SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 201

PRIMJENA JEZIČNIH UGRAĐIVANJA ZA SEMANTIČKU SEGMENTACIJU

Ivan Martinović

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 201

PRIMJENA JEZIČNIH UGRAĐIVANJA ZA SEMANTIČKU SEGMENTACIJU

Ivan Martinović

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

DIPLOMSKI ZADATAK br. 201

Pristupnik:	**Ivan Martinović (0036513627)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Primjena jezičnih ugrađivanja za semantičku segmentaciju**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate postižu duboki modeli koji znatan dio zaključivanja provode na poduzorkovanoj reprezentaciji. Međutim, standardno nadzirano učenje osjetljivo je na odabir taksonomije skupa za učenje kao i na pomak domene između skupova za učenje i ispitivanje. Ove probleme možemo umanjiti učenjem vizualnih koncepata na jezičnim ugrađivanjima. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće segmentacijske arhitekture utemeljene na konvolucijama i pažnji. Oblikovati postupak ugađanja dubokih modela koji se temelji na prednaučenim modelima koji povezuju vizualne koncepte s jezičnim ugrađivanjima. Odabrati slobodno dostupne skupove slika te oblikovati podskupove za učenje, validaciju i testiranje. Odabrati prikladni model te validirati hiperparametre. Vrednovati naučene modele te prikazati i ocijeniti postignutu točnost. Procijeniti prikladnost modela za učenje s prilagođavanjem domene. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 23. lipnja 2023.

Mojoj majci. U mom srcu ostat ćeš zauvijek.

SADRŽAJ

1. Uvod

Strojno učenje (engl. *machine learning*) razmatra računalne programe koji poboljšavaju svoju funkcionalnost iskorištavanjem podataka. Osnovni su elementi svakog algoritma strojnog učenja: model, funkcija gubitka te optimizacijski postupak. Često raspolažemo modelom koji je definiran do na neke parametre, a učenje modela svodi se na optimizaciju tih parametara s ciljem minimizacije vrijednosti funkcije gubitka [1]. Duboko učenje (engl. *deep learning*) područje je strojnog učenja koje se bavi optimizacijom modela koje opisujemo kompozicijom većeg broja nelinearnih transformacija. Takve modele možemo zvati i dubokim neuronskim mrežama. Primjena paradigme dubokog učenja, odnosno primjena dubokih neuronskih mreža široko je rasprostranjena u području računalnog vida (engl. *computer vision*).

Područje računalnog vida bavi se crpljenjem korisnih informacija iz digitalnih slika. Velik broj dubokih modela koji se danas koriste u području računalnog vida zasniva se na konvolucijskim slojevima, a takve modele nazivamo dubokim konvolucijskim modelima. Neki od najpoznatijih zadataka kojima se bavi područje računalnog vida su: klasifikacija slika (engl. *image classification*), detekcija objekata (engl. *object detection*), virtualna stvarnost (engl. *virtual reality*) te segmentacija slike (engl. *image segmentation*). U ovome radu naglasak je na zadatku semantičke segmentacije (engl. *semantic segmentation*) koji za cilj ima odrediti semantički razred svakog slikovnog elementa (engl. *pixel*) ulazne slike. Posebno zanimljivo područje u okviru računalnoga vida razumijevanje je scena vožnje sa svrhom razvoja algoritama koji bi omogućili autonomnu vožnju. Upravo je semantička segmentacija scena vožnje središnja tema ovog rada.

Nakon pojave vizualnih transformera [15], u računalnom vidu sve se češće koriste modeli koji se temelje na mehanizmu pažnje (engl. *attention*) [79]. Takvi modeli zapažene rezultate prvo su ostvarili u području obrade prirodnog jezika (engl. *natural language processing*, NLP), prije svega u zadacima strojnog prevođenja i klasifikacije

teksta. Uspjeh arhitektura koje se temelje na mehanizmu pažnje teško je ostvariv bez prijenosnog učenja (engl. *transfer learning*). Osnovna je ideja prednaučiti model na velikom skupu podataka i određenom zadatku, npr. na zadatku klasifikacije slika, a potom parametre naučenog modela iskoristiti kao inicijalizaciju modela za rješavanje nekog srodnog zadatka, npr. semantičke segmentacije.

Jedan od prvih modela koji je u fazi predučenja povezao računalni vid i obradu prirodnog jezika, odnosno vizualne koncepte i jezična ugrađivanja (engl. *language embeddings*) jest CLIP (*Constrastive Language-Image Pre-Training*) [61]. Korištenjem kontrastnog učenja, jezičnih ugrađivanja i velikog skupa podataka koji se sastojao od parova slika i odgovarajućih jezičnih opisa, CLIP postiže jednaku točnost klasifikacije na ImageNet [13] skupu podataka kao i arhitektura ResNet-50 [22] koja je naučena na skupu podataka ImageNet. Takav rezultat zaslužio je pažnju među istraživačima te ponukao na daljnje istraživanje doprinosa i robusnosti jezičnih reprezentacija u području računalnoga vida.

Primjena jezičnih latentnih reprezentacija za zadatak semantičke segmentacije te ispitivanje robusnosti tih reprezentacija predstavlja važan dio ovog rada. Ideja je naučiti modele za semantičku segmentaciju koristeći pritom okosnicu (engl. *backbone*) koja je prednaučena na parovima slika i odgovarajućih jezičnih opisa. Nadalje, cilj je iskoristiti latentne jezične reprezentacije odgovarajućih semantičkih razreda i vidjeti kako njihovo korištenje doprinosi generalizaciji. Sposobnost generalizacije ispitat ćemo korištenjem skupa za provjeru, ali i vrednovanjem naučenog modela na podacima iz druge domene. Radi toga ćemo modele naučiti na sintetičkim, a vrednovati na stvarnim scenama vožnje. Osim toga, prikazat ćemo i performanse modela kojega učimo na stvarnim scenama iz vožnje, a vrednujemo na stvarnim scenama u otežanim vremenskim uvjetima.

Ovaj rad podijeljen je u tri glavna dijela. Prvi i središnji dio rada semantička je segmentacija primjenom latentnih jezičnih reprezentacija. Arhitekture koje koristimo u ovom dijelu opisane su u poglavlju 4, a temeljni cilj ispitati je korisnost i robusnost jezičnih reprezentacija za zadatak semantičke segmentacije. Drugi dio rada reprodukcija je rezultata koji predstavljaju stanje tehnike (engl. *state of the art*, SOTA) iz područja adaptacije domene (engl. *domain adaptation*), a pregled modela koje koristimo u okviru ovoga dijela dan je u poglavlju 5. Posljednji dio vezan je uz sudjelovanje na natje-

canju ACDC[1] (*Adverse Conditions Dataset with Correspondences*) koje se održava u sklopu konferencije CVPR[2] (*Computer Vision and Pattern Recognition*). Pregled metoda koje koristimo u okviru ovog dijela prikazan je u poglavlju 3. Korišteni skupovi podataka za sva tri dijela navedeni su u poglavlju 6, dok su provedeni eksperimenti i dobiveni rezultati prikazani u poglavlju 7. Poglavlje 2 daje kratak pregled osnovnih i dobro poznatih metoda računalnog vida.

[1] `https://acdc.vision.ee.ethz.ch/`
[2] `https://cvpr2023.thecvf.com/`

2. Osnovne metode računalnog vida

U ovom poglavlju navodimo metode računalnoga vida koje predstavljaju osnovicu za sve provedene eksperimente u okviru ovoga rada. Prvo ćemo predstaviti operator diskretne konvolucije i konvolucijski sloj kao gradivni blok arhitektura u računalnom vidu. Nakon toga opisat ćemo tehniku normalizacije nad grupom, pojavu koja je omogućila stabilnije učenje dubokih modela. Potom ćemo predstaviti rezidualne jedinice s preskočnim vezama čija je pojava omogućila učenje još dubljih modela. Zatim ćemo dati pregled mehanizma pažnje te arhitekture *Transformer* koja se na tom mehanizmu temelji, a potom ćemo predstaviti ViT [15] kao pionira korištenja *Transformer* arhitekture u području računalnog vida.

2.1. Konvolucijski modeli

Model LeNet [39] smatra se prvom modernom konvolucijskom arhitekturom, a primijenjen je na zadatku prepoznavanja rukom pisanih znamenki. Pokretač razvoja modela koji se temelje na operatoru konvolucije bila je objava modela AlexNet [38]. Model AlexNet prvi je duboki konvolucijski model koji je pobijedio na ImageNet natjecanju klasifikacije slika [66], ostvarivši znatan napredak nad drugim metodama. Najjednostavnije rečeno, konvolucijski su modeli oni modeli koji imaju barem jedan konvolucijski sloj. Uz konvolucijske slojeve, često koristimo slojeve sažimanja te aktivacijske funkcije, od kojih danas prednjači korištenje zglobnice (engl. *Rectified Linear Unit*, ReLU) [50]. Konvolucijski modeli specijalizirani su za podatke s topologijom rešetke, a tipičan primjer takvih podataka su slike.

Operator konvolucije općenito definiramo kao skalarni umnožak jedne funkcije s obzirom na posmaknutu i reflektiranu drugu funkciju:

$$h\left(t\right) = \left(w * x\right)\left(t\right) = \int_{\mathcal{D}(w)} w\left(\tau\right) x\left(t - \tau\right) d\tau. \tag{2.1}$$

Kod konvolucijskih modela pod pojmom konvolucije najčešće podrazumijevamo unakrsnu korelaciju [99]:

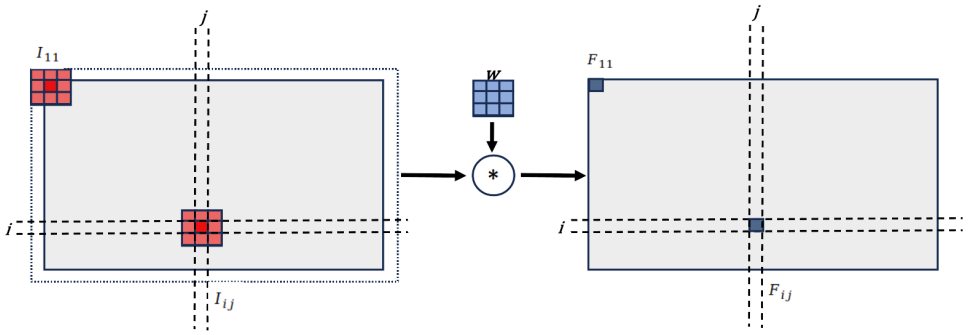
$$h\left(t\right) = \left(w * x\right)\left(t\right) = \int_{\mathcal{D}(w)} w\left(\tau\right) x\left(t+\tau\right) d\tau. \tag{2.2}$$

U kontekstu konvolucijskih modela funkciju x možemo promatrati kao ulaznu sliku, funkciju w nazivamo jezgrom, a rezultat h nazivamo mapom značajki. Budući da se bavimo slikama, odnosno diskretnim ulazima, integral možemo zamijeniti zbrojem, a pretpostavimo li da su vrijednosti funkcija x i w izvan domene jednake 0, slijedi:

$$h\left(t\right) = \left(w * x\right)\left(t\right) = \sum_{\tau=\tau_{min}}^{\tau=\tau_{max}} w\left(\tau\right) x\left(t+\tau\right) d\tau. \tag{2.3}$$

Pretpostavimo da raspolažemo sivom ulaznom slikom $\mathbf{I} \in \mathbb{R}^{H \times W}$, te parametrima jezgre $\mathbf{w} \in \mathbb{R}^{k \times k}$. Neka na izlazu dobivamo mapu značajki $\mathbf{F}$. Unakrsnu korelaciju ulazne slike $\mathbf{I}$ i jezgre $\mathbf{w}$ na lokaciji (i, j) dobivamo kao (slika 2.1):

$$\mathbf{F}_{ij} = \sum_{u=1}^{k} \sum_{v=1}^{k} I_{i+u-\left\lfloor \frac{k}{2} \right\rfloor - 1, j+v-\left\lfloor \frac{k}{2} \right\rfloor - 1} \cdot \mathbf{w}_{uv}. \tag{2.4}$$



Slika 2.1: Ilustracija diskretne konvolucije (unakrsne korelacije) na primjeru sive ulazne slike $\mathbf{I}$ i konvolucijske jezgre $\mathbf{w}$ dimenzije $k = 3$. Kako bismo zadržali rezoluciju izlaza, slika je po rubovima nadopunjena nulama, što je ilustrirano bijelim pravokutnikom s isprekidanim obrubom. Slika nastala po uzoru na [99].

Parametri jezgre $\mathbf{w}$ slobodni su parametri konvolucijskog sloja koje prilagođavamo za vrijeme učenja modela s ciljem minimizacije vrijednosti funkcije gubitka. U općenitom slučaju operator konvolucije provodit ćemo nad semantički bogatim mapama značajki u dubljim slojevima konvolucijskog modela. Pretpostavimo da raspolažemo s C_{in} mapa značajki koje označavamo s $\mathbf{F}_{in} \in \mathbb{R}^{C_{in} \times H_1 \times W_1}$. Provedemo li operaciju

unakrsne korelacije između tenzora $\mathbf{F}_{in}$ i konvolucijskih filtara $\mathbf{w} \in \mathbb{R}^{C_{out} \times C_{in} \times k \times k}$, na izlazu dobijamo tenzor $\mathbf{F}_{out} \in \mathbb{R}^{C_{out} \times H_2 \times W_2}$. Za r-tu od ukupno C_{out} mapa značajki na izlazu vrijedi:

$$\mathbf{F}_{out}^{(r)} = \sum_{c=1}^{C_{in}} \mathbf{F}_{in}^{(c)} * \mathbf{w}^{(r,c)}. \tag{2.5}$$

Unakrsnu korelaciju pod sumom u izrazu 2.5 na pojedinoj lokaciji (i, j) sada možemo računati jednako kao u izrazu 2.4. Pretpostavimo li da je vrijednost koraka (engl. *stride*) konvolucije u izrazu 2.5 jednaka 1, za prostornu rezoluciju izlaznog tenzora vrijedi:

$$H_2 = H_1 - (k-1), W_2 = W_1 - (k-1). \tag{2.6}$$

Vrijednost parametra k u primjenama je najčešće 3, 5 ili 7, a ako želimo izbjeći smanjenje rezolucije na izlazu svaku ravninu ulaznog tenzora $\mathbf{F}_{in}$ nadopunjujemo s $k-1$ nula po rubovima, kao što prikazuje slika 2.1.

2.1.1. Normalizacija nad grupom

Normalizacija nad grupom (engl. *batch normalization*) [31] često je korištena tehnika koja je imala veliki utjecaj na stvaranje boljih optimizacijskih uvjeta za učenje dubokih modela. Pojava normalizacije nad grupom omogućila je bržu konvergenciju i stabilnije učenje modela te mogućnost korištenja većih vrijednosti koraka učenja za vrijeme optimizacijskog postupka. Ideja normalizacije nad grupom normirati je vrijednosti aktivacija dubokih modela na način da se svaka značajka zasebno normira tako da joj je srednja vrijednost kroz sve raspoložive podatke jednaka 0, a standardna devijacija 1. Pretpostavimo li da je ulaz u jedan od slojeva dubokog modela $\mathbf{x} = (x_1, x_2, ..., x_d) \in \mathbb{R}^d$, tada normaliziramo svaku dimenziju ulaza kako slijedi:

$$\hat{x}_k = \frac{x_k - \mathbb{E}(x_k)}{\sqrt{\text{Var}(x_k)}}. \tag{2.7}$$

Očekivanje i disperziju svake pojedine značajke u idealnom slučaju računali bismo pomoću svih dostupnih primjera iz skupa za učenje. S obzirom na dubinu modela te veličinu skupova podataka koje danas koristimo, takvo računanje očekivanja i disperzije učinilo bi tehniku normalizacije nad grupom netraktabilnom. Autori članka [31] stoga uvode pojednostavljenje, na način da se aktivacije normiraju samo s obzirom na trenutnu mini-grupu podataka za učenje. Stoga, osim računalne efikasnosti, dobivamo i regularizacijski učinak normalizacije nad grupom s obzirom da će vrijednosti aktivacija modela za isti podatak ovisiti o ostalim primjerima u grupi. Tako dobivamo

rastresanje (engl. *jittering*) podataka na razini epohe. Kako bi omogućili modeliranje različitih distribucija aktivacija, kao i mogućnost modeliranja funkcije identiteta, autori članka [31] uvode slobodne parametre skaliranja i pomaka, pa konačna vrijednost normalizirane značajke iz izraza 2.7 postaje:

$$y_k = \gamma_k \hat{x}_k + \beta_k. \tag{2.8}$$

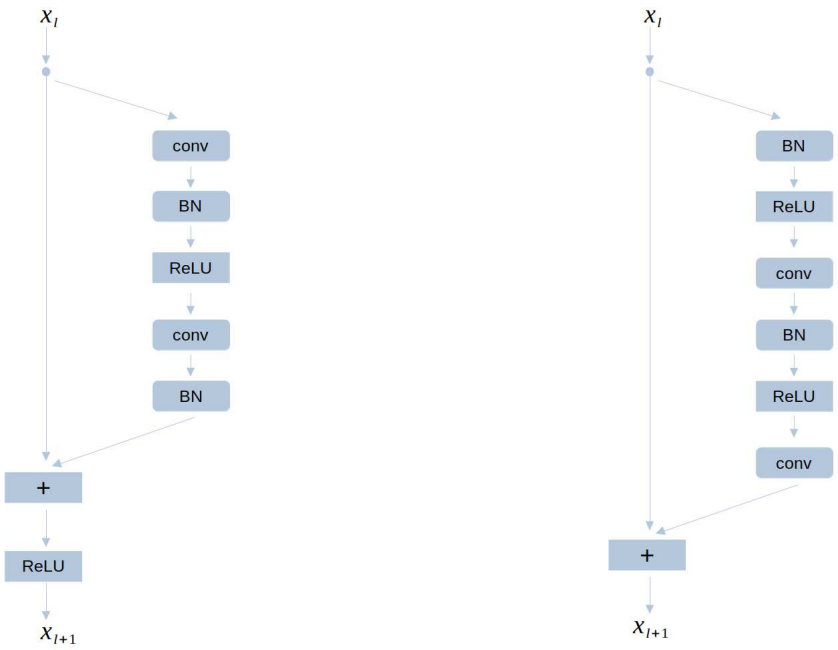
Neka raspolažemo s mini-grupom $\mathcal{B}$ od m primjera: $\mathcal{B} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)}\right\}$. Normalizaciju k-te značajke unutar trenutne mini-grupe provodimo tako da izračunamo procjene očekivanja i varijance [31]:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_k^{(i)}, \quad \sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_k^{(i)} - \mu_{\mathcal{B}}\right)^2. \tag{2.9}$$

Koristeći izračunate procjene očekivanja i standardne devijacije računamo normalizirane vrijednosti aktivacija prema izrazu 2.8. Slobodni parametri γ i β parametri su koje učimo za vrijeme optimizacijskog postupka. Bitno je istaknuti da kod dubokih konvolucijskih modela koji koriste normalizaciju nad grupom najčešće koristimo iste vrijednosti parametara γ i β za sve značajke iste mape značajki.

Korištenje normalizacije nad grupom u fazi zaključivanja (engl. *inference*) različito je nego korištenje za vrijeme učenja s obzirom da želimo biti u mogućnosti iskoristiti naš model na samo jednom primjeru. Stoga u fazi zaključivanja najčešće koristimo pomični prosjek (engl. *moving average*) očekivanja i standardne devijacije kojega pamtimo iz faze učenja. Druga mogućnost izračunati je statističke pokazatelje populacije, odnosno svih dostupnih primjera skupa za učenje. Kako bismo osigurali stabilno učenje koristeći tehniku normalizacije nad grupom, bitno je osigurati dovoljan broj primjera unutar mini-grupe kako bi dobivene procjene očekivanja i varijance bile pouzdane. Osim toga, korištenje normalizacije nad grupom povećava memorijski otisak učenja modela, a može dovesti i do nestabilnosti u slučaju pomaka u distribuciji između podataka za učenje i podataka na kojima model iskorištavamo.

2.1.2. Rezidualne mreže



(a) Izvorna građa rezidualne jedinice [22].

(b) Poboljšana građa rezidualne jedinice [23].

Slika 2.2: Ilustracija slijeda operacija izvorne [22] i poboljšane [23] verzije rezidualne jedinice. Normalizacija nad grupom i zglobnica izvorne rezidualne jedinice nalaze se nakon konvolucijskog sloja, a zglobnica se primjenjuje nakon zbrajanja s ulazom, dok poboljšana verzija rezidualne jedinice normalizaciju nad grupom te zglobnicu premješta prije konvolucijskog sloja. Slika nastala po uzoru na [23].

Pojava rezidualnih mreža (engl. *residual network*, ResNet) [22] predstavlja značajan događaj u razvoju dubokih modela računalnoga vida. Rezidualne mreže omogućile su učenje značajno dubljih arhitektura nego što je to bio slučaj prije njihove pojave. Povećanje dubine, odnosno povećanje broja slojeva našeg modela, pozitivno korelira s performansama modela, ali samo do određene dubine. Nakon dosezanja određenog broja slojeva, dolazi do degradacije performansi modela [22]. Ova pojava kontraintuitivna je s obzirom da dublji model može imitirati plitki model koji ima bolje performanse na način da nepotrebni slojevi obavljaju funkciju identiteta. Ideja je rezidualnih jedinica da jedan sloj modelira aditivnu pogrešku (rezidual), odnosno da dublji slojevi aditivno poboljšavaju predikcije prethodnih slojeva. Pretpostavimo kako je sloj modela l zadužen za učenje transformacije $\mathcal{H}_l(x_l)$, pri čemu x_l predstavlja ulaz l-tog sloja. Umjesto da nastojimo naučiti sloj našega modela da obavlja preslikavanje $\mathcal{H}_l$, ideja je da sloj

uči rezidualno preslikavanje $\mathcal{F}_l$, pri čemu općenito vrijedi:

$$\mathcal{H}_l\left(x_l\right) = \mathcal{F}_l\left(x_l\right) + h\left(x_l\right), \tag{2.10}$$

$$x_{l+1} = f(\mathcal{H}_l\left(x_l\right)). \tag{2.11}$$

Preslikavanje iz izraza 2.10 možemo ostvariti korištenjem preskočnih veza (engl. *skip connections*). Za izvornu rezidualnu jedinicu [22] vrijedi da je h funkcija identiteta, funkcija f je zglobnica, dok je preslikavanje $\mathcal{F}_l$ definirano kao niz transformacija prikazanih na slici 2.2a. Naknadno je pokazano [23] kako rezidualne jedinice postižu bolje performanse ako je i f funkcija identiteta, a normalizacija nad grupom i zglobnica se premjeste prije operacije konvolucije, kao što prikazuje slika 2.2b. Slijednom kompozicijom rezidualnih jedinica nastaju rezidualne mreže koje često nalazimo kao sastavne dijelove modernih arhitektura, uglavnom u ulozi ekstraktora značajki.

2.2. Preko strojnog prevođenja do klasifikacije slika

U ovom odjeljku napravit ćemo kratak izlet u područje obrade prirodnog jezika i pokazati na koji način je razvoj arhitektura u tom području pomogao razvoju metoda u računalnom vidu. Kako bi računalni program bio sposoban razumjeti jezik, jasno je da riječi moramo prikazati u obliku pogodnom za računalo. Stoga ćemo riječi prikazivati numeričkim reprezentacijama. Danas su često korištene distribucijske reprezentacije riječi (engl. *word-embeddings*) naučene pomoću jednostavnog modela dubokog učenja, a nazivamo ih *Word2Vec* [49] reprezentacijama. Distribucijska jezična ugrađivanja temelje se na distribucijskoj pretpostavci, prema kojoj riječi koje se pojavljuju u sličnom kontekstu imaju slično značenje [20]. Danas najčešće koristimo prednaučene vektorske reprezentacije riječi izgrađene nad velikim korpusima teksta. Odjeljci 2.2.1 i 2.2.2 inspirirani su radom [98].

2.2.1. Mehanizam pažnje

Do pojave arhitekture *Transformer* [79] glavnu riječ u području obrade prirodnog jezika, u kombinaciji s prednaučenim vektorskim reprezentacijama, imale su povratne neuronske mreže (engl. *recurrent neural network*, RNN), od kojih su najznačajniji predstavnici ćelije LSTM (*Long-Short Term Memory*) [26] i GRU (*Gated Recurrent Unit*) [10]. Važna karakteristika povratnih ćelija postojanje je povratne veze koja modelira ovisnost trenutnog skrivenog stanja (engl. *hidden state*) o prethodnom skrivenom stanju. Takva ovisnost zahtijeva slijedno računanje skrivenih stanja kroz vremenske

korake, što za posljedicu ima onemogućavanje paralelnog izvođenja.

Uzmimo za primjer zadatak strojnog prevođenja (engl. *machine translation*). Standardne koder-dekoder arhitekture [9, 73] u prvom koraku ulaznu rečenicu kodiraju u vektor kojega nazivamo vektorom konteksta te označavamo s $\mathbf{c}$. Pretpostavimo da na ulazu imamo rečenicu $\mathbf{X}$ sastavljenu od T riječi koje su predstavljene vektorskim reprezentacijama $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$. Koder, najčešće povratna neuronska mreža, reprezentaciju ulazne rečenice $\mathbf{X}$ transformira u vektor konteksta $\mathbf{c}$ prema izrazima [4]:

$$\mathbf{h}_t = f\left(\mathbf{h}_{t-1}, \mathbf{x}_t\right), \quad \mathbf{c} = g\left(\{\mathbf{h}_1, \cdots, \mathbf{h}_T\}\right). \tag{2.12}$$

Funkcije f i g najčešće su nelinearne funkcije, dok je $\mathbf{h}_t$ vektor skrivenog stanja u koraku t. S druge strane, zadaća je dekodera generiranje prijevoda, odnosno maksimizacija vjerojatnosti izlaznog niza $Y = (y_1, y_2, ..., y_{T'})$:

$$p\left(Y\right) = \prod_{t=1}^{T'} p\left(y_t \mid \{y_1, \cdots, y_{t-1}\}, \mathbf{c}\right). \tag{2.13}$$

Uvjetnu vjerojatnost iz izraza 2.13 moguće je modelirati pomoću povratne ćelije [4]:

$$p\left(y_t \mid \{y_1, \cdots, y_{t-1}\}\right) = q\left(y_{t-1}, \mathbf{s}_t, \mathbf{c}\right). \tag{2.14}$$

U izrazu 2.14 q predstavlja nelinearnu funkciju, dok je $\mathbf{s}_t$ skriveno stanje dekodera u koraku t. Jasno je da je vektor konteksta $\mathbf{c}$ usko grlo ovakve arhitekture, osobito ako koder na ulazu dobiva izrazito duge rečenice, a kodira ih u vektor fiksne dimenzije.

Kako bi riješili problem uskog grla vektora konteksta, autori članka [4] predstavljaju mehanizam pažnje (engl. *attention mechanism*). Za razliku od izraza 2.14, uvođenjem sloja pažnje vjerojatnost izlaza y_t uvjetujemo uvijek novim vektorom konteksta $\mathbf{c}_t$, stoga izraz 2.14 postaje:

$$p\left(y_t \mid \{y_1, \cdots, y_{t-1}\}\right) = q\left(y_{t-1}, \mathbf{s}_t, \mathbf{c}_t\right). \tag{2.15}$$

Vektor konteksta $\mathbf{c}_t$ računa se kao težinski prosjek skrivenih stanja kodera [4]:

$$\mathbf{c}_t = \sum_{j=1}^{T} \alpha_{tj} \mathbf{h}_j. \tag{2.16}$$

Težine α_{tj} računamo pomoću izraza:

$$e_{tj} = a\left(\mathbf{s}_{t-1}, \mathbf{h}_j\right), \tag{2.17}$$

$$\alpha_{tj} = \frac{exp\left(e_{tj}\right)}{\sum_{k=1}^{T} exp\left(e_{tk}\right)}. \tag{2.18}$$

U izrazu 2.17 funkciju a modeliramo potpuno povezanim slojem, dok vrijednosti α_{tj} iz izraza 2.18 možemo interpretirati kao važnost koju pridajemo skrivenom stanju kodera na poziciji j pri generiranju riječi u trenutku t. Upravo uz takvu interpretaciju govorimo o uvođenju sloja pažnje (engl. *attention layer*).

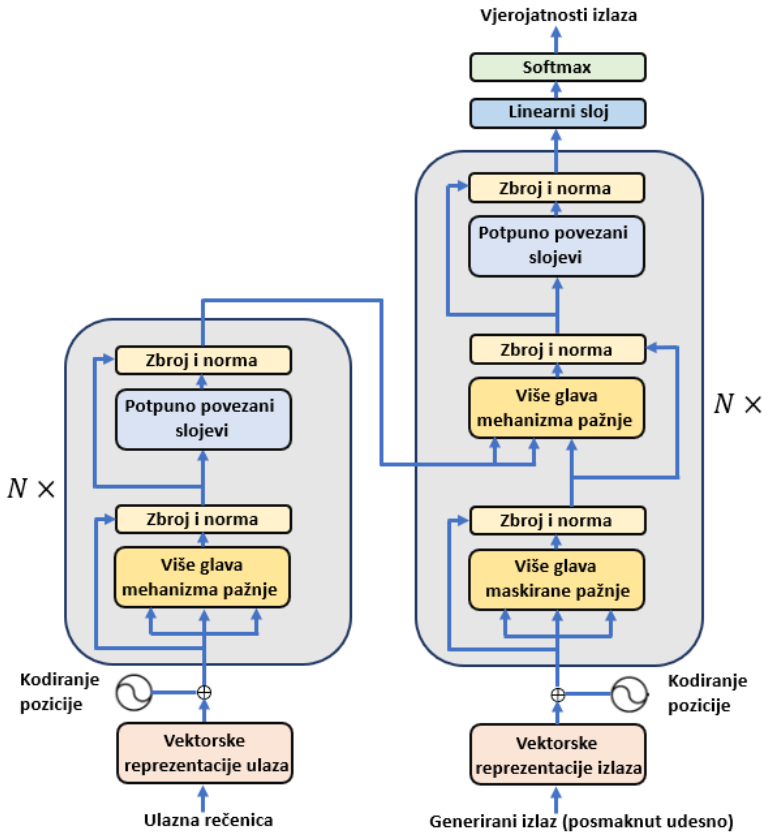
2.2.2. Arhitektura *Transformer*

Nemogućnost paralelizacije računanja skrivenih stanja kod povratnih neuronskih mreža stvarala je problem u području obrade prirodnog jezika sve do pojave arhitekture *Transformer* [79] koja se u potpunosti temelji na mehanizmu pažnje. Ova arhitektura prati koder-dekoder strukturu kao što je opisano u odjeljku 2.2.1, ali umjesto povratnih mreža koriste se slojevi pažnje te potpuno povezani slojevi, koji zajedno predstavljaju jedan blok *Transformera*. Zadaća je dekodera autoregresivno generirati riječ za vremenski korak t, pri čemu se dekoder koristi dostupnim reprezentacijama kodera i jezičnim ugrađivanjima generiranima do koraka $t-1$ [18], kao u izrazu 2.15. Slika 2.3 prikazuje *Transformer* arhitekturu.

Koder modela *Transformer* sastoji se od $N = 6$ blokova, pri čemu se svaki blok kodera sastoji od dva podsloja. Prvi podsloj svakog bloka predstavlja više glava mehanizma pažnje (engl. *multi-head self-attention*), a drugi podsloj čine dva potpuno povezana sloja i zglobnica. Rezidualnim vezama (engl. *residual connections*) [22] ulazi u svaki od podslojeva zbrajaju se s njihovim izlazima (2.10) te normiraju (engl. *layer normalization*) [2]. Kao i koder, dekoder modela *Transformer* sastoji se od $N = 6$ jednakih blokova. Osim dva već spomenuta podsloja koja predstavljaju sastavne dijelove kodera, dekoder sadrži i dodatni podsloj koji predstavlja više glava mehanizma pažnje. Ključevi i vrijednost dodatnog podsloja dolaze iz izlaza kodera, pa ovakav tip pažnje nazivamo unakrsnom pažnjom (engl. *cross-attention*). Prvi podsloj mehanizma višestruke pažnje dekodera modificiran je tako da prilikom generiranja sljedeće riječi sloj pažnje ne vidi buduće riječi, odnosno kažemo kako su buduće riječi maskirane (engl. *masked self-attention*). Ovakva modifikacija osigurava da dekoder u svakom koraku doista i modelira uvjetnu vjerojatnost:

$$p\left(y_t \mid \{y_1, y_2, \cdots, y_{t-1}\}, \mathbf{H}\right). \tag{2.19}$$

Pritom $\mathbf{H} = (\mathbf{h}_1, ..., \mathbf{h}_n)$ u izrazu 2.19 predstavlja izlaz kodera. Sloj pažnje kod *Transformer* arhitekture predstavlja transformaciju upita (engl. *query*) i skupa parova ključ-vrijednost (engl. *key-value*) u izlaz, pri čemu su upit, ključevi i vrijednosti vektori. Izlaz računamo kao težinsku sumu vektora vrijednosti, pri čemu težine računamo ne-



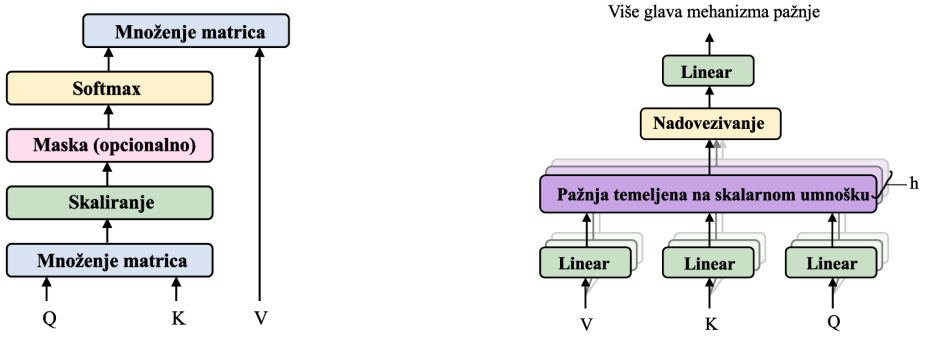
Slika 2.3: Prikaz arhitekture *Transformer* s vizalno razdvojenim koderom (lijevo) i dekoderom (desno). Oba dijela, koder i dekoder, sastoje se od N blokova. Svaki blok kodera čini više glava mehanizma pažnje te potpuno povezani slojevi, a oko svakog podsloja dodane su i preskočne veze [22] te normalizacija sloja [2]. Sličnu građu ima i blok dekodera, a razlika je u tom što postoje dva podsloja pažnje, pri čemu prvi podsloj predstavlja više glava maskirane pažnje, a u drugom podsloju pažnje ključevi i vrijednosti dolaze iz izlaza kodera. Slika nastala po uzoru na [79].

kom funkcijom koja predstavlja sličnost između vektora upita i odgovarajućeg vektora ključa [79]. Označimo vektor upita s $\mathbf{q} \in \mathbb{R}^{d_{model}}$, skup ključeva smjestimo u matricu $\mathbf{K} \in \mathbb{R}^{N \times d_{model}}$, a skup vrijednosti u matricu $\mathbf{V} \in \mathbb{R}^{N \times d_{model}}$, pri čemu N predstavlja broj parova ključ-vrijednost. Zbog mogućnosti paralelnog izračuna mehanizma pažnje za sve upite, u praksi ćemo i vektore upita $\mathbf{q}$ smjestiti u matricu upita $\mathbf{Q} \in \mathbb{R}^{N \times d_{model}}$. Mehanizam pažnje temeljen na skaliranom skalarnom umnošku (engl. *scaled dot-product attention*) [79] tada računamo prema izrazu:

$$attention\,(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.20}$$

Možemo primijetiti kako izraz 2.20 zahtijeva samo da je dimenzija upita jednaka dimenziji ključa, ali kako bismo mehanizam pažnje mogli ugraditi u model prikazan na

slici 2.3, treba vrijediti da je i dimenzija vektora vrijednosti jednaka dimenziji ključa i upita, odnosno jednaka d_{model}. Napravimo li usporedbu s mehanizmom pažnje opisanim u odjeljku 2.2.1, upit bi predstavljao vrijednost skrivenog stanja dekodera $\mathbf{s}_{t-1}$, vektori ključeva i vrijednosti bili bi $(\mathbf{h}_1, ..., \mathbf{h}_T)$, a mjera sličnosti između upita i ključa modelirana je potpuno povezanim slojem a iz izraza 2.17. Ako govorimo o maskiranom mehanizmu pažnje, kao što je to slučaj u dekoderu, prije nego što provedemo funkciju $softmax$, na pozicije na koje ne želimo obraćati pažnju postavljamo vrijednost $-\infty$ što će rezultirati time da vrijednosti težina na mjestima na kojima ne želimo obraćati pažnju poprime vrijednost 0. Mehanizam pažnje temeljen na skalarnom umnošku ilustriran je na Slici 2.4a, dok je mehanizam pažnje s više glava ilustriran na Slici 2.4b.



(a) Ilustracija mehanizma skalirane pažnje. **(b)** Ilustracija više glava mehanizma pažnje.

Slika 2.4: Ilustracija mehanizma skalirane pažnje iz izraza 2.20 te mehanizma pažnje s više glava iz izraza 2.21. Slika nastala po uzoru na [79].

Autori članka [79] bolje performanse dobili su zamjenom jednostavnog mehanizma pažnje (izraz 2.20) s više glava mehanizma pažnje. Umjesto računanja pažnje s d_{model}-dimenzionalnim ključevima, vrijednostima i upitima, ideja je korištenjem potpuno povezanih slojeva projicirati ključeve, vrijednosti i upite h puta i dobiti dimenzije odgovarajućih vektora d_k, d_v i d_q. Nakon toga obavljamo h paralelnih računanja pažnje nad dobivenim projekcijama ključeva, vrijednosti i upita. Naposljetku dobivene projekcije nadovežemo (engl. *concat*), dobijemo izlazne reprezentacije dimenzija $h \times d_v$, a potom primjenom još jedne linearne projekcije dobijemo odgovarajuću dimenziju d_{model}. Opisanu modifikaciju formalno možemo predstaviti izrazima [79]:

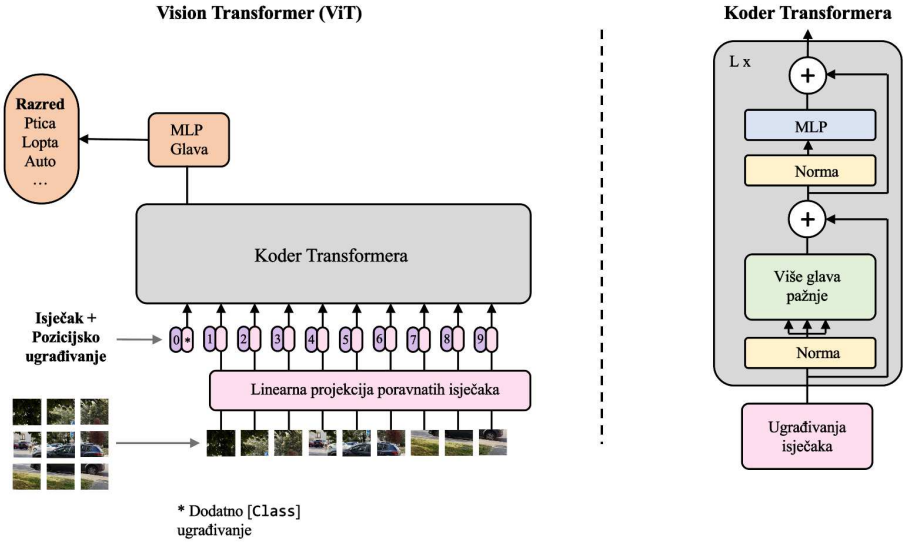
$$multihead\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = concat\left(head_1, head_2, ..., head_h\right)\mathbf{W}^O, \tag{2.21}$$

$$head_i = attention\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right). \tag{2.22}$$

Pritom za linearne projekcije vrijedi $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$ te $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$. Važno je naglasiti da je *Transformer* invarijantan na permutacije ulaznog slijeda, zbog čega gubimo informaciju o poretku koju smo imali kod povratnih neuronskih mreža. Kako bismo doskočili tom problemu, koristimo pozicijsko kodiranje (engl. *positional encoding*), što je i prikazano na Slici 2.3. Pozicijsko kodiranje temelji se na nekoj periodičnoj funkciji (npr. sinus) čiji se iznos zbraja sa svakom dimenzijom ugrađivanja ulaznog slijeda.

2.2.3. ViT

Pojava arhitekture *Transformer* u početku se odrazila uglavnom u području obrade prirodnog jezika. Brojne arhitekture koje se temelje na *Transformeru* postale su praktički standard [14, 58, 59], a korištenjem paradigme prijenosa znanja (engl. *transfer learning*) takve arhitekture podigle su granicu uspješnosti modela u području obrade prirodnog jezika. S druge strane, primjena *Transformera* u području računalnoga vida nije bila u prvom planu sve do pojave modela ViT (Vision Transformer) [15]. Model ViT u potpunosti se temelji na ranije opisanoj arhitekturi *Transformer*, konkretnije na koderu te arhitekture. Prepreka za korištenje *Transformera* u području računalnoga vida prije svega bilo je pitanje kako ulaznu sliku predstaviti kao niz ugrađivanja. Naivan pristup bio bi predstaviti sliku kao niz pojedinačnih piksela. Osim memorijskih problema (složenost mehanizma samopažnje kvadratno skalira s duljinom ulaza), postavlja se pitanje značenja pojedinačnog piksela izvan konteksta slike. Autori članka [15] predlažu stvaranje slikovnih ugrađivanja iz isječaka ulazne slike. Neka je na ulazu modela slika $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$. Ideja je ulaznu sliku pretvoriti u niz poravnatih dvodimenzionalnih isječaka (engl. *flattened 2D patches*) $\mathbf{x}_p \in \mathbb{R}^{N \times \left(P^2 \cdot C\right)}$, pri čemu je $P \times P$ veličina svakog isječka, C je broj kanala, dok $N = \frac{HW}{P^2}$ predstavlja konačan broj isječaka. U radu [15] koristi se veličina isječka $P = 16$.



Slika 2.5: Prikaz modela ViT. Prvi korak podjela je slike u isječke fiksnih dimenzija te njihovo poravnavanje, a potom radimo linearnu projekciju isječaka te dodajemo pozicijska ugrađivanja. Dobivena slikovna ugrađivanja dovodimo na ulaz kodera *Transformera*. Prvo ugrađivanje u slijedu predstavlja dodatno [Class] ugrađivanje iz kojega određujemo konačne predikcije u slučaju zadatka klasifikacije slike. Važno je naglasiti kako se koder *Transformera* razlikuje od onog opisanog u odjeljku 2.2.2 po tomu što se norma premješta prije mehanizma pažnje. Slika nastala po uzoru na [15].

3. Metode za semantičku segmentaciju

Zadatak semantičke segmentacije odrediti je semantički razred za svaki piksel ulazne slike, odnosno radi se o gustoj (engl. *dense*) predikciji. Ako radimo semantičku segmentaciju scena vožnje, primjeri semantičkih razreda mogli bi biti automobil, pješak ili cesta. Obično ovaj zadatak učimo nadzirano, tako da imamo na raspolaganju parove slika i pripadnih segmentacijskih mapi, a jedan primjer prikazuje Slika 3.1.

Slika 3.1: Primjer ulazne slike (lijevo) i pripadne segmentacijske mape (desno). Na segmentacijskoj mapi možemo jasno povezati boje i odgovarajuće semantičke razreda, npr. crveno bi bili ljudi. Slike su preuzete iz skupa podataka *Cityscapes* [11].

Odjeljak 3.1 daje uvid u najvažnija postignuća za zadatak semantičke segmentacije iz literature, a u odjeljcima nakon toga predstavit ćemo arhitekture i tehnike za semantičku segmentaciju koje koristimo u okviru ovoga rada.

3.1. Prethodni rad

Nakon pojave članka [44] konvolucijske mreže postale su standardni pristup u rješavanju zadatka semantičke segmentacije. Dva su problema s kojima se susrećemo pri rješavanju zadatka semantička segmentacije: povećanje receptivnog polja te rekonstrukcija rezolucije ulazne slike na izlazu. Jedna mogućnost za rekonstrukciju ulazne rezolucije učenje je puta naduzorkovanja, a tada govorimo o arhitekturi koder-dekoder (engl. *encoder-decoder*). U takvim arhitekturama koder je zadužen za prepoznavanje (engl. *recognition*), a dekoder je zadužen za naduzorkovanje te određivanje

prostornih granica između semantičkih razreda. Početne arhitekture karakterizirala je simetrična struktura kodera i dekodera [3], a potom i ljestvičasto (engl. *ladder-style*) povezivanje kodera i dekodera s ciljem sjedinjavanja (engl. *blend*) semantički bogatih mapa značajki kodera s prostorno bogatim značajkama dekodera [62, 64]. S obzirom na vjerovanje da je zadatak određivanja semantike (koder) teži od zadatka određivanja prostornih granica (dekoder) [48], prema [37, 48] ima smisla koristiti asimetričnu koder-dekoder arhitekturu, pri čemu koder ima značajno veći kapacitet. Što se tiče tehnika za povećanje receptivnog polja, rani pristupi temeljili su se na korištenju dilatiranih konvolucija (engl. *dilated convolutions*) [87, 88]. Povećanje receptivnog polja moguće je i korištenjem modula SPP (Spatial Pyramid Pooling) [21, 37, 48] ili izgradnjom piramide značajki na način da na ulaz modela dovodimo slike različitih rezolucija (engl. *resolution pyramid*) [16, 48]. Moderni pristupi rješavanju zadatka semantičke segmentacije temelje se na arhitekturi *Transformer* [79]. Radovi [51, 55] pokazuju kako su *Transformeri* robusniji na šum, maskiranje regija slike ili pomake u domeni od konvolucijskih modela. Poznate arhitekture za semantičku segmentaciju koje se temelje na *Transformeru* svakako su *Swin* [42] i *SegFormer* [84], a u posljednje vrijeme izvrsne performanse postižu metode koje se temelje na klasifikaciji maski [7, 32].

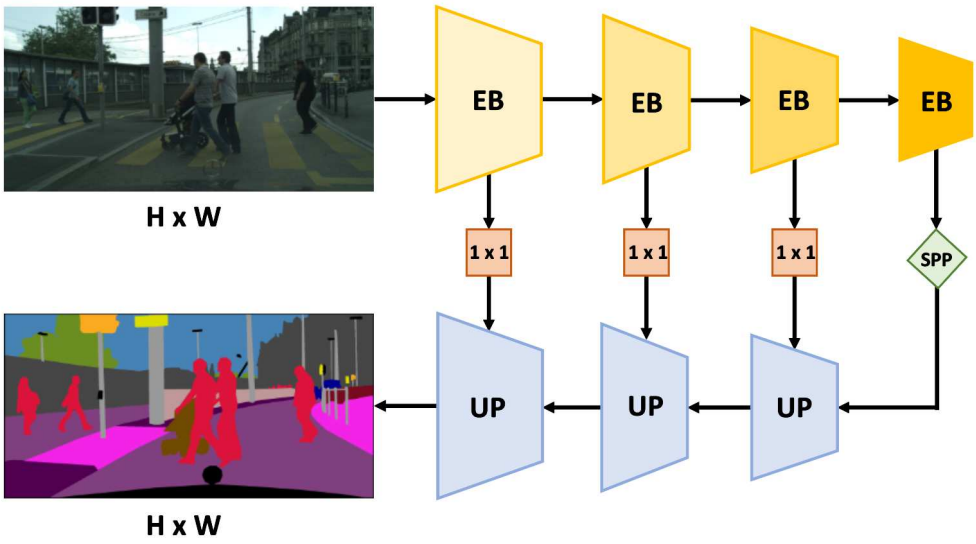
3.2. *SwiftNet*

Arhitektura *SwiftNet* predstavlja asimetričnu koder-dekoder arhitekturu za rješavanje zadatka semantičke segmentacije. Koder arhitekture zadužen je za učenje semantičkih značajki, dok je dekoder zadužen za naduzorkovanje semantički bogatih značajki niske rezolucije na originalnu rezoluciju ulaza (engl. *upsampling decoder*). Osim ova dva dijela, ideja autora iskoristiti je tehnike za povećanje receptivnog polja, modul SPP [21] te piramidu značajki [16].

3.2.1. Jednorezolucijski *SwiftNet*

Jednorezolucijska (engl. *single-scale*) inačica *SwiftNeta* transformira ulaznu sliku u segmentacijsku mapu na izlazu koristeći pritom koder koji na izlazu daje hijerarhiju značajki, modul SPP te dekoder zadužen za naduzorkovanje. Sika 3.2 prikazuje arhitekturu jednorezolucijskog *SwiftNeta*. Kao koder obično koristimo model prednaučen na *ImageNetu* (npr. *ResNet-18* [22]) koji na izlazu daje hijerarhiju značajki. Koder se sastoji od 4 bloka, pri čemu i-ti blok kodera daje značajke s brojem kanala C_i kojima

je rezolucija 2^{i+1} puta poduzorkovana s obzirom na rezoluciju ulazne slike. Obično vrijedi da blokovi kodera na dubljim razinama daju značajke s više kanala, odnosno $C_{i+1} > C_i$. Modul SPP [21] provodi sažimanje prosjekom nad značajkama F_4 koje predstavljaju izlaz posljednjeg bloka kodera. Sažimanje provodimo 4 puta, tako da na izlazu imamo značajke s rezolucijama 1×1, 2×2, 4×4 te 8×8 (pod pretpostavkom kvadratnih ulaznih isječaka). Dobivene značajke nakon sažimanja provodimo kroz blok koji se sastoji od normalizacije nad grupom, zglobnice i 1×1 konvolucije (blok *BNReLUConv*), a potom bilinearno naduzorkujemo na rezoluciju značajki F_4. Posljednji korak nadovezivanje je dobivenih značajki po dimenziji koja predstavlja broj kanala te redukcija broja kanala još jednim blokom *BNReLUConv*.



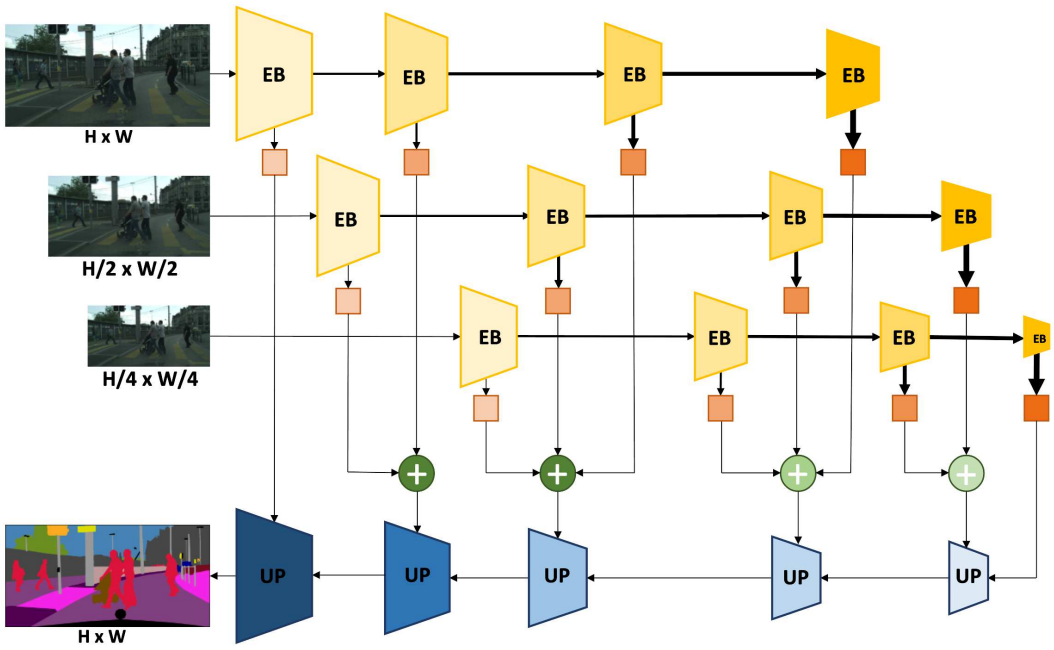
Slika 3.2: Jednorezolucijski *SwiftNet*. Blokovi EB (Encoder Block) predstavljaju blokove kodera, a različitim nijansama žute označavamo različit broj parametara. Različite veličine blokova predstavljaju različite rezolucije značajki na izlazu iz bloka. Zeleni kvadrat predstavlja modul SPP, crveni kvadratići predstavljaju 1×1 konvolucije zadužene za izjednačavanje broja kanala značajki kodera i dekodera, dok svjetloplavi blokovi predstavljaju blokove dekodera koji su zaduženi za naduzorkovanje. Slika nastala po uzoru na [48].

Zadaća dekodera naduzorkovanje je semantički bogatih značajki koje dobijemo na izlazu modula SPP. Prva tri bloka kodera povezana su poprečnim vezama (engl. *lateral connections*) s odgovarajućim blokovima dekodera (Slika 3.2). Svaki blok dekodera na ulazu dobije semantički bogate značajke iz odgovarajućeg bloka kodera te bilinearno naduzorkovane značajke prethodnog bloka dekodera. Značajke iz kodera prije dolaska na dekoder provodimo kroz blok *BNReLUConv* kako bismo osigurali jednak broj kanala značajki. Značajke iz odgovarajućeg bloka kodera i bilinearno naduzorkovane

značajke prethodnog bloka dekodera potom zbrojimo te provedemo kroz još jedan blok *BNReLUConv*, ali ovoga puta koristimo konvoluciju s veličinom jezgre 3. Nakon posljednjeg bloka dekodera na izlazu imamo značajke poduzorkovane 4 puta s obzirom na rezoluciju ulazne slike. Takve značajke provodimo kroz blok *BNReLUConv* kako bismo dobili broj kanala jednak broju mogućih razreda, a dobivene značajke potom bilinearno naduzorkujemo 4 puta kako bismo rekonstruirali rezoluciju ulazne slike.

3.2.2. Višerezolucijski *SwiftNet* s piramidom značajki

Višerezolucijski (engl. *multi-scale*) *Swiftnet* [48] s povezivanjem piramide (engl. *pyramid fusion*) značajki nadogradnja je jednorezolucijskog *SwiftNeta*. Za povećanje receptivnog polja više ne koristimo modul SPP, nego stvaramo piramidu značajki tako da na ulaz kodera osim ulazne slike originalne rezolucije dovedemo 2 odnosno 4 puta poduzorkovanu sliku, što se može vidjeti na Slici 3.2.2. Ovim nastojimo omogućiti prepoznavanje objekata na različitim skalama.



Slika 3.3: Višerezolucijski *SwiftNet* s piramidom značajki. Blokovi EB (Encoder Block) predstavljaju blokove kodera, a blokovi koji su obojani istom nijansom žute dijele parametre. Isto vrijedi i za 1×1 konvolucije obojane različitim nijansama narančaste. Različite veličine blokova predstavljaju različite rezolucije značajki na izlazu iz bloka. Plavi blokovi predstavljaju blokove dekodera koji su zaduženi za naduzorkovanje. Slika nastala po uzoru na [48].

Značajke kodera koje su na istoj rezoluciji, nakon svođenja na jednak broj kanala korištenjem 1×1 konvolucija, međusobno zbrajamo i dovodimo na ulaz odgovarajućeg

bloka dekodera, kao što prikazuje Slika 3.2.2. Blokovi kodera u piramidi dijele parametre, što je prikazano istim nijansama narančaste na Slici 3.2.2. Blokovi dekodera jednaki su kao kod jednorezolucijske inačice, a sastoje se od bilinearnog naduzorkovanja značajki prošlog sloja dekodera, zbrajanja tih značajki sa značajkama kodera i jednog *BNReLUConv* bloka, pri čemu je veličina konvolucijske jezgre $k = 3$.

Semantička segmentacija svjesna granica

Autori članka [48] zabilježili su loše performanse *SwiftNeta* na sitnim objektima slike, kao što je npr. prometni znak. Razlog bi mogao biti to što se model prenauči (engl. *overfit*) na značajke niske rezolucije te dobro raspoznaje samo veće objekte slike. Kako bismo doskočili tom problemu koristimo gubitak svjestan granica (engl. *boundary-aware loss*) [94], što je modifikacija fokalnog gubitka (engl. *focal loss*) [41]. Fokalni gubitak nastoji naglasiti gubitak nad teškim primjerima, a definiramo ga kao:

$$\mathcal{L}_f = -(1 - p_t)^{\lambda} \log (p_t) . \tag{3.1}$$

Postavljanjem vrijednosti $\lambda = 1$ fokalni gubitak postaje gubitak negativne log-izglednosti. Član p_t u izrazu 3.1 predstavlja vjerojatnost koju je model dodijelio točnom razredu. Gubitak svjestan granica definiramo pomoću poznavanja udaljenosti svakog pojedinog piksela slike od najbliže semantičke granice, a tu informaciju dobivamo iz stvarnih oznaka (engl. *ground truth*). Neka je d^{ij} udaljenost piksela na lokaciji (i, j) od najbliže semantičke granice. Tada definiramo faktor modulacije α^{ij} prema izrazu:

$$\alpha_{ij} = \begin{cases} 8, & \text{ako } d^{ij} \in [0, 15] \\ 4, & \text{ako } d^{ij} \in [16, 63] \\ 2, & \text{ako } d^{ij} \in [64, 127] \\ 1, & \text{ako } d^{ij} > 127. \end{cases} \tag{3.2}$$

Tada možemo definirati gubitak svjestan granica za piksel (i, j) kao:

$$\mathcal{L}_{BA}^{ij} = \alpha^{ij} \mathcal{L}_f^{ij}. \tag{3.3}$$

Autori članka [94] eksperimentalno su utvrdili da bolje performanse postižu tako da modulacijski faktor $(1 - p_t)^{\lambda}$ zamijene s $e^{\lambda(1-p_t)}$, pa tako dobivamo novu definiciju gubitka za piksel na lokaciji (i, j):

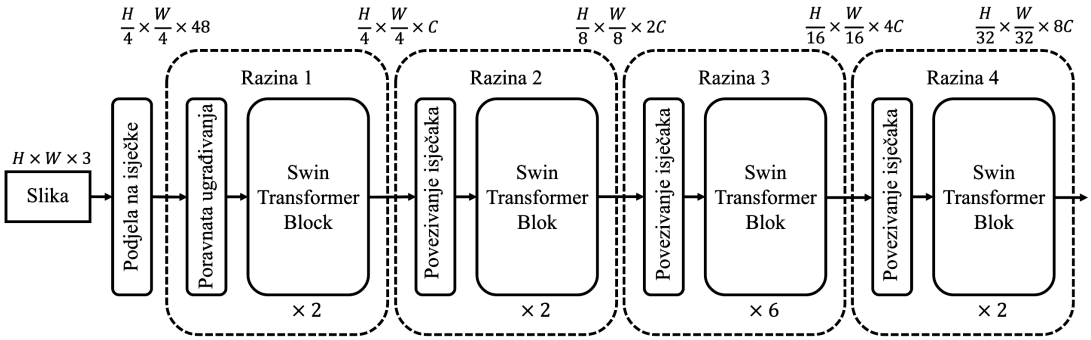
$$\mathcal{L}_{BA}^{ij} = -\alpha^{ij} e^{\lambda\left(1-p_t^{ij}\right)} \log p_t^{ij}. \tag{3.4}$$

3.3. Korištene okosnice

U ovom odjeljku dat ćemo kratak uvid u arhitekture koje smo koristili kao prednaučene okosnice (engl. *backbone*) u više eksperimenata u okviru ovoga rada. U odjeljku 3.3.1 predstavljamo arhitekturu *Swin* [42], a u odjeljcima 3.3.2 i 3.3.2 opisat ćemo konvolucijske arhitekture *Convnext* [43] te *Convnext v2* [81].

3.3.1. *Swin*

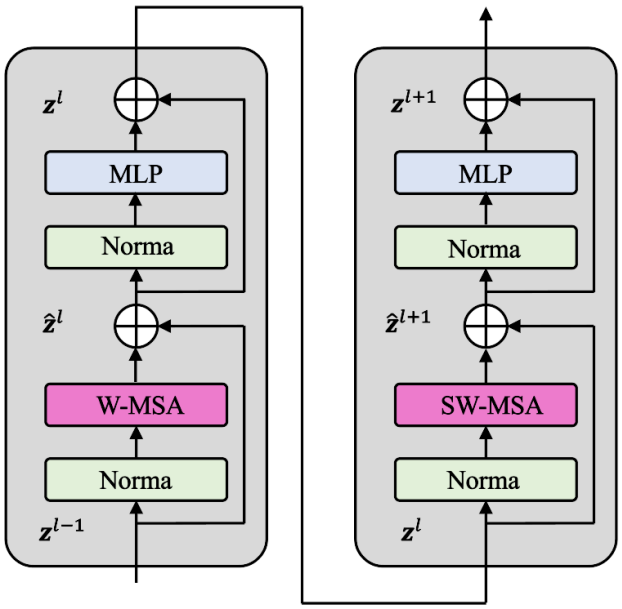
Arhitektura *Swin* [42] temelji se na arhitekturi *Transformer* [79], a može poslužiti kao općenita okosnica za više zadataka u području računalnog vida. Metoda se temelji na lokalnom računanju mehanizma pažnje te korištenju pomičnih prozora (engl. *shifted windows*). Korištenje pomičnih prozora omogućava modeliranje veza između lokalnih isječaka.



Slika 3.4: Model *Swin* sastoji se od 4 razine, a svaka razina sastoji se od više *Swin Transformer* blokova te sloja povezivanja isječaka. Nakon svake razine rezolucija je dvaput poduzorkovana s obzirom na ulaznu rezoluciju, a broj kanala dvostruko je veći s obzirom na ulazni broj kanala. Tako dobijemo hijerarhiju značajki, što je karakteristično za standardne konvolucijske okosnice kao što je npr. *ResNet* [22]. Zadaća sloja povezivanja isječaka poduzorkovati je rezoluciju te povećati broj kanala, dok blokovi *Swin Transformer* zadržavaju rezoluciju i broj kanala ulaza. Slika prikazuje *tiny* verziju modela, gdje sve razine osim treće imaju 2 bloka *Swin Transformer*, dok treća razina ima 6 takvih blokova. Slika nastala po uzoru na [42].

Pregled arhitekture *tiny* verzije modela *Swin* prikazuje Slika 3.4. Neka na ulazu imamo sliku dimenzija $H \times W \times 3$. Prvi korak podjela je slike u nepreklapajuće isječke, a dimenzija isječaka koja se koristi kod modela *Swin* jest 4×4. Svaki isječak predstavlja jedan ulaz *Transformera*, pri čemu je broj kanala jednak $4 \times 4 \times 3 = 48$. Prije ulaza u prvi blok *Swin Transformera* koristimo linearni sloj ugrađivanja (engl. *linear embedding layer*) kako bismo napravili projekciju ulaza u proizvoljan broj kanala C

na izlazu. Svaka od 4 razine *Swina* (Slika 3.4) sastoji se od više *Swin Transformer* blokova te sloja povezivanja isječaka (engl. *patch merging layer*). Sloj povezivanja isječaka zadužen je za stvaranje hijerhije značajki, odnosno svaki sloj povezivanja isječaka dvostruko smanjuje rezoluciju ulaza u taj blok, a dvostruko povećava broj kanala na izlazu iz bloka. Dvostruko smanjenje rezolucije provodi se na način da se ulazne značajke dimenzija $\frac{H}{4} \times \frac{W}{4} \times C$ podijele na manje isječke 2×2, a dobivene isječke nadovežemo po dimenziji kanala te dobijemo značajke dimenzija $\frac{H}{8} \times \frac{W}{8} \times 4C$, što potom projiciramo na broj kanala $2C$ primjenom linearnog sloja. Blok *Swin Transformer* na Slici 3.4 sličan je kao kod ViTa (odjeljak 2.2.3), osim što sada koristimo više glava mehanizma samopažnje nad lokalnim prozorima (engl. *window based multi-head self-attention*, W-MSA) ili nad pomičnim lokalnim prozorima (engl. *shifted window based multi-head self-attention*, SW-MSA). Dva slijedna bloka *Swin Transformer* prikazuje Slika 3.5.

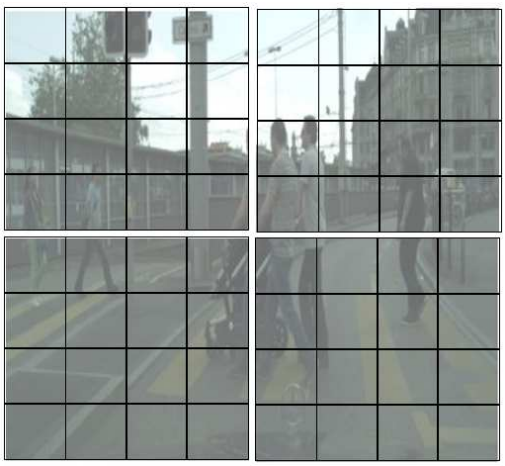


Slika 3.5: Ilustracija dva slijedna bloka *Swin Transformera*. Koristi se standardni *Transformer* [79] blok kodera kao kod arhitekture ViT [15], s tim da kod *Swina* standardni mehanizam pažnje mijenjamo mehanizmom pažnje unutar lokalnih – regularnih ili posmaknutih – prozora. Koristimo normalizaciju sloja [2], koju postavljamo prije mehanizma pažnje, kao kod arhitekture ViT. Slika nastala po uzoru na [42].

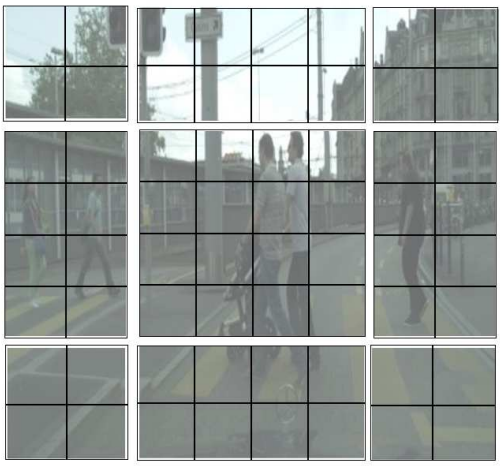
Mehanizam samopažnje nad prozorima

Složenost globalnog računanja mehanizma pažnje [79, 15] kvadratno skalira s duljinom ulaza. Autori članka [42] predlažu računanje mehanizma pažnje lokalno, unutar

manjih prozora ulazne slike ili kasnijih mapa značajki. Pretpostavimo da se svaki prozor sastoji od $M \times M$ isječaka (engl. *patches*). Tada je složenost računanja linearna po duljini ulaznog slijeda, a kvadratna po veličini prozora M. S obzirom da je veličina prozora konstantna (podrazumijevana vrijednost u radu [42] iznosi 7), možemo zaključiti kako mehanizam pažnje unutar lokalnih prozora linearno skalira s duljinom ulaza, što značajno povećava efikasnost u radu sa slikama većih rezolucija. Računanje mehanizma pažnje samo unutar prozora unaprijed određenih dimenzija ima jasan nedostatak u tomu što više ne postoje veze između isječaka u različitim dijelovima slike. Kako bi doskočili tom problemu, autori članka [42] predlažu korištenje pomičnih prozora pri računanju mehanizma pažnje.



(a) Podjela ulaza po prozorima za prvi od dva slijedna bloka *Swin Transformer* kao na Slici 3.5.



(b) Podjela ulaza po prozorima za drugi od dva slijedna bloka *Swin Transformer* kao na Slici 3.5.

Slika 3.6: Ilustracija pristupa temeljenog na pomičnim prozorima za računanje mehanizma samopažnje. U prvom od dva slijedno povezana bloka *Swin Transformera* računamo mehanizam pažnje unutar pravilnih prozora (W-MSA), dok drugi blok računa mehanizam pažnje nad posmaknutim prozorima (SW-MSA) kao što je prikazano na Slici 3.6b. Uvođenjem pomičnih prozora modeliramo vezu između između lokalnih prozora. Slika inspirirana radom [42].

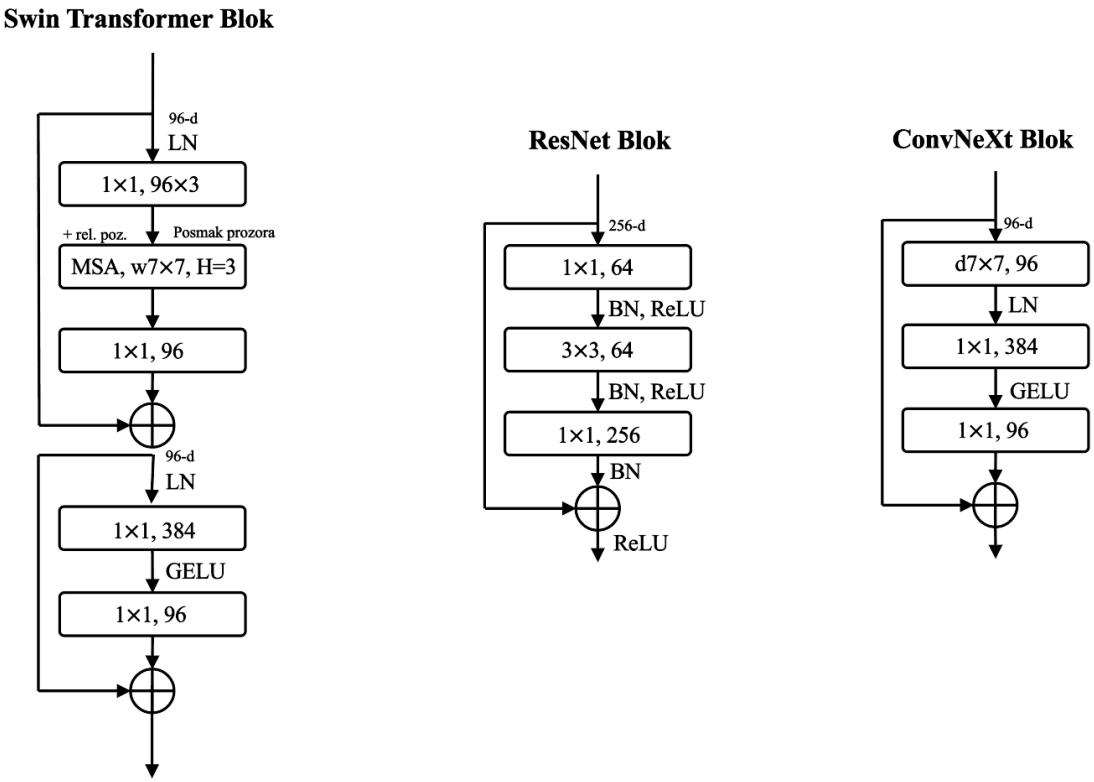
Kao što prikazuje Slika 3.6, prvi od dva slijedna bloka *Swin Transformera* koristi pravilnu podjelu na prozore, počevši od gornjeg lijevog slikovnog elementa, pri čemu mapu značajki dimenzija 8×8 dijelimo na 4 prozora dimenzija 4×4 ($M = 4$). Naredni blok potom koristi konfiguraciju prozora koja je nastala posmicanjem konfiguracije prethodnog sloja tako da se prozori posmaknu za $\left(\left\lfloor \frac{M}{2} \right\rfloor, \left\lfloor \frac{M}{2} \right\rfloor\right)$ elemenata s obzirom na pravilno postavljene prozore.

3.3.2. *Convnext*

Model *Convnext* [43] modernizacija je standardnog modela *ResNet* [22], a cilj autora stvaranje je modela koji ostvaruje jednake ili bolje performanse od konkurentnih modela koji se temelje na arhitekturi *Transformer* [79], prije svega od modela *Swin* [42]. Autori članka [43] kreću od modela *ResNet-50*. Prvi korak bilo je učenje modela *ResNet-50* koristeći moderne tehnike učenja karakteristične za učenje modela koji se temelje na arhitekturi *Transformer*. Autori članka [43] koriste tehnike učenja bliske modelima *DeiT* [77] i *Swin* [42]. Postupak učenja produžen je s 90 epoha (originalni *ResNet*) na 300 epoha, a kao optimizator koristi se AdamW [46]. Dodane su različite perturbacije (engl. *augmentation*) ulazne slike: *CutMix* [90], slučajno brisanje (engl. *random erasing*) [95], *mixup* [93] te *RandAugment* [12]. Dodatno, umjesto jednojediničnih oznaka koriste se zaglađene oznake (engl. *label smoothing*) [74], što daje regularizacijski učinak.

Modifikacije arhitekture

Prva promjena s obzirom na arhitekturu *ResNet-50* promjena je količine računanja u pojedinoj razini (engl. *stage*) hijerarhije, po uzoru na model *Swin* [42]. Broj je blokova u svakoj od 4 razine modela *ResNet-50* jednak $(3, 4, 6, 3)$, a kod *Convnexta* raspored blokova mijenja se u $(3, 3, 9, 3)$. Naredna promjena zamjena je početne 7×7 konvolucije praćene sažimanjem maksimumom koja rezultira $4\times$ poduzorkovanim značajkama. Taj početni blok (*stem* blok) mijenja se jednom 4×4 konvolucijom s korakom konvolucije 4 (*patchify* blok). Zamjena starog bloka novim, jednostavnijim slojem, nije uzrokovala degradaciju performansi modela. Iduća modifikacija korištenje je grupnih konvolucija u svim blokovima po uzoru na *ResNeXt* [85], ali je broj grupa jednak broju izlaznih kanala (engl. *depthwise convolution*). Nakon provođenja takve grupne konvolucije, provodi se 1×1 konvolucija zadužena za miješanje kanala. Naredna promjena uvođenje je inverznog uskog grla (engl. *inverted bottleneck*) standardnog *ResNet* bloka te premještanje grupne konovlucije na početak bloka. Dodatno, za razliku od *ResNet* blokova gdje koristimo 3×3 konvoluciju, kod *Convnexta* koristimo grupnu konvoluciju s veličinom jezgre 7. Što se tiče promjena na najnižoj razini, kod *Convnexta* više ne koristimo zglobnicu nego njezinu zaglađenu varijantu GELU (Gaussian Error Linear Unit) [25]. Osim toga, umjesto normalizacije nad grupom koristimo normalizaciju sloja [2], a dodatno smanjujemo i broj primjena aktivacijske funkcije u svakom bloku. Ilustraciju *Swin*, *ResNet* i *Convnext* bloka prikazuje Slika 3.3.2.



Slika 3.7: Ilustracija *Swin Transformer* bloka, *ResNet* bloka te *Convnext* bloka. Kod *Convnext* bloka na početku se nalazi grupna 7×7 konvolucija, a potom slijedi 1×1 konvolucija s većim brojem izlaznih kanala (inverted bottleneck). Potom slijedi GELU [25], 1×1 konvolucija te zbrajanje s ulaznim aktivacijama korištenjem rezidualne veze. Slika nastala po uzoru na [43].

3.3.3. *Convnext v2*

Model *Convnext v2* [81] nadogradnja je modela *Convnext* [43]. Zadatak na kojemu je model *Convnext v2* prednaučen rekonstrukcija je maskiranih dijelova slike, po uzoru na MAE [24]. Uz korištenje novog zadatka predučenja, promjena *Convnext v2* modela dodavanje je sloja globalne normalizacije aktivacija (engl. *Global Response Normalization*, GRN). Sloj GRN sastoji se od tri faze: globalna agregacija značajki, normalizacija značajki te kalibracija značajki. U okviru ovoga rada model *Convnext v2* koristimo kao okosnicu u različitim eksperimentima za nadziranu semantičku segmentaciju (odjeljak 7.1) te prilagodbu domene (odjeljak 7.3).

3.4. *Segment Anything Model* - SAM

Model SAM (*Segment Anything Model*) [36] naučen je na skupu podataka SA-1B [36] koji se sastoji od 11M slika i 1.1B segmentacijskih maski. Skup podataka prikupljen je u okviru rada [36]. Prikupljanje podataka sastojalo se od tri faze, a posljednja faza potpuno je automatska u kojoj model SAM generira maske bez dodatne ljudske intervencije. Model SAM sposoban je odrediti segmentacijsku mapu za objekte na slici, ali ne može odrediti semantiku tih segmentacijskih mapa. Model se sastoji od 3 dijela, a to su: koder slike (engl. *image encoder*), koder predloška (engl. *prompt encoder*) i dekoder maski (engl. *mask decoder*). Ideja je predati ulaznu sliku u koder slike te dobiti slikovno ugrađivanje na izlazu. Koder predložaka zadužen je za kodiranje različitih predložaka koji daju informaciju o objektima koje želimo segmentirati na slici. Predlošci mogu biti ključne točke objekta ili pravokutna regija (engl. *bounding box*) unutar koje želimo segmentirati objekt. Dobiveno slikovno ugrađivanje zajedno s ugrađivanjima predložaka predajemo dekoderu maske te na izlazu dobivamo segmentacijske mape objekta koji sadrži predane ključne točke ili se nalazi unutar zadane pravokutne regije. Za koder slike SAM koristi ViT [15] prednaučen na zadatku rekonstrukcije maskiranih regija slike [24]. Model SAM u ovom radu iskoristili smo tako da mu damo niz ključnih točaka (rešetku točaka), a SAM nam na izlazu daje sve maske pokrivene rešetkom točaka. Ilustraciju tog postupka prikazuje Slika 3.8.

Slika 3.8: Primjer generiranja maski koristeći model SAM. Na ulaznoj slici (lijevo) zadamo rešetku ključnih točaka, a SAM na izlazu daje sve pronađene maske (desno) pokrivene nekom od zadanih točaka. Prikazana ulazna slika preuzeta je iz skupa podataka *Cityscapes* [11].

4. Primjena latentnih jezičnih reprezentacija za semantičku segmentaciju

U ovom poglavlju opisujemo osnovne metode koje ćemo koristiti za rješavanje zadatka semantičke segmentacije primjenom latentnih jezičnih reprezentacija (ugrađivanja) (engl. *language embeddings*). Odjeljak 4.1 daje kratak uvid u prethodni rad iz literature povezan s korištenjem jezičnih ugrađivanja, a ostali odjeljci redom predstavljaju CLIP [60], OpenCLIP [8] te CLIPSeg [47].

4.1. Prethodni rad

Korištenje jezičnih ugrađivanja u računalnom vidu relativno je novo područje, a zaživjelo je pojavom modela CLIP [61]. To je model naučen kontrastno na parovima slika i odgovarajućih opisa, a bez ugađanja (engl. *zero-shot*) na *ImageNetu* postigao je točnost jednaku točnosti modela *ResNet-50* [22] koji je naučen na *ImageNetu*. Uspjeh modela CLIP koji je naučen na 400M parova slika i pripadajućih opisa potaknuo je daljnje istraživanje [33, 56, 92, 89], ali zajedničko svim metodama korištenje je iznimno velikih skupova podataka koji su privatni. S obzirom da je skup podataka na kojemu je učen originalni model CLIP privatan, autori članka [8] naučili su isti model korištenjem kontrastnog učenja na parovima slika i teksta, ali pritom koristeći javne skupove podataka *LAION-400M* [71] i *LAION-5B* [72]. U okviru ovoga rada koristimo upravo OpenCLIP modele naučene na javno dostupnim skupovima podataka, a zadatak koji rješavamo semantička je segmentacija. U članku [86] autori mijenjaju jednojedinične oznake jezičnim ugrađivanjima opisa razreda, a ova ideja najsličnija je provedenim eksperimentima u okviru ovoga rada. U radu [40] autori također koriste jezična ugrađivanja kao oznake, ali uz to koriste i jednostavan dekoder, prije kojega provode skalarni umnožak jezičnih ugrađivanja i slikovnih značajki. Model CLIP [61]

često se koristi za rješavanje zadatka semantičke segmentacije na neviđenim razredima (engl. *zero-shot semantic segmentation*), a poznate su metode MaskCLIP [96] i njegovo poboljšanje pod imenom ZegCLIP [97].

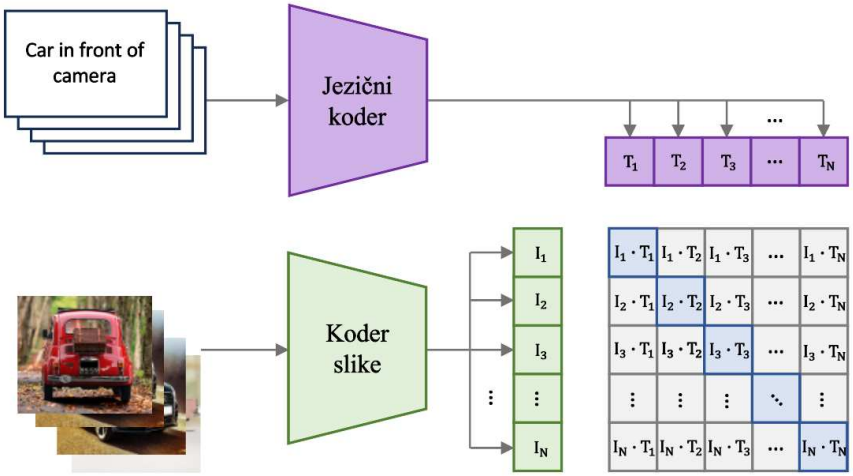
4.2. CLIP

U računalnom vidu često učimo model (npr. za klasifikaciju slika) s unaprijed određenim brojem mogućih razreda. Takav oblik nadziranoga učenja nepogodan je za klasifikaciju novih vizualnih koncepata, odnosno nepogodan je za promjenu taksonomije razreda na kojima učimo. Učenje iz jezičnih opisa slika način je kako možemo doskočiti tom problemu, a upravo to je glavna ideja modela CLIP [61]. Želimo biti sposobni učiti uz nadzor prirodnog jezika (engl. *natural language supervision*). Ideja autora članka [61] naučiti je model na parovima slika i jezičnih koncepata koji opisuju što se na slici nalazi. Kako bi to uspjeli, autori [61] su s različitih izvora Interneta skupili 400M parova slika i pripadajućih jezičnih opisa. Nakon što su prikupili skup podataka, autori članka [61] kontrastno su prednaučili jezični koder koji se temelji na arhitekturi *Transformer* [79] i koder slika koji se temelji na arhitekturi ViT [15].

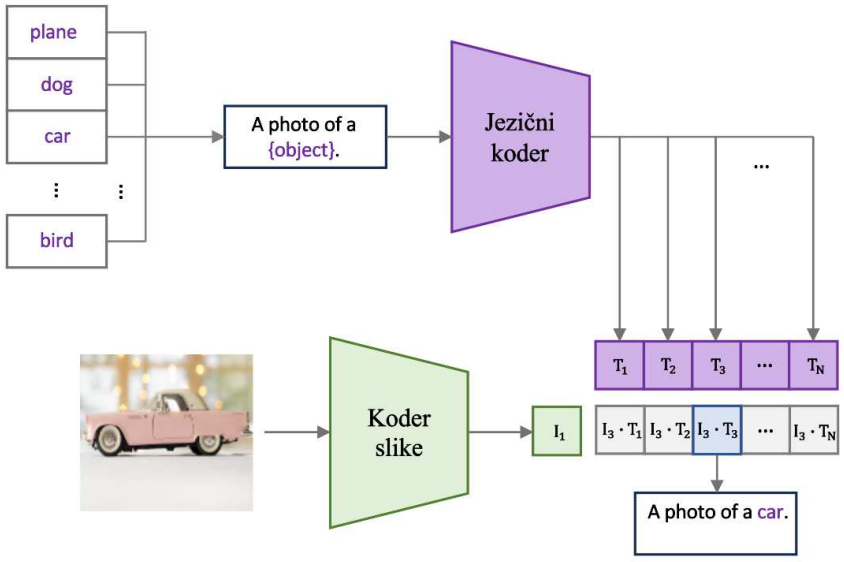
U nastavku ćemo objasniti kontrastno predučenje na parovima slika i jezičnih opisa. Neka raspolažemo mini-grupom od N parova (slika, opis) koje predajemo slikovnom odnosno jezičnom koderu. Svakoj slici odgovara točno jedan tekstualni opis. Radimo li s minigrupom veličine N, postoji ukupno $N \times N$ mogućih uparivanja slike i jezičnog opisa unutar minigrupe. Od tih uparivanja samo N parova je ispravno, dok je $N^2 - N$ parova neispravno. Model CLIP učimo tako da maksimiziramo kosinusnu sličnost između N valjanih uparivanja, a minimiziramo kosinusnu sličnost između $N^2 - N$ pogrešnih uparivanja. U praksi ovo postižemo optimizacijom gubitka unakrsne entropije nad dobivenim sličnostima. Slika 4.1 prikazuje pseudokod postupka kontrastnog predučenja.

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Slika 4.1: Pseudokod kontrastnog predučenja na parovima slika i pripadnih jezičnih opisa. Nakon što dobijemo slikovne i jezične reprezentacije, dobivene reprezentacije projiciramo u prostor ugrađivanja te normiramo. Potom izračunamo sličnost tako da izračunamo skalarni umnožak između normiranih jezičnih i slikovnih ugrađivanja. Nakon toga računamo gubitak unakrsne entropije nad dobivenom matricom sličnosti. Slika je preuzeta od [61].



(a) Faza predučenja. Mini-grupu od N slika predamo koderu slike, a N pripadajućih jezičnih opisa predamo jezičnom koderu. Dobijemo ugrađivanja slika $\mathbf{I}_i$ te jezična ugrađivannja $\mathbf{T}_i$. Nastojimo maksimizirati skalarni umnožak normiranih ugrađivanja $\mathbf{I}_i\mathbf{T}_i$, a minimizirati skalarni umnožak normiranih ugrađivanja $\mathbf{I}_i\mathbf{T}_j$, pri čemu je $i \neq j$.



(b) Faza zaključivanja na zadatku klasifikacije slika. Sliku koju želimo klasificirati predamo koderu slike, dok sve potencijalne nazive razreda predamo jezičnom koderu te dobijemo odgovarajuća jezična ugrađivanja. Izračunamo kosinusnu sličnost između slikovnog i svih jezičnih ugrađivanja razreda. Kao konačni razred odaberemo onaj za koji je kosinusna sličnost jezičnog ugrađivanja sa slikovnim ugrađivanjem najveća.

Slika 4.2: Ilustracija faze predučenja (4.2a) te faze zaključivanja (4.2b) modela CLIP. Slika nastala po uzoru na [61].

4.3. OpenCLIP

S ozbirom da je skup podataka na kojemu je naučen model CLIP [61] privatan, autori članka [8] iskoristili su javno objavljene skupove podataka *LAION-400M* [71] te *LAION-5B* [72] kako bi reproducirali rezultate originalnog modela CLIP. Programski kod za učenje i vrednovanje, kao i svi prednaučeni modeli, javno su objavljeni na repozitoriju.[1] U originalnom članku [61] autori kao koder slike koriste *ResNet* [22] i ViT [15], dok autori OpenCLIPa uče i modele s *Convnext* [43] okosnicom. Upravo OpenCLIP *Convnext* prednaučeni model intenzivno koristimo u okviru ovoga rada. Pogodnost korištenja konvolucijskog modela kao okosnice leži u tomu što možemo dobiti standardnu hijerarhiju značajki i ugraditi tu okosnicu u veliki broj arhitektura za semantičku segmentaciju (npr. *SwiftNet* [48]).

4.4. CLIPSeg

Model CLIPSeg [47] rješava zadatak segmentacije sa zadanim izrazima (engl. *referring expression segmentation*), zadatak segmentacije na neviđenim razredima (engl. *zero-shot segmentation*) te zadatak segmentacije s jednim pokaznim primjerom (engl. *one-shot segmentation*). Model CLIPSeg koristi CLIP [61] kao okosnicu te dekoder koji omogućava dobivanje gustih predikcija na izlazu. Autori su model učili na skupu podataka *PhraseCut* [82] koji se sastoji od 340000 parova slika i jezičnih fraza koje opisuju neki vizualni koncept koji se pojavljuje na ulaznoj slici. Za vrijeme učenja ažuriramo samo parametre dekodera, a CLIP okosnicu zamrznemo. Ulaznu sliku predamo CLIP slikovnom koderu, a pripadajući izraz predamo CLIP jezičnom koderu. Na izlazu dobivamo binarnu mapu koja pokazuje na kojim mjestima ulazne slike jezična fraza odgovara vizualnom konceptu sa slike. Jedna mogućnost predavanje je jezične fraze jezičnom koderu te dobivanje ugrađivanja koje potom tražimo na ulaznoj slici, dok je druga mogućnost predavanje pokaznog primjera vizualnog koncepta kojega želimo segmentirati te dobivanje slikovnog ugrađivanja tog pokaznog vizualnog koncepta kojega potom segmentiramo na ulaznoj slici.

[1] `https://github.com/mlfoundations/open_clip`

5. Nenadzirana prilagodba domene

Učenje modela na zadatku semantičke segmentacije zahtijeva veliki skup označenih slika na razini piksela. Proces označavanja slika iz stvarnog svijeta, osobito iz scena vožnje, dugotrajan je. Jedan način kako bismo doskočili tom problemu učenje je modela na umjetnim podacima [63, 65]. Međutim, zbog pomaka u domeni, takav model općenito ostvaruje slabe performanse na stvarnim podacima [28]. Upravo ovim problemom bavi se zadatak iz područja računalnoga vida pod imenom *Nenadzirano prilagođavanje domene* (engl. *Unsupervised domain adaptation*, UDA), gdje je cilj prilagoditi model naučen na izvornoj domeni (npr. umjetne scene vožnje) tako da ostvaruje dobre rezultate na ciljnoj domeni (npr. stvarne scene vožnje), bez da je za vrijeme učenja imao pristup stvarnim oznakama (engl. *ground truth*) slika ciljne domene.

5.1. Formalizam samonadziranog pristupa prilagodbi domene

U okviru ovoga rada bavimo se reprodukcijom metoda iz samonadziranog pristupa prilagodbe domene. U ovom odjeljku ukratko opisujemo formalizam (prema članku [28]) koji ćemo koristiti za prilagodbu domene u ovome radu. Pretpostavimo kako raspolažemo skupom slika izvorne domene $\mathcal{X}_S = \{x_S^{(i)}\}_{i=1}^{N_S}$ i pripadnim jednojediničnim (engl. *one-hot*) oznakama $\mathcal{Y}_S = \{y_S^{(i)}\}_{i=1}^{N_S}$. Cilj nenadzirane prilagodbe domene za semantičku segmentaciju učiti je model g_θ na slikama izvorne domene, a pritom postići dobre performanse na slikama ciljne domene $\mathcal{X}_T = \{x_T^{(i)}\}_{i=1}^{N_T}$, pri čemu model za vijeme učenja nema pristup oznakama ciljne domene $\mathcal{Y}_T = \{y_T^{(i)}\}_{i=1}^{N_T}$. Učenje modela g_θ s kategoričkim gubitkom unakrsne entropije nameće se kao jedno rješenje [28]:

$$\mathcal{L}_S^{(i)} = -\sum_{j=1}^{H \cdot W} \sum_{c=1}^{C} y_S^{(i,j,c)} \log g_\theta \left(x_S^{(i)} \right)^{(j,c)} . \tag{5.1}$$

Međutim, ovakav pristup najčešće ne daje dobre rezultate na skupu slika ciljne domene $\mathcal{X}_T$. Za rješavanje problema degradacije performansi na slikama ciljne domene, samo-

nadzirani pristup prilagođavanja domene koristi mrežu učitelja (engl. *teacher network*) h_ϕ. Mreža učitelja h_ϕ služi za generiranje pseudooznaka za slike ciljne domene:

$$p_T^{(i,j,c)} = \left[\!\left[c = \arg\max_{c'} h_\phi \left(x_T^{(i)} \right)^{(j,c')} \right]\!\right]. \tag{5.2}$$

Za svaku pseudooznaku generiramo i procjenu pouzdanosti, a u članku [28] za mjeru pouzdanosti koristi se udio piksela koji premašuje prag τ nakon *softmaxa* [78]:

$$q_T^{(i)} = \frac{\sum_{j=1}^{H \cdot W} \left[\!\left[\max_{c'} h_\phi \left(x_T^{(i)} \right)^{(j,c')} > \tau \right]\!\right]}{H \cdot W}. \tag{5.3}$$

Generirane pseudooznake, kao i procjene njihove pouzdanosti, koriste se kako bi model g_θ učili i na slikama ciljne domene, a gubitak definiramo kako slijedi [28]:

$$\mathcal{L}_T^{(i)} = - \sum_{j=1}^{H \cdot W} \sum_{c=1}^{C} q_T^{(i)} p_T^{(i,j,c)} \log g_\theta \left(x_T^{(i)} \right)^{(j,c)}. \tag{5.4}$$

Parametre modela učitelja h_ϕ ažuriramo na temelju modela g_θ za vrijeme učenje, a najčešće vrijednosti parametara učitelja h_ϕ računamo kao eksponencijalni pomični prosjek (engl. *exponential moving average*) modela g_θ nakon svake iteracije učenja t [76]:

$$\phi_{t+1} = \alpha \cdot \phi_t + (1 - \alpha)\, \theta_t. \tag{5.5}$$

Praksa je model g_θ učiti na perturbiranim (engl. *augmented*) slikama ciljne domene $\mathcal{X}_T$. S druge strane, model h_ϕ pseudooznake generira na slikama ciljne domene bez perturbacija. U svim metodama koje se koriste u ovom radu za perturbacije ulazne slike koristi se: rastresanje boje (engl. *color jittering*), Gaussovo zaglađivanje (engl. *Gaussian blur*) te *ClassMix* [54] po uzoru na članak [78].

5.2. *SegFormer*

Okosnica arhitekture za semantičku segmentaciju *SegFormer* [84] koristi se kao okosnica u svim metodama koje ćemo predstaviti u ostatku ovog poglavlja. Glavne su značajke arhitekture *SegFormer*:

- dizajniranje hijerarhijski strukturiranog *Transformera* kao kodera koji na izlazu daje značajke na više rezolucija (engl. *multiscale*),
- izbacivanje pozicijskog kodiranja iz kodera tako da se između potpuno povezanih slojeva u koderu *Transformera* doda konvolucijski sloj,

- povezivanje preklapajućih isječaka (engl. *overlapped patch merging*), što omogućuje dobivanje značajki na različitim rezolucijama te
- korištenje laganog (engl. *lightweight*) dekodera koji se sastoji samo od potpuno povezanih slojeva.

Orijentirat ćemo se samo na značajke *SegFormera* koje se tiču kodera (okosnice), s obzirom da metode u nastavku ovog poglavlja koriste samo koder arhitekture *SegFormer*. Ideja *SegFormera* podijeliti je ulaznu sliku dimenzija $H \times W \times 3$ na isječke dimenzija 4×4. Potom se dobiveni isječci koriste kao ulaz u hijerarhijski koder temeljen na *Transformeru* te se na izlazu kodera dobivaju značajke na nekoliko rezolucija, konkretno na $\left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\right\}$ originalne rezolucije. Iz dobivenih značajki, korištenjem dekodera temeljenog na potpuno povezanim slojevima (engl. *All-MLP decoder*) na izlazu dobivamo tenzor dimenzija $\frac{H}{4} \times \frac{W}{4} \times N_{cls}$, pri čemu N_{cls} predstavlja broj razreda. Autori članka [84] koder ove arhitekture nazivaju *MiX Transformer* te predstavljaju 6 različitih verzija kodera (B0-B5) koji imaju jednaku arhitekturu, a različit broj parametara. Kompletnu arhitekturu *SegFormera* prikazuje Slika 5.1. Težine kodera prednaučene na *ImageNet-1k* skupu podataka dostupne su na službenom repozitoriju.[1]

Hijerarhijske značajke

Koder *SegFormera* na izlazu daje značajke na rezolucijama jednakima $\left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\right\}$ rezolucije ulazne slike, jednako kao dobro poznata okosnica *ResNet* [22]. Koder se sastoji od 4 bloka. Ako je na ulazu slika dimenzija $H \times W \times 3$, izlaz iz svakog bloka kodera F_i je dimenzija $\frac{H}{2^{i+1}} \times \frac{H}{2^{i+1}} \times C_i$, pri čemu vrijedi $i \in \{1, 2, 3, 4\}$ te $C_{i+1} > C_i$.

Efikasno računanje mehanizma pažnje

Problem s uvođenjem hijerarhijskih značajki visoka je složenost provođenja mehanizma pažnje. Složenost mehanizma pažnje kvadratna je s obzirom na duljinu ulaznog slijeda, što u slučaju slika predstavljaju poravnati isječci (engl. *flattened patch*). U izrazu 2.20 za računanje mehanizma pažnje prepostavimo da vrijedi $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times C}$ gdje je N duljina slijeda, dok je C broj kanala. Složenost mehanizma pažnje iz 2.20 tada je $O\left(N^2\right)$. Kako bi smanjili složenost mehanizma pažnje, autori članka [84] primijenili su ideju redukcije slijeda (engl. *sequence reduction*) [80]. Ideja je da reduciramo broj ključeva $\mathbf{K}$ i vrijednosti $\mathbf{V}$ transformacijama:

$$\hat{\mathbf{K}} = reshape\left(\frac{N}{R}, C \cdot R\right)(\mathbf{K}), \hat{\mathbf{V}} = reshape\left(\frac{N}{R}, C \cdot R\right)(\mathbf{V}), \tag{5.6}$$

[1] `https://github.com/NVlabs/SegFormer`

$$\mathbf{K_{red}} = Linear\left(C \cdot R, C\right)(\hat{\mathbf{K}}), \mathbf{V_{red}} = Linear\left(C \cdot R, C\right)(\hat{\mathbf{V}}). \tag{5.7}$$

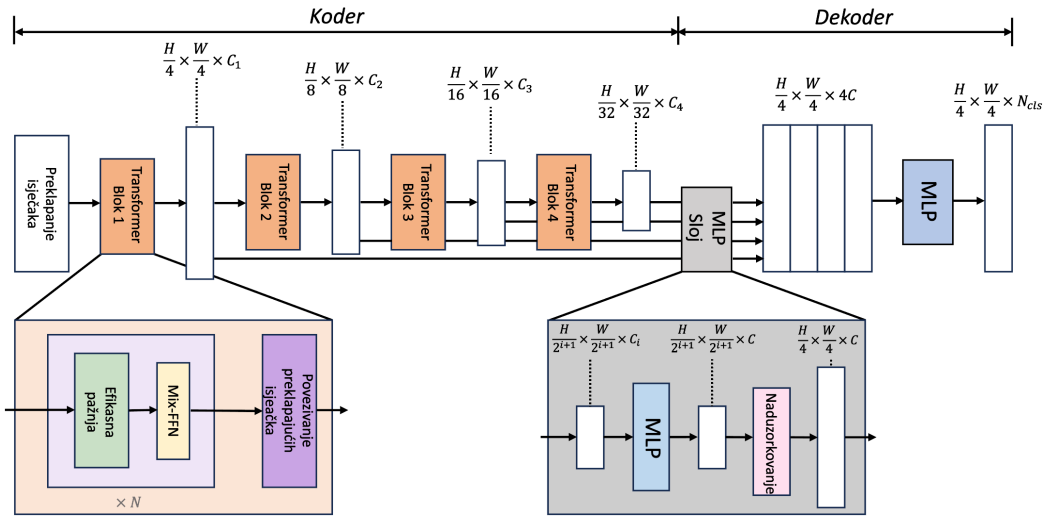
Reducirane verzije ključeva $\mathbf{K_{red}}$ i vrijednosti $\mathbf{V_{red}}$ nakon navedenih su transformacija dimenzija $\frac{N}{R} \times C$. Tako smo složenost računanja mehanizma pažnje smanjili s $O\left(N^2\right)$ na $O\left(\frac{N^2}{R}\right)$. Vrijednosti parametra R kod *SegFormera* jednake su $[64, 16, 4, 1]$ počevši od prvog do posljednjeg, četvrtog bloka. Redukciju složenosti platili smo dodavanjem novih parametara u vidu potpuno povezanog sloja $Linear$ iz izraza 5.7.

Povezivanje preklapajućih isječaka

U procesu povezivanja isječaka (engl. *patch merging*) kod *ViTa* isječak dimenzija $N \times N \times 3$ transformiramo u isječak dimenzija $1 \times 1 \times C$. Ovaj koncept možemo poopćiti tako da isječke mapa značajki dimenzija $2 \times 2 \times C_i$ transformiramo u isječke dimenzija $1 \times 1 \times C_{i+1}$ i tako dobijemo hijerarhiju značajki počevši od F_1 dimenzije $\left(\frac{H}{4} \times \frac{W}{4} \times C_1\right)$ do značajki F_2 dimenzije $\left(\frac{H}{8} \times \frac{W}{8} \times C_2\right)$. Taj proces možemo primijeniti iterativno, na bilo koju mapu značajki F_i. Ipak, ovim postupkom gubimo na lokalnom kontinuitetu isječaka. Kako bi doskočili tom problemu, autori članka [84] koriste preklapajuće isječke primjenom operatora konvolucije, pri čemu vrijedi da je veličina jezgre veća od veličine koraka (engl. *stride*) konvolucije jer tako dobivamo preklapajuće isječke.

Uklanjanje pozicijskog kodiranja

Pozicijsko kodiranje kao što koristi *ViT* zbog svoje fiksne dimenzije problematično je kada želimo zaključivati na slikama čija je rezolucija različita od slika na kojima je model naučen. Standardno rješenje tog problema interpolacija je pozicijskog kodiranja, ali tako dolazi do degradacije performansi prednaučenog modela. Autori članka [84] uklanjaju pozicijsko kodiranje, a kako bi nadomjestili informaciju o lokaciji, između potpuno povezanih slojeva u koderu *Transformera* dodaju konvolucijski sloj s veličinom grupe koja je jednaka broju izlaznih kanala (engl. *depth-wise convolution*). Takav modificirani podsloj jednog bloka *Trasnformera* autori nazivaju *Mix-FFN*.



Slika 5.1: Arhitektura modela *SegFormer*. Koder se sastoji od 4 bloka, a izlaz i-tog bloka predstavlja mape značajki F_i dimenzija $\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i$. Svaki blok kodera sastoji se od više blokova *Transformera*, ali izmijenjenih tako da se dva potpuno povezana sloja zamijene slojem *Mix-FFN*, koji osim dva potpuno povezana sloja dodaje jedan konvolucijski sloj kojemu je zadaća kodiranje lokacije. Slika nastala po uzoru na [84].

5.3. *DAFormer*

Arhitektura *DAFormer* [28] arhitektura je specijalizirana za rješavanje zadatka semantičke segmentacije u okviru nenadzirane prilagodbe domene. Ova arhitektura koristi ranije navedeni koder *SegFormera* kao okosnicu, dok za dekoder koristi posebno razvijeni dekoder svjestan konteksta koji radi sa značajkama na više rezolucija (engl. *multi-level context-aware fusion decoder*). Doprinos ove metode, osim u modeliranju dekodera, predstavljanje je tehnika za stabilizaciju učenja i sprečavanje prenaučenosti na izvornu domenu u okviru nenadzirane prilagodbe domene, a te tehnike su:

- uzorkovanje rijetkih razreda (engl. *rare class sampling*),
- udaljenost od značajki modela prednaučenog na *ImageNetu* (engl. *Thing-Class ImageNet Feature Distance*) te
- zagrijavanje stope učenja (engl. *learning rate warmup*).

U nastavku ovog odjeljka predstavit ćemo svaku od navedenih tehnika koje pospješuju učenje u okviru nenadzirane prilagodbe domene, a potom ćemo ukratko predstaviti korišteni dekoder. Oznake u nastavku jednake su kao oznake koje koristimo u odjeljku 5.1.

Uzorkovanje rijetkih razreda

Autori članka [28] primijetili su kako performanse modela na razredima koji su slabije zastupljeni u skupu podataka izvorne domene značajno variraju u više pokretanja učenja istog modela, s obzirom na različit redoslijed kojim uzorkujemo podatke u minigrupe. Što kasnije vidimo neki rijetki razred za vrijeme učenja, performanse konačnog modela na tom razredu su to gore jer je model do tog trenutka razvio pristranost prema zastupljenijim razredima skupa podataka [28]. Kako bi doskočili tom problemu, autori članka [28] predlažu uvođenje uzorkovanja rijetkih razreda (engl. *Rare Class Sampling*, RCS) za vrijeme stvaranja mini-grupa. Prvo je potrebno izračunati udio piksela f_c za svaki razred c iz skupa podataka:

$$f_c = \frac{\sum_{i=1}^{N_S} \sum_{j=1}^{H \cdot W} \left[\!\left[y_S^{(i,j,c)} \right]\!\right]}{N_S \cdot H \cdot W}. \tag{5.8}$$

Vjerojatnost uzorkovanja P_c razreda c potom definiramo kao funkciju izračunate frekvencije:

$$P_c = \frac{e^{\frac{1-f_c}{T}}}{\sum_{c'=1}^{C} e^{\frac{1-f_{c'}}{T}}}. \tag{5.9}$$

Prilikom stvaranja minigrupe prvo ćemo uzorkovati razred $c \sim P_c$, a potom ćemo iz skupa svih slika koji sadržavaju odabrani razred $\mathcal{X}_{S,c}$ slučajno uzorkovati jednu sliku $x_S \sim \mathit{Uniform}\left(\mathcal{X}_{S,c}\right)$ i dodati odabranu sliku u trenutnu minigrupu. Vrijednost parametra T iz izraza 5.9 omogućava nam modulaciju glatkoće distribucije uzorkovanja razreda, a vrijednost tog paramtera eksperimentalno je određena [28]. Ovim postupkom češće ćemo uzorkovati slike koje sadrže piksele sa semantikom rijetkih razreda iz skupa podataka. S obzirom na to da se rijetki razredi (npr. vlak ili autobus) često pojavljuju na istoj slici sa zastupljenijim razredima (npr. cesta ili građevina), opisanim postupkom neizravno ćemo učiti i semantiku zastupljenijih razreda.

Udaljenost od značajki modela prednaučenog na *ImageNetu*

Praksa u računalnom vidu korištenje je prijenosa znanja, a vrlo često se u zadacima kao što je semantička segmentacija koristi okosnica koja je prednaučena na skupu podataka *ImageNet*. Autori članka [28] pokazali su kako korištenje značajki prednaučenog modela može pomoći u zadatku nenadzirane prilagodbe domene, tako da značajke kodera približavaju značajkama modela prednaučenog na *ImageNetu*. Ideju za korištenjem ove tehnike potaklo je to što se model u prvim iteracijama učenja na određenim razredima ponašao dobro u vidu mIoU metrike, a potom bi performanse modela na istim

razredima degradirale. Zbog toga su autori članka [28] došli do zaključka da početne značajke modela prednaučenog na *ImageNetu* imaju utjecaj i na zadatak nenadzirane prilagodbe domene. Ako imamo model g_θ koji učimo te model g_{IN} koji je prednaučen na *ImageNetu*, tada udaljenost značajki F_θ kodera i značajki F_{IN} modela prednaučenog na *ImageNetu* definiramo [28]:

$$d^{(i,j)} = \left\| F_\theta \left(x_S^{(i)} \right)^{(j)} - F_{IN} \left(x_S^{(i)} \right)^{(j)} \right\|_2 . \tag{5.10}$$

S obzirom da je *ImageNet* model uglavnom prednaučen na razredima koji predstavljaju stvari (engl. *Things-class*) umjesto na razredima koji predstavljaju pozadinu (npr. nebo ili cesta), udaljenost značajki računamo samo za razrede koji predstavljaju stvari $\mathcal{C}_{things}$ koje opisujemo binarnom maskom $\mathcal{M}_{things}$. Dimenzije značajki $H_F \times W_F$ razlikuju se od dimenzija oznaka $H \times W$. Zbog toga koristimo sažimanje prosjekom za svaki kanal c nad oznakama y_S^c kako bi ujednačili dimenzije značajki i oznaka, odnosno dobili oznake $Y_{S,small}^c$:

$$y_{S,small}^c = \left[\!\left[avgpool \left(y_S^c, \frac{H}{H_F}, \frac{W}{W_F} \right) > r \right]\!\right] . \tag{5.11}$$

Pritom uzimamo samo one razrede za koje dobivene vrijednosti nakon sažimanja premašuju prag r. Sada možemo definirati:

$$\mathcal{M}_{things}^{(i,j)} = \sum_{c'=1}^{C} y_{S,small}^{i,j,c'} \cdot [\![c' \in \mathcal{C}_{things}]\!] . \tag{5.12}$$

Konačno, gubitak koji se temelji na udaljenosti od značajki modela prednaučenog na *ImageNetu* računamo prema izrazu:

$$\mathcal{L}_{FD}^{(i)} = \frac{\sum_{j=1}^{H_F \cdot W_F} d^{(i,j)} \cdot \mathcal{M}_{things}^{(i,j)}}{\sum_j \mathcal{M}_{things}^{(i,j)}}. \tag{5.13}$$

Konačni gubitak za nenadziranu prilagodbu domene, uzveši u obzir izraze 5.1 i 5.4, računamo prema izrazu:

$$\mathcal{L} = \mathcal{L}_S + \mathcal{L}_T + \lambda_{FD} \mathcal{L}_{FD}. \tag{5.14}$$

Zagrijavanje stope učenja

Ideja iza zagrijavanja stope učenja (engl. *learning rate warmup*) [17] spriječiti je velike promjene parametara modela u početnim iteracijama učenja, a ova tehnika korisna je za nenadziranu prilagodbu domene jer spriječava naglo zaboravljanje vrijednosti

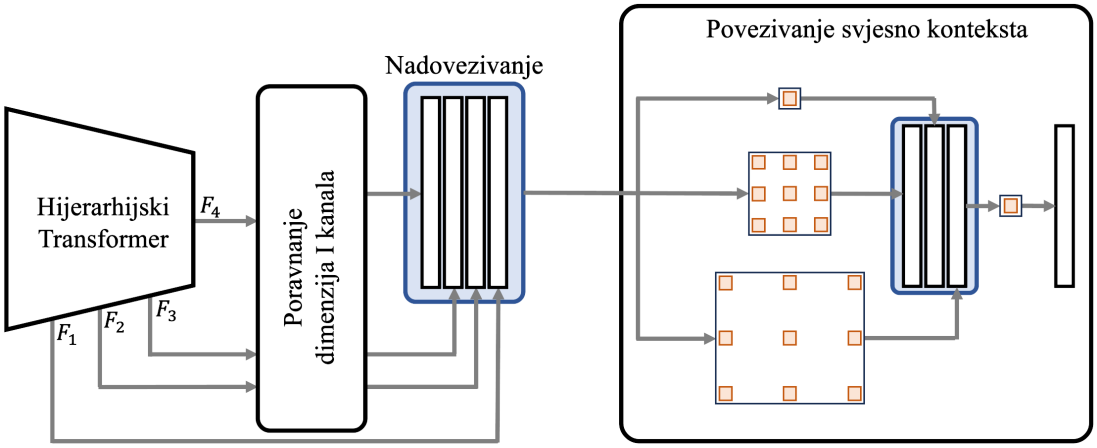
parametara prednaučenog modela okosnice. Ideja je prvih t_{warm} iteracija povećavati stopu učenja η prema izrazu:

$$\eta_t = \eta_{base} \cdot \frac{t}{t_{warm}}. \tag{5.15}$$

Pritom vrijedi da je η_{base} početna vrijednost stope učenja koju se dostiže nakon t_{warm} iteracija, a nakon t_{warm} iteracija vrijednost stope učenja prilagođavamo ovisno o odabranom raspoređivaču.

Arhitektura dekodera

Arhitekturu dekodera modela *DAFormer* [28] ilustrira Slika 5.2.



Slika 5.2: Arhitektura dekodera modela *DAFormer*. Značajke svake razine hijerarhijskog kodera korištenjem 1×1 konvolucija poravnamo s unaprijed određenim brojem kanala dekodera C_e. Poravnate značajke bilinearno naduzorkujemo do rezolucije značajki F_1, a potom sve značajke nadovežemo po dimenziji kanala. Na tako nadovezanim značajkama potom primjenjujemo više dubinski odvojivih konvolucija (engl. *depthwise separable*) s različitim faktorom dilatacije kako bismo povećali receptivno polje. Nakon primjene dubinski odvojivih konvolucija dobivamo 4 tenzora značajki s dimenzijom kanala C_e. Dobivene značajke ponovno nadovežemo po dimenziji kanala te 1×1 konvolucijom projiciramo u dimenziju C_e. Nakon toga dobivene značajke bilinearno naduzorkujemo i 1×1 konvolucijom projiciramo na broj kanala jednak broju razreda.

5.4. *HRDA*

Metoda *HRDA* [27] nadogradnja je prethodno opisane arhitekture *DAFormer*. Arhitektura *DAFormer* uči na isječcima dimenzija 512×512 koji su nastali poduzorkovanjem

isječka ulazne slike dimenzija 1024×1024. Autori članka [27] navode kako učenje na poduzorkovanim isječcima može biti problematično u segmentaciji finih detalja ulazne slike. Jedno rješenje svakako je učenje nad originalnim isječcima dimenzija 1024×1024, ali u tom slučaju okvirno trebamo 4 puta više memorijskih resursa. S druge strane, u percepciji velikih regija slike (npr. cesta, pločnik) detalji nam nisu presudni. Upravo je kombiniranje prednosti koje sa sobom nosi korištenje isječaka velike rezolucije (segmentacija finih detalja) te korištenje poduzorkovanih isječaka (razumijevanje većih regija zanemarujući detalje) osnovna ideja metode *HRDA*. U nastavku opisujemo ideju na najvišoj razini.

Prvi korak generiranje je slučajnog isječka visoke rezolucije iz ulazne slike koji potom poduzorkujemo odabranim faktorom poduzorkovanja, a ovaj isječak nazivamo isječkom konteksta (engl. *context crop*). Drugi korak slučajno je generiranje isječka iz prvog generiranog isječka velike rezolucije, a ovaj isječak bit će isječak zadužen za učenje detalja (engl. *detail crop*). U slučaju metode *HRDA* faktor poduzorkovanja iznosi 2, pa su dimenzije isječka konteksta nakon poduzorkovanja te drugog detaljnog isječka jednake. Nakon toga dovodimo isječak konteksta te detaljni isječak na ulaz *DAFormera*, a na izlazu dobivamo predikcije modela. Jasno je da ćemo ovim postupkom za detaljni isječak dobiti i segmentacijsku mapu s finim detaljima, dok ćemo za isječak konteksta dobiti segmentacijsku mapu s većim naglaskom na razumijevanje većih regija. Na izlazu je potrebno nekako spojiti dobivene predikcije, a to činimo korištenjem dodatne glave temeljene na konvolucijskim slojevima koju nazivamo skaliranom pažnjom (engl. *learned scale attention*) [6]. Dodatna glava zadužena je za određivanje u kojim regijama vjerujemo predikcijama dobivenim nad isječkom konteksta, odnosno nad detaljnim isječkom. Nakon što smo odredili regije povjerenja, koristeći dobivene težine spajamo (engl. *fuse*) predikcije. Važno je naglasiti kako i ova metoda koristi slike izvorne domene s pripadajućim stvarnim oznakama, ali i slike ciljne domene s pseudooznakama koje generiramo od strane modela učitelja h_ϕ.

5.5. *MIC*

Metoda *MIC* (*Masked Image Consistency*) [29] nadogradnja je već predstavljene metode *HRDA*. Metoda se, kao i prije, temelji na korištenju arhitekture *DAFormer*. *MIC* je zapravo modul koji se može iskoristiti i u drugim metodama nenadzirane prilagodbe domene. Temeljna ideja modula maskirati je manje odsječke (engl. *patch masking*) slika iz ciljne domene, a na izlazu pokušati rekonstruirati segmentacijsku mapu cijele ulazne slike, neovisno o tom što neke dijelove ulazne slike maskiramo. Ovim postup-

kom pokušavamo učiti kontekstualne ovisnosti (engl. *context relations*) slika iz ciljne domene, s obzirom na to da modeli za semantičku segmentaciju u okviru nenadzirane prilagodbe domene često griješe na razredima koji su slični (npr. cesta i pločnik) [29]. Jedini dodatak u odnosu na prethodno opisanu metodu *HRDA* dodavanje je još jednog gubitka $\mathcal{L}_M$ koji nazivamo maskirani gubitak (engl. *masked loss*), a koji nastoji približiti predikcije nastale nad maskiranom ulaznom slikom pseudooznakama koje daje model učitelj h_ϕ. Važno je naglasiti kako su pseudooznake modela učitelja h_ϕ nastale iz nemaskirane i neperturbirane slike.

6. Skupovi podataka

U ovom poglavlju predstavit ćemo korištene skupove podataka u okviru ovoga rada. Detaljnije ćemo opisati skupove podataka *Cityscapes* [11] (odjeljak 6.1), *GTA5* [63] (odjeljak 6.2) te ACDC [70] (odjeljak 6.3). U odjeljku 6.3 predstavit ćemo ostale skupove podataka koje smo koristili u odjeljku 7.1.

6.1. Cityscapes

Skup podataka *Cityscapes* [11] jedan je od najkorištenijih skupova podataka za semantičku segmetnaciju scena vožnje. Skup podataka sastoji se od slika vožnje i pripadnih segmentacijskih oznaka. Postoji ukupno 30 različitih razreda, a u primjenama za semantičku segmentaciju koristi se uglavnom taksonomija od 19 razreda (npr. zanemarujemo razred *parking* ili *rail track*). Popis razreda s pripadnim identifikatorom prikazuje Tablica 6.1.

Naziv razreda	ID	Naziv razreda	ID	Naziv razreda	ID
road	0	*traffic light*	6	*rider*	12
sidewalk	1	*traffic sign*	7	*car*	13
building	2	*vegetation*	8	*truck*	14
wall	3	*terrain*	9	*bus*	15
fence	4	*sky*	10	*train*	16
pole	5	*person*	11	*motorcycle*	17
bicycle	18				

Tablica 6.1: Popis 19 razreda s pripadnim identifikatorom razreda za standardnu *Cityscapes* taksonomiju.

Skup podataka *Cityscapes* podijeljen je u skup za učenje (2975 slika), skup za provjeru (500 slika) te skup za ispitivanje (1525 slika). Stvarne segmentacijske oznake dostupne su za skup za učenje te skup za provjeru. Prvi redak Slike 6.1 prikazuje primjer ulazne

slike i pripadne segmentacijske mape. Važno je napomenuti da je rezolucija slika iz skupa podataka *Cityscapes* jednaka 1024×2048.

6.2. GTA5

Skup podataka *GTA5* [63] skup je slika prikupljenih iz računalne igrice GTA5.[1] Skup podataka *GTA5* sadrži 24966 slika. Rezolucija slika u skupu podataka jest 1052×1914. Analizom skupa podataka uočili smo kako 62 para slika i pripadnih segmentacijskih mapa nemaju iste dimenzije, a te slike izbacili smo iz skupa podataka i u provedenim eksperimentima nismo koristili. Taksonomija razreda jednaka je kao kod *Cityscapesa*. Skup podataka objavljen je prije svega za rješavanje zadatka semantičke segmentacije u okviru prilagodbe domene s umjetnih na stvarne slike vožnje. Primjer slike i pripadne segmentacijske mape skupa podataka *GTA5* možemo vidjeti u drugom retku Slike 6.1.

6.3. ACDC

Skup podataka *ACDC* [70] predstavlja skup slika iz vožnje u teškim uvjetima. Sastoji se od ukupno 4006 slika, od čega je 1600 slika za učenje, 406 slika za vrednovanje, a 2000 slika nalazi se u ispitnom skupu. Svaka slika dolazi s pripadnom segmentacijskom mapom, pri čemu je taksonomija razreda kao kod *Cityscapesa* [11]. Slike su ravnomjerno raspoređene u 4 vremenska uvjeta: magla, kiša, snijeg te noćni uvjeti. Primjer slike sa snježnim uvjetima te pripadne segmentacijske mape skupa podataka *ACDC* prikazuje posljednji redak Slike 6.1.

Ostali skupovi podataka

Ostali skupovi podataka korišteni u ovom radu skupovi su podataka koji sadrže scene iz vožnje. U okviru ovoga rada te skupove podataka iskoristili smo za učenje modela u sklopu sudjelovanja na natjecanju ACDC2023 (odjeljak 7.1). Uz već navedene skupove podataka *Cityscapes* [11], *GTA5* [63] i *ACDC* [70] dodatno su korišteni skupovi podataka: *Dark Zurich* [69], *Foggy Zurich* [68], *Wilddash2* [91], *BDD-100k*[2], *Mapillary Vistas* [52], *Foggy Cityscapes* [67], *Rain Cityscapes* [30], *NightOwls*[3], *NightCity* [75], *SeeingThroughFog* [5] te skup podataka *CADCD* [57].

[1] `https://www.rockstargames.com/gta-v`
[2] `https://bdd-data.berkeley.edu/`
[3] `https://www.nightowls-dataset.org/`

Slika 6.1: Primjeri ulaznih slika i pripadnih segmentacijskih mapa. Prvi redak prikazuje sliku iz skupa podataka *Cityscapes* [11], drugi je redak primjer slike iz skupa podataka *GTA5* [63], dok je treći redak primjer slike iz skupa podataka *ACDC* [70].

7. Eksperimenti

U ovom poglavlju predstavljamo eksperimente provedene u okviru ovoga rada te dobivene rezultate. Odjeljak 7.1 predstavlja eksperimente koje smo proveli u okviru sudjelovanja na ACDC2023[1] natjecanju, odjeljak 7.2 daje uvid u provedene eksperimente i dobivene rezultate primjene jezičnih ugrađivanja za semantičku segmentaciju, dok odjeljak 7.3 prikazuje eksperimente provedene u području generalizacije i prilagodbe domene koristeći metodu MIC [29].

7.1. Semantička segmentacija - natjecanje ACDC

Natjecanje ACDC2023 natjecanje je u semantičkoj percepciji u teškim vremenskim uvjetima (noć, magla, kiša i snijeg) organizirano kao radionica u sklopu konferencije CVPR2023 pod nazivom *Vision for All Seasons*[2]. Natjecanje se temelji na skupu podataka ACDC [70] koji smo predstavili u odjeljku 6.3. Ukupno postoji 5 različitih disciplina u okviru natjecanja:

1. prilagodba domene na zadatku semantičke segmentacije Cityscapes→ACDC,
2. nadzirana semantička segmentacija,
3. nadzirana semantička segmentacija svjesna nesigurnosti (engl. *uncertainty-aware*),
4. prilagodba domene na zadatku detekcije objekata Cityscapes→ACDC te
5. nadzirana panoptička segmentacija (engl. *panoptic segmentation*) [35].

U okviru ovoga rada bavili smo se drugom disciplinom, a predikcije dobivenog modela su, uz sitne modifikacije, iskorištene i za treću disciplinu. Metrika vrednovanja za drugu disciplinu je mIoU, dok je za treću disciplinu metrika vrednovanja AvgU-IoU (Average Uncertainty-aware Intersection over Union). Važno je naglasiti da je na

[1] `https://acdc.vision.ee.ethz.ch/benchmarks`
[2] `https://vision4allseason.net/`

prošlogodišnjem izdanju natjecanja ACDC2022 najbolji rezultat u drugoj disciplini iznosio 82.82% mIoU. Kako bi model predan na vrednovanje bio valjan za ovogodišnje natjecanje ACDC2023, morao je nadmašiti prošlogodišnji najbolji rezultat.

7.1.1. ACDC2022 - pregled metode

Na prošlogodišnjem izdanju ovoga natjecanja u drugoj disciplini najbolji rezultat ostvario je doktorand s Fakulteta elektrotehnike i računarstva mag. ing. Matej Grcić[3], a njegov prošlogodišnji pristup i model polazne su točke u ovome radu. Njegova pobjednička metoda idejno je vrlo jednostavna. Prvi korak prikupljanje je velike količine slika iz vožnje, po mogućnosti u teškim vremenskim uvjetima. Za slike za koje su dostupne stvarne oznake (engl. *ground truth*), prilikom učenja iskorištavamo dostupne oznake, a za slike za koje ne postoje stvarne oznake generiramo pseudooznake iterativno, na temelju pouzdanosti izlaza modela koji je naučen na slikama za koje su oznake dostupne. Taj postupak iterativno ponavljamo, ali u drugom i svim sljedećim koracima model učimo i na slikama za koje smo generirali pseudooznake pomoću pouzdanosti modela. Arhitektura koja se koristi za semantičku segmentaciju je *SwiftNet* [48] s piramidom značajki koju smo opisali u odjeljku 3.2.2, a kao okosnica koristi se *Convnext large* [43] koju smo predstavili u odjeljku 3.3.2. Za gubitak koristimo gubitak svjestan granica koji smo opisali u odjeljku 3.2.2. Kao što smo već rekli, rezultat modela dobivenog opisanom metodom na ispitnom skupu ACDC skupa podataka iznosio je 82.82% mIoU, dok je rezultat na skupu za provjeru iznosio 85.21% mIoU. Ovaj model iskoristili smo kao polaznu točku za generiranje pseudooznaka za novu iteraciju učenja.

7.1.2. Validacija arhitekture

Prvi je korak u poboljšanju prošlogodišnjeg modela bila validacija korištene arhitekture, odnosno pokušaj pronalaska okosnice ili puta naduzorkovanja koji bi dali bolje rezultate u smislu mIoU metrike. Skupovi podataka za učenje koje smo iskoristili kako bismo odabrali najpogodniju arhitekturu su: *Cityscapes* [11], *ACDC* [70], *Wilddash2* [91] te 50 označenih slika skupa podataka *Dark Zurich* [69]. Na kombinaciji navedenih skupova podataka učili smo modele, a njihove performanse vrednovali smo na podskupu za provjeru skupa podataka *ACDC*. Različite vrste okosnica koje smo isprobali su: *Convnext* [43], *Convnext v2* [81] i *Swin Transformer* [42], dok smo za put

[3] `https://matejgrcic.github.io/files/ACDC_Challenge_Matej_Grcic.pdf`

naduzorkovanja koristili: *SwiftNet* s piramidom značajki (*SwiftNet-pyr*), jednorezolucijski (engl. *single-scale*) *SwiftNet* (*SwiftNet-ss*) te *UperNet* [83].

Okosnica	**Naduzorkovanje**	**Broj parametara**	**mIoU(%)**
convnext-tiny-384-22k_1k	SwiftNet-pyr-256	$33.3M$	78.89
	SwiftNet-ss-256	$30.9M$	**79.09**
	UperNet-256	$37.8M$	77.58
convnextv2-tiny-fcmae-384-22k_1k	SwiftNet-pyr-256	$32.5M$	78.23
	SwiftNet-ss-256	$30.2M$	76.53
	UperNet-256	$37.1M$	77.68
swin-tiny-p4-w7-224-22k	SwiftNet-pyr-256	$32.2M$	77.03
	SwiftNet-ss-256	$29.9M$	76.65
	UperNet-256	$36.7M$	76.67

Tablica 7.1: Rezultati dobiveni korištenjem *tiny* verzija okosnice. Prva tri retka prikazuju rezultate korištenja *Convnext* okosnice prednaučene na *ImageNet-22k* i ugađane na *ImageNet-1k* skupu podataka na rezoluciji koja iznosi 384, te pripadnog puta naduzorkovanja. Iduća tri retka prikazuju rezultate za *Convnext v2* okosnicu, učenu na istim skupovima i rezoluciji kao i *Convnext*. Posljednja tri retka prikazuju rezultate dobivene korištenjem *Swin Transformera* prednaučenog na *ImageNet-22k* na rezoluciji od 224. Svi eksperimenti provedeni su s veličinom grupe 10, početna vrijednost stope učenja za put naduzorkovanja iznosila je $4\mathrm{e}{-4}$, dok je za okosnicu početna vrijednost stope učenja postavljena na $1\mathrm{e}{-4}$. Dimenzija puta naduzorkovanja iznosila je 256. Svaki eksperiment pokrenut je na dvije NVIDIA RTX A4500 grafičke kartice s 20GiB dostupne radne memorije. Modele vrednujemo na ACDC skupu za provjeru.

Prvi eksperiment proveli smo s *tiny* verzijama navedenih modela koji su predstavljali okosnice, a sve korištene okosnice prednaučene su na skupu podataka *ImageNet-22k*, a *Convnext* i *Convnext v2* okosnice dodatno su ugađane (engl. *fine-tune*) na skupu podataka *ImageNet-1k*. Prednaučene težine za *Convnext* i *Swin* preuzete su s odgovarajućih repozitorija[4][5], dok su prednaučene težine za *Convnext v2* preuzete pomoću knjižnice `timm`.[6] Svaki model učen je 50 epoha, veličina je grupe iznosila 10, početna je vrijednost stope učenja za put naduzorkovanja iznosila $4\mathrm{e}{-4}$, dok je za okosnicu početna stopa učenja 4 puta manja. Kao raspoređivač stope učenja (engl. *learning rate scheduler*) koristili smo kosinusno kaljenje (engl. *cosine annealing*) [45] po uzoru na

[4] `https://github.com/facebookresearch/ConvNeXt`
[5] `https://github.com/microsoft/Swin-Transformer`
[6] `https://github.com/huggingface/pytorch-image-model`

članak [48], dok je za optimizator odabran *Adam* [34] s originanim vrijednostima parametara koje su predložene u članku. Minimalna vrijednost stope učenja postavljena je na $1\mathrm{e}{-7}$. Jednak optimizacijski postav korišten je i za sve ostale eksperimente u ovom odjeljku. Također, sve eksperimente iz ovog odjeljka učili smo koristeći 16-bitnu preciznost (engl. *Automatic Mixed Precision*, AMP). U svim daljnjim eksperimentima, ako nije drugačije navedeno, korišteno je rastresanje skale (engl. *scale jittering*) u rasponu $[0.5, 2]$, vodoravno zrcaljenje (engl. *horizontal flip*), a učenje je provedeno na isječcima (engl. *crop*) veličine 1024×1024. Rezultate u smislu mIoU metrike na *ACDC* podskupu za validaciju dobivene u prvom eksperimentu prikazuje Tablica 7.1. Možemo uočiti kako smo najbolje rezultate uz *tiny* verzije modela postigli koristeći *Convnext* kao okosnicu uz jednorezolucijski *SwiftNet* za put naduzorkovanja.

Drugi eksperiment koji smo proveli učenje je *base* verzija istih okosnica kao u prethodnom eksperimentu. Dodatno, osim okosnica naučenih na *ImageNet* skupu podataka, proveli smo eksperimente s *Convnext* okosnicom prednaučenom na podskupu *LAION-5B* [72]. Dodatno, isprobali smo i *Convnext* okosnicu prednaučenu na podskupu *LAION-5B* skupa podataka te na *ImageNet-12k*[7] skupu, a ugađanu na skupu podataka *ImageNet-1k*. Prednaučene vrijednosti parametara za *Convnext* prednaučen na *LAION-5B* podskupu dostupne su na repozitoriju[8], a za okosnicu koja je dodatno učena na *ImageNet* skupu podataka težine su, kao i prije, dostupne preko programrske knjižnice `timm`. Optimizacijski postav jednak je kao u prethodnom eksperimentu, osim što smo za jedan eksperiment sa *Swin Transformerom* kao okosnicom smanjili stopu učenja okosnice 10 puta s obzirom na početnu vrijednost stope učenja puta naduzorkovanja. Rezultate drugog eksperimenta pokazuje Tablica 7.2. Rezultati ukazuju da i za *base* verzije modela vrijedi da *Convnext* kao okosnica u suradnji sa *SwiftNetom* daje najbolje rezultate u smislu mIoU metrike. Za razliku od prethodnog eksperimenta, u ovom eksperimentu *SwiftNet* s piramidom značajki nadmašuje jednorezolucijski *SwiftNet* za 1.55 mIoU, što nije u skladu s prethodnim eksperimentom. Zanimljivo je uočiti kako *Convnext* okosnice prednaučene na *LAION-5B* podskupu podataka ne daju bolje rezultate od onih naučenih na *ImageNet* skupu podataka, iako je *LAION-5B* značajno veći skup podataka. Zbog rezultata koje smo u ovom eksperimentu ostvarili korištenjem *SwiftNet-pyr* puta naduzorkovanja, za put naduzorkovanja konačnog modela odabrali smo upravo *SwiftNet-pyr*, a u narednim eksperimentima *UperNet* i *SwiftNet-ss* više ne razmatramo.

[7] `https://github.com/rwightman/imagenet-12k`
[8] `https://github.com/mlfoundations/open_clip`

Okosnica	**Naduzorkovanje**	**Broj parametara**	**mIoU(%)**
convnext-base-384-22k_1k	SwiftNet-pyr-256	$93.6M$	**81.88**
convnext-base-384-22k_1k	SwiftNet-ss-256	$91.1M$	80.33
convnextv2-base-fcmae-384-22k_1k	SwiftNet-pyr-256	$92.7M$	80.72
convnext-base-laiona-augreg-384-12k_1k	SwiftNet-pyr-256	$93.6M$	79.95
convnext-base-laiona-augreg-320	SwiftNet-pyr-256	$93.6M$	80.60
swin-base-p4-w12-384-22k	SwiftNet-pyr-256	$91.9M$	79.52
swin-base-p4-w12-384-22k_lr/10	SwiftNet-pyr-256	$91.9M$	79.68
swin-base-p4-w12-384-22k	UperNet-512	$120M$	79.52

Tablica 7.2: Rezultati dobiveni korištenjem *base* verzija okosnica. Prvi stupac predstavlja korištenu okosnicu. Prvi dio imena u prvom stupcu predstavlja verziju okosnice, a ostatak predstavlja skup podataka i rezoluciju slika prilikom učenja. Tako 'convnext-base-laiona-augreg-384-12k_1k' predstavlja eksperiment s *base* verzijom *Convnext* okosnice, koja je prednaučena na podskupu *LAION-5B* skupa podataka i na *ImageNet-12k* skupu, a ugađana na *ImageNet-1k* skupu podataka, s rezolucijom slika pri učenju 384. Svi eksperimenti provedeni su s veličinom grupe 12, početna vrijednost stope učenja za put naduzorkovanja iznosila je $4\mathrm{e}{-4}$, dok je za okosnicu početna vrijednost stope učenja postavljena na $1\mathrm{e}{-4}$. Dimenzija puta naduzorkovanja iznosila je 256, osim u eksperimentu gdje se koristi *UperNet*, gdje je dimenzija puta naduzorkovanja iznosila 512. Svaki eksperiment pokrenut je na 4 NVIDIA Tesla V100 grafičke kartice s 32GiB dostupne radne memorije. Modele vrednujemo na ACDC skupu za provjeru.

Posljednji eksperiment koji smo provjerili s ciljem validacije arhitekture pokretanje je ranije spomenutih okosnica, ali ovoga puta s *large* verzijama modela. Optimizacijski postav jednak je kao u prethodnim eksperimentima. U ovom eksperimentu za put naduzorkovanja koristimo samo *SwiftNet-pyr*, zbog ranije ostvarenih rezultata. Rezultate koje smo dobili u ovom eksperimentu prikazuje Tablica 7.3. Ponovno možemo uočiti kako najbolje rezultate u smislu metrike mIoU dobivamo korištenjem *SwiftNeta* s piramidom značajki u suradnji s *Convnext* okosnicom. Upravo tu arhitekturu odabrali smo kao arhitekturu konačnog modela. Zanimljivo je kako je odabrana arhitektura jednaka prošlogodišnjem pobjedničkom modelu, odnosno kako nijedna nova kombinacija okosnice i puta naduzorkovanja nije uspjela nadmašiti prošlogodišnju pobjedničku arhitekturu.

Nakon što smo odabrali arhitekturu, provjerili smo doprinos dodavanja rastresanja boje (engl. *color jittering*) te uniformnog uzorkovanja (engl. *uniform sampling*)[9] slika

[9] `https://github.com/NVIDIA/semantic-segmentation`

Okosnica	**Naduzorkovanje**	**Broj parametara**	**mIoU(%)**
convnext-large-384-22k_1k	SwiftNet-pyr-320	$206M$	**81.91**
convnextv2-large-fcmae-384-22k_1k	SwiftNet-pyr-320	$204M$	80.11
swin-large-p4-w12-384-22k	SwiftNet-pyr-320	$203M$	80.53

Tablica 7.3: Rezultati dobiveni korištenjem *large* verzija okosnica. Prvi stupac predstavlja korištenu okosnicu, a '384-22l_1k' u nazivu predstavlja korištenje okosnice koja je prednaučena na *ImageNet-22k* skupu, dodatno ugađana na *ImageNet-1k* skupu, a rezolucija prilikom učenja iznosila je 384. Svi eksperimenti provedeni su s veličinom grupe 16, početna vrijednost stope učenja za put naduzorkovanja iznosila je $4\mathrm{e}{-4}$, dok je za okosnicu početna vrijednost stope učenja postavljena na $1\mathrm{e}{-4}$. Dimenzija puta naduzorkovanja iznosila je 320. Svaki eksperiment pokrenut je na 8 NVIDIA Tesla V100 grafičkih kartica s 32GiB dostupne radne memorije. Evaluacija napravljena nad ACDC skupom za provjeru.

Okosnica	**Svjetlina**	**Kontrast**	**Zasićenje**	**mIoU(%)**
convnext-tiny-384-22k_1k	✗	✗	✗	78.89
convnext-tiny-384-22k_1k+col_jit_v1	$[0, 1]$	$[0, 1]$	$[0, 1]$	79.01
convnext-tiny-384-22k_1k+col_jit_v2	$[0.9, 1.1]$	$[0.9, 1.1]$	$[0.9, 1.1]$	78.63
convnext-tiny-384-22k_1k+uniform	✗	✗	✗	78.64

Tablica 7.4: Rezultati dobiveni korištenjem *tiny* verzije *Convnext* okosnice te *SwiftNet-pyr-256* puta naduzorkovanja. Prvi stupac predstavlja korištenu okosnicu, a '384-22l_1k' u nazivu predstavlja korištenje okosnice koja je prednaučena na *ImageNet-22k* skupu, dodatno ugađana na *ImageNet-1k* skupu, a rezolucija prilikom učenja iznosila je 384. Svi eksperimenti provedeni su s veličinom grupe 10, početna vrijednost stope učenja za put naduzorkovanja iznosila je $4\mathrm{e}{-4}$, dok je za okosnicu početna vrijednost stope učenja postavljena na $1\mathrm{e}{-4}$. Svaki eksperiment pokrenut je na dvije NVIDIA RTX A4500 grafičke kartice s 20GiB dostupne radne memorije. Modele vrednujemo na ACDC skupu za provjeru.

prilikom učenja. U ovom eksperimentu koristimo *tiny* verziju *Convnexta* te *SwiftNet-pyr* s dimenzijom puta naduzorkovanja 256. Optimizacijski postav jednak je kao u prethodnim eksperimentima, a uz već postojeće perturbacije dodano je i rastresanje boje. Isprobali smo rastresanje boje s dvije različite vrijednosti raspona parametara, a isprobane kombinacije te dobivene rezultate možemo vidjeti u Tablici 7.4. U postupku učenja konačnog modela odlučili smo dodati rastresanje bojom, i to prvu varijantu iz Tablice 7.4, a također smo koristili i uniformno uzorkovanje slika za učenje, iako

je njegovo uključivanje, koristeći *tiny* verziju okosnice, dalo nešto slabije rezultate u smislu mIoU metrike.

7.1.3. Konačni model

Na temelju eksperimenata iz prethodnog odjeljka za arhitekturu smo odabrali *Convnext large* okosnicu uz *SwiftNet-pyr-320* put naduzorkovanja. Naredni korak predstavljalo je generiranje pseudooznaka pomoću modela koji je pobijedio na prošlogodišnjem izdanju istog natjecanja, s obzirom na to da imamo pristup prednaučenim težinama tog modela.

Okosnica	**Naduzorkovanje**	**mIoU(%)**
convnext-large-384-22k_1k	SwiftNet-pyr-320	81.91
convnext-large-384-22k_1k+DZ_Night_pseudo	SwiftNet-pyr-320	84.01

Tablica 7.5: Usporedba modela s obzirom na korištenje *Dark Zurich Night* pseudooznaka. Vrijednosti hiperparametara (stopa učenja, broj epoha, veličina grupe i dr.) jednaki su kao za Tablicu 7.3. Oba eksperimenta pokrenuta su na 8 NVIDIA Tesla V100 grafičkih kartica s 32GiB dostupne radne memorije. Modele vrednujemo na ACDC skupu za provjeru.

Pseudooznake za skupove koji nemaju stvarne oznake generiramo na način da uzmemo samo one piksele za koje model s pouzdanošću iznad 99.9% kaže da pripadaju određenom razredu. Pritom izlaze iz $softmaxa$ tumačimo kao pouzdanost modela. Sve ostale piksele označimo razredom *void*, a na tim pikselima model ne učimo. Kako bismo na neki način pokušali validirati ovakav pristup, generirali smo pseudooznake samo za *Night* podskup *Dark Zurich* skupa podataka (ukupno 2416 slika), te smo učili novi model s identičnim optimizacijskim i podatkovnim postavom (kombinacija *Cityscapes*, *ACDC*, *Wilddash2* i *Dark Zurich Val*) kao u prethodnom odjeljku, s tim da smo tim podacima dodali još i pseudooznačeni *Dark Zurich Night*. Tablica 7.5 prikazuje rezultate tog eksperimenta i usporedbu s istim modelom koji je učen bez pseudooznaka. Iz dobivenih rezultata možemo zaključiti da ima smisla dodati pseudooznake i da njihovim dodavanjem možemo očekivati poboljšanje modela i na *ACDC* ispitnom skupu.

Generiranje pseudooznaka

Zbog navedenog zaključka da učenje s pseudooznakama pomaže, generirali smo pseudooznake za različite skupove podataka, popis kojih možemo vidjeti u prvom stupcu

Tablice 7.6. Za neke skupove podataka (*NightOwls, CADCD, BDD100k*) odabrani su samo manji podskupovi za pseudooznačavanje. Iz skupa podataka *NightOwls* [53] odabrano je 10000 slika iz *Train* podskupa te 2593 slike iz *Val* podskupa. Slike su odabrane na način da su uzimane slike koje su nastale u što razmaknutijim vremenskim trenucima (ta informacija dostupna je u skupu podataka), a n je odabran tako da na kraju imamo ranije navedeni broj slika. Tako smo pokušali osigurati da slike za pseudooznačavanje budu što različitije. Sličan postupak ponovili smo i za *CADCD*[10] skup podataka, pri čemu smo odabrali 10% dostupnih slika, odnosno 5600 od ukupno 56000 dostupnih slika. Za skup podataka *BDD100k*[11] postoje podskupovi s kišnim, snježnim i maglovitim uvjetima, a upravo ti podskupovi odabrani su za pseudooznačavanje, s obzirom da su takve slike teže za segmentirati nego slike nastale u sunčanim dnevnim uvjetima. Nekoliko primjera pseudooznačenih slika možemo vidjeti na Slici 7.1.

Učenje modela

U konačnici, pokrenuli smo 8 različitih eksperimenata učenja *Convnext large* okosnice i *SwiftNet-pyr-320* puta naduzorkovanja. Eksperimente smo pokrenuli na superračunalu *Supek*[12], gdje je dostupno 80 NVIDIA V100 grafičkih kartica. Eksperimenti su se razlikovali s obzirom na skupove označenih i pseudooznačenih slika na kojima su učeni, prema korištenju uniformnog uzorkovanja i rastresanja boje, kao i po veličini grupe i početnoj stopi učenja. Više detalja o pokrenutim eksperimentima kao i dobivene rezultate u vidu mIoU metrike na *ACDC* skupu podataka za provjeru prikazuje Tablica 7.8. Uvid u skupove podataka s postojećim originalnim oznakama kao i uvid u pseudooznačene skupove podataka koji su korišteni u okviru navedenih eksperimenata prikazuju redom Tablice 7.7 i 7.6. Svi ostali detalji vezani uz postupak učenja modela i korištene perturbacije jednaki su kao u odjeljku 7.1.2. Iz Tablice 7.8 zaključujemo kako smo najbolji rezultat na *ACDC* skupu za provjeru dobili učenjem modela s veličinom grupe 40, početnom stopom učenja puta naduzorkovanja 0.0005, a model smo učili kroz 100 epoha na svim dostupnim označenim i pseudooznačenim skupovima podataka izuzev skupa *Cityscapes Val*.

[10] `cadcd.uwaterloo.ca`
[11] `https://bdd-data.berkeley.edu/`
[12] `https://wiki.srce.hr/display/NR/Supek`

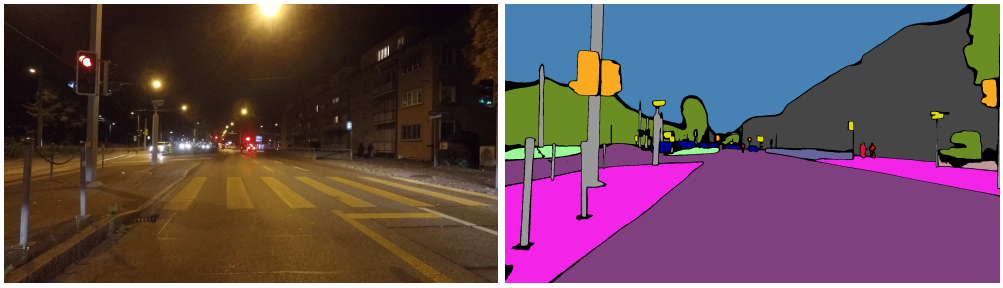
Korišteni pseudooznačeni skupovi podataka	Redni broj eksperimenta								Broj slika
	1.	2.	3.	4.	5.	6.	7.	8.	
Dark Zurich Night	✓	✓	✓	✓	✓	✓	✓	✓	2416
Dark Zurich Twilight	✓	✓	✓	✓	✓	✓	✓	✓	2920
NightOwls Train	✓	✗	✗	✓	✓	✗	✗	✗	10000
NightOwls Val	✓	✓	✗	✓	✓	✗	✗	✗	2593
NightCity Train	✓	✗	✗	✓	✓	✗	✗	✗	2997
NightCity Val	✗	✗	✗	✓	✓	✗	✗	✗	1300
BDD100k-Rain Train	✓	✓	✓	✓	✓	✓	✓	✗	5070
BDD100k-Rain Val	✓	✓	✓	✓	✓	✓	✓	✗	738
Seeing Through Fog	✓	✓	✓	✓	✓	✗	✓	✗	12997
BDD100k-Fog Train	✓	✗	✗	✓	✓	✓	✗	✗	130
BDD100k-Snow Train	✓	✗	✗	✓	✓	✓	✗	✗	5549
BDD100k-Snow Val	✓	✗	✗	✓	✓	✓	✗	✗	769
CADCD	✓	✓	✓	✓	✓	✗	✓	✗	5600
Dark Zurich Day	✓	✓	✓	✓	✓	✓	✓	✓	3041

Tablica 7.6: Za dobivanje konačnog modela pokrenuto je 8 različitih eksperimenata, pri čemu se u svakom eksperimentu koristi *Convnext large* okosnica i *SwiftNet-320-pyr* put naduzorkovanja. Eksperimenti se razlikuju po skupovima podataka na kojima su učeni. Prvi stupac označava naziv skupa podataka za kojega smo generirali pseudooznake, posljednji stupac predstavlja broj slika u svakom od tih skupova podataka, a ostali stupci predstavljaju redni broj eksperimenta i oznaku ✓ ako je skup podataka korišten u tom eksperimentu, a ✗ u suprotnom.

Seeing Through Fog

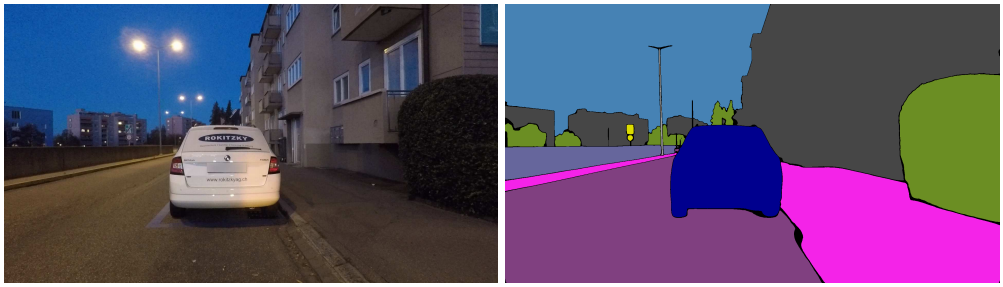
CADCD

Dark Zurich Night



Night City Train

Dark Zurich Twilight



Slika 7.1: Nekoliko primjera generiranih pseudooznaka. Prvi stupac sadrži originalne slike, dok drugi stupac predstavlja pseudooznake. Pseudooznake se generiraju tako da se u obzir uzmu samo oni pikseli za koje je izlaz iz *softmaxa* prednaučenog modela veći od 0.999.

Korišteni označeni skupovi podataka	Redni broj eksperimenta								Broj slika
	1.	2.	3.	4.	5.	6.	7.	8.	
ACDC	✓	✓	✓	✓	✓	✓	✓	✓	1600
Cityscapes Train	✓	✓	✓	✓	✓	✓	✓	✓	2975
Dark Zurich Val	✓	✓	✓	✓	✓	✓	✓	✓	50
Foggy Zurich	✓	✓	✓	✓	✓	✓	✓	✓	40
Wilddash2	✓	✓	✓	✓	✓	✓	✓	✓	4256
BDD10k Train	✓	✓	✗	✓	✓	✓	✗	✓	7000
BDD10k Val	✓	✓	✗	✓	✓	✓	✗	✓	1000
Mapillary Vistas Train	✗	✓	✗	✓	✓	✓	✗	✗	18000
Mapillary Vistas Val	✓	✓	✗	✓	✓	✓	✗	✗	2000
Mapillary Vistas Snow Train	✓	✗	✓	✓	✓	✗	✓	✗	652
Mapillary Vistas Snow Val	✓	✗	✓	✓	✓	✗	✓	✗	71
Foggy Driving	✓	✓	✗	✓	✓	✓	✗	✗	33
Foggy Cityscapes Train	✗	✗	✗	✓	✓	✓	✗	✗	8878
Foggy Cityscapes Val	✗	✗	✗	✓	✓	✓	✗	✗	1500
Rain Cityscapes	✗	✗	✗	✓	✓	✗	✗	✗	9425
Cityscapes Val	✗	✗	✗	✗	✗	✓	✗	✗	500

Tablica 7.7: Za dobivanje konačnog modela pokrenuto je 8 različitih eksperimenata, pri čemu se u svakom eksperimentu koristi *Convnext large* okosnica i *SwiftNet-320-pyr* put naduzorkovanja. Eksperimenti se razlikuju po skupovima podataka na kojima su učeni. Prvi stupac označava naziv skupa podataka za učenje (za kojega imamo originalne oznake), posljednji stupac predstavlja broj slika u svakom od tih skupova podataka, a ostali stupci predstavljaju redni broj eksperimenta i oznaku ✓ ako je skup podataka korišten u tom eksperimentu, a ✗ ako to nije slučaj.

	Redni broj eksperimenta							
	1.	2.	3.	4.	5.	6.	7.	8.
Broj označenih slika za učenje	19677	36954	9644	57480	57480	47832	9644	16921
Broj pseudooznačenih slika za učenje	54820	35375	32782	56120	56120	20633	32782	8377
Veličina grupe na pojedinačnom GPU	5	5	5	5	5	5	5	5
Broj grafičkih kartica	4	4	8	4	8	4	8	8
Uniformno uzorkovanje	✓	✓	✗	✓	✓	✓	✓	✗
Rastresanje boje	✓	✓	✓	✗	✓	✓	✓	✓
Početna vrijednost stope učenja	4e−4	4e−4	4e−4	4e−4	5e−4	4e−4	4e−4	4e−4
Broj epoha	80	80	80	60	100	80	100	80
mIoU(%)	85.48	86.27	85.83	86.00	**86.50**	85.65	86.05	82.80

Tablica 7.8: Za dobivanje konačnog modela pokrenuto je 8 različitih eksperimenata, pri čemu se u svakom eksperimentu koristi *Convnext large* okosnica i *SwiftNet-320-pyr* put naduzorkovanja. Eksperimenti pokrenuti na 4 grafičke kartice koriste veličinu grupe 16, dok eksperimenti pokrenuti na 8 grafičkih kartica koriste veličinu grupe 40, odnosno 5 minigrupa po grafičkoj kartici. Korištene su grafičke kartice NVIDIA A100 s 40GiB dostupne memorije. Stopa učenja za okosnicu 4 puta je manja od početne vrijednosti stope učenja za put naduzorkovanja (redak 7). Jednako kao u odjeljku 7.1.2 koristimo *Adam*, kosinusno kaljenje te rastresanje skale i horizontalno zrcaljenje uz uniformno uzorkovanje slika po razredima (redak 5), rastresanje boje (redak 6). Dimenzija korištenih isječaka je 1024×1024.

Korištenje ansambla

Uzimajući u obzir da smo eksperimente iz Tablice 7.8 učili na različitim kombinacijama dostupnih skupova podataka, ideja za koju smo pretpostavili da će poboljšati rezultate uvođenje je jednostavnog ansambla (engl. *ensemble*) naučenih modela. Ideja je iskoristiti nekoliko naučenih modela iz provedenih eksperimenata tako da za istu ulaznu sliku generiramo logite za svaki od modela, potom uprosječimo logite i tek onda provedemo kroz *softmax* te dobijemo konačnu predikciju. Valja napomenuti da smo za vrijeme učenja modela imali značajne poteškoće s dostupnim resursima pa je čitav postupak učenja navedenih eksperimenata pokrenut kasno, odnosno učenje svih modela nije se stiglo završiti prije roka za službeno vrednovanje modela na *ACDC* ispitnom skupu. Zbog toga su odabrani modeli koji su u trenutku generiranja predikcija imali najveću vrijednost mIoU na *ACDC* skupu za provjeru. Validirano je više kombinacija naučenih modela, a ansambl koji je ostvario najbolji rezultat sastojao se od 4 modela. Modele koji su bili dio tog ansambla kao i dobivene rezultate prikazuje Tablica 7.9.

Redni broj eksperimenta	Odabrana epoha	Pojedinačni mIoU(%)	Ansambl mIoU (ss)	Ansambl mIoU (ms)
2.	77	86.27	87.21	87.74
5.	74	86.33		
6.	76	85.71		
7.	92	86.05		

Tablica 7.9: Rezultati pojedinačnih modela kao i ansambla u vidu mIoU metrike na *ACDC* skupu za provjeru. Prvi stupac prikazuje redni broj eksperimenta iz Tablice 7.8. Drugi stupac predstavlja broj epohe učenja odgovarajućeg eksperimenta iz koje su preuzete vrijednosti parametara. S obzirom na to da se modeli nisu stigli naučiti do kraja prije roka evaluacije, odabrani su najbolji modeli do trenutka generiranja predikcija. Treći stupac predstavlja pojedinačnu vrijednost metrike mIoU svakog odabranog modela, a četvrti stupac predstavlja mIoU koju postiže ansambl, pri čemu je generiranje predikcija ostvareno jednorezolucijski (engl. *single-scale*). Posljednji stupac predstavlja mIoU vrijednost ansambla, pri čemu su konačne predikcije nastale uprosječivanjem predikcija za više skala (engl. *multi-scale*), konkretno: 1, 1.25, 1.5, 1.75 te 2.

Službeno vrednovanje

Konačni model koji je na službenom vrednovanju ostvario najbolji rezultat na *ACDC* ispitnom skupu jest ansambl koji opisuje tablica 7.9. Međutim, prvi model koji je poslan na vrednovanje bio je sličan ansambl, ali su modeli uzeti iz puno ranijih epoha učenja (konkretno, $30.$ epoha). Taj model na *ACDC* ispitnom skupu ostvario je rezultat od 83.05 mIoU. Iako je nadmašio prošlogodišnji najbolji rezultat od 82.82 mIoU, taj model ostvario je značajno slabiji rezultat (84.01 mIoU) na slikama sa snježnim uvjetima ispitnog skupa *ACDC* u usporedbi s prošlogodišnjim najboljim modelom na snježnim uvjetima (85.94 mIoU). Svaki model ansambla opisanog u Tablici 7.9 zbog toga je dodatno ugađan na kombinaciji *ACDC* skupova za učenje i provjeru (engl. *trainval*), pri čemu su sve slike sa snježnim uvjetima u svakoj epohi uključene dvaput. Optimizacijski postav ovog eksperimenta sličan je kao onaj opisan u odjeljku 7.1.2, ali je ovoga puta početna vrijednost stope učenja puta naduzorkovanja smanjena na vrijednost $1e-5$, dok je početna vrijednost stope učenja za okosnicu 4 puta manja. Svaki od modela ansambla učen je kroz 40 epoha. Svaki eksperiment pokrenut je na 4 NVIDIA A100 grafičke kartice s 40GiB dostupne radne memorije, a veličina grupe iznosila je 16 (4 minigrupe po grafičkoj kartici). U konačnici su ansambl predstavljala upravo 4 modela iz Tablice 7.9, dodatno ugađana na *ACDC trainval* skupu. Nekoliko predikcija takvog ansambla na *ACDC* ispitnom skupu prikazuje Slika 7.2.

Slika 7.2: Po jedan primjer generiranih predikcija za svaki vremenski uvjet *ACDC* ispitnog skupa. Predikcije su generirane pomoću konačnog ansambla modela iz Tablice 7.9, a svaki model dodatno je ugađan na *ACDC trainval* skupu podataka.

Konačni ansambl evaluiran je višerezolucijski (engl. *multi-scale*), a rezultat koji je taj ansambl ostvario putem službenog vrednovanja na *ACDC* ispitnom skupu iznosio je 83.82 mIoU, za bod više od prošlogodišnjeg najboljeg rezultata u drugoj disciplini. Ovim rezultatom ostvarili smo drugi rezultat na ovogodišnjem natjecanju, a najbolji rezultat na službenom poretku[13] iznosio je 84.01 mIoU, kao što možemo vidjeti na Slici 7.3.

[13] `https://acdc.vision.ee.ethz.ch/benchmarks#semanticSegmentation`

Name	IoU ↓	Supervision	Code	Date
SUTech-MT-SemSeg_all	84.01	supervised	no	2023/06/07 22:31
helen1c_Ens4_CJ	83.82	supervised	no	2023/06/07 21:08
UniZG-FER_VSITE_SN-CNL-pyr	82.82	supervised	yes	2022/06/03 06:07

Slika 7.3: Službeni poredak *ACDC2023* natjecanja u nadziranoj semantičkoj segmentaciji. Naš model pod nazivom *helen1c_Ens4_CJ* ostvario je drugi rezultat. Važno je naglasiti da su samo dva modela predana na službeno vrednovanje ostvarila bolji rezultat od prošlogodišnjeg najboljeg modela.

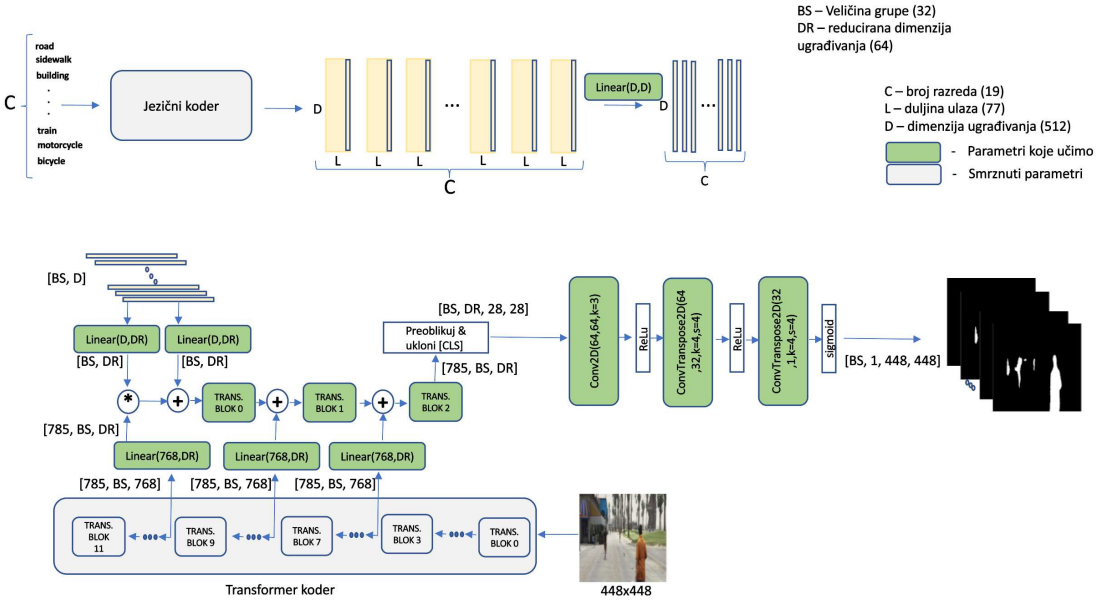
7.2. Primjena jezičnih ugrađivanja za semantičku segmentaciju

U ovom poglavlju bavimo se primjenom latentnih jezičnih ugrađivanja (engl. *language embeddings*) za rješavanje zadatka semantičke segmentacije. Motivacija iza eksperimenata koje smo proveli u okviru ovoga poglavlja vidjeti je može li korištenje jezičnih ugrađivanja poboljšati robusnost modela u smislu generalizacije domene. U tom smislu, modele smo učili na umjetnom skupu podataka prikupljenom iz računalne igre GTA5 [63], a vrednovali na skupu podataka *Cityscapes* [11] koji se sastoji od stvarnih slika vožnje. Proveli smo i eksperimente vrednovanja kontrastno prednaučenih modela bez dodatnog ugađanja (engl. *zero-shot*) na skupovima podataka GTA5 i *Cityscapes* (7.2.2), a dodatno smo vrednovali i kontrastno prednaučeni model u suradnji s modelom SAM [36] bez dodatnog ugađanja na skupovima podataka *Cityscapes* i *ACDC* [70] (7.2.2). Potom smo proveli eksperimente u kojima smo modele doista i učili na umjetnim slikama GTA5, a vrednovali na stvarnim slikama. Pritom smo za arhitekturu odabrali jednorezolucijski *SwiftNet*, a kao okosnicu koristili smo različite inicijalizacije modela *Convnext* [43]. Prve modele učili smo bez dodanih perturbacija ulaza te koristeći gubitak unakrsne entropije, a potom smo proveli i eksperimente s dodavanjem perturbacija i korištenjem gubitka svjesnog granica (engl. *boundary-aware loss*) [94] (3.2.2).

7.2.1. Primjena CLIPSega

Prvi eksperiment koji smo proveli prilagodba je modela CLIPSeg [47] za rješavanje zadatka semantičke segmentacije. U tu svrhu naučili smo model na umjetnom skupu podataka scena vožnje *GTA5* [63], a model smo vrednovali na *Cityscapesu* [11] (generalizacija domene). Za učenje modela koristili smo 20000 slučajno odabranih slika skupa podataka *GTA5* (odjeljak 7.2.3). Svi parametri vezani uz postupak učenja pre-

uzeti su iz originalnog članka [47]. Učenje smo proveli kroz 50000 iteracija, a veličina grupe iznosila je 32. Koristimo kosinusno kaljenje te optimizator AdamW [46]. Početna stopa učenja iznosi 0.001. Ulazna slika iz skupa podataka *GTA5* prvo se bilinearno poduzorkuje na dimenzije 720×1280, a za vrijeme učenja na ulaz modela predajemo slučajno odabrane isječke dimenzija 448×448. Model CLIPSeg omogućava dobivanje binarnih maski za razred predstavljen jezičnim opisom. Jedan primjer za učenje predstavljao je jedan isječak ulazne slike te binarna maska za jedan odabrani razred od 19 mogućih razreda (*Cityscapes* taksonomija razreda).



Slika 7.4: Primjena modela CLIPSeg za rješavanje zadatka semantičke segmentacije u okviru generalizacije domene *GTA5→Cityscapes*. Za koder slike koristimo ViT [15] koji je kontrastno prednaučen zajedno s jezičnim koderom. Prednaučene težine slikovnog i jezičnog kodera preuzete su sa službenog repozitorija modela CLIP [61]. Parametri kodera slike te jezičnog kodera za vrijeme učenja su smrznuti, odnosno učimo samo komponente koje su na slici prikazane zelenom bojom. Iz poduzorkovane ulazne slike slučajno odaberemo isječak dimenzija 448×448 te ga predamo slikovnom koderu. Jedan par za učenje čine ulazni isječak te pripadne binarne segmentacijske mape za svaki mogući razred. Nazive razreda predamo jezičnom koderu te za svaki razred dobijemo pripadno jezično ugrađivanje. Dobivena jezična ugrađivanja koristimo kao latentno znanje u početnoj fazi dekodera.

Dakle, iz jedne originalne ulazne slike nastaje 19 primjera za učenje u svakoj epohi, a ti primjeri se kroz epohe mijenjaju s obzirom da se za svaki razred mijenja slučajno odabrani isječak gdje taj razred segmentiramo. Ovaj pristup ima nedostataka, a najveći je u tomu što će se često dogoditi da slikovni elementi koji odgovaraju odabranom raz-

redu ne postoje u slučajno odabranom isječku. Postupak učenja modela CLIPSeg na skupu podataka *GTA5* prikazuje Slika 7.4. Za koder slike koristimo *base* verziju modela ViT [15], a jezični je koder *Transformer* [79]. Nakon završetka učenja vrednovali smo model na *Cityscapesu* i 600 slika iz izdvojenog podskupa *GTA5* (odjeljak 7.2.3). Dobivene rezultate prikazuje Tablica 7.10. S obzirom da smo učili model na isječcima dimenzije 448, vrednovanje smo proveli koristeći tehniku pomičnog prozora (engl. *sliding window*). Veličina prozora iznosila je 448, dok je korak (engl. *stride*) između dva uzastopna prozora iznosio 336.

Okosnica	Veličina isječka	Veličina grupe	GTA5 mIoU (%)	Cityscapes mIoU (%)
ViT-B_16	448	32	71.86	49.54

Tablica 7.10: Rezultati dobiveni korištenjem modela CLIPSeg [47] za rješavanje zadatka semantičke segmetnacije. Svi parametri učenja preuzeti su iz originalnog članka [47], izuzev veličine grupe koju smo postavili na 32, a model smo učili 50000 iteracija. Model smo učili na 20000 slika iz skupa podataka *GTA5*, a vrednovali na *Cityscapes* skupu za provjeru. Prikazanu mIoU vrijednost na skupu podataka *GTA5* dobili smo vrednovanjem na 600 izdvojenih slika tog skupa (odjeljak 7.2.3). Eksperiment je pokrenut na jednoj grafičkoj kartici NVIDIA RTX A4500 s 20GiB dostupne radne memorije.

7.2.2. Primjena prednaučenih OpenCLIP modela

U nastavku ovog poglavlja koristimo kontrastno prednaučene modele OpenCLIP [8], a parametre tih modela preuzeli smo preko službenog repozitorija.[14] Koristit ćemo dvije verzije modela *Convnext* [43] (*base* i *large*). U nastavku, ako drugačije nije navedeno, pod *base*[15] verzijom kontrastno prednaučenog modela smatramo model prednaučen na skupu podataka *LAION-A*, što je podskup od $\sim 900M$ primjera originalnog skupa podataka *LAION-5B* [72]. S druge strane, *large*[16] verzija modela koju koristimo u nastavku prednaučena je na skupu podataka *LAION-2B*, što je podskup od $\sim 2B$ primjera iz originalnog skupa podataka *LAION-5B*. Rezolucija je ulazne slike za vrijeme predučenja obiju verzija modela iznosila 320×320. Za kodiranje riječi, odnosno jezičnih predložaka, koristimo koder *Transformera* [79] koji je učen kontrastno zajedno s odgovarajućim modelom za kodiranje slike. Dimenzija jezičnih ugrađivanja koju na izlazu daje jezični koder za *base* verziju modela jest 640, dok je za *large* verziju modela dimenzija jezičnih ugrađivanja jednaka 768.

[14] `https://github.com/mlfoundations/open_clip`

[15] Naziv verzije: `convnext_base_w_320.laion_aesthetic_s13b_b82k_augreg`.

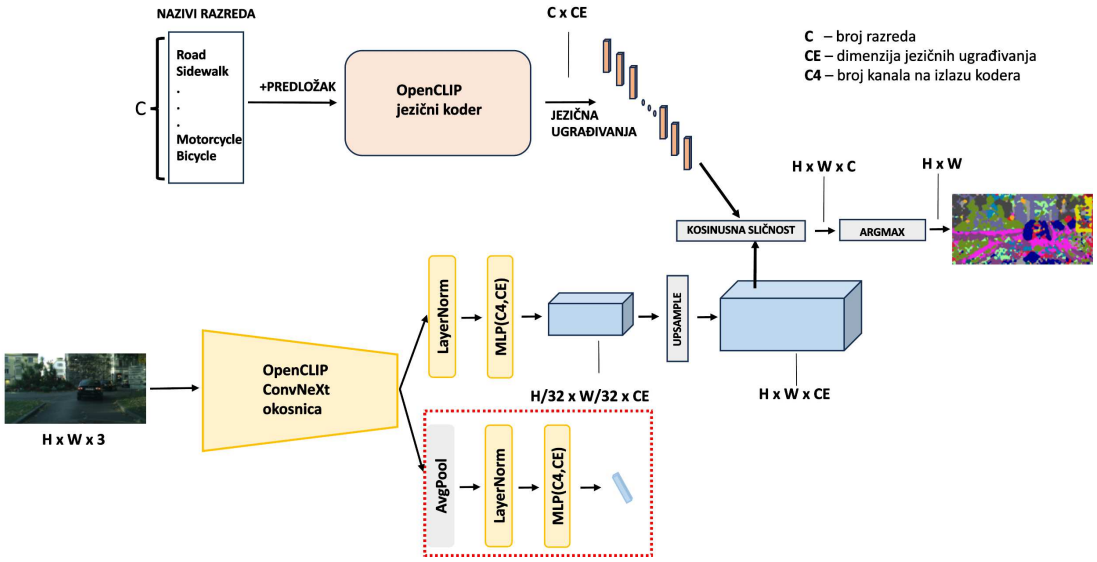
[16] Naziv verzije: `convnext_large_d_320.laion2b_s29b_b131k-ft`.

OpenCLIP bez ugađanja

Prvi eksperiment koji smo proveli vrednovanje je performansi OpenCLIP modela bez dodatnog ugađanja (engl. *zero-shot*) na *Cityscapes* [11] podskupu za provjeru te na 500 slučajno uzorkovanih slika iz skupa podataka *GTA5* [63]. Skupovi podataka *Cityscapes* i *GTA5* imaju istu taksonomiju razreda kojih ukupno ima 19. Nazive tih razreda ugradili smo u jednostavni tekstualni predložak (engl. *template*): `"A photo of _"` i predali jezičnom koderu. Tako smo na izlazu jezičnog kodera dobili jezična ugrađivanja $\mathbf{E} \in \mathbb{R}^{C \times C_E}$ za svaki razred. Pretpostavimo da želimo odrediti segmentacijsku mapu za sliku na ulazu dimenzija $H \times W \times 3$. Tu sliku prvo predajemo koderu slike koji se temelji na prednaučenom modelu (OpenCLIP *Convnext*). Originalni model CLIP [61] učen je na zadatku klasifikacije slike. Kako bismo riješili zadatak semantičke segmentacije izostavljamo klasifikacijsku glavu, odnosno na izlazu kodera slike dobivamo poduzorkovane značajke $\mathbf{F}_4 \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times C_4}$. Dobivene značajke potom prednaučenim preslikavanjem projiciramo u dimenziju ugrađivanja C_E te na izlazu dobivamo značajke dimenzija $\frac{H}{32} \times \frac{W}{32} \times C_E$. Potom dobivene značajke bilinearno naduzorkujemo do dimenzija originalne slike $H \times W$, a nakon toga računamo kosinusnu sličnost između jezičnih ugrađivanja $\mathbf{E}$ i projiciranih naduzorkovanih značajki $\mathbf{F}_4$. Potom, nakon provođenja operacije $argmax$ po dimenziji kanala, na izlazu dobivamo segmentacijsku mapu ulazne slike. Ilustraciju ovog postupka prikazuje Slika 7.5.

Okosnica	Dodatni predlošci	Cityscapes mIoU (%)	GTA5 mIoU (%)
convnext-base-laiona_320	✗	18.74	15.90
convnext-base-laiona_320	✓	21.04	17.43
convnext-large-laion2b_320	✓	19.59	15.97

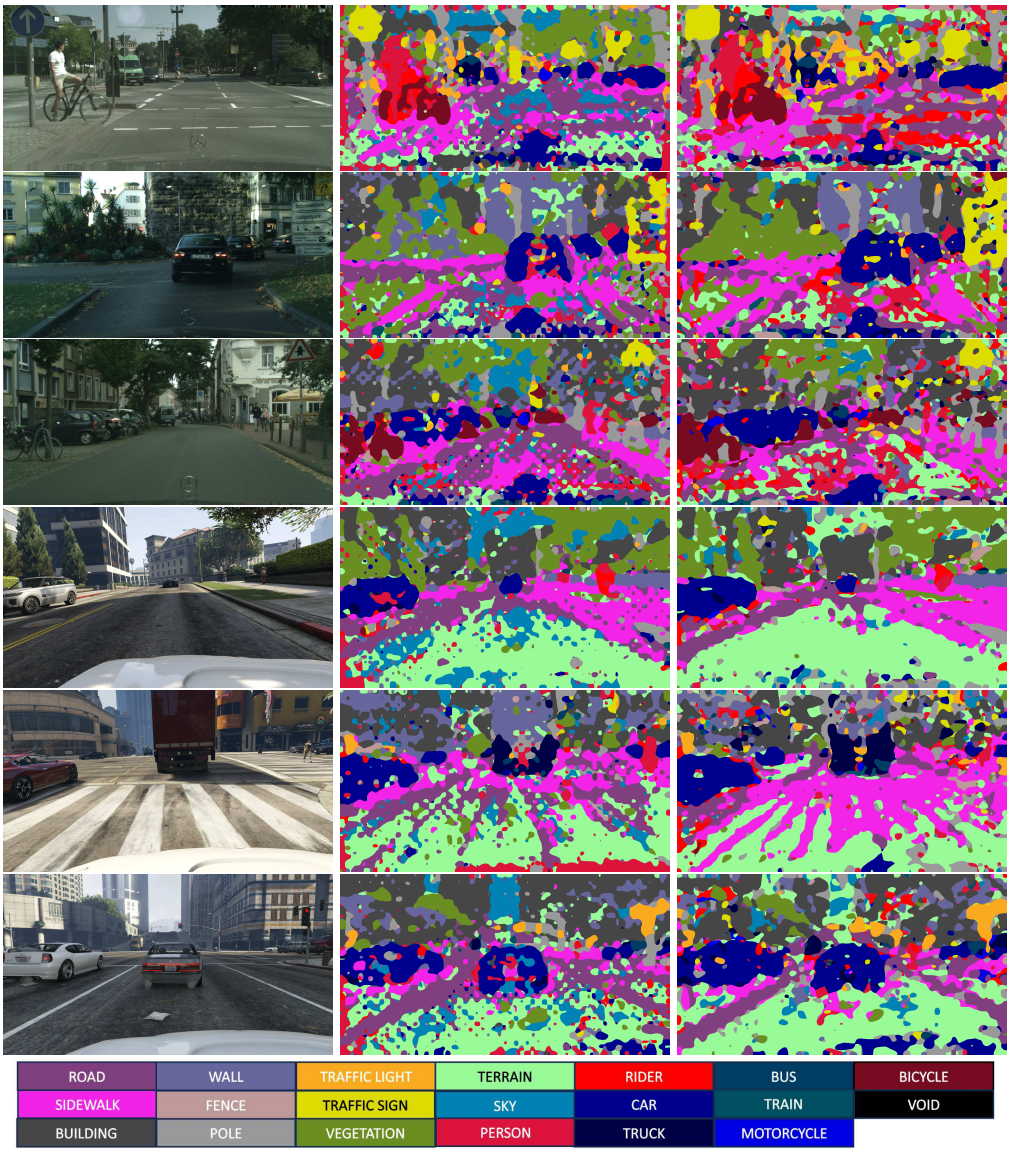
Tablica 7.11: Rezultati semantičke segmentacije korištenjem OpenCLIP *Convnext* modela kao okosnice. Prvi redak predstavlja korištenje *base* verzije modela sa samo jednim tekstualnim predloškom. Kod naredna dva retka koristili smo 85 predložaka čija ugrađivanja smo za svaki razred uprosječili. Prikazani rezultati dobiveni su vrednovanjem nad *Cityscapes* skupom za provjeru i 500 slučajnih slika iz skupa podataka *GTA5*.



Slika 7.5: Korištenje prednaučenog OpenCLIP kodera za semantičku segmentaciju bez ugađanja (engl. *zero-shot*). Nazive semantičkih razreda ugradimo u tekstualni predložak i predamo jezičnom koderu. Nakon toga za svaki razred na izlazu jezičnog kodera imamo jezično ugrađivanje koje predstavlja taj razred. Ulaznu sliku koju želimo semantički segmentirati predamo OpenCLIP *Convnext* okosnici, a izlazne značajke projiciramo u jezični semantički prostor korištenjem prednaučene projekcije (MLP na slici). Dobivene značajke potom bilinearno naduzorkujemo na rezoluciju ulazne slike, a nakon toga računamo kosinusnu sličnost jezičnih ugrađivanja i ugrađivanja svakog slikovnog elementa. Konačni razred za svaki slikovni element dobijemo tako što odaberemo njemu najsličnije jezično ugrađivanje.

Rezultate prvog eksperimenta možemo vidjeti u prvom retku Tablice 7.11. U prvom eksperimentu koristili smo samo jedan predložak u koji smo umetali naziv razreda: `"A photo of _"`. Po uzoru na članak [19], u drugom eksperimentu koristili smo listu od 85 predložaka (npr. `"There is a _ in the scene"`). Kao konačno jezično ugrađivanje za svaki razred koristimo prosjek jezičnih ugrađivanja nastalih od 85 ispunjenih predložaka. U drugom eksperimentu pokrenuli smo vrednovanje *base* i *large* verzija *Convnext* modela, a rezultate prikazuje Tablica 7.11. Dobiveni rezultati značajno su lošiji u vidu mIoU metrike od stadardnih nadziranih metoda, ali s obzirom na jednostavnost opisanog pristupa, možemo zaključiti da korištenje jezičnih ugrađivanja i prednaučenih OpenCLIP modela daje rezultate poticajne za buduće istraživanje primjene jezičnih ugrađivanja u semantičkoj segmetaciji. Posebna pogodnost ovog pristupa leži u činjenici da možemo mijenjati taksonomiju razreda, što uglavnom nije slučaj u nadziranim metodama gdje je broj razreda unaprijed definiran i fiksan. Kvalitativne rezultate opisanih eksperimenata na nekoliko primjera prikazuje Slika 7.6.

U svim sljedećim eksperimentima gdje koristimo kodiranje naziva razreda u jezična ugrađivanja koristimo prosjek 85 predložaka, s obzirom da je korištenje više predložaka poboljšalo rezultate bez ugađanja.

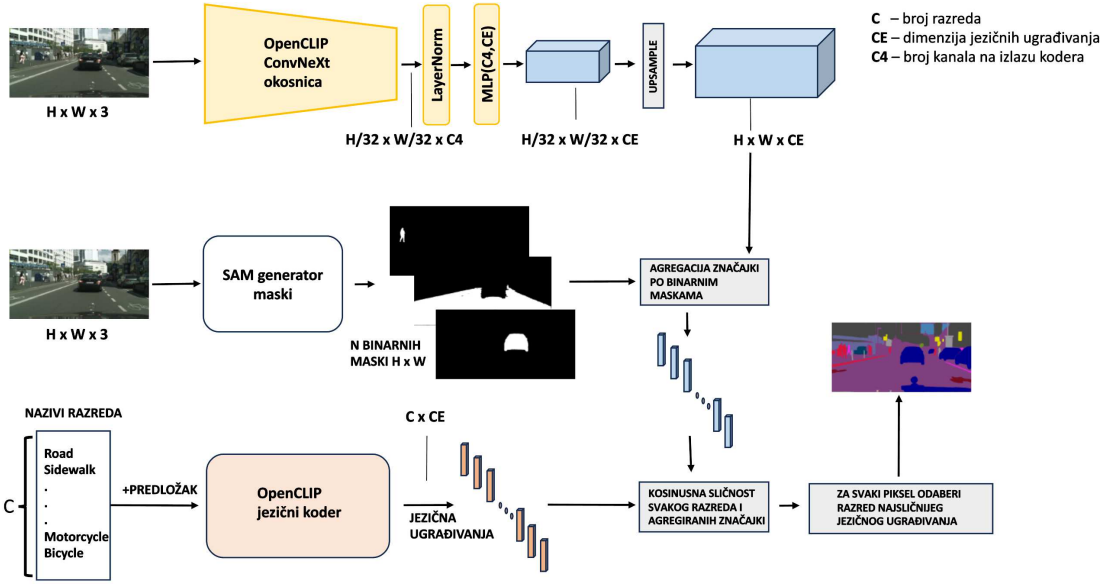


Slika 7.6: Primjeri semantičke segmentacije koristeći prednaučeni *Convnext* bez dodatnog ugađanja. Prvi stupac predstavlja ulazne slike. Prva tri reda predstavljaju slike iz skupa podataka *Cityscapes*, a ostali redovi slike iz skupa podataka *GTA5*. Drugi stupac predikcije su *base* verzije modela, dok su četvrti stupac predikcije *large* verzije modela.

OpenCLIP u suradnji s modelom SAM

U ovom odjeljku iskoristili smo model SAM [36] zajedno s prednaučenim OpenCLIP *Convnext* okosnicama za semantičku segmentaciju scena vožnje bez dodatnog ugađanja. Model SAM omogućava nam dobivanje niza binarnih maski za ulaznu sliku, ali

pritom nemamo bilo kakvu informaciju o semantici te maske. Kako bi doznali semantiku maske iskoristili smo prednaučeni OpenCLIP jezični i slikovni koder. Neka raspolažemo ulaznom slikom dimenzija $H \times W \times 3$. Predamo li tu sliku modelu SAM, na izlazu dobivamo niz od N binarnih maski dimenzija $H \times W$. Istu sliku predamo i prednaučenom OpenCLIP *Convnext* modelu te na izlazu dobijemo poduzorkovane značajke projicirane u semantički prostor jezičnih ugrađivanja. Dobivene značajke potom bilinearno naduzorkujemo na dimenzije ulazne slike. Nakon tog postupka raspolažemo nizom od N binarnih maski te značajkama $\mathbf{F}_4 \in \mathbb{R}^{H \times W \times C_E}$. Potom uprosječimo značajke $\mathbf{F}_4$ za svaku masku koju je na izlazu dao model SAM tako da na izlazu dobijemo niz ugrađivanja $\mathbf{SE} \in \mathbb{R}^{N \times C_E}$. Nakon toga izračunamo kosinusnu sličnost između jezičnih ugrađivanja $\mathbf{E} \in \mathbb{R}^{C \times C_E}$ i uprosječenih značajki za svaku binarnu masku $\mathbf{SE}$.



Slika 7.7: Korištenje prednaučenog modela SAM i OpenCLIP kodera za semantičku segmentaciju bez dodatnog ugađanja. Ulaznu sliku predamo modelu SAM te na izlazu dobijemo niz binarnih maski. Istu sliku predamo i OpenCLIP *Convnext* okosnici, a izlazne značajke projiciramo u jezični semantički prostor i bilinearno naduzorkujemo do rezolucije ulazne slike. S druge strane, nazive semantičkih razreda ugradimo u tekstualne predloške i predamo jezičnom koderu, a na izlazu dobijemo jezična ugrađivanja za svaki semantički razred. Potom za svaku binarnu masku koju je dao model SAM uprosječimo naduzorkovane značajke dobivene iz *Convnext* okosnice na mjestima gdje binarna maska poprima vrijednost 1. Tako dobijemo niz slikovnih ugrađivanja koje potom usporedimo s jezičnim ugrađivanjima izračunavanjem kosinusne sličnosti. Za svaku binarnu masku odaberemo razred najsličnijeg jezičnog ugrađivanja uprosječenoj značajci te maske.

Za svaku binarnu masku potom odaberemo najsličnije jezično ugrađivanje. S obzirom

da model SAM na izlazu daje binarne maske koje se međusobno mogu preklapati, ako za ista područja ulazne slike dobijemo pripadnost različitim razredima, kao konačni razred odabiremo onaj koji je imao veću sličnost s odgovarajućim jezičnim ugrađivanjem. Opisani postupak ilustrira Slika 7.7.

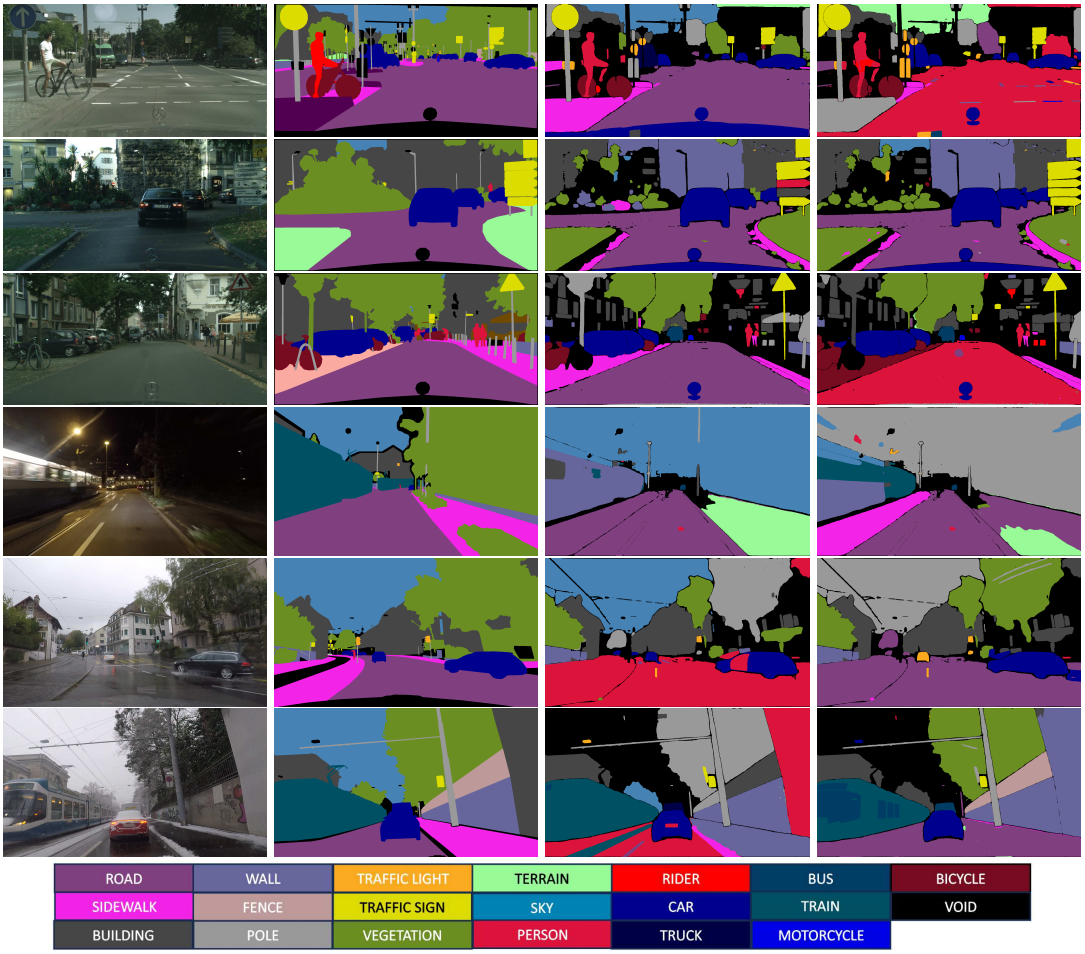
Okosnica	Cityscapes mIoU (%)	ACDC mIoU (%)
convnext-base-laiona_320	39.44	25.40
convnext-large-laion2b_320	32.74	26.68

Tablica 7.12: Dobiveni rezultati primjene OpenCLIP *Convnext* prednaučenih modela u suradnji s modelom SAM, kao što prikazuje Slika 7.7. Navedene rezultate dobili smo vrednovanjem na skupovima za provjeru.

Rezultate eksperimenata koje smo ranije opisali prikazuje Tablica 7.12. U oba eksperimenta koristili smo *huge*[17] verziju modela SAM. Zanimljivo je da *base* verzija modela radi značajno bolje (7 mIoU bodova) od *large* verzije modela na podskupu za provjeru skupa podataka *Cityscapes*. S druge strane, uočavamo kako *large* verzija modela ostvaruje bolji rezultat (1 mIoU bod) na podskupu za provjera skupa podataka *ACDC*. S obzirom da su navedeni rezultati dobiveni bez dodatnog ugađanja korištenih modela, možemo reći da rezultat od 40% mIoU na skupu podataka *Cityscapes* izgleda poticajno za nastavak istraživanja.

Pogledamo li kvalitativne rezultate provedenih eksperimenata (Slika 7.8), možemo uočiti kako sama lokalizacija maski i objekata radi prilično dobro (zadaća modela SAM). S druge strane, vidimo kako dobivene maske često nisu dobro klasificirane, odnosno najsličnije jezično ugrađivanje ne predstavlja točan razred. Nastavak istraživanja u smjeru popravka klasifikacije dobivenih maski dio je budućeg rada, s obzirom da i kvalitativni rezultati izgledaju obećavajuće.

[17] `https://github.com/facebookresearch/segment-anything#model-checkpoints`



Slika 7.8: Primjeri semantičke segmentacije koristeći SAM u suradnji s kontrastno prednaučenim *Convnextom*. Prvi stupac predstavlja ulazne slike. Prva tri reda predstavljaju slike iz skupa podataka *Cityscapes*, a ostali redovi slike iz skupa podataka *ACDC*. Drugi stupac stvarne su oznake, treći stupac predikcije su *base* verzije modela, dok četvrti stupac predstavlja predikcije *large* verzije modela.

7.2.3. Kombinacija OpenCLIPa i *SwiftNeta*

Podjela skupa podataka GTA5

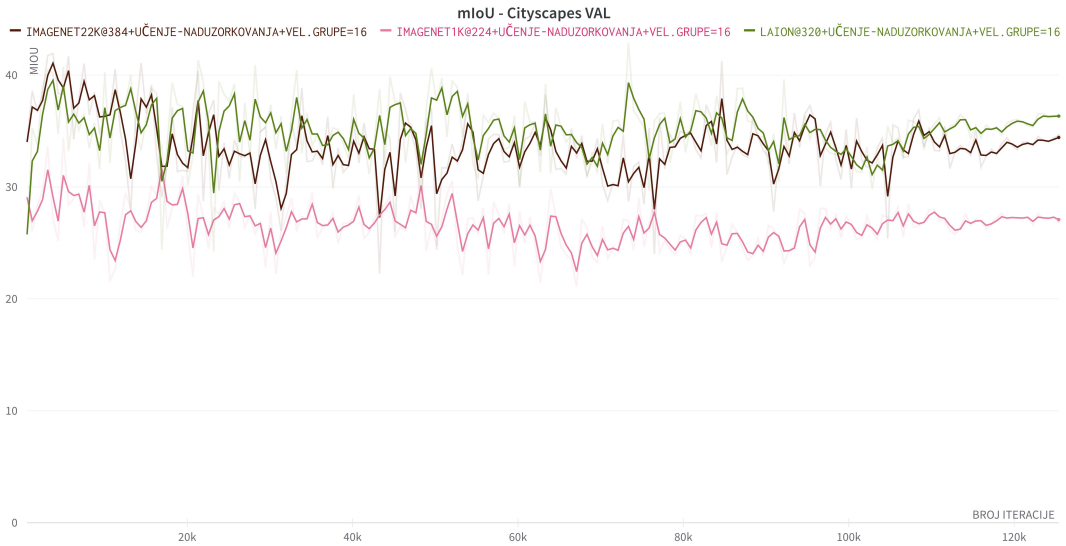
Za provedbu eksperimenata u ovom poglavlju slučajno smo odabrali podskup od 20000 slika skupa podataka *GTA5* na kojemu smo učili naše modele. Dio ostatka slika iskoristili smo za vrednovanje naših modela. Ovakva podjela nije najbolja s obzirom da se može dogoditi da slične slike završe u skupu za učenje i skupu za vrednovanje. U svakom slučaju, naš je fokus generalizacija na stvarne slike skupa podataka *Cityscapes*. Ipak, kako bi rezultati na skupu *GTA5* bili ispravni, dodatno smo proveli eksperimente s originalnom podjelom skupa podataka *GTA5*.

Provedeni eksperimenti

Prvi eksperiment koji smo proveli učenje je jednorezolucijske *SwiftNet* [48] arhitekture (odjeljak 3.2.1) s *Convnext base* [42] okosnicom na 20000 slika skupa podataka *GTA5* (odjeljak 7.2.3). U okviru ovog eksperimenta proveli smo tri postupka učenja, a razlika između svakog bila je početna inicijalizacija okosnice. Koristili smo 3 različite inicijalizacije parametara: prednaučeno na *ImageNet1k*, na *ImageNet22k* te *LAION-A* [72]. Učili smo samo put naduzorkovanja, a parametre okosnice za vrijeme učenja smo smrznuli (engl. *freeze*). Svako učenje proveli smo kroz 100 epoha. Veličina grupe iznosila je 16, stopa učenja $4\mathrm{e}-4$, kao raspoređivač stope učenja koristimo kosinusno kaljenje s minimalnom vrijednošću stope učenja $4\mathrm{e}-6$, a kao optimizator koristimo Adam [34] s podrazumijevanim vrijednostima parametara. Dimenzija puta naduzorkovanja iznosila je 256. Modele smo vrednovali na *Cityscapes* skupu za provjeru. Osim toga, iz izdvojenog podskupa za provjeru (odjeljak 7.2.3) skupa podataka *GTA5* odabrali smo 600 slika na kojima smo dodatno vrednovali naše modele. Odabrali smo samo podskup od 600 slika kako bi skratili vrijeme vrednovanja, s obzirom da smo vrednovanje provodili na kraju svake epohe. Ako ne navedemo drugačije, ovi parametri podrazumijevani su parametri za sve eksperimente u ovom odjeljku. Rezultate ovog eksperimenta prikazuje Tablica 7.13. Možemo uočiti kako smo najbolji rezultat na *Cityscapes* skupu za provjeru ostvarili korištenjem okosnice koja je prednaučena na skupu podataka *LAION-A* [72]. Također je zanimljivo uočiti kako okosnica prednaučena na skupu podataka *ImageNet1k* ostvaruje značajno lošije performanse na *Cityscapesu* u usporedbi s druge dvije inicijalizacije. Na Slici 7.9 možemo vidjeti kretanje mIoU metrike na *Cityscapes* skupu za provjeru po epohama.

Okosnica	Cityscapes mIoU (%)	GTA5 mIoU (%)
convnext-base-in1k_224	26.91	80.48
convnext-base-in22k_384	34.56	82.95
convnext-base-laiona_320	36.35	80.82

Tablica 7.13: Rezultati semantičke segmentacije korištenjem *Convnext base* okosnica koje su prednaučene na različitim skupovina podataka (*ImageNet1k*, *ImageNet22k* te *LAION-A*). Dimenzije isječaka na kojima smo učili modele iznosi 512. Sve detalje vezane uz postupak učenja opisali smo u odjeljku 7.2.3. Prikazani rezultati dobiveni su vrednovanjem modela iz posljednje epohe na *Cityscapes* podskupu za provjeru te na 600 slika iz izdvojenog podskupa za provjeru skupa podataka *GTA5*.



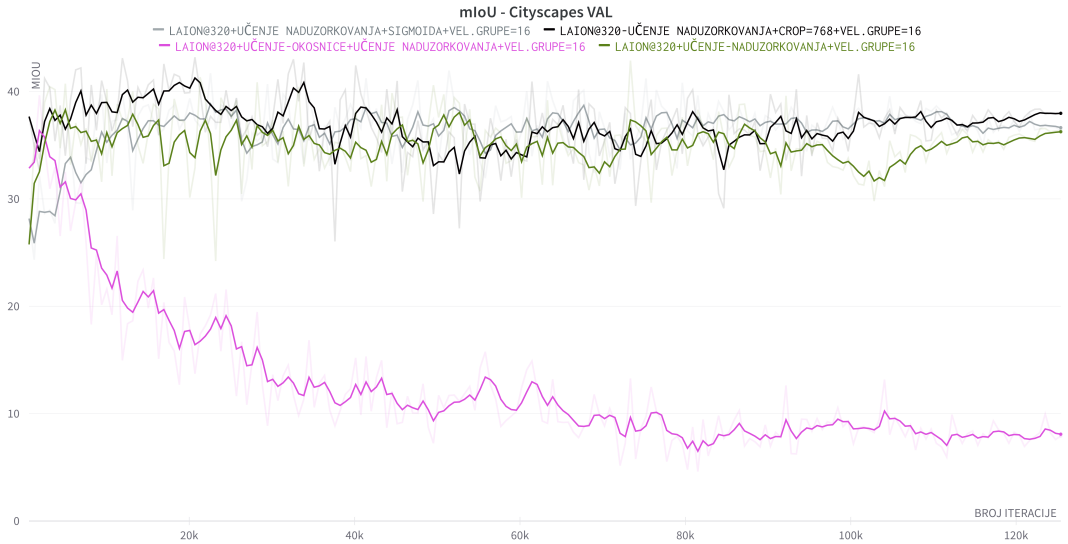
Slika 7.9: Kretanje mIoU po epohama za tri različite inicijalizacije *Convnext base* okosnice. Zelenom je prikazana inicijalizacija prednaučena na skupu podataka *LAION-A*, tamnosmeđom inicijalizacija prednaučena na skupu podataka *ImageNet22k*, dok je rozom prikazana inicijalizacija prednaučena na skupu podataka *ImageNet1k*.

Dodatno smo proveli eksperimente u kojima koristimo OpenCLIP *Convnext* kao koder, ali pritom učimo okosnicu, povećamo veličinu isječka ili $softmax$ na izlazu zamijenimo $sigmoidom$. Rezultate tih eksperimenata prikazuje Tablica 7.14. Možemo zaključiti kako povećanje dimenzije isječka na kojima učimo model poboljšava performanse u smislu mIoU metrike, a također kako korištenje $sigmoide$ ne doprinosi performansama. Dodatno, učenje okosnice značajno degradira performanse na *Cityscapes* skupu za provjeru, dok je doprinos učenja okosnice na 600 slika *GTA5* podskupa za provjeru manji od jednog mIoU boda.

Okosnica	Učenje okosnice	Veličina isječka	$sigmoid$	mIoU (%) Cityscapes	mIoU (%) GTA5
convnext-base-laiona_320	✗	512	✗	36.35	80.82
convnext-base-laiona_320	✓	512	✗	7.92	81.64
convnext-base-laiona_320	✗	768	✗	38.01	82.26
convnext-base-laiona_320	✗	512	✓	36.40	74.00

Tablica 7.14: Rezultati semantičke segmentacije korištenjem OpenCLIP *Convnext base* okosnice. Sve detalje vezane uz postupak učenja opisali smo u odjeljku 7.2.3. Dodatno, stopa učenja okosnice 4 puta je manja od stope učenja puta naduzorkovanja. Prikazani rezultati dobiveni su vrednovanjem modela iz posljednje epohe na *Cityscapes* podskupu za provjeru te na 600 slika iz izdvojenog podskupa za provjeru skupa podataka *GTA5*.

Zanimljivo je pogledati i kretanje mIoU metrike na *Cityscapes* skupu za provjeru (Slika 7.10), osobito za eksperiment u kojemu učimo okosnicu. Možemo uočiti kako u početnim epohama model s učenjem okosnice ostvaruje rezultate slične ostalim modelima kod kojih je okosnica smrznuta, ali već nakon 5 epoha performanse počinju naglo degradirati. Dolazimo do zaključka kako prednaučene težine OpenCLIP *Convnext* okosnice imaju veliki doprinos na generalizaciju modela na slikama ciljne domene. Dodatno, kako ne bismo zaboravili važne koncepte naučene za vrijeme predučenja, ima smisla pokrenuti eksperiment s manjim vrijednostima stope učenja za okosnicu.



Slika 7.10: Kretanje mIoU po epohama za OpenCLIP *Convnext base* okosnicu i različite modifikacije. Crnom je prikazano kretanje metrike mIoU za eksperiment s veličinom isječka 768, rozom je prikazan eksperiment s učenjem okosnice, sivom je prikazano kretanje mIoU metrike za eksperiment u kojemu koristimo *sigmoidu* umjesto *softmaxa* na izlazu, dok je zelenom prikazan standardni eksperiment bez dodatnih modifikacija. Sve detalje vezane uz postupak učenja opisali smo u odjeljku 7.2.3.

Korištenje jezičnih ugrađivanja

U ovom odjeljku opisat ćemo postupak korištenja jezičnih ugrađivanja za semantičku segmentaciju. Neka raspolažemo OpenCLIP *Convnext* okosnicom te pripadajućim jezičnim koderom koji su kontrastno prednaučeni, kao što smo opisali u odjeljku 4.2. S obzirom da rješavamo zadatak semantičke segmentacije, imamo niz razreda koje želimo raspoznati na ulaznoj slici. Nazive tih razreda, nakon umetanja u 85 jezičnih predložaka (odjeljak 7.2.2), predamo jezičnom koderu, a na izlazu kodera dobijemo jezično ugrađivanje za svaki razred. Označimo dobivena ugrađivanja s $\mathbf{E} \in \mathbb{R}^{C \times C_E}$, pri

čemu je C broj razreda, a C_E dimenzija svakog ugrađivanja. Sliku koju želimo semantički segmentirati dovodimo na ulaz OpenCLIP *Convnext* kodera te na izlazu dobijemo poduzorkovane značajke $\mathbf{F}_4 \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times C_4}$, pri čemu je C_4 dimenzija značajki na izlazu okosnice, a H i W predstavljaju visinu i širinu ulazne slike. Dobijene značajke potom prednaučenom 1×1 konvolucijom projiciramo u dimenziju jezičnih ugrađivanja C_E, a na izlazu dobijemo značajke $\hat{\mathbf{F}}_4 \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times C_E}$. Dobivene značajke predamo putu naduzorkovanja (dimenzija jednaka C_E) jednorezolucijskog *SwiftNeta* [48], a na izlazu dobijemo naduzorkovane značajke $\mathbf{I} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C_E}$. Dobivene naduzorkovane značajke normiramo po dimenziji kanala, a isto napravimo i s jezičnim ugrađivanjima $\mathbf{E}$. Potom izračunamo skalarni umnožak između normiranih značajki $\mathbf{I}$ i jezičnih ugrađivanja $\mathbf{E}$ (kosinusna sličnost), te na izlazu dobijemo tenzor $\mathbf{SE} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C}$:

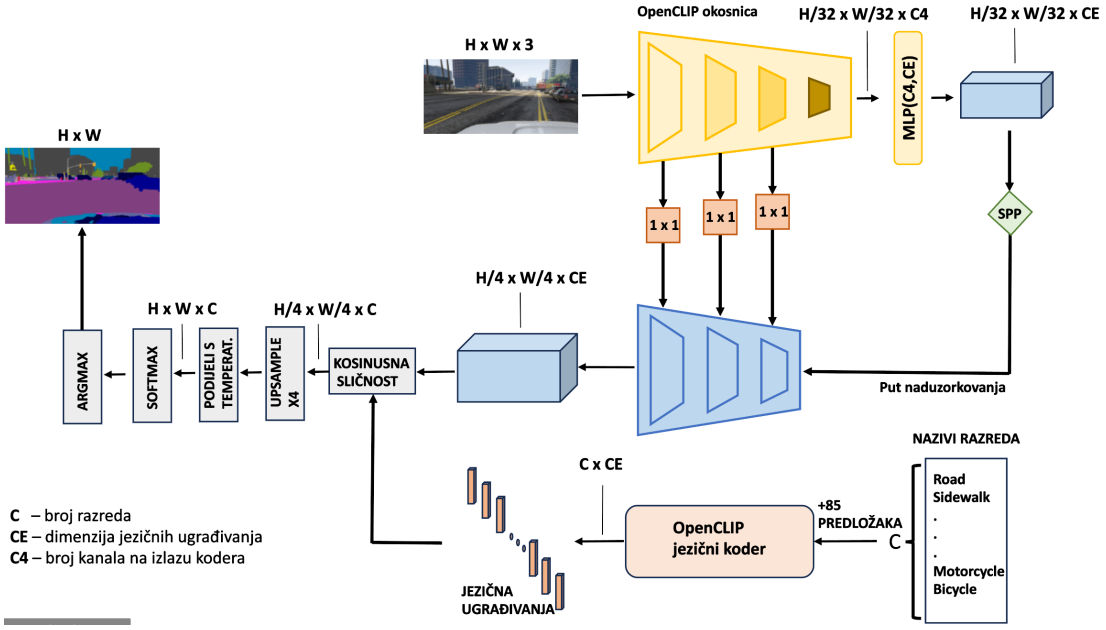
$$\mathbf{SE}_{ijc} = \mathbf{E}_c \cdot \mathbf{I}_{ij} = \sum_{k=1}^{C_E} \mathbf{E}_{ck} \cdot \mathbf{I}_{ijk}. \tag{7.1}$$

Dobiveni tenzor bilinerano naduzorkujemo na dimenziju originalne slike, podijelimo s temperaturom t, a nad tako dobivenim logitima izračunamo $softmax$ po dimenziