Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gasper Spagnolo

# Lokalizacija brezpilotnih letalnikov

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

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

Ljubljana, 2023



*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

**Kandidat:** Gasper 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:**
todo

**Title:** UAV localization

**Description:**
todo

*Zahvaljujem se mami in proskiju.*

Svoji dragi proskici.

# 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:** Gasper 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:** Gasper 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

Droni in sateliti se uporabljajo za iskanje in reševanje, kartiranje terena, kmetijsko spremljanje, navigacijo dronov in podobne naloge. Raznolikost platform za tehnologijo oddaljenega zaznavanja prinaša veliko prednosti na teh področjih. Ljudje lahko ne le prek satelitov pridobijo podatke velikega obsega, temveč tudi s pomočjo platform dronov pridobijo bolj jasne lokalne slike.

Trenutno se droni v glavnem zanašajo na satelitske signale za navigacijo in določitev položaja med letom. Vendar se v praksi satelitski signal močno oslabi po dolgi razdalji, kar lahko povzroči motnje sprejetega satelitskega signala na dronu. Zlasti na vojaškem področju je izguba satelitskega signala pogosta. Samozadostna lokalizacija in navigacija dronov v okoljih, kjer so satelitski signali omejeni ali moteni, postaja vse pomembnejša.

Da bi rešili problem avtonomne navigacije brezpilotnih letalnikov v okolju, kjer je uporaba satelitskih signalov omejena, so bile prejšnje metode večinoma izvedene z uporabo prepoznavanja slik. Lokalizacija naprave se doseže z ujemanjem slike letalnika z vsako sliko v satelitski slikovni bazi. Med procesom ucenja so neprestano skrajševali razdaljo med slikami letalnikov in satelitskimi slikami podobnih regij preko metrike učenja. Metoda prepoznavanja slik je na nekaterih naborih podatkov dosegla odlične rezultate. Vendar ima več problemov:

- Pred praktično uporabo je treba vnaprej pripraviti slikovno bazo za prepoznavanje, in vse slike v bazi so poslane modelu za izvleček značilnosti.
- Za doseganje bolj natančne pozicioniranja, mora baza pokrivati čim večji obseg in slika poizvedbe mora biti izračunana z vsemi slikami v bazi. To prinaša večji skladiščenjski in računalniški pritisk na računalnik.
- Ko se model posodobi, je treba posodobiti tudi ustrezno bazo.

Povzemajoč, metoda prepoznavanja slik zahteva veliko predobdelovalnih operacij. Hkrati pa so tudi zahteve za skladiščno zmogljivost in računalniško moč precej velike.

S hitrim razvojem računalniškega vida se pojavlja geolokacija dronov na podlagi satelitskih slik. Ta metoda, podobna sinergiji med človeškimi očmi in možgani, omogoča iskanje ustrezne lokacije v iskalnem zemljevidu (satelitska slika) na podlagi slike drona. Ko je lokacija iskanja najdena na iskalnem zemljevidu, lahko iz podatkov o zemljepisni širini in dolžini iskalnega zemljevida sklepamo o trenutnem položaju drona.

# Poglavje 2

# Teoreticno Ozadje

## 2.1 Pregled literature

Tuki bos opisal o sorodhin delih in kake clanke si use si pogledu.

## 2.2 Osnovni pojmi in terminologija

### 2.2.1 Brezpilotni letalnik

Brezpilotni letalniki ali droni so zračna plovila, ki se lahko upravljajo na daljavo ali avtonomno preko programske opreme, ki je integrirana s senzorji in GPS sistemi. Droni imajo široko paleto uporabe v različnih industrijah, vključno z vojaško, komercialno, znanstveno in rekreativno uporabo. Droni so postali izjemno pomembni za zbiranje podatkov v realnem času, izvajanje raziskav in analiz na terenu, kot tudi za opravljanje nalog, ki so lahko za človeka nevarne ali nedostopne. Zaradi svoje fleksibilnosti in prilagodljivosti se uporabljajo v nalogah, kot so iskanje in reševanje, zaznavanje okoljskih sprememb, kmetijski nadzor, inspekcija infrastrukture, filmska produkcija, dostava paketov in še veliko več. Trg dronov je hitro rastoč in vključuje široko paleto proizvajalcev, ki ponujajo različne modele za različne namene in proračune. Kot tehnologija napreduje, se pojavljajo tudi novi izzivi, vključno

