkcija:

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.4}$$

Za optimizacijo uteži se uporablja gradientni sestop. Za vsako iteracijo se uteži posodobijo v smeri negativnega gradienta kriterijske funkcije:

$$w_{\text{novi}} = w_{\text{stari}} - \eta \nabla J(w_{\text{stari}}) \tag{2.5}$$

2.1.3 Značilnosti in prednosti

Konvolucijske mreže so sposobne avtomatskega zaznavanja hierarhičnih značilnosti. Na nižjih ravneh mreže se zaznavajo nizkonivojske značilnosti, kot so robovi in teksture, na višjih ravneh pa se zaznavajo kompleksnejše strukture, kot so oblike in objekti. Ta hierarhična značilnost je tisto, kar omogoča CNN, da doseže izjemno natančnost pri različnih nalogah obdelave slik.

2.2 Transformerska arhitektura

2.2.1 Predhodni mehanizmi

Preden so obstajali transformatorji, so bile najpogostejše metode za obvladovanje zaporedij v jezikovnih modelih rekurentne nevronske mreže (RNN) in njihove različice, kot so dolgokratni kratkotrajni spomini (LSTM) in obogatene RNN (GRU). Najpogostejša uporaba teh modelov v kontekstu strojnega

prevajanja ali drugih nalog pretvarjanja zaporedja v zaporedje je bila uporaba strukture kodirnik-dekodirnik. V tej strukturi je bilo zaporedje vhodnih besed ali tokenov kodirano v latentni prostor z uporabo RNN (kodirnik), ta latentni vektor pa je bil nato uporabljen za generiranje zaporedja izhodnih besed ali tokenov z uporabo drugega RNN (dekodirnik). Problem s to strukturo je bil, da je bil latentni prostor omejen na velikost fiksne dolžine in je moral vsebovati vse informacije iz izvornega zaporedja, ki so potrebne za generiranje ciljnega zaporedja. To je omejevalo model pri obvladovanju dolgih zaporedij, saj je bilo težko ohraniti informacije iz zgodnjega dela zaporedja do konca. Da bi to težavo rešili, so raziskovalci vključili mehanizem pozornosti, ki je omogočil dekodirniku, da se osredotoči na različne dele izvornega zaporedja na različnih stopnjah generiranja ciljnega zaporedja. To je bil velik napredek, ki je omogočil boljše obvladovanje dolgih zaporedij.

Članek, ki je predstavil to idejo za strojno prevajanje, je bil "Neural Machine Translation by Jointly Learning to Align and Translate"[1], objavljen leta 2015. To je bil ključni korak k razvoju transformerske arhitekture, ki je bila kasneje predstavljena v članku "Attention is All You Need"[7] leta 2017.

2.2.2 Razlaga RNN kodirnik-dekodirnik arhitekture

Definirajmo problem strojnega prevajanja kot iskanje najboljše ciljne sekvence $\vec{E} = (e_0, e_1, ..., e_m)$ glede na dane izvorne besede $\vec{F} = (f_0, f_1, ..., f_n)$. Ta problem lahko izrazimo kot optimizacijo pogojne verjetnosti $P(\vec{E}|\vec{F})$. Začnimo z opisom RNN-kodirnik-dekodirnik arhitekture. Imamo dva RNN modela, kodirnik RNNenc in dekodirnik RNNdec. Kodirnik z zaporedjem vektorjev $\vec{F}$ proizvede skrito stanje h_n:

$$h_n = \text{RNNenc}(f_n, hn - 1) \tag{2.6}$$

Začetno stanje h_0 je pogosto postavljeno na nič ali se ga mreža nauči. Dekodirnik nato uporablja to skrito stanje, da generira ciljno zaporedje $\vec{E}$:

$$e_t = \text{RNNdec}(et - 1, h_{t-1}) \tag{2.7}$$

Opomba: pri učenju se za e_{t-1} pogosto uporablja dejanska vrednost iz ciljnega zaporedja (ne izhod modela), kar je znano kot "teacher forcing". Izvorna zaporedja besed $\vec{F}$ se tako vnašajo v kodirnik, ki generira skrita stanja za vsako besedo:

$$\vec{H} = \text{Kodirnik}(\vec{F}) \tag{2.8}$$

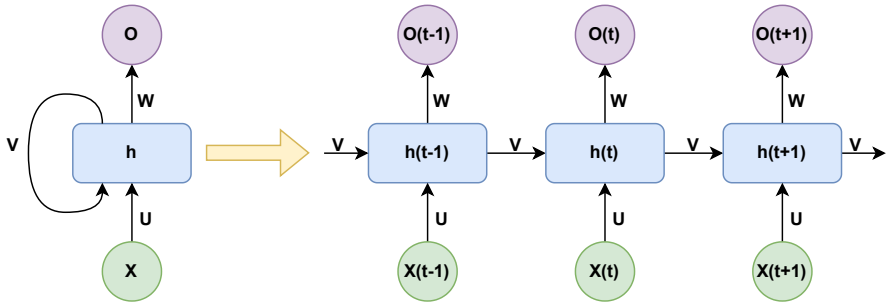
Za vsako besedo v ciljnem zaporedju $\vec{E}$ se potem izračuna ponderirana vsota skritih stanj iz kodirnika:

$$\vec{a}t = \text{Pozornost}(\vec{H}, et - 1) \tag{2.9}$$

Potem se ta vektor uporabi za napoved ciljne besede:

$$e_t = \text{Dekodirnik}(\vec{a}t, et - 1) \tag{2.10}$$

Ta pristop omogoča, da dekodirnik upošteva vse besede v izvornem zaporedju, ne samo prejšnje besede v ciljnem zaporedju, kar izboljša kakovost prevoda. Vendar je to zgolj matematična formulacija koncepta. Dejanski detajli, kot so vrste in struktura kodirnika in dekodirnika, so odvisni od specifičnega modela, ki ga uporabljamo.



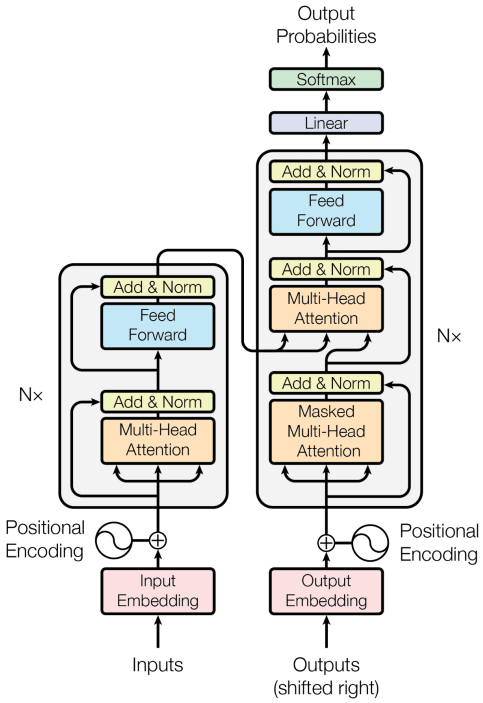
Slika 2.1: Skica RNN modela

2.3 Zgradba transformerja

V kontekstu strojnega prevajanja so avtorji v članku "Attention is all you need"[7] o pozornosti predstavili novo vrsto arhitekture, ki se izogiba mnogim

pastem modelov, ki temeljijo na RNN. Kljub vsem napredkom pri kodirnikih-dekodirnikih RNN, ki smo jih obravnavali zgoraj, je ostalo dejstvo, da so RNN težko paralelizirani, ker zaporedno obdelujejo vhod. Ključna inovacija tega članka je, da so RNN in njihova skrita stanja v celoti nadomeščeni z operacijami na osnovi pozornosti, ki so v mnogih problematičnih režimih bolj učinkoviti.

Transformer model je model kodirnika-dekodirnika. Kodirnik sestavljajo N blokov na levi, dekodirnik pa N blokov na desni, vidno na sliki 2.2.



Slika 2.2: Izgled transformerja, iz članka "Attention is all you need"[7].

Med učenjem se vhodne besede $\vec{F} = (f_0, ..., f_n)$ hkrati prenesejo v prvi blok kodirnika, izhod tega bloka pa se nato prenese v njegovega naslednika. Postopek se ponavlja, dokler vseh N blokov kodirnika ni obdelalo vhoda. Vsak blok ima dve komponenti: plast večglave samopozornosti, ki ji sledi polno povezana plast z aktivacijami ReLU, ki obdeluje vsak element vhodne sekvence vzporedno. Tako večglav sloj pozornosti kot polno povezana plast

sledita koraku *Dodaj in Normiraj* - *dodaj* se nanaša na residualno povezavo, ki doda vhod vsake plasti na izhod, *normiraj* pa se nanaša na normalizacijo plasti. Ko je vhod prešel skozi vse bloke kodiranja, ostane kodirana predstavitev $\vec{F}$.

Dekodirnik pa sestoji iz treh korakov: maske večglave samopozornosti, večglave plasti pozornosti, ki povezuje kodirano izvorno predstavitev z dekodirnikom, in polno povezane plasti z aktivacijami ReLU. Tako kot v kodirniku, vsaki plasti sledi plast *Dodaj in Normiraj*. Dekodirnik sprejme vse ciljne besede $\vec{E} = (e_0, ..., e_m)$ kot vhod. V procesu napovedovanja besede e_i ima dekodirnik dostop do prej generiranih besed. Ne more pa imeti dostopa do besed, ki sledijo e_i, saj te še niso bile generirane. Maskiranje med učenjem nam omogoča, da posnemamo pogoje, s katerimi se bo model soočil med sklepanjem. Obstaja nekaj ključnih razlik od kodirnika - ena je, da so vhodi v prvo operacijo pozornosti v blokih dekodirnika maskirani, zato ime plasti. To pomeni, da se lahko katera koli beseda v ciljnem izhodu nanaša samo na besede, ki so prišle pred njo. Razlog za to je preprost: med sklepanjem generiramo predvideni prevod $\vec{E}$ besedo za besedo z uporabo izvornega stavka $\vec{F}$.

Druga razlika od kodirnika je druga večglava plast pozornosti, ki se imenuje tudi plast pozornosti kodirnika-dekodirnika. Za razliko od plasti pozornosti na začetku blokov kodirnika in dekodirnika ta plast ni plast samopozornosti.

2.3.1 Utežena točkovna produktna pozornost

Utežena točkovna produktna pozornot (angl. Scaled Dot-Product Attention) se uporablja v vseh plasteh pozornosti v transformatorju. Za zdaj bomo razčlenili matematiko za to operacijo, le da dobimo občutek, katera števila gredo kam. Kasneje se bomo osredotočili na njegove aplikacije v članku [7].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.11}$$

Scaled Dot-Product Attention je skoraj identičen Dot-Product Attention-u, omenjenem prej pri Luongu [1]. Edina razlika je, da je vhod v softmax skaliran s faktorjem $\frac{1}{\sqrt{d_k}}$.

V članku in predhodni literaturi se vrstice $Q \in \mathbb{R}^{m \times d_k}$ imenujejo "poizvedbe", vrstice $K \in \mathbb{R}^{n \times d_k}$ "ključi", in končno vrstice $V \in \mathbb{R}^{n \times d_v}$ "vrednosti". Upoštevati je potrebno, da se za izvedbo mora število ključev in vrednosti n ujemati, vendar se lahko število poizvedb m razlikuje. Prav tako se mora dimenzionalnost ključev in poizvedb ujemati, vendar se lahko dimenzionalnost vrednosti razlikuje.

Izpeljimo izračun pozornosti. Začeli bomo z zapisom posameznih vrstic Q in K, nato pa bomo izrazili produkt QK^T v smislu teh vrstic:

$$\begin{aligned} Q &= \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_m \end{pmatrix}, \\ K^T &= \begin{pmatrix} k_0 & k_1 & \cdots & k_n \end{pmatrix}, \\ QK^T &= \begin{pmatrix} q_0 \cdot k_0 & q_1 \cdot k_0 & \cdots & q_m \cdot k_0 \\ q_0 \cdot k_1 & q_1 \cdot k_1 & \cdots & q_m \cdot k_1 \\ \vdots & \vdots & \ddots & \vdots \\ q_0 \cdot k_n & q_1 \cdot k_n & \cdots & q_m \cdot k_n \end{pmatrix}. \end{aligned} \tag{2.12}$$

Nato pridobimo naše uteži pozornosti tako, da vsak element delimo z $\sqrt{d_k}$ in uporabimo funkcijo softmax nad vsako vrstico:

$$\begin{aligned}\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) &= \begin{pmatrix}\text{softmax}\left(\frac{1}{\sqrt{d_k}}\cdot\langle q_0\cdot k_0, q_1\cdot k_0, \ldots, q_m\cdot k_0\rangle\right)\\ \text{softmax}\left(\frac{1}{\sqrt{d_k}}\cdot\langle q_0\cdot k_1, q_1\cdot k_1, \ldots, q_m\cdot k_1\rangle\right)\\ \vdots\\ \text{softmax}\left(\frac{1}{\sqrt{d_k}}\cdot\langle q_0\cdot k_n, q_1\cdot k_n, \ldots, q_m\cdot k_n\rangle\right)\end{pmatrix}\\ &= \begin{pmatrix}s_{0,0} & s_{1,0} & \cdots & s_{m,0}\\ s_{0,1} & s_{1,1} & \cdots & s_{m,1}\\ \vdots & \vdots & \ddots & \vdots\\ s_{0,n} & s_{1,n} & \cdots & s_{m,n}\end{pmatrix}^T\end{aligned} \tag{2.13}$$

kjer za vsako vrstico i, kot rezultat operacije softmax, velja $\sum_{j=0}^{n} s_{i,j} = 1$. Zadnji korak je množenje te matrike z V:

$$\begin{aligned}\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V &= \begin{pmatrix}s_{0,0} & s_{1,0} & \cdots & s_{m,0}\\ s_{0,1} & s_{1,1} & \cdots & s_{m,1}\\ \vdots & \vdots & \ddots & \vdots\\ s_{0,n} & s_{1,n} & \cdots & s_{m,n}\end{pmatrix}\begin{pmatrix}v_0\\ v_1\\ \vdots\\ v_n\end{pmatrix}\\ &= \begin{pmatrix}\sum_{i=0}^{m} s_{i,0}v_0\\ \sum_{i=0}^{m} s_{i,1}v_1\\ \vdots\\ \sum_{i=0}^{m} s_{i,n}v_n\end{pmatrix}\end{aligned} \tag{2.14}$$

Tukaj lahko opazimo, da mehanizem pozornosti je rezultat serije uteženih povprečij vrstic V, kjer uteži določajo vhodne poizvedbe in ključe. Vsaka od m poizvedb v Q rezultira v specifični uteženi vsoti vektorskih vrednosti. Pomembno je omeniti, da v tem posebnem postopku ni nobenih učljivih parametrov - sestavljen je izključno iz matričnih in vektorskih operacij. Poglejmo si še posamezno vrstico i uteži pozornosti:

$$\text{softmax}\left(\frac{1}{\sqrt{d_k}}\cdot Q\cdot K_i\right) = \frac{1}{S}\cdot\exp\left(\frac{Q\cdot K_i}{\sqrt{d_k}}\right) \tag{2.15}$$

kjer je S normalizacijska konstanta:

$$S = \sum_{j=0}^{m} \exp\left(\frac{q_j \cdot k_i}{\sqrt{d_k}}\right) \tag{2.16}$$

Če pogledamo, kako so uteži konstruirane, je izvor imen "poizvedbe", "ključi"in "vrednosti"jasnejši. Tako kot v zgostitveni tabeli ta operacija izbira želene vrednosti preko ustrezajočih, ena-na-ena ključev. Ključi, ki jih iščemo, so označeni s poizvedbami - skalarni produkt med danim ključem in poizvedbo lahko izrazimo kot θ med njima:

$$q_i \cdot k_j = |q_i||k_j| \cos(\theta) \tag{2.17}$$

Uporaba eksponentne funkcije povečuje pozitivne vrednosti kosinusa in zmanjšuje negativne. To pomeni, da če sta dva vektorja bolj poravnana (manjši kot med njima), bo njihova zastopanost v vektorju pozornosti večja. Nasprotno pa, če sta dva vektorja manj poravnana (večji kot med njima), bo njihova zastopanost v vektorju pozornosti manjša. To je smiselno, saj želimo, da model daje večjo pozornost tistim ključem, ki so bolj relevantni za dano poizvedbo. Zato je bližje kot sta si ključ $\vec{k_j}$ in poizvedba $\vec{q_i}$ po kotu, večja je njihova zastopanost v vektorju pozornosti.

Še ena pomanjkljivost, ki so jo raziskovalci opazili pri modelih, ki temeljijo na RNN (Recurrent Neural Networks) arhitekturi, je, da imajo težave z uporabo informacij iz elementov, ki so bili opaženi daleč v preteklosti. To je posledica tega, kar se imenuje "problem dolgih časovnih razdalj", kjer se informacije iz preteklih korakov postopoma izgubljajo skozi čas. Bolj splošno, RNN imajo težave s povezovanjem zaporednih informacij, ki so med seboj daleč narazen. Tehnike, kot so pozornost na skritih stanjih (attention on hidden states) in dvosmerni modeli (bidirectional models), so bile poskusi za odpravo te težave in so služile kot naravni prehod k tehnikam v tem članku.

Avtorji pozornosti omenjajo, da delijo vhode v softmax funkcijo z $\sqrt{(d_k)}$, da bi ublažili učinke velikih vhodnih vrednosti, ki bi vodile do majhnih gradientov med učenjem. Za lažje razumevanje, zakaj veliki argumenti softmax vodijo do majhnih gradientov, lahko konstruiramo primer. Začnimo z definicijo softmax funkcije:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.18}$$

Gradient softmax funkcije je izračunan kot:

$$\frac{\partial \text{softmax}(x_i)}{\partial x_j} = \text{softmax}(x_i) * (\delta_{ij} - \text{softmax}(x_j)) \tag{2.19}$$

kjer ∂_{ij} je Kroneckerjev delta, ki je enak 1, če sta i in j enaka, in 0 sicer. Če upoštevamo skaliranje, dobimo:

$$\frac{\partial \text{softmax}(x_i/C)}{\partial x_j} = \frac{1}{C} * \text{softmax}(x_i/C) * (\delta_{ij} - \text{softmax}(x_j/C)) \tag{2.20}$$

kjer C je faktor skaliranja. Iz tega lahko vidimo, da skaliranje zmanjšuje velikost gradientov, kar lahko pomaga pri stabilizaciji učenja.

2.3.2 Večglava pozornost

Večglava pozornost (Multi-Head Attention) je razširitev mehanizma pozornosti Scaled Dot-Product Attention. V večglavi pozornosti se vhodni podatki (poizvedbe, ključi in vrednosti) najprej transformirajo v več različnih prostorov z uporabo linearnih preslikav. Nato se za vsak niz izračuna funkcija pozornosti Scaled Dot-Product Attention. Rezultati teh funkcij pozornosti se nato združijo skupaj v eno matriko. Končno se ta matrika preslika nazaj

v izviren prostor z uporabo druge linearne preslikave, da se pridobi končni rezultat večglave pozornosti. Avtorji to izrazijo v spodnji obliki:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O \tag{2.21}$$

Vsak head_i je rezultat izvajanja Scaled Dot-Product Attention na i-tem nizu transformiranih poizvedb, ključev in vrednosti:

$$\text{head}_i = \text{Attention}(QW_{Qi}, KW_{Ki}, VW_{Vi}) \tag{2.22}$$

kjer so $Q \in \mathbb{R}^{m \times d_{\text{model}}}$, $K \in \mathbb{R}^{n \times d_{\text{model}}}$, in $V \in \mathbb{R}^{n \times d_{\text{model}}}$. Poleg tega, ob upoštevanju hiperparametra h, ki označuje število glav pozornosti, velja: $W_{Qi} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_{Ki} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_{Vi} \in \mathbb{R}^{d_{\text{model}} \times d_v}$, in $W_O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Dokažimo, da se matrično množenje izide. Najprej vemo iz prejšnjega razdelka, da bo vsaka matrika head_i imela enako število vrstic kot QW_{Qi} in enako število stolpcev kot VW_{Vi}. Ker velja $QW_{Qi} \in \mathbb{R}^{m \times d_k}$ in $VW_{Vi} \in \mathbb{R}^{n \times d_v}$, to pomeni, da je $\text{head}_i \in \mathbb{R}^{m \times d_v}$. Ko združimo h takih matrik, dobimo matriko v $\mathbb{R}^{m \times hd_v}$. Množenje z W_O daje matriko v $\mathbb{R}^{m \times d_{\text{model}}}$. Kar drži - začeli smo z m poizvedbami v Q in končali z m odgovori v izhodu operatorja.

Vsak izračun glave ima drugačno linearno preslikavo za matrike ključev, poizvedb in vrednosti. Vsaka od teh preslikav se nauči med učenjem.

Maskiranje vhodov

En način za maskiranje vhodov je preprosto dodajanje matrike M k argumentu ki vsebuje 0 v spodnjem trikotniku in $-\infty$ povsod drugje:

$$\frac{1}{\sqrt{d_k}} Q' K'^T + M = \tag{2.23}$$

$$\begin{pmatrix} \frac{1}{\sqrt{d_k}} \vec{q}'_0 \cdot \vec{k}'_0 & \vec{q}'_0 \cdot \vec{k}'_1 & \cdots & \vec{q}'_0 \cdot \vec{k}'_n \\ \vec{q}'_1 \cdot \vec{k}'_0 & \vec{q}'_1 \cdot \vec{k}'_1 & \cdots & \vec{q}'_1 \cdot \vec{k}'_n \\ \vdots & \vdots & \ddots & \vdots \\ \vec{q}'_n \cdot \vec{k}'_0 & \vec{q}'_n \cdot \vec{k}'_1 & \cdots & \vec{q}'_n \cdot \vec{k}'_n \end{pmatrix} + \begin{pmatrix} 0 & -\infty & -\infty & \cdots & -\infty \\ -\infty & 0 & 0 & \cdots & -\infty \\ -\infty & \vdots & \vdots & \vdots & \vdots \\ -\infty & 0 & 0 & \cdots & 0 \\ -\infty & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$= \frac{1}{\sqrt{d_k}} \begin{pmatrix} \vec{q}_0 \cdot \vec{k}'_0 & -\infty & -\infty & \cdots & -\infty \\ \vec{q}_1 \cdot \vec{k}'_0 & \vec{q}_1 \cdot \vec{k}'_1 & -\infty & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots & \\ \vec{q}_{n-1} \cdot \vec{k}'_0 & \vec{q}_{n-1} \cdot \vec{k}'_1 & \vec{q}_{n-1} \cdot \vec{k}'_2 & \cdots & -\infty \\ \vec{q}_n \cdot \vec{k}'_0 & \vec{q}_n \cdot \vec{k}'_1 & \vec{q}_n \cdot \vec{k}'_2 & \cdots & \vec{q}_n \cdot \vec{k}'_n \end{pmatrix}$$

Nato ima izvajanje softmaxa na vsaki vrstici učinek pošiljanja vseh $-\infty$ celic na 0, pri čemer ostanejo samo veljavni izrazi za pozornost.

2.3.3 Pozornost kodirnik-dekodirnik

Tretja in zadnja uporaba pozornosti v članku [7] je pozornost kodirnik-dekodirnik, ki se uporablja v blokih dekodirnika neposredno po sloju maske večglave pozornosti, da se povežejo izvorne in ciljne sekvence. Medtem ko so pri samopozornosti vsi trije vhodi enaka matrika, to tukaj ne velja.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O, \quad \text{head}_i = f(Q, K, V)$$

Ko govorimo o pozornosti med kodirnikom in dekodirnikom, je edina razlika od prej v tem, da Q izhaja iz sloja maske večglave pozornosti, medtem ko sta K in V kodirani predstavitvi $\vec{F}$. Lahko bi razmišljali o tem tako, da model zastavlja vprašanje o tem, kako se vsak položaj v ciljni sekvenci nanaša na izvor, in pridobiva predstavitve izvora za uporabo pri generiranju naslednje besede v cilju. Pomembno je poudariti, da vsi bloki dekodirnika prejmejo enake podatke od kodirnika. Od prvega do N-tega bloka dekodirnika vsak uporablja kodirano izvorno sekvenco kot ključe in vrednosti.

2.4 Vision Transformer (ViT)

Transformerji so prvotno bili omejeni na obdelavo zaporedij, kar je idealno za jezik, vendar ne nujno za slike, ki so običajno dvodimenzionalne. To se

je spremenilo z razvojem Vision Transformerja (ViT) s strani Google-a [5]. Namesto da bi slike obdelovali kot dvodimenzionalne mreže pikslov (kot to počnejo konvolucijske nevronske mreže), Vision Transformer slike obravnava kot zaporedje majhnih kvadratov ali "obližev". To omogoča uporabo istih tehnik samo-pozornosti, ki so bile učinkovite v jezikovnih modelih, tudi za obdelavo slik. Ta pristop je pokazal obetavne rezultate, saj je Vision Transformer dosegel ali presegel učinkovitost konvolucijskih nevronskih mrež na številnih nalogah računalniškega vida.

ViT arhitektura

- Razdelitev slike na obliže: Slika velikosti $H \times W \times C$ se razdeli na kvadrate (obliže) velikosti $P \times P$, kjer je H višina, W širina, C število barvnih kanalov in P velikost obliža. To ustvari $(H \cdot W)/P^2$ obližev. Vsak obliž se nato zravna v 1D vektor dolžine $P^2 \cdot C$.

- Linearne projekcije: Vsak 1D vektor x se prenese skozi enostaven linearni model (npr. polno povezano plast), da se pretvori v vektorski vložek. To se lahko zapiše kot:

$$z = Wx + b$$

  kjer sta W in b uteži in pristranskost linearne plasti.

- Dodajanje pozicijskih vložkov: Ker transformatorji ne vsebujejo nobene inherentne informacije o relativni ali absolutni poziciji vložkov v zaporedju, se dodajo pozicijski vložki. To so enaki vektorji, ki se dodajo vložkom obližev, da bi modelu dali nekaj informacij o tem, kje se obliž nahaja v sliki. Če je z_i vložek i-tega obliža in p_i pozicijski vložek, potem je končni vložek e_i določen kot:

$$e_i = z_i + p_i$$

- Transformerjevi bloki: Zaporedje vložkov (zdaj z dodanimi pozicijskimi vložki) se nato prenese skozi več blokov transformatorjev. Ti bloki vsebujejo večglavo samopozornost in mreže feed-forward, ki omogočajo

modelu, da se nauči, kako povezati različne dele slike. Večglava samopozornost se lahko zapiše kot:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O$$

kjer je $\text{head}_i = \text{Attention}(QWQi, KW_{Ki}, VW_{Vi})$, Q, K in V so poizvedbe, ključi in vrednosti, W_{Qi}, W_{Ki}, W_{Vi} in W_O so uteži, ki se naučijo, in Attention je funkcija samopozornosti.

- Klasifikacijska glava: Na koncu se uporabi klasifikacijska glava (ponavadi ena polno povezana plast), da se izračuna končna napoved za dano nalogo (npr. klasifikacija slik). To se lahko zapiše kot:

$$y = \text{softmax}(W_2\text{ReLU}(W_1 e))$$

kjer sta W_1 in W_2 uteži polno povezanih plasti, e je vložek, ki izhaja iz transformatorjevih blokov, in ReLU in softmax sta aktivacijski funkciji.

2.5 Piramidni vision transformer (PVT)

Piramidni Vision Transformer (PVT) [9] je bil razvit z namenom vključitve piramidne strukture v okviru Transformerja, kar omogoča generiranje večrazsežnih značilnostnih map za naloge goste napovedi, kot so zaznavanje objektov in semantična segmentacija. Arhitektura PVT je razdeljena na štiri stopnje. Vsaka od teh stopenj je sestavljena iz plasti za vdelavo obližev, imenovane "patch embedding", in iz več plasti Transformer kodirnika. Značilnost te arhitekture je, da se izstopna ločljivost štirih stopenj postopoma zmanjšuje, kar sledi piramidni strukturi. Na najvišji stopnji je ločljivost značilnostne mape največja, medtem ko se na najnižji stopnji zmanjša.

Za boljše razumevanje si poglejmo podrobneje prvo stopnjo: Vhodna slika velikosti $H \times W \times 3$ je razdeljena na obliže velikosti $4 \times 4 \times 3$. To pomeni, da je število obližev enako $HW/4^2$. Vsak obliž je nato sploščen in prenesen v linearno projekcijo, kar rezultira v vdelavi obližev velikosti $HW/4^2 \times C1$.

Ti vdelani obliži, skupaj z dodano vdelavo položaja, prehajajo skozi Transformer kodirnik z $L1$ plastmi. Izhod iz tega kodirnika je nato preoblikovan v značilnostno mapo $F1$ velikosti $H/4 \times W/4 \times C1$.

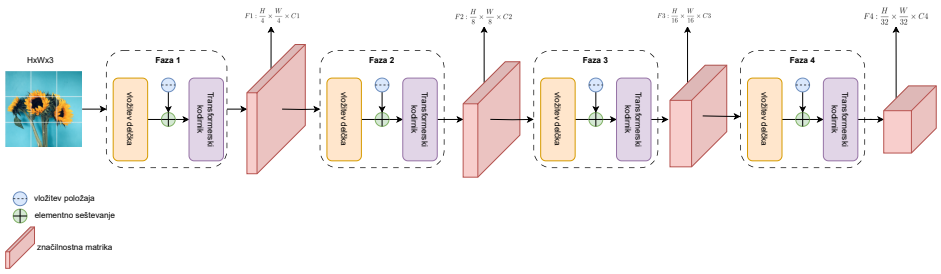
Matematično to lahko izrazimo kot:

$$F1 = \frac{H}{4} \times \frac{W}{4} \times C1 \tag{2.24}$$

Naslednje stopnje PVT sledijo podobnemu pristopu, vendar z različnimi ločljivostmi in dimenzijami. Na primer, značilnostne mape $F2, F3$ in $F4$ so pridobljene z različnimi koraki, ki so 8, 16 in 32 slikovnih pik glede na vhodno sliko.

Ena izmed ključnih inovacij v PVT je uporaba pozornosti za zmanjšanje prostorskega obsega (SRA) namesto tradicionalne večglave pozornostne plasti (MHA). Ta pristop omogoča PVT-u, da učinkovito obdela značilnostne mape visoke ločljivosti.

V primerjavi z Vision Transformerjem (ViT), PVT prinaša večjo prilagodljivost, saj lahko generira značilnostne mape različnih meril/kanalov v različnih fazah. Poleg tega je bolj vsestranski, saj se lahko enostavno vključi in uporabi v večini modelov za spodnje naloge. Prav tako je bolj prijazen do računalniških virov in spomina, saj lahko obdela značilnostne mape višje ločljivosti ali daljše sekvence.



Slika 2.3: Skica PVT modela

2.6 Piramidni vision transformer z uporabo lokalnih značilnosti (PCPVT)

Twins-PCPVT [3] je zasnovan na osnovi PVT in CPVT [2]. Glavna razlika med Twins-PCPVT in PVT je v načinu uporabe pozicijskih kodiranj. V PVT so uporabljena absolutna pozicijska kodiranja, medtem ko Twins-PCPVT uporablja pogojna pozicijska kodiranja (CPE), ki so bila predlagana v CPVT.

PVT je uvedel piramidni večstopenjski dizajn z namenom boljšega obravnavanja nalog goste napovedi, kot so zaznavanje objektov in semantična segmentacija. Vendar je bilo ugotovljeno, da je manjša učinkovitost PVT-ja v veliki meri posledica uporabe absolutnih pozicijskih kodiranj. Absolutna pozicijska kodiranja se soočajo s težavami pri obdelavi vhodov različnih velikosti, kar je pogosto v nalogah goste napovedi.

V Twins-PCPVT so absolutna pozicijska kodiranja nadomeščena s pogojnimi pozicijskimi kodiranji (CPE), ki so odvisna od vhodov in se tako lahko naravno izognejo zgoraj omenjenim težavam. Generator pozicijskega kodiranja (PEG), ki generira CPE, je postavljen za prvim kodirnim blokom vsake stopnje. Uporablja najpreprostejšo obliko PEG, tj. 2D globinsko konvolucijo brez normalizacije serij.

$$CPE = f(PEG(E_1, E_2, ..., E_n)) \tag{2.25}$$

Kjer je CPE pogojno pozicijsko kodiranje, f je funkcija, ki generira kodiranje na podlagi vhodnih značilnosti, in E_i so značilnosti iz različnih stopenj kodirnika. Twins-PCPVT združuje prednosti tako PVT-ja kot CPVT-ja, kar ga naredi enostavnega za učinkovito implementacijo. Eksperimentalni rezultati so pokazali, da ta preprosta zasnova lahko doseže zmogljivost nedavno predlaganega Swin transformerja [6].

2.7 Siamska nevronska mreža za primerjavo vzorcev

Pri obdelavi slik in vizualni analitiki je ena izmed ključnih nalog primerjava ali ujemanje vzorcev, znano tudi kot "template matching". Tradicionalne metode, ki temeljijo na neposrednem ujemanju ali korelaciji, so občutljive na spremembe svetlobe, rotacije, deformacije in druge variacije v sliki. S prihodom globokih nevronskih mrež, zlasti siamskih mrež, se je učinkovitost te naloge znatno povečala.

2.7.1 Osnovna arhitektura siamske mreže za primerjavo vzorcev

Klasična siamska mreža za primerjavo vzorcev sestoji iz dveh identičnih podmrež, ki delijo enake uteži. Vsaka podmreža prejme sliko: ena je ciljna slika, druga pa je iskana slika. Oba vhoda se preoblikujeta v značilnostne vektorje prek teh podmrež. Nato se izračuna razdalja med obema vektorjema, običajno z evklidsko razdaljo, da se ugotovi, kako podobni sta sliki.

Matematično, za dve sliki x_1 in x_2, podmreži proizvedeta predstavitve $f(x_1;\theta)$ in $f(x_2;\theta)$. Razdalja D med tema dvema predstavitvama je določena kot:

$$D(f(x_1;\theta), f(x_2;\theta)) = |f(x_1;\theta) - f(x_2;\theta)|_2 \tag{2.26}$$

2.7.2 Učenje siamske mreže za primerjavo vzorcev

Da bi siamsko mrežo usposobili za učinkovito primerjavo vzorcev, potrebujemo nabor učnih podatkov, ki vsebuje pare podobnih in različnih slik. Med učenjem je cilj zmanjšati razdaljo med podobnimi slikami in povečati razdaljo med različnimi slikami. Kriterijska funkcija, običajno uporabljena pri

učenju siamskih mrež za primerjavo vzorcev, je kontrastna kriterijska funkcija, definirana kot:

$$L(y, D(f(x_1;\theta), f(x_2;\theta))) = y \cdot \frac{1}{2}D^2 + (1-y) \cdot \frac{1}{2}\max(0, m-D)^2 \quad (2.27)$$

Kjer y označuje oznako podobnosti (1 za podobne in 0 za različne), m pa je prag, ki določa mejo med podobnimi in različnimi slikami.

2.7.3 Aplikacije in prednosti

Siamske mreže za primerjavo vzorcev so se izkazale za izjemno koristne v številnih aplikacijah, kot so prepoznavanje in sledenje objektom, biometrija ter varnost in nadzor. V primerjavi s tradicionalnimi metodami imajo siamske mreže večjo odpornost na variacije v svetlobi, rotaciji, lestvici in drugih deformacijah. Zaradi globje hierarhične predstavitve slike so sposobne zaznati in primerjati kompleksne značilnosti, ki jih manj kompleksne metode morda ne bi opazile.

Poglavje 3

Podatkovna množica

V raziskovalnem svetu je podatkovna množica ključnega pomena za razvoj, testiranje in validacijo modelov. Kljub pomembnosti modela WAMF-FPI avtorji niso javno delili originalne podatkovne množice, kar je postavilo pred nas izziv pri pripravi ustreznih podatkov za našo analizo. Za doseganje konsistentnosti in kakovosti rezultatov smo se odločili samostojno kreirati in kurirati našo lastno podatkovno množico, ki odraža realne pogoje in scenarije uporabe.

Podatkovna množica, ki smo jo oblikovali, temelji na dveh glavnih virih vizualnih podatkov. Prvi vir predstavljajo slike, pridobljene z brezpilotnimi letalniki. Te slike so bile pridobljene preko orodja Google Earth Studio, ki omogoča natančno in realno reprezentacijo terenskih značilnosti iz ptičje perspektive. Te slike nudijo bogate detajle in so ključne za razumevanje fine strukture terena.

Drugi vir podatkov predstavljajo satelitske slike, pridobljene preko Mapbox API. Satelitske slike prinašajo širši pogled na regijo in omogočajo razumevanje večjih geografskih in prostorskih vzorcev. V kombinaciji z dronskimi slikami te slike nudijo celovito sliko terena z različnih višin in ločljivosti.

Skupno naša podatkovna množica vključuje več kot 11.000 dronskih slik in njihovih pripadajočih satelitskih slik. Ta obsežna zbirka podatkov nam omogoča, da model WAMF-FPI testiramo in validiramo v številnih različnih

scenarijih in pogojih, s čimer zagotavljamo njegovo robustnost in splošno uporabnost.

3.1 Slike brezpilotnega letalnika

Nabor podatkov, ki ga predstavljamo, je bil zasnovan z namenom raziskovanja in analize dronov v različnih mestnih scenarijih. Osredotoča se na dve ključni območji:

1. Gosto pozidana mestna območja z zgradbami in
2. odprte zelene površine, kot so parki in travniki.

Zajem slik je bil izveden na naključnih poteh po mestu, kar omogoča širok spekter scenarijev in situacij. V mestnih območjih je poudarek na razumevanju, kako se brezpilotni letalniki lokalizirajo in navigirajo med visokimi zgradbami, kjer so lahko GPS signali zmanjšani ali moteni. V zelenih območjih je cilj razumeti, kako se brezpilotni letalniki obnašajo v okoljih, kjer so vizualni vzorci manj raznoliki in se teren lahko zdi monoton. V naboru podatkov za učenje je 10.000 slik iz desetih mest, pri čemer vsako mesto prispeva 1.000 slik. Brezpilotni letalniki so bili kalibrirani na višini 150 metrov nad navedeno nadmorsko višino mesta. Kamere na brezpilotnih letalnikih imajo vidno polje 80 stopinj in so usmerjene pravokotno na središče Zemlje. Vse slike so bile ustvarjene z uporabo orodja Google Earth Studio.

Mesta, vključena v učni nabor podatkov, so:

- **Maribor:** Nadmorska višina: 272m, Višina drona: 150m, Skupaj: 422m nad morsko gladino.
- **Trst:** Nadmorska višina: 23m, Višina drona: 150m, Skupaj: 173m nad morsko gladino.
- **Zagreb:** Nadmorska višina: 158m, Višina drona: 150m, Skupaj: 308m nad morsko gladino.

- **Gradec:** Nadmorska višina: 353m, Višina drona: 150m, Skupaj: 503m nad morsko gladino.

- **Celovec:** Nadmorska višina: 446m, Višina drona: 150m, Skupaj: 596m nad morsko gladino.

- **Videm:** Nadmorska višina: 113m, Višina drona: 150m, Skupaj: 263m nad morsko gladino.

- **Pula:** Nadmorska višina: 17m, Višina drona: 150m, Skupaj: 167m nad morsko gladino.

- **Pordenone:** Nadmorska višina: 24m, Višina drona: 150m, Skupaj: 174m nad morsko gladino.

- **Szombathely:** Nadmorska višina: 212m, Višina drona: 150m, Skupaj: 362m nad morsko gladino.

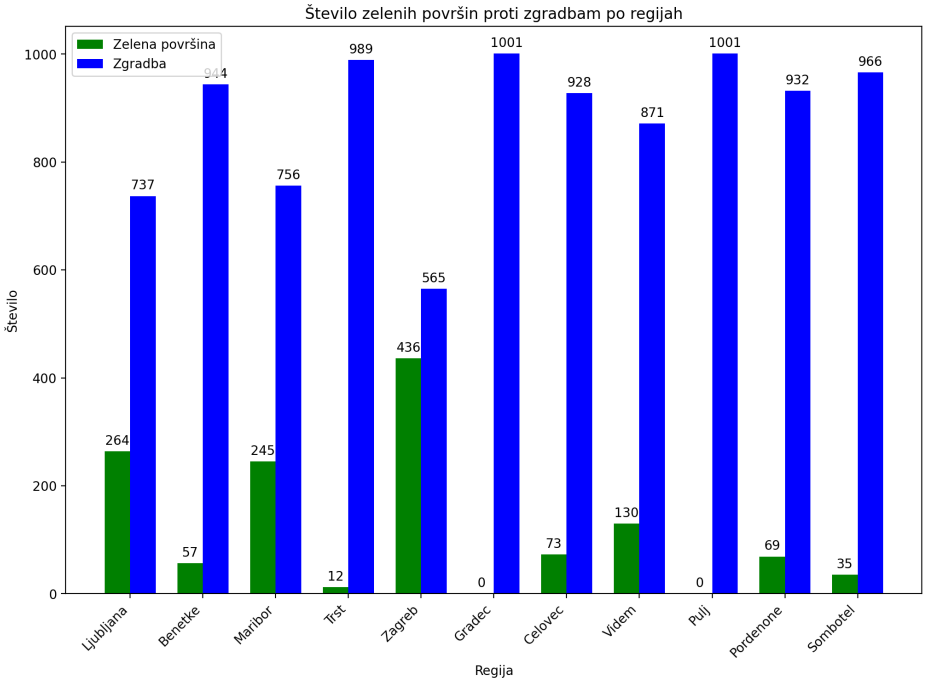
- **Benetke:** Nadmorska višina: -1m, Višina drona: 150m, Skupaj: 149m nad morsko gladino.

Dodatno je bil v nabor dodan tudi testni nabor podatkov za Ljubljano, ki vključuje 1.000 slik. Vsaka slika je opremljena z oznakami lokacije kamere v sistemu ECEF. Sistem ECEF (Earth Centered, Earth Fixed) je globalni koordinatni sistem z izhodiščem v središču Zemlje.

Slika 3.1: Slika prikazuje lokacije mest, ki so vključena v nabor podatkov.

Na Sliki 3.2 je prikazana vizualna razdelitev zelenih površin in stavb za različna mesta, temelječa na analizi slik, ki smo jih zajeli v našem podatkovnem naboru. Vsako mesto razkriva svojo edinstveno strukturo in raven urbanizacije. Te razlike so ključnega pomena pri razumevanju izzivov, s katerimi se srečujejo brezpilotni letalniki pri lokalizaciji in navigaciji v različnih mestnih okoljih.

Nekatera mesta, kot sta Gradec in Pula, kažejo višjo stopnjo urbanizacije z minimalno prisotnostjo zelenih površin. To pomeni, da bodo droni v teh okoljih večinoma navigirali med zgradbami. Na drugi strani pa mesta, kot je Zagreb, predstavljajo večjo mešanico zgradb in zelenih površin. Takšne razlike lahko vplivajo na algoritme lokalizacije in navigacije dronov, saj se morajo prilagajati različnim scenarijem in oviram.



Slika 3.2: Graf prikazuje razmerje med zelenimi površinami in stavbami za vsako mesto.

Na sliki 3.3 je prikazan primer dronske slike, zajete v Ljubljani.

Slika 3.3: Primer dronske slike.

3.2 Satelitske slike

Za vsako dronsko sliko smo poiskali ustrezen satelitski "tile"ali ploščico. Ta korak je bil ključnega pomena, saj je zagotovil, da so satelitske slike popolnoma usklajene z dronskimi slikami v smislu geografske lokacije. Ko smo identificirali ustrezno satelitsko ploščico, smo jo prenesli neposredno iz Mapbox API-ja, priznanega vira za visokokakovostne satelitske slike. Da bi zagotovili dodatno globino in kontekst za vsako lokacijo, nismo prenesli samo osrednje ploščice, temveč tudi vse njene sosednje ploščice. Te sosednje ploščice smo nato združili z osrednjo ploščico za ustvarjanje enotne TIFF datoteke.

Ta pristop nam je omogočil, da smo imeli na voljo širšo regijo za analizo in učenje. Ko smo imeli pripravljene TIFF datoteke, smo začeli z učnim procesom. Za vsako iteracijo učenja smo iz vsake TIFF datoteke naključno izrezali regijo velikosti 400x400 pikslov. Ključnega pomena je bilo, da se je točka lokalizacije vedno nahajala nekje znotraj te izrezane regije. Ta metoda nam je zagotovila, da je bil model izpostavljen širokemu naboru scenarijev in kontekstov, hkrati pa smo ohranili natančnost in relevantnost lokalizacijskih podatkov. S tem pristopom smo uspešno sestavili nabor podatkov, ki združuje najboljše iz obeh svetov: detajlnost dronskih slik in širino satelitskih slik, kar omogoča poglobljeno analizo in učinkovito učenje.

Ko govorimo o ploščicah v kontekstu kartografije in GIS (Geografski informacijski sistem), se običajno nanašamo na kvadratne segmente, ki pokrivajo Zemljo in se uporabljajo za hitrejše in učinkovitejše prikazovanje zemljevidov na spletu. Sistem ploščic je zelo priljubljen v spletnih kartografskih aplikacijah, kot je Google Maps.

Za pretvorbo geografskih koordinat (latitudo in longitudo) v ploščične koordinate (x, y) na določeni ravni povečave z uporabo Mercatorjeve projekcije, lahko izrazimo:

- Pretvorba geografskih koordinat v radiane:

$$\begin{aligned} \text{lat}_{\text{rad}} &= \text{latitude} \times \frac{\pi}{180}, \\ \text{lon}_{\text{rad}} &= \text{longitude} \times \frac{\pi}{180} \end{aligned}$$

- Pretvorba radianov v normalizirane koordinate Mercatorja:

$$\begin{aligned} x &= \frac{\text{lon}_{\text{rad}} = +\pi}{2\pi}, \\ y &= \frac{\pi - \log(\tan(\frac{\pi}{4} + \frac{\text{lat}_{\text{rad}}}{2}))}{2\pi} \end{aligned}$$

- Pretvorba normaliziranih koordinat v ploscicne koordinate:

$$\begin{aligned} \text{tile}_{\text{x}} &= \text{floor}(x \times 2^z), \\ \text{tile}_{\text{y}} &= \text{floor}(y \times 2^z) \end{aligned}$$

Na sliki 3.4 je prikazan primer pripadajoče satelitske slike za dronsko sliko.

Slika 3.4: Primer pripadajoče satelitske slike za dronsko sliko.

Poglavje 4

Rezultati

V tem poglavju so podrobno predstavljeni rezultati, doseženi v različnih fazah implementacije in optimizacije modela WAMF-FPI. Naš izhodiščni korak je bil zagotoviti stabilno osnovo, kar smo dosegli z implementacijo modela skladno z metodologijo, opisano v izvirnem članku. Ta pristop nam je zagotovil referenčno točko, od katere smo izvajali nadaljnje optimizacije in izboljšave.

Model smo trenirali na računalniku, opremljenem z Intel(R) Xeon(R) CPU E5-2690 v3 in NVIDIA GeForce RTX 3060, uporabljali pa smo programsko okolje PyTorch.

Med optimizacijo modela smo se posvetili iskanju optimalne kriterijske funkcije. Da bi bolje razumeli, katera funkcija bi lahko prinesla najboljše rezultate v našem primeru, smo izvedli serijo eksperimentov z različnimi funkcijami ter jih evalvirali glede na njihovo učinkovitost in zanesljivost.

Kot naslednji korak smo preučili stratificirano vzorčenje, tehniko, ki bi lahko pripomogla k izboljšanju natančnosti in robustnosti modela z zagotavljanjem bolj uravnoteženega učnega nabora.

Pregledali smo tudi vpliv Hanningovega okna ter analizirali, kako različne velikosti tega okna vplivajo na končne rezultate modela.

V zaključni fazi naših eksperimentov smo se osredotočili na regularizacijo, predvsem na tehniko izpuščanja nevronov. Zaradi kompleksnosti modelov globokega učenja smo želeli razumeti, kako bi taka regularizacija lahko po-

magala preprečiti prekomerno prilagajanje ter izboljšala splošno učinkovitost modela.

Vsako od teh področij je v nadaljevanju podrobno obravnavano, pri čemer so podane analize, interpretacije in ključne ugotovitve, ki smo jih pridobili v tem procesu.

4.1 Implementacija

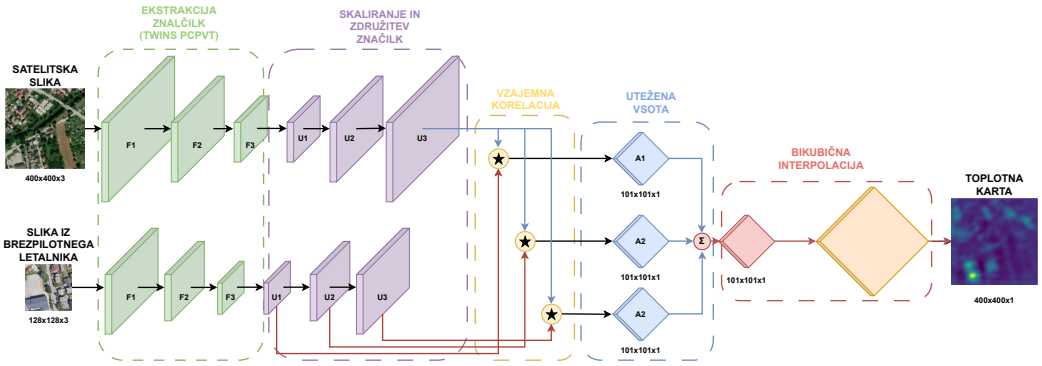
Sledenje objektov v okviru računalniškega vida običajno temelji na izračunu podobnosti med referenčno in iskalno podobo v trenutnem okviru. Medtem ko temeljna metoda za iskanje točk znotraj slike izhaja iz metodologije sledenja objektov, je prva v primerjavi z drugo bolj zapletena. To je posledica različnih perspektiv med predlogo (dronsko sliko) in iskalno sliko (satelitsko sliko), ki povzročajo veliko variacijo.

Metoda iskanja točk uporablja satelitsko sliko kot referenčno in dronsko sliko kot poizvedbeno. Obe sliki – posneto z dronom in satelitsko sliko relevantnega območja – se nato prenesejo v end-to-end omrežje. Po obdelavi je rezultat toplotna karta, kjer točka z najvišjo vrednostjo predstavlja lokacijo drona, kot jo predvideva model. Lokacijo nato preslikamo na satelitsko sliko, pri čemer položaj drona določimo na podlagi geografske širine in dolžine, ki jih vsebuje satelitska slika. V FPI avtorji kot modul za ekstrakcijo značilnosti uporabljajo dva Deit-S brez deljenih uteži za vertikalne poglede dronske in satelitske slike [4]. Ekstrahirane značilnosti nato uporabimo za izračun podobnosti in izdelavo toplotne karte. Lokacijo z najvišjo vrednostjo toplotne karte nato preslikamo na satelitsko sliko, da določimo lokacijo drona.

V FPI je za izračun podobnosti uporabljena zadnja plast zemljevidnih značilnosti [4]. Zaradi 16-kratne stiskalne rate končnega izhodnega zemljevida model izgubi veliko prostorskih informacij, kar vodi v znatno izgubo natančnosti pozicioniranja.

Da bi izboljšali lokalizacijske sposobnosti modela, smo uporabili strukturo piramidnih značilnosti (Twins-PCPVT) in modul utežno prilagodlji-

vega združevanja večznačilnostnih lastnosti (WAMF). K osnovnemu modelu so bile dodane izboljšave z vključitvijo dveh močnejših PCPVT-S modulov za ekstrakcijo značilnosti iz dronskih in satelitskih slik. Da bi bolje zajeli informacije na različnih ločljivostih in ohranili več prostorskih informacij, so bile prvotno ekstrahirane značilnosti poslane v omrežje značilnostne piramide za nadaljnjo obdelavo. Modul WAMF je bil nato uporabljen za izračun podobnosti in združevanje različnih značilnosti. Končne združene značilnosti so bile razširjene za izdelavo končne izhodne napovedne mape. Rezultat je toplotna karta iste velikosti kot vhodna satelitska slika v modelu WAMF-FPI.



Slika 4.1: Skica arhitekture modela

4.1.1 Modul za ekstrakcijo značilnosti

WAMF-FPI temelji na strukturi, ki je podobna Siamese omrežju, vendar se od tradicionalnega sledenja objektom loči v ključnih aspektih. Zaradi občutne razlike med satelitskimi slikami in slikami brezpilotnega letalnika, ki izvirajo iz različnih naprav, veji modela WAMF-FPI za vsako od teh vrst slik ne uporabljata metode deljenja uteži.

Konkretno, WAMF-FPI kot vhod uporablja satelitske slike dimenzij 400 $\times$ 400 $\times$ 3 in slike brezpilotnega letalnika dimenzij 128 $\times$ 128 $\times$ 3. Značilnosti obeh vrst slik so ekstrahirane s pomočjo PCPVT-S.

atančneje, v modelu smo odstranili zadnjo stopnjo PCPVT-S in uporabili

samo prve tri stopnje za ekstrakcijo značilnosti. Pri dimenzijah vhodnih slik 400 x 400 x 3 in 128 x 128 x 3 oba pristopa pridobita značilnostne mape z obliko 25 x 25 x 256 in 8 x 8 x 320 oziroma.

V primerjavi z Deit-S, ki je bil uporabljen v FPI [4], ima PCPVT-S piramidno strukturo. Ta struktura je bolj prilagodljiva za naloge goste napovedi. Pravzaprav uporaba piramidne strukture zagotavlja osnovo za kasnejšo integracijo modula WAMF. Poleg tega omrežje z piramidno strukturo lahko zmanjša obseg potrebnih izračunov in s tem izboljša hitrost procesiranja, kar je ključno za učinkovito uporabo metode v praksi.

Po ekstrakciji informacij iz slike s pomočjo PCPVT-S se podobnost neposredno izračuna na zadnjih značilnostnih mapah. Kljub temu je končni izhod stisnjen samo za faktor štiri v primerjavi z vhodom, kar je potem s bikubično interpolacijo povečano nazaj na velikost vhodne satelitske slike.

Pristranskost, ki je posledica nizke ločljivosti značilnostne mape, je bila odstranjena že na samem začetku. Ker značilnostna mapa z visoko ločljivostjo vsebuje več prostorskih informacij, je bila združena z globoko značilnostno mapo, bogato s semantičnimi informacijami, preko lateralne povezovalne strukture.

WAMF-FPI uporablja konvolucijske mreže za izluščenje značilnosti iz vhodnih slik. Konvolucija je ključna operacija, ki modelu omogoča, da "vidi"in prepoznava vzorce in značilnosti v slikah.

Prva faza obdelave v WAMF-FPI je uporaba konvolucijskega jedra velikosti ena, ki prilagodi kanalsko dimenzijo tri-stopnjske značilnostne mape, pridobljene s pomočjo PCPVT-S. Število izhodnih kanalov je bilo nastavljeno na 64, kar zagotavlja kompaktno in učinkovito zastopanje značilnosti. Po tej fazi sledi upsampling operacija na značilnostnih mapah zadnjih dveh stopenj, ki poveča njihovo ločljivost in s tem omogoča bolj precizno lokalizacijo. Te mape se nato kombinirajo z značilnostnimi mapami istega merila iz osnovnega modela.

Končno, značilnosti se dodatno ekstrahirajo s pomočjo konvolucijskega jedra velikosti 3, kar modelu omogoča izluščenje bolj kompleksnih značilnosti

iz združenih map. Rezultat je združena značilnostna mapa, ki združuje plitve (prostorske) in globoke (semantične) informacije. Ta bogata kombinacija modelu omogoča učinkovito prepoznavanje in lokalizacijo objektov na vhodnih slikah.

4.1.2 Arhitektura utežno-prilagodljivega združevanja večznačilnostnih lastnosti (WAMF)

Modul za združevanje značilnosti je zasnovan tako, da združuje informacije iz dveh ločenih vhodnih tokov, v tem primeru iz UAV (brezpilotnega letalnika) in SAT (satelita). Ta modul uporablja piramido značilnosti iz obeh in izračuna korelacije med njimi, da jih združi v en sam izhod.

Za začetek se izvedejo konvolucijske operacije na značilnostnih mapah UAV in SAT. Konvolucijske operacije so izvedene s konvolucijskimi jedri velikosti 1×1, kar omogoča prilagoditev kanalskih dimenzij značilnostnih map.

Za UAV značilnostne mape:

$$U1_{\text{UAV}} = \text{Conv1UAV}(s3\text{UAV}) \tag{4.1}$$

$$U2_{\text{UAV}} = \text{Upsample}(U1_{\text{UAV}}) + \text{Conv2UAV}(s2\text{UAV}) \tag{4.2}$$

$$U3_{\text{UAV}} = \text{Upsample}(U2_{\text{UAV}}) + \text{Conv3UAV}(s1\text{UAV}) \tag{4.3}$$

Za SAT značilnostne mape:

$$U1_{\text{SAT}} = \text{Conv1SAT}(s3\text{SAT}) \tag{4.4}$$

$$U2_{\text{SAT}} = \text{Upsample}(U1_{\text{SAT}}) + \text{Conv2SAT}(s2\text{SAT}) \tag{4.5}$$

$$U3_{\text{SAT}} = \text{Upsample}(U2_{\text{SAT}}) + \text{Conv3SAT}(s1\text{SAT}) \tag{4.6}$$

Kjer je "Upsample"funkcija, ki poveča prostorsko resolucijo značilnostne mape z uporabo bikubične interpolacije.

$$A1 = \text{corr}(U1_{\text{UAV}}, U3_{\text{SAT}}) \tag{4.7}$$

$$A2 = \text{corr}(U2_{\text{UAV}}, U3_{\text{SAT}}) \tag{4.8}$$

$$A3 = \text{corr}(U3_{\text{UAV}}, U3_{\text{SAT}}) \tag{4.9}$$

Kjer je corr funkcija za izračun korelacije med dvema značilnostnima mapama.

Korelacija v kontekstu obdelave slik je postopek izračuna podobnosti med dvema slikama ali značilnostnima mapama. V osnovi ena značilnostna mapa (poimenovana "poizvedba") "drsi"čez drugo značilnostno mapo (poimenovana "iskalna regija") in izračuna podobnost med njima na vsaki lokaciji. Rezultat tega postopka je nova značilnostna mapa, imenovana korelacijska mapa, kjer vsaka vrednost predstavlja stopnjo podobnosti med poizvedbo in delom iskalne mape na določeni lokaciji.

Matematično je korelacija med dvema funkcijama f in g definirana kot:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f^*(\tau) g(t + \tau) d\tau \tag{4.10}$$

Kjer je f^* kompleksno konjugirana funkcija f.

V kontekstu diskretnih signalov, kot so slike ali značilnostne mape, je korelacija definirana kot:

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m] g[n + m] \tag{4.11}$$

Nazadnje se izvede uteženo združevanje teh treh koreliranih značilnostnih map s pomočjo naučljivih uteži:

$$\text{fused\_map} = w_1 \cdot A1 + w_2 \cdot A2 + w_3 \cdot A3 \tag{4.12}$$

Za dokončanje postopka se uporabi bikubična interpolacija, da se fused_map poveča na velikost vhodne satelitske slike. Na izhodu dobimo toplotno karto iste velikosti kot vhodna satelitska slika v WAMF-FPI.

4.1.3 RDS metrika

Da bi lahko ovrednotili in primerjali zmogljivost našega modela, uporabljamo metriko RDS [8]. Zaradi različnih meril podatkov v naboru podatkov vsak piksel v različnih satelitskih slikah predstavlja različno razdaljo. Čeprav model morda najde točko, ki je na satelitski sliki blizu dejanske lokacije, lahko v resničnem prostoru povzroči veliko napako. Da bi se izognili težavam zaradi spremembe merila, RDS izračuna relativno razdaljo na ravni pikslov med napovedano in dejansko točko.

Enačba za izračun RDS je naslednja:

$$RDS = e^{-k \times \frac{\sqrt{\left(\frac{dx}{w}\right)^2 + \left(\frac{dy}{h}\right)^2}}{2}} \tag{4.13}$$

Kjer so:

- w širina piksla satelitske slike,
- h višina piksla satelitske slike,
- dx pikselska razdalja med vodoravnimi koordinatami napovedane pozicije in dejanske pozicije,
- dy pikselska razdalja med navpičnimi koordinatami napovedane pozicije in dejanske pozicije,
- k je faktor merila, ki je v tem delu postavljen na 10.

4.2 Učenje modela

Model smo učili na računalniški konfiguraciji, opremljeni z visokozmogljivim procesorjem Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz s 12 jedri. Dodatno je računalnik vseboval grafično kartico NVIDIA GeForce RTX 3060 s 12 GB pomnilnika, kar je omogočalo efektivno paralelizacijo in optimizacijo operacij, ki jih zahteva model med treningom. Naš razvoj je temeljil na platformi Ubuntu z uporabo python knjižnice PyTorch.

Da bi povečali produktivnost in optimizirali proces, smo razvili avtomatizirane skripte, imenovane `push-ml-node` in `train-ml-node`. Prva skripta je bila uporabljena za sinhronizacijo virov z računalnikom, medtem ko je bila druga skripta uporabljena za zagon samega učenja.

Za dosego optimalnih rezultatov smo uporabili specifične hiperparametre in nastavitve:

Naprava: Učenje je potekalo na `cuda:0`, ki se nanaša na uporabo NVIDIA grafične kartice.

Hitrost učenja: Uporabljena sta bila dva različna parametra: $lr\_fusion = 0.0004$ za združevanje in $lr\_backbone = 0.0001$ za osnovno arhitekturo.

Prilagajanje hitrosti učenja: $gamma = 0.2$ z mejniki na epohah 9, 13 in 15.

Delovni procesi: Skupno 24 hkratnih delovnih procesov (`num_workers = 24`).

Epoh: Model je bil učen skozi 24 epoh.

Velikost serije: `batch_size = 16`.

Mešanje podatkov: Podatki so bili premešani pred vsako epoho.

Funkcija izgube: Uporabljena je bila `hanning` funkcija.

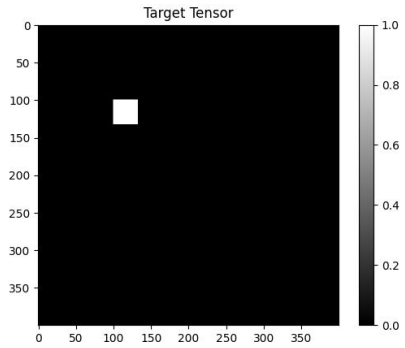
Vizualizacija: Vključena za spremljanje napredka učenja.

4.3 Izbira kriterjiske funkcije

Zanimalo nas je kako se bo model obnesel, ko izbiramo razlicne kriterijske funkcije.

4.3.1 Hanningova kriterijska funkcija

V clanku WAMF-FPI [8] so avtorji predlagali uporabo Hanningove kriterijske funkcije. Prvi pomemben vidik te funkcije izgube je dodelitev uteži vzorcem. Namesto enakega pomena vseh pozitivnih vzorcev, funkcija izgube Hanning dodeli različne uteži glede na lokacijo vzorca.



Slika 4.2: Primer vzorca, sredisce je tocka lokacije vzorca.

To je zato, ker je pomembnost središčnega položaja veliko večja kot pomembnost robovih položajev, kar v kontekstu satelitskih slik logično smiselno. Za normalizacijo teh pozitivnih uteži se uporablja Hanningovo okno, za normalizacijo negativnih utezi, pa

$$1/\#\text{negativnih vzorcev}$$

. Uteži so dodeljene tako, da je vsota uteži pozitivnih in negativnih vzorcev enaka 1. Toda ker je število negativnih vzorcev običajno večje od števila pozitivnih vzorcev, postane utež negativnih vzorcev manjša. Da bi to popravili, se uvede hiperparameter , imenovan Negativna utež (NG), ki prilagodi utež negativnih vzorcev.

Hanningova funkcija:

$$\text{Hanning}(n) = \begin{cases} 0.5 + 0.5\cos\left(\frac{2\pi n}{M-1}\right) & \text{za } 0 \leq n \leq M-1 \\ 0 & \text{sicer} \end{cases} \tag{4.14}$$

Utezi primerov:

- Utez negativnih vzorcev:

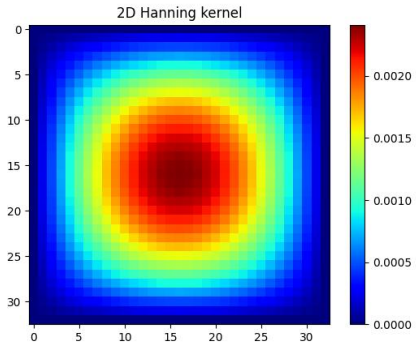
$$w_{pos} = NG/(NN(NW+1))$$

- Utez pozitivnih vzorcev:

$$w_{neg} = HN(n)/(NW+1)$$

Kjer je:

- **NG** je Negativna utež
- **NN** je stevilo vseh uzorcev
- **NW** je normalizacijski faktor
- **HN(n)** je vrednost Hanningove funkcije na lokaciji n



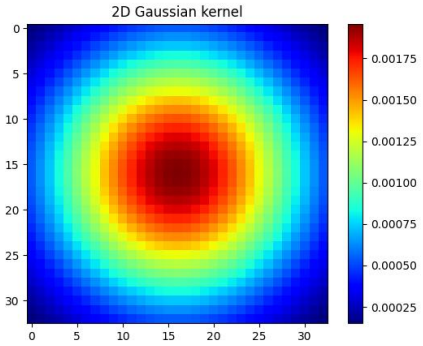
Slika 4.3: Hanningovo jedro

4.3.2 Gaussovo utezena srednja kvadratna napaka

Gaussova utežena srednja kvadratna napaka (Gaussian Weighted Mean Squared Error - GWMSE) je modificirana funkcija izgube, namenjena izboljšanju modelov, ki obravnavajo podatke, kot so satelitske slike. Glavna značilnost GWMSE je dodeljevanje uteži vzorcem, na zelo podoben nacin kot pri Hanningovi funkciji izgube. Namesto enakega pomena vseh pozitivnih vzorcev, GWMSE različnim vzorcem dodeljuje različne uteži glede na njihovo lokacijo. Za normalizacijo teh uteži se uporablja Gaussova funkcija.

Gaussova funkcija:

$$\text{Gauss}(n) = \begin{cases} \exp\left(-\frac{(n-\mu)^2}{2\sigma^2}\right) & \text{za } 0 \leq n \leq M-1 \\ 0 & \text{sicer} \end{cases} \tag{4.15}$$



Slika 4.4: Gaussovo jedro

4.3.3 Hanningovo utežena srednja kvadratna napaka

Hanningova utežena srednja kvadratna napaka (Hanning Weighted Mean Squared Error - HWMSE) je spremenjena funkcija izgube, namenjena izboljšanju modelov, ki obravnavajo podatke, kot so satelitske slike. Glavna značilnost HWMSE je dodeljevanje uteži vzorcem na zelo podoben način kot pri Gaussovi funkciji izgube. Namesto enakega pomena vseh pozitivnih

vzorcev, HWMSE različnim vzorcem dodeljuje različne uteži glede na njihovo lokacijo. Za normalizacijo teh uteži se uporablja Hanningovo okno.

Hanningova funkcija je podana kot:

$$\text{Hanning}(n) = \begin{cases} 0.5 + 0.5\cos\left(\frac{2\pi n}{M-1}\right) & \text{za } 0 \leq n \leq M-1 \\ 0 & \text{sicer} \end{cases} \tag{4.16}$$

4.3.4 Križno utežena srednja kvadratna napaka

Funkcija izgube križno utežena srednja kvadratna napaka (Cross-Weighted Mean Squared Error - CW-MSE) je napredna različica standardne srednje kvadratne napake (Mean Squared Error - MSE), ki vključuje uteževanje dveh različnih skupin vzorcev: tistih, katerih resnična vrednost je večja od 0 (t.i. "resničnih"vzorcev) in tistih, katerih resnična vrednost je manjša ali enaka 0 (t.i. "ne-resničnih"vzorcev). Končna funkcija izgube se izračuna kot utežena kombinacija srednjih kvadratnih napak za "resnične"in "ne-resnične"vzorce, pri čemer se uteži vzorcev različnih skupin prekrižajo. Ta pristop se formalno izraža z naslednjo enačbo:

$$\text{loss} = \frac{\text{true\_weight} \cdot N_{\text{true}} \cdot \text{MSE}_{\text{false}} + \text{false\_weight} \cdot N_{\text{false}} \cdot \text{MSE}_{\text{true}}}{N_{\text{all}}} \tag{4.17}$$

- N_{true}: število vzorcev, katerih resnična vrednost je večja od 0.
- N_{false}: število vzorcev, katerih resnična vrednost je enaka ali manjša od 0.
- N_{all}: skupno število vzorcev.
- MSEtrue = $\frac{1}{N\text{true}} \sum_{i=1}^{N_{\text{true}}} (y_i - \hat{y}_i)^2$ za vzorce, katerih resnična vrednost je večja od 0.
- MSEfalse = $\frac{1}{N\text{false}} \sum_{i=1}^{N_{\text{false}}} (y_i - \hat{y}_i)^2$ za vzorce, katerih resnična vrednost je enaka ali manjša od 0.
- true_weight in false_weight: uteži, dodeljene skupinama *true* in *false*.

Python implementacija funkcije izgube CW-MSE je naslednja:

```
class CrossWeightedMSE(nn.Module):
    def __init__(self, true_weight=1, false_weight=1):
        super(CrossWeightedMSE, self).__init__()
        self.mse_loss = nn.MSELoss(reduction="none")
        self.true_weight = true_weight
        self.false_weight = false_weight

    def forward(self, input, target):
        N_all = target.numel()
        N_true = torch.sum(target > 0.0).item()
        N_false = N_all - N_true

        true_mask = target > 0.0
        false_mask = torch.logical_not(true_mask)

        MSE_true = torch.mean(self.mse_loss(input[true_mask], target[
            true_mask]))
        MSE_false = torch.mean(self.mse_loss(input[false_mask], target[
            false_mask]))

        loss = (
            self.true_weight * N_true * MSE_false
            + self.false_weight * N_false * MSE_true
        ) / N_all

        return loss
```

4.3.5 Primerjava rezultatov

Kriterijska funkcija	vrednost	RDS_{train}	RDS_{val}
Hanningova kriterijska funkcija	8.49	0.893	0.709
Gaussovo utezena srednja kvadratna napaka	0.001	0.077	0.074
Hanningovo utežena srednja kvadratna napaka	4.04e-06	0.061	0.059
Križno utežena srednja kvadratna napaka	0.007	0.07	0.06

Tabela 4.1: Rezultati ob uporabi razlicnih kriterijskih funkij.

4.3.6 Analiza rezultatov

Hanningova kriterijska funkcija

Hanningova kriterijska funkcija, znana tudi po svoji značilnosti dodeljevanja uteži vzorcem glede na njihovo lokacijo, je v testiranju pokazala dobre rezultate. S skupno vrednostjo 8.49 in RDS_{train} vrednostjo 0.893 na učni množici se je izkazala kot izredno učinkovita za trening set. Čeprav je bila njena učinkovitost na validacijski množici, kjer je dosegla RDS_{val} vrednost 0.709, nekoliko nižja, so rezultati še vedno zelo obetavni. Primer je viden na sliki 4.5.

Ključna prednost Hanningove funkcije je v njeni zmožnosti prilagajanja uteži vzorcem glede na njihov položaj, kar se zdi še posebej primerno pri analizi satelitskih slik. V teh slikah je središčni položaj pogosto bistven, medtem ko robovi morda niso tako pomembni. To naravno prilagodljivost Hanningove funkcije lahko opazimo v njenih rezultatih, ki jih dosegla v obravnavanem primeru.

Gaussovo utežena srednja kvadratna napaka

Čeprav je Gaussova utežena srednja kvadratna napaka prav tako zasnovana na principu dodeljevanja uteži glede na lokacijo vzorca, rezultati kažejo, da ne dosega enake uspešnosti kot Hanningova funkcija. Z RDS_{train} vrednostjo 0.077 na učni množici in RDS_{val} vrednostjo 0.74 na validacijski množici so njeni rezultati precej slabši v primerjavi s Hanningovo funkcijo. Primer je viden na sliki 4.6.

Čeprav obe funkciji temeljita na podobnem principu, se zdi, da Hanningova funkcija bolje odraža posebnosti in značilnosti satelitskih slik.

Hanningovo utežena srednja kvadratna napaka

Pri tej funkciji se je izkazalo, da mreža ni dosegla želenih rezultatov. Namesto, da bi se mreža naučila prepoznati in interpretirati relevantne značilnosti satelitskih slik, se je večinoma učila šuma. Praktično, model se ni naučil nič

koristnega, kar nakazuje, da Hanningovo utežena srednja kvadratna napaka morda ni primerna za to vrsto podatkov ali za uporabljeni model. Primer je viden na sliki 4.7.

Križno utežena srednja kvadratna napaka

Podobno kot pri Hanningovi uteženi srednji kvadratni napaki se je tudi pri Križno uteženi srednji kvadratni napaki pokazalo, da mreža večinoma prepoznava in se uči šuma. Rezultati so bili nezadovoljivi in kažejo na to, da ta funkcija ni najbolj primerna za analizo satelitskih slik s tem pristopom. Primer je viden na sliki 4.8.

Zaključek: Hanningova kriterijska funkcija se je v obravnavanem primeru izkazala kot najbolj učinkovita. Njena edinstvena sposobnost prilagajanja uteži glede na lokacijo vzorca se zdi še posebej primerna za obravnavo satelitskih slik, kar je morda razlog za njeno premoč nad ostalimi obravnavanimi funkcijami izgube.

4.4 Učenje s Stratificiranim Vzorčenjem

4.4.1 Stratificirano vzorčenje

Stratificirano vzorčenje je metoda vzorčenja, pri kateri se celoten nabor podatkov razdeli na ločene podskupine ali strate. Vsak stratum predstavlja določeno kategorijo ali razred v naboru podatkov. V kontekstu mest bi lahko vsako mesto predstavljalo svoj stratum. Namen stratificiranega vzorčenja je zagotoviti, da je vsak vzorec reprezentativen za celoten nabor podatkov.

Zakaj je stratificirano vzorčenje pomembno?

1. **Ohranjanje Distribucije**: Stratificirano vzorčenje zagotavlja, da se razmerje vzorcev v vsakem stratumu ohranja enako kot v celotnem naboru podatkov. To je še posebej pomembno, ko je distribucija podatkov v vsakem stratumu (v tem primeru mesto) ključnega pomena za analizo. Na primer, če želimo, da je naš vzorec reprezentativen za

različna mesta, bi uporabili stratificirano vzorčenje, da zagotovimo, da so vsa mesta ustrezno zastopana.

2. **Natancnost**: Stratificirano vzorčenje lahko poveča natančnost ocen, saj zmanjšuje variabilnost znotraj vsakega strata. To pomeni, da so vzorci iz vsakega strata bolj homogeni, kar lahko vodi do natančnejših rezultatov.

Slabosti stratificiranega vzorčenja:

1. **Omejena Generalizacija**: Čeprav stratificirano vzorčenje zagotavlja, da so vse kategorije ali razredi v naboru podatkov ustrezno zastopani v vzorcu, to lahko pomeni, da model morda ni tako dobro pripravljen na povsem nove, nevidene podatke. Model je lahko optimiziran za specifično distribucijo podatkov, ki je bila uporabljena med ucenjem in validacijo.

2. **"In-Distribution"Validacija** Ker se vzorci za ucenje in validacijo izbirajo iz iste distribucije (stratificirane distribucije), model morda ne bo dobro deloval na "out-of-distribution"podatkih. To pomeni, da čeprav model morda kaže visoko natančnost na validacijskem naboru, to ne zagotavlja, da bo enako dobro deloval na podatkih, ki se močno razlikujejo od originalne distribucije.

4.4.2 Rezultati

Nacin	Hanningova izguba	RDS_{train}	RDS_{val}
Train test split ucenje	8.49	0.893	0.709
Ucenje s stratificiranim uzorcenjem	3.17	0.750	0.731

Tabela 4.2: Rezultati ob uporabi stratificiranega uzorcenja.

Iz rezultatov 4.2 je razvidno, da je uporaba stratificiranega vzorčenja pozitivno vplivala na rezultate.

Za boljše razumevanje uspešnosti modelov je ključno upoštevati tudi njihovo zmogljivost na validacijskih naborih podatkov. To je še posebej pomembno, saj nam validacija daje vpogled v to, kako dobro model predvideva rezultate na nevidenih podatkih. Če primerjamo rezultate RDS_{val} med obema pristopoma, opazimo, da je model, ki je bil naučen s stratificiranim vzorčenjem, dosegel rahlo višjo uspešnost (0.731) v primerjavi z modelom, ki je bil naučen s tradicionalno metodo "train-test split"(0.709). To kaže, da se je model, ki je bil naučen s stratificiranim vzorčenjem, nekoliko bolje spoprijel s generalizacijo na nevidenih podatkih. To dejstvo podkrepi tudi zmanjšana razlika med uspešnostjo na učni in validacijski množici v primeru stratificiranega vzorčenja. Večja konsistentnost rezultatov med učno in validacijsko množico je lahko pokazatelj, da model ni pretirano prilagojen in se lahko bolje generalizira na nove podatke. Torej, medtem ko je tradicionalna "train-test split"metoda dosegla višjo uspešnost na učni množici, se zdi, da stratificirano vzorčenje ponuja bolj zanesljive in stabilne rezultate na validacijski množici, kar je ključnega pomena za ocenjevanje realne zmogljivosti modela. V našem primeru se zdi, da stratificirano vzorčenje ponuja bolj robusten in stabilen model za obravnavane satelitske slike. Vendar pa je pomembno upoštevati tudi omejitve stratificiranega vzorčenja, kot so omejena generalizacija in potencialne težave pri "out-of-distribution"podatkih.

4.5 Vpliv velikosti Hanningovega okna na rezultate

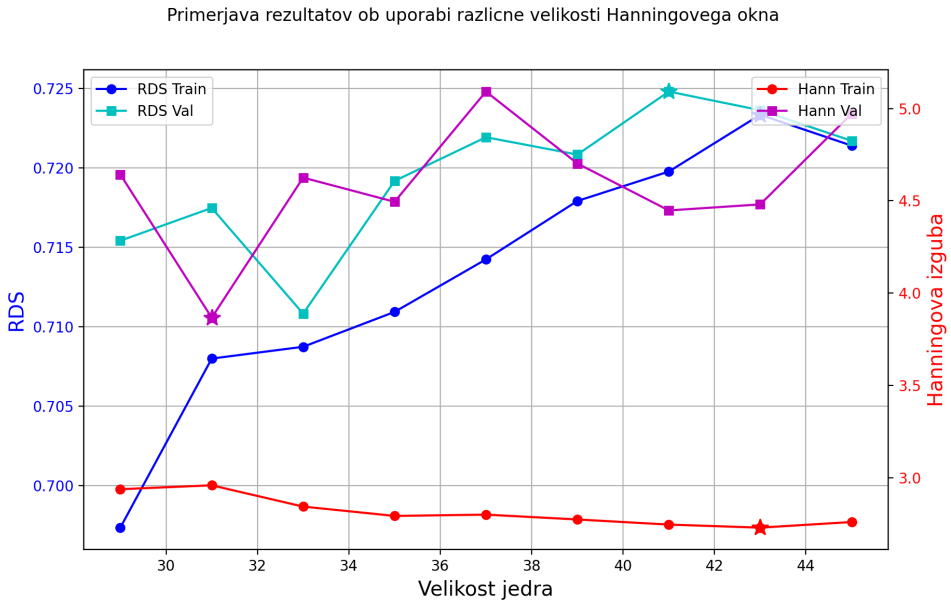
Velikost Hanningovega okna igra ključno vlogo pri določanju uteži vzorcev. Zaradi narave Hanningove kriterijske funkcije velikost okna neposredno vpliva na razporeditev in obliko uteži, dodeljenih vzorcem v satelitskih slikah.

4.5.1 Dinamika različnih velikosti Hanningovih oken

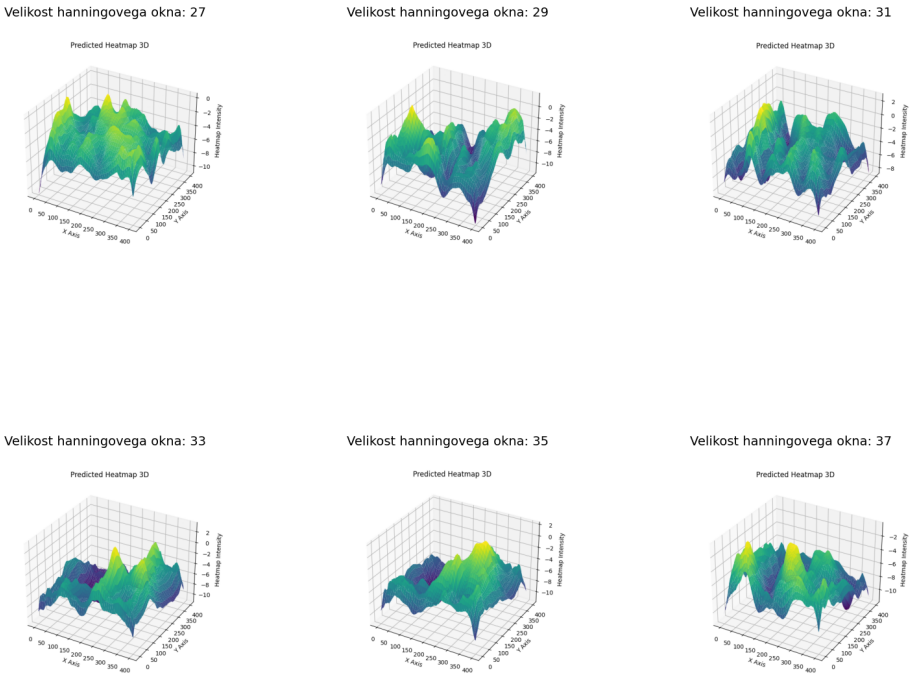
Ko je velikost okna majhna, bo okno zajelo ožji del vzorcev, kar lahko povzroči težave s povratnim razširjanjem med učenjem modela. Če je območje, ki ga pokriva okno, premajhno, kriterijska funkcija ne more efektivno vplivati na celotno mrežo, kar vodi do potencialno slabše uspešnosti modela. Nasprotno, preveliko okno lahko privede do izgube natančnosti. Čeprav kriterijska funkcija zajema širši del vzorcev, lahko pomembni detajli postanejo zamegljeni, kar vodi do manj natančnih rezultatov.

4.5.2 Eksperimentalni rezultati

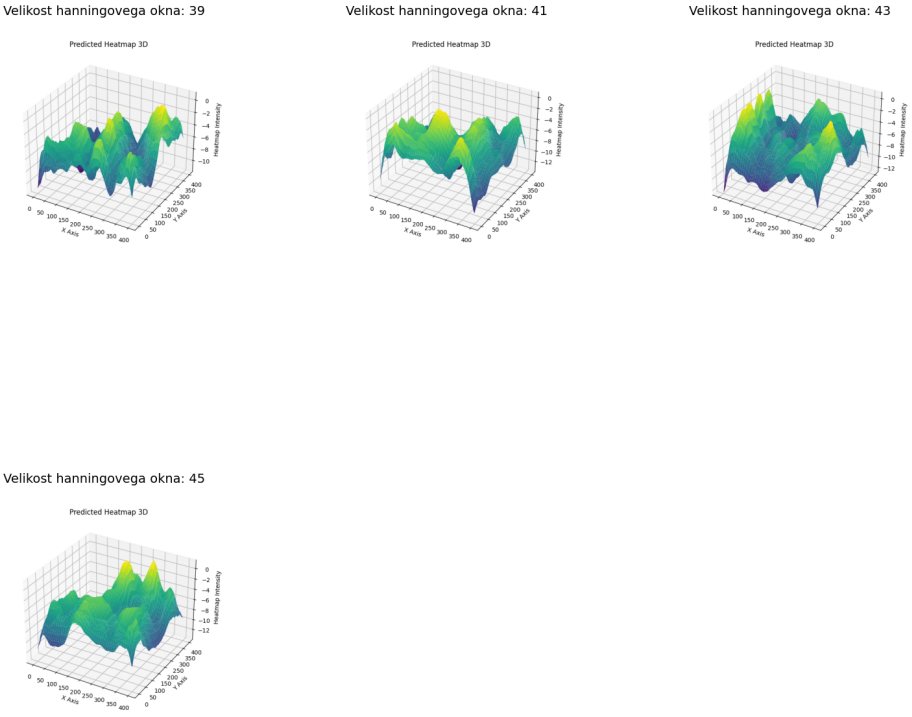
V naših testiranjih smo ugotovili, da je najbolje najti uravnoteženo velikost Hanningovega okna, ki omogoča modelu, da efektivno uči in hkrati ohranja natančnost pri predikcijah. V ta namen smo izvedli več iteracij, kjer smo eksperimentirali z različnimi velikostmi oken.



Slika 4.9: Primerjava rezultatov ob uporabi razlicnih velikosti Hanningovega okna.



Slika 4.10: Primerjava toplotnih map ob uporabi razlicnih velikosti Hanningovega okna.



Slika 4.11: Primerjava toplotnih map ob uporabi razlicnih velikosti Hanningovega okna.

4.6 Regularizacija v modelu z uporabo izpuščanja nevronov

4.6.1 Izpuščanje nevronov

V svetu strojnega učenja je regularizacija ključna tehnika, ki se uporablja za preprečevanje prekomernega prilagajanja modela. Prekomerno prilagajanje se pojavi, ko model postane preveč specifičen za učni nabor podatkov, kar pomeni, da se "preveč nauči"podrobnosti in šuma v učnih podatkih, kar vodi v slabo zmogljivost na novih, nevidenih podatkih. Med različnimi tehnikami regularizacije je "izpuščanje nevronov"(v angleščini "dropout") ena izmed najbolj priljubljenih in učinkovitih metod za nevronske mreže. Koncept izpuščanja nevronov je preprost, a močan: med ucenjem se določen odstotek nevronov v mreži naključno "izklopi"ali izpusti. To pomeni, da se med posameznim prehodom naprej določeni nevroni (in njihove povezave) začasno odstranijo iz mreže.

V modelu sem uporabil izpuščanje nevronov na več ključnih mestih:

1. **Izpuščanje Nevronov v Modelu Twins**: Izpuščanje nevronov je bilo uporabljeno za regulacijo različnih komponent modela, vključno z deli, kot so `attn_drop`, `proj_drop`, `head_drop`, `mlp_drop1`, `mlp_drop2` in `pos_drops`.. Vsaka od teh komponent ima svojo specifično vlogo v arhitekturi modela. Z dodajanjem izpuščanja nevronov na te komponente sem dodal dodatno raven regularizacije, ki pomaga preprečiti prekomerno prilagajanje.

2. **Izpuščanje Nevronov v Modulu za Združevanje Značilnosti**: Po vsaki konvolucijski operaciji v fuzijskem delu modela sem dodal izpuščanje nevronov. Konvolucijske plasti lahko hitro postanejo kompleksne in se prekomerno prilagodijo podatkom, zlasti ko delujejo na visokodimenzionalnih značilnostih. Z dodajanjem izpuščanja nevronov po vsaki konvolucijski plasti sem zmanjšal to tveganje in povečal robustnost modela.

Izpuščanje nevronov je ena izmed najbolj učinkovitih tehnik regularizacije za nevronske mreže. Z njegovo uporabo v modelu sem zagotovil, da je model bolj robusten in manj nagnjen k prekomernemu prilagajanju na učne podatke. V kompleksnih modelih, kot je Twins, kjer je veliko komponent, ki se lahko prekomerno prilagodijo podatkom, je uporaba izpuščanja nevronov ključnega pomena za zagotavljanje natančnih in zanesljivih rezultatov.

4.6.2 Rezultati

Parameter	**UAV**	**Satelit**	**Združevanje**
dropout	0.1	0.1	0.1
attn_drop	0.1	0.1	-
proj_drop	0.1	0.1	-
head_drop	0.1	0.1	-
mlp_drop1	0.1	0.1	-
mlp_drop2	0.1	0.1	-
pos_drops	0.05	0.05	-

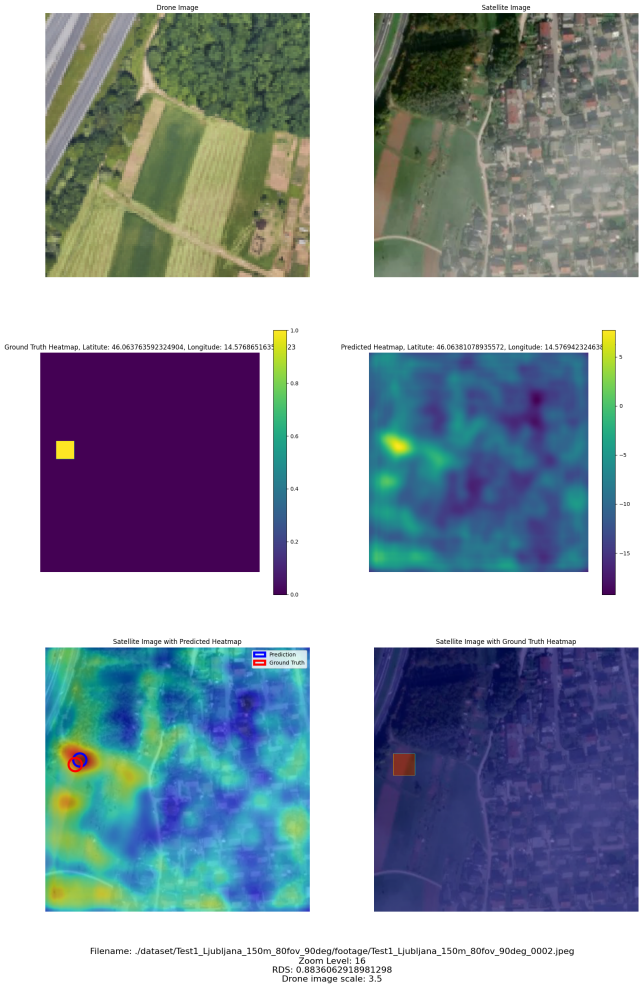
Tabela 4.3: Parametri z uravnovesenim izpustom nevronov.

Parameter	**UAV**	**Satelit**	**Združevanje**
dropout	0.15	0.05	0.05
attn_drop	0.15	0.05	-
proj_drop	0.15	0.05	-
head_drop	0.15	0.05	-
mlp_drop1	0.15	0.05	-
mlp_drop2	0.15	0.05	-
pos_drops	0.1	0.05	-

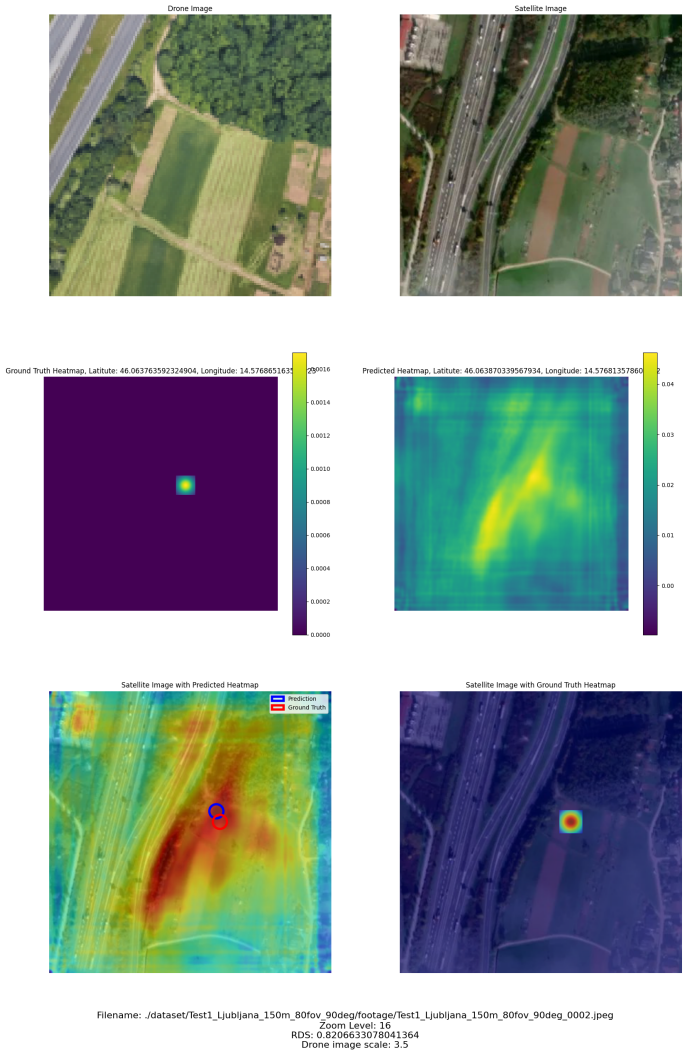
Tabela 4.4: Parametri z neuravnovesenim izpustom nevronov.

Nacin	Hanningova izguba	RDS_{train}	RDS_{val}
Brez izpuscanja nevronov	8.49	0.893	0.709
Z uravnovesenim izpuscanjem nevronov	5.49	0.725	0.690
Z neuravnovesenim izpuscanjem nevronov	5.42	0.725	0.719

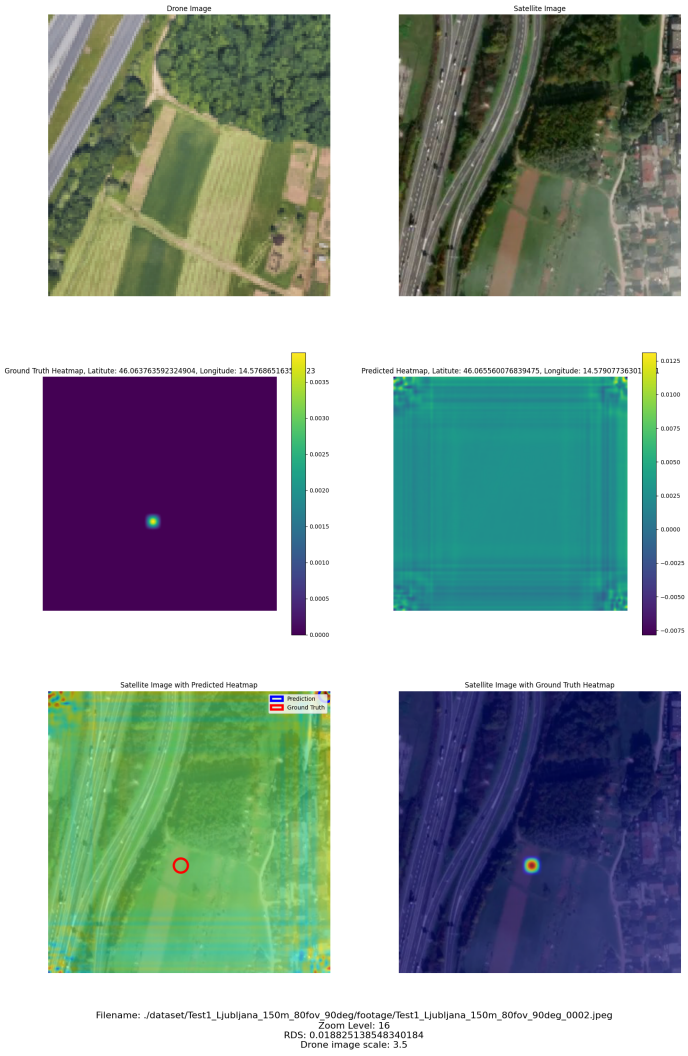
Tabela 4.5: Rezultati ob uporabi razlicnih izpustov.



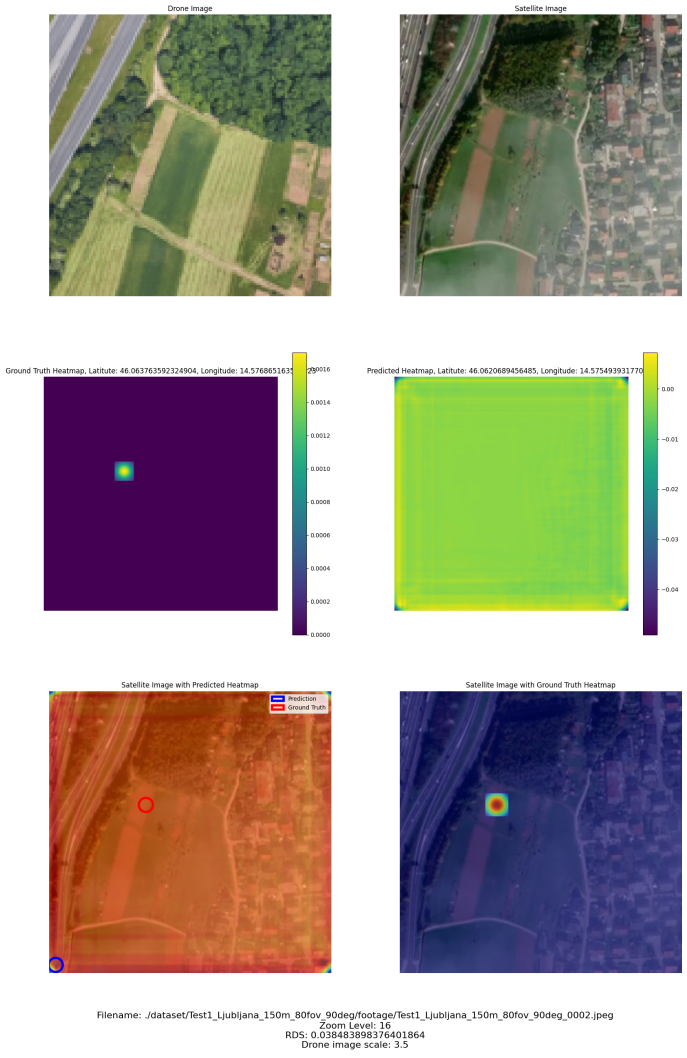
Slika 4.5: Primer izhoda ob uporabi Hanningove kriterijske funkcije



Slika 4.6: Primer izhoda ob uporabi Gaussovo utežene srednje kvadratne napake



Slika 4.7: Primer izhoda ob uporabi Hanningove utežene srednje kvadratne napake



Slika 4.8: Primer izhoda ob uporabi Križno uteženе srednje kvadratne napake

Poglavje 5

Sklepne ugotovitve

Uporaba LaTeXa in BibLaTeXa je v okviru Diplomskega seminarja **obvezna**! Izbira – LaTeX ali ne LaTeX – pri pisanju dejanske diplomske naloge pa je prepuščena dogovoru med diplomantom in njegovim mentorjem.

Res je, da so prvi koraki v LaTeXu težavni. Ta dokument naj služi kot začetna opora pri hoji. Pri kakršnihkoli nadaljnih vprašanjih ali napakah pa svetujemo uporabo Googla, saj je spletnih strani za pomoč pri odpravljanju težav pri uporabi LaTeXa ogromno.

Preden diplomo oddate na sistemu STUDIS, še enkrat preverite, če so slovenske besede, ki vsebujejo črke s strešicami, pravilno deljene in da ne segajo preko desnega roba. Poravnavo po vrsticah lahko kontrolirate tako, da izvorno datoteko enkrat testno prevedete z opcijo `draft`, kar vam pokaže predolge vrstice.

Literatura

[1] Dzmitry Bahdanau, Kyunghyun Cho in Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". V: *arXiv preprint arXiv:1409.0473* (2015).

[2] Xiangxiang Chu in sod. "Conditional positional encodings for vision transformers". V: *arXiv preprint arXiv:2102.10882* (2021).

[3] Xiangxiang Chu in sod. "Twins: Revisiting the design of spatial attention in vision transformers". V: *Advances in Neural Information Processing Systems* 34 (2021), str. 9355–9366.

[4] Ming Dai in sod. "Finding Point with Image: An End-to-End Benchmark for Vision-based UAV Localization". V: *arXiv preprint arXiv:2208.06561* (2022).

[5] Alexey Dosovitskiy in sod. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". V: *arXiv preprint arXiv:2010.11929* (2020).

[6] Ze Liu in sod. "Swin transformer: Hierarchical vision transformer using shifted windows". V: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, str. 10012–10022.

[7] Ashish Vaswani in sod. "Attention is all you need". V: *Advances in neural information processing systems* 30 (2017).

[8] Guirong Wang in sod. "WAMF-FPI: A Weight-Adaptive Multi-Feature Fusion Network for UAV Localization". V: *Remote Sensing* 15.4 (2023), str. 910.

[9] Wenhai Wang in sod. "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions". V: *arXiv preprint arXiv:2102.12122* (2021).