z vprašanji zasebnosti, varnosti in zakonodajnimi ureditvami. Zato je to področje postalo predmet intenzivnih raziskav in razvoja, s ciljem optimizacije zmogljivosti, zanesljivosti in dostopnosti brezpilotnih letalnikov.

### 2.2.2 Satelit

Sateliti so objekti, ki krožijo okoli Zemlje ali drugih nebesnih teles in se uporabljajo za številne namene, vključno z komunikacijo, opazovanjem vremena, znanstvenimi raziskavami, navigacijo in še veliko več. Za komercialno in vojaško uporabo je pomembna zlasti komunikacijska satelitska tehnologija. Ta omogoča globalno povezljivost in prenos podatkov, kot so televizijski signali, telefonski klici in internet. Vremenski sateliti so ključni za napovedovanje vremena in spremljanje okoljskih sprememb, saj zagotavljajo nenehne in natančne podatke o atmosferskih razmerah. Navigacijski sateliti, kot je sistem Global Positioning System (GPS), omogočajo določanje položaja in časa na skoraj katerem koli mestu na Zemlji. Ta tehnologija je ključna za številne aplikacije, od vojaške navigacije do vsakodnevnega usmerjanja v prometu. Znanstveni sateliti se uporabljajo za študij nebesnih teles, vključno z Zemljo. Ta opazovanja lahko pomagajo pri razumevanju podnebnih sprememb, geoloških procesov in drugih pomembnih znanstvenih vprašanj. Lansiranje satelita je kompleksno in drago opravilo, ki zahteva natančno načrtovanje in usklajevanje. Sateliti morajo biti postavljeni na natančno določeno orbito, da bi zagotovili optimalno delovanje in izogibanje trkom z drugimi objekti v vesolju.

### 2.2.3 Geolokalizacija

Geolokalizacija je proces določanja geografske lokacije objekta, kot je mobilni telefon, računalnik, vozilo ali kateri koli druga povezana naprava. Ta postopek je postal ključen del sodobnih tehnologij in se uporablja v številnih aplikacijah in storitvah. Geolokalizacija je bistvena za sistem GPS in druge navigacijske sisteme, ki voznikom, pohodnikom in drugim omogočajo na-

tančno navigacijo po poti do cilja. Podjetja uporabljajo geolokalizacijo za ciljno usmerjanje oglasov glede na lokacijo uporabnikov, kar omogoča, da so oglasi prilagojeni lokalnim zanimanjem in potrebam. Geolokalizacija se uporablja tudi v varnostnih aplikacijah, kot so sledenje vozil, iskanje izgubljenih ali ukradenih naprav in nadzor nad dostopom do določenih storitev na podlagi lokacije. V socialnih omrežjih in aplikacijah, ki uporabljajo lokacijo, je geolokalizacija omogočila uporabnikom, da delijo svojo lokacijo, najdejo prijatelje v bližini ali odkrijejo lokalne dogodke in atrakcije. Geolokalizacija se uporablja tudi v različnih znanstvenih raziskavah, kot je spremljanje selitve živali, raziskovanje tektonskih premikov in analiza podnebnih sprememb. Kljub številnim uporabam obstajajo tudi izzivi pri uporabi geolokalizacije. Natančnost geolokalizacije je odvisna od številnih dejavnikov, vključno z dostopnostjo satelitskih signalov, gostoto urbanih območij in uporabljeno tehnologijo. Prav tako se pojavljajo vprašanja glede zasebnosti in varnosti, saj lahko nepooblaščeno sledenje vodi v zlorabo informacij o lokaciji.

### 2.2.4 Iskanje tocke v sliki

Iskanje točke v sliki je postopek identifikacije in določanja posebnih točk ali predmetov v določeni sliki ali seriji slik. Ta tehnologija se je izkazala za pomembno v različnih aplikacijah, vključno z navigacijo in geolokalizacijo.

FPI je pozicijski standard, kjer je vhodna slika, ki jo je treba pozicionirati, poimenovana kot *query*, in slika, ki jo je treba pridobiti, se imenuje *search map*. Ta proces se lahko uporablja za različne naloge lokalizacije, kot so lokalizacija brezpilotnih letalnikov (UAV) in prečno geolokalizacijo. Glavni cilj je najti ustrezno lokacijo v iskalnem zemljevidu. FPI neposredno vnese poizvedbo in iskalni zemljevid v model, ki nato izpiše zemljevid toplote, ki predstavlja napovedano lokacijsko porazdelitev poizvedbe v iskalnem zemljevidu. Ena od ključnih prednosti metode FPI je, da ne zahteva veliko pripravljalnih podatkov ali operacij ekstrakcije značilnosti vnaprej. Edino shranjevanje, ki je potrebno, je iskalni zemljevid. Ta metoda omogoča hitro in natančno določanje lokacij v kompleksnih slikah in lahko služi številnim

namenom. Na primer, v scenarijih brezpilotnih letalnikov bi FPI lahko uporabili za identifikacijo in sledenje specifičnih lokacij ali objektov na tleh iz zraka. V geolokacijskih aplikacijah bi FPI lahko uporabili za določanje lokacije na satelitskih posnetkih. Skupno gledano iskanje točke v sliki, še posebej s pomočjo FPI metode, predstavlja pomemben korak naprej v tehnologijah lokalizacije in navigacije. Ponuja elegantno rešitev za težave, ki jih lahko tradicionalne metode imajo pri obdelavi kompleksnih slik in informacij, in je primerna za široko paleto aplikacij in industrije.

### 2.2.5 Konvolucijska nevronska mreza

Konvolucijska nevronska mreža (CNN - Convolutional Neural Network) je posebna vrsta umetnih nevronskih mrež, zasnovana za obdelavo vizualnih podatkov. S svojo zmožnostjo avtomatičnega in prilagodljivega učenja hierarhičnih značilnosti iz vhodnih podatkov se CNN pogosto uporablja v nalogah strojnega vida, kot so razpoznavanje vzorcev, klasifikacija slik in iskanje točk v slikah. Struktura CNN vključuje konvolucijske plasti, ki izvajajo konvolucijsko operacijo s pomočjo majhnih filtrirnih matrik za odkrivanje lokalnih značilnosti, kot so robovi, teksture in oblike. To sledi združevalnim plastem, ki zmanjšajo dimenzionalnost slike, hkrati pa ohranijo pomembne informacije. Na koncu se uporabljajo popolnoma povezane plasti, ki združijo lokalne značilnosti v globalno razumevanje slike, kar omogoča klasifikacijo ali regresijo. CNN se izkaže za zelo učinkovito v primerjavi z drugimi tipi nevronskih mrež v nalogah, povezanih z obdelavo slik, zlasti zaradi sposobnosti zajemanja prostorskih hierarhij značilnosti. To pomeni, da so sposobne razumeti in reprezentirati sliko na več ravneh abstrakcije.

### 2.2.6 Toplotna karta

Toplotna karta je grafična predstavitev podatkov, kjer vrednosti v matriki predstavljajo različne barve. Ponavadi se uporablja za prikazovanje, kako se določena spremenljivka razporedi po dvodimenzionalnem prostoru. To je

zelo uporabno pri vizualizaciji razmerij, povezav ali gostote v velikih naborih podatkov. Toplotne karte so priljubljene v mnogih znanstvenih in poslovnih aplikacijah. V statistiki in strojnem učenju se lahko uporabljajo za prikaz korelacij med različnimi značilnostmi. V biologiji se pogosto uporabljajo za prikazovanje izražanja genov, v geografiji pa za vizualizacijo gostote prebivalstva ali druge geolokacijske podatke. Ena od prednosti toplotne karte je, da omogoča hitro in intuitivno razumevanje kompleksnih naborov podatkov. Vizualizacija barvnih prehodov pomaga opazovalcu, da hitro zazna vzorce in trende, ki bi jih bilo težje zaznati v tabelaričnih ali tekstovnih prikazih.

# Poglavje 3

# Transformerske nevronske mreze

## 3.1 Transformerska arhitektura

### 3.1.1 Predhodni mehanizmi

Preden so obstajali transformerji, 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 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 Transformer arhitekture, ki je bila kasneje predstavljena v članku "Attention is All You Need"[7] leta 2017.

### 3.1.2 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 $\text{RNN}_{\text{enc}}$ in dekodirnik $\text{RNN}_{\text{dec}}$. Kodirnik z zaporedjem vektorjev $\vec{F}$ proizvede skrito stanje $h_n$:

$$h_n = \text{RNN}_{\text{enc}}(f_n, h_{n-1}) \tag{3.1}$$

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

$$e_t = \text{RNN}_{\text{dec}}(e_{t-1}, h_{t-1}) \tag{3.2}$$

Opomba: pri treniranju 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:

$$\vec{H} = \text{Encoder}(\vec{F}) \tag{3.3}$$

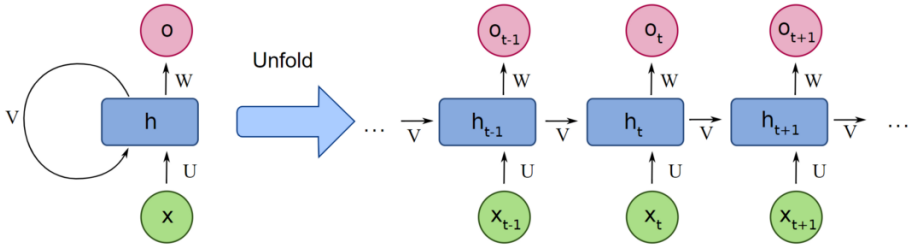
Za vsako besedo v ciljnem zaporedju $\vec{E}$ se potem izračuna ponderirana vsota skritih stanj iz kodirnika:

$$\vec{a}_t = \text{Attention}(\vec{H}, e_{t-1}) \tag{3.4}$$

Potem se ta vektor uporabi za napoved ciljne besede:

$$e_t = \text{Decoder}(\vec{a}_t, e_{t-1}) \tag{3.5}$$

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.



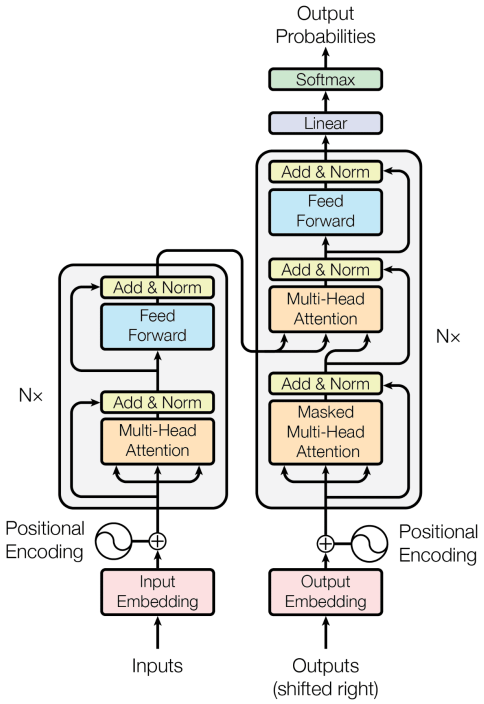

Slika 3.1: Rnn manjka citacija

## 3.2 Zgradba transformerja

V kontekstu strojnega prevajanja so avtorji v članku "Attention is all you need"[7] o pozornosti predstavili novo vrsto arhitekture, ki se loteva mnogih pasti modelov, ki temeljijo na RNN. Kljub vsem napredkom pri kodirnikih-dekodirnikih RNN, ki smo jih obravnavali zgoraj, je ostalo dejstvo, da so RNN težko paralelizabilni(parallel), ker zaporedno obdelujejo vhod. Ključna inovacija tega članka je, da so RNN in njihova skrita stanja v celoti nadomeščena z operacijami na osnovi pozornosti, ki so v mnogih problematičnih režimih bolj učinkovite.

Transformerski model je model kodirnika-dekodirnika. Kodirnik sestavljajo $N$ blokov na levi, dekodirnik pa $N$ blokov na desni.

Med ucenjem 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.




Slika 3.2: Izgled transformerskega modela, iz clanka "Attention is all you need"[7].

Postopek se ponavlja, dokler vseh $N$ blokov kodirnika ni obdelalo vhoda. Vsak blok ima dve komponenti: plast večglave samopozornosti, ki ji sledi popolnoma povezana plast z aktivacijami ReLU, ki obdeluje vsak element vhodne sekvence vzporedno. Tako večglavi sloj pozornosti kot popolnoma 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 popolnoma 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 ucenjem 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.

## 3.3 Scaled Dot-Product Attention

Ta funkcija se uporablja v vseh plasteh pozornosti v transformerju. Za zdaj bomo razčlenili matematiko za to operacijo, samo da dobimo občutek, katera števila gredo kam. Kasneje se bomo osredotočili na njegove aplikacije v članku.

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

Scaled Dot-Product Attention je skoraj identičen Dot-Product Attention, 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 izracun pozornosti. Zaceli bomo z zapisom posameznih vrstic $Q$ in $K$, nato pa bomo izrazili produkt $QK^T$ v smislu teh vrstic:

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

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

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

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 $V$:

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

Tukaj lahko opazimo, da mehanizem pozornosti rezultira v seriji 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, da v tem posebnem postopku ni nobenih učljivih parametrov - sestavljen je izključno iz matričnih in vektorskih operacij. Poglejmo si se posamezno vrstico $i$ uteži pozornosti:

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

kjer je $S$ normalizacijska konstanta:

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

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

$$q_i \cdot k_j = |q_i||k_j| \cos(\theta)$$

Uporaba eksponentne funkcije povečuje pozitivne vrednosti kosinusa in zmanjšuje negativne. 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), 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 z 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 bili poskusi za odpravo te težave in so služili kot naravni prehod v tehnike v tem članku.

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

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

Gradient softmax funkcije je izračunan kot:

$$\frac{\partial \text{softmax}(x_i)}{\partial x_j} = \text{softmax}(x_i) * (\delta_{ij} - \text{softmax}(x_j)) \tag{3.6}$$

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

$$\frac{\partial \text{softmax}(x_i/C)}{\partial x_j} = \frac{1}{C} * \text{softmax}(x_i/C) * (\delta_{ij} - \text{softmax}(x_j/C)) \tag{3.7}$$

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

### 3.3.1 Multi-Head Attention

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, ta matrika se preslika nazaj v izvirni prostor z uporabo druge linearne preslikave, da se pridobi končni rezultat večglave pozornosti. Avtorji to izrazijo v spodnji obliki:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O \tag{3.8}$$

Vsak $\text{head}_i$ je rezultat izvajanja Scaled Dot-Product Attention na $i$-tem nizu transformiranih poizvedb, ključev in vrednosti:

$$\text{head}_i = \text{Attention}(QW_{Qi}, KW_{Ki}, VW_{Vi}) \tag{3.9}$$

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 $W_O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Dokazimo, da se matrično množenje izide. Najprej vemo iz prejšnjega razdelka, da bo vsaka matrika head$i$ imela enako število vrstic kot $QWQi$ 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 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 drzi - 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 ucenjem.

#### Maskiranje vhodov

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

$$\frac{1}{\sqrt{d_k}}Q'K'^T + M = \tag{3.10}$$

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

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

Nato ima izvajanje softmaxa na vsaki vrstici učinek pošiljanja vseh $-\infty$ celic na 0, pri čemer ostanejo samo veljavni izrazi za pozornost. Tretja in zadnja uporaba večglave pozornosti v članku 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.

## 3.4 Vision Transformer (ViT)

Transformerji so prvotno bili omejeni na obdelavo zaporedij, kar je idealno za jezik, vendar ne nujno za slike, ki so običajno 2D. To se je spremenilo z razvojem Vision Transformerja (ViT) s strani Google-a [5]. Namesto da bi slike obdelovali kot 2D mreže pikslov (kot to počnejo konvolucijske nevronske mreže), Vision Transformer slike obravnava kot zaporedje majhnih kvadratov ali "oblizev". To omogoča uporabo enakih 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 oblize: Slika velikosti $H \times W \times C$ se razdeli na kvadrate (oblize) velikosti $P \times P$, kjer je $H$ višina, $W$ širina, $C$ število barvnih kanalov in $P$ velikost obliza. To ustvari $(H \cdot W)/P^2$ oblizev. Vsak obliz 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 transformerji 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 oblizev, da bi modelu dali nekaj informacij o tem, kje se obliz nahaja v sliki. Če je $z_i$ vložek $i$-tega obliza 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 transformerjev. Ti bloki vsebujejo večglavo samo-pozornost in mreže feed-forward, ki omogočajo modelu, da se nauči, kako povezati različne dele slike. Večglava samo-pozornost se lahko zapiše kot:

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

kjer je head$i$ = 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 samo-pozornosti.

- 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 transformerjevih blokov, in ReLU in softmax sta aktivacijski funkciji.

## 3.5 Piramidni vision transformer (PVT)

Piramidni Vision Transformer (PVT) [9] je bil razvit z namenom vključitve piramidne strukture v okvir 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 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, 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 vdelanih obližih 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:

$$F1 = \frac{H}{4} \times \frac{W}{4} \times C1 \tag{3.11}$$

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, da učinkovito obdela značilnostne mape visoke ločljivosti.

V primerjavi z Vision Transformer (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 računalništva in spomina, saj lahko obdela značilnostne mape višje ločljivosti ali daljše sekvence.

## 3.6 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, kako se uporabljajo pozicijski kodiranji. 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, da bi bolje obravnaval naloge goste napovedi, kot so zaznavanje objektov in semantična segmentacija. Vendar je bilo presenetljivo ugotovljeno, da je manjša učinkovitost PVT-ja predvsem 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 pogojena na vhodih in se 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 serije.

$$CPE = f(PEG(E_1, E_2, ..., E_n)) \tag{3.12}$$

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 kot CPVT, kar ga naredi enostavnega za učinkovito implementacijo. Eksperimentalni rezultati so pokazali, da ta preprosta zasnova lahko doseže zmogljivost nedavno predlaganega Swin transformerja [6].

# Poglavje 4

# Podatkovni set

## 4.1 Dronske slike

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 droni lokalizirajo in navigirajo med visokimi zgradbami, kjer so lahko GPS signali zmanjšani ali moteni. V zelenih območjih je cilj razumeti, kako se droni obnašajo v okoljih, kjer so vizualni vzorci manj raznoliki in se teren lahko zdi monoton. V naboru podatkov za ucenje je 10.000 slik iz desetih mest, pri čemer vsako mesto prispeva 1.000 slik. Droni so bili kalibrirani na višini 150 metrov nad navedeno nadmorsko višino mesta. Kamere na dronih 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 ucni 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.

Ta nabor podatkov ponuja vpogled v izzive in možnosti, ki jih droni srečujejo v različnih mestnih okoljih, in je ključnega pomena za razvoj naprednih algoritmov za lokalizacijo in navigacijo.


Slika 4.1: Primer dronske slike.

## 4.2 satelitske slike

Za vsako dronsko sliko sem poiskal 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 sem identificiral ustrezen satelitsko ploscico, sem jo prenesel neposredno iz Mapbox API-ja, priznanega vira za visokokakovostne satelitske slike. Da bi zagotovil dodatno globino in kontekst za vsako lokacijo, nisem prenesel samo osrednje ploscice, temveč tudi vse njegove sosednje ploščice. Te sosednje ploščice so bile nato združene z osrednjo ploscico za ustvarjanje enotne TIFF datoteke. Ta pristop je omogočil, da sem imel na voljo širšo regijo za analizo in učenje.

Ko sem imel pripravljene TIFF datoteke, sem začel z učnim procesom. Za vsako iteracijo učenja sem iz vsake TIFF datoteke naključno izrezal regijo velikosti 400x400 pikslov. Ključnega pomena je bilo, da se je točka lokalizacije

vedno nahajala nekje znotraj te izrezane regije. Ta metoda je zagotovila, da je bil model izpostavljen širokemu naboru scenarijev in kontekstov, hkrati pa je ohranila natančnost in relevantnost lokalizacijskih podatkov. S tem pristopom sem uspešno sestavil 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 ploscicah 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 ploscic je zelo priljubljen v spletnih kartografskih aplikacijah, kot je Google Maps.

Za pretvorbo geografskih koordinat (latitude in longitude) v ploscicne 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:

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

Slika 4.2: Primer pripadajoče satelitske slike za dronsko sliko.

# Poglavje 5

# Implementacija

### 5.0.1 Motivacija

Pri sledenju objektov raziskovalci sledenje izvajajo z izračunom podobnosti med predlogo in iskalnim območjem v trenutnem okviru. Metoda iskanja točk znotraj slike izhaja iz metode na področju sledenja objektov, vendar je ta metoda bolj zapletena kot sledenje objektov. To je zato, ker sta predloga (dronska slika) in iskalna slika (satelitske slike) iz različnih pogledov, kar povzroča veliko variabilnost.

Metoda iskanja točk z uporabo slike uporablja satelitsko sliko kot iskalno sliko in dronsko sliko kot poizvedbeno sliko. Nato se slike, posnete z dronom, in satelitske slike ustreznih območij prenesejo v end-to-end (celovito?) omrežje. Po obdelavi je rezultat toplotni zemljevid, točka z najvišjo vrednostjo na toplotnem zemljevidu pa je lokacija drona, kot jo napove model. Nato to lokacijo preslikamo na satelitsko sliko. Položaj drona lahko določimo glede na informacije o geografski širini in dolžini, ki jih ohranja satelitska slika. V FPI avtorji uporabljajo dva Deit-S brez deljenih uteži kot modula za ekstrakcijo značilnosti za vertikalne poglede slik drona in satelitskih slik [4]. Nato se ekstrahirane značilnosti podvržejo izračunu podobnosti, da se pridobi toplotni zemljevid. Končno preslikamo lokacijo z najvišjo vrednostjo toplotnega zemljevida na satelitsko sliko, da določimo lokacijo UAV.

V FPI se za izračun podobnosti uporablja zadnja plast zemljevidov značilnosti

[4]. Ker je končni izhodni zemljevid stisnjen 16-krat, model izgubi veliko prostorskih informacij. Izguba prostorskih informacij prinese nepopravljivo izgubo končne natančnosti pozicioniranja.

# Poglavje 6

# Eksperimenti

V tem poglavju se bom osredotocil na eksperimente, ki sem jih izvedel.

## 6.1 Izbira kriterjiske funkcije

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

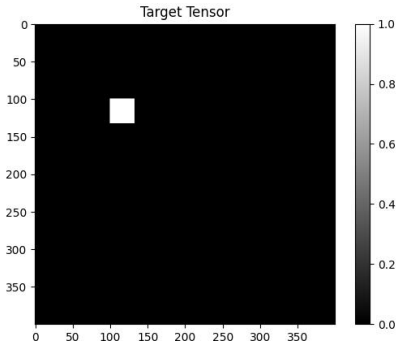
### 6.1.1 Hanningova kriterijska funkcija

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.

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 po-




Slika 6.1: Primer vzorca, sredisce je tocka lokacije vzorca.

zitivnih vzorcev, postane utež negativnih vzorcev manjša. Da bi to popravili, se uvede hiperparameter , imenovan Negativna utež (NG), ki prilagodi utež negativnih vzorcev.

Hanningova funkcija:

$$\text{Hanning}(n) = \begin{cases} 0.5 + 0.5\cos\left(\frac{2\pi n}{M-1}\right) & \text{za } 0 \leq n \leq M-1 \\ 0 & \text{sicer} \end{cases} \tag{6.1}$$

Utezi primerov:

- Utez negativnih vzorcev:

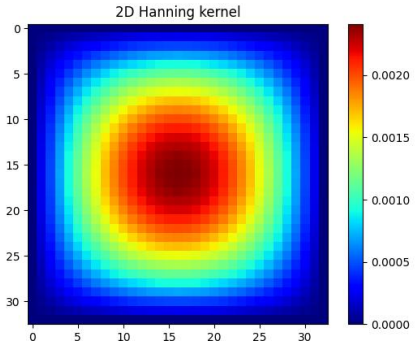
$$w_{pos} = NG/(NN(NW+1))$$

- 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 6.2: Hanningovo jedro

### 6.1.2 Gaussovo utezena srednja kvadratna napaka

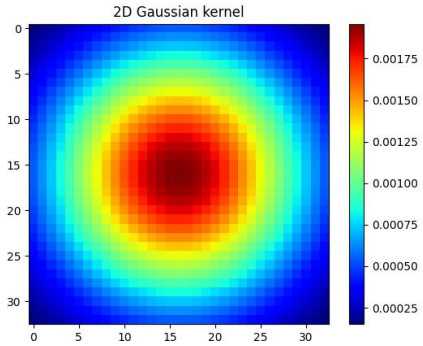
Gaussova utežena srednja kvadratna napaka (Gaussian Weighted Mean Squared Error - GWMSE) je modificirana funkcija izgube, namenjena izboljšanju modelov, ki obravnavajo podatke, kot so satelitske slike. Glavna značilnost GWMSE je dodeljevanje uteži vzorcem, na zelo podoben nacin kot pri Hanningovi funkciji 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:

$$\text{Gauss}(n) = \begin{cases} \exp\left(-\frac{(n-\mu)^2}{2\sigma^2}\right) & \text{za } 0 \leq n \leq M-1 \\ 0 & \text{sicer} \end{cases} \tag{6.2}$$

### 6.1.3 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




Slika 6.3: Gaussovo jedro

lokacijo. Za normalizacijo teh uteži se uporablja Hanningovo okno.

Hanningova funkcija je podana kot:

$$\text{Hanning}(n) = \begin{cases} 0.5 + 0.5\cos\left(\frac{2\pi n}{M-1}\right) & \text{za } 0 \leq n \leq M-1 \\ 0 & \text{sicer} \end{cases} \tag{6.3}$$

### 6.1.4 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:

$$\text{loss} = \frac{\text{true\_weight} \cdot N_{\text{true}} \cdot \text{MSE}_{\text{false}} + \text{false\_weight} \cdot N_{\text{false}} \cdot \text{MSE}_{\text{true}}}{N_{\text{all}}} \tag{6.4}$$

- $N_{\text{true}}$: število vzorcev, katerih resnična vrednost je večja od 0.

- $N_{\text{false}}$: število vzorcev, katerih resnična vrednost je enaka ali manjša od 0.
- $N_{\text{all}}$: 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
```

### 6.1.5 Primerjava rezultatov

## 6.2 Testiranje zmoznosti lokalizacije na testnih podatkih

pass

# Poglavje 7

# 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.

# Članki v revijah


[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).

[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).

# Članki v zbornikih

[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.

# 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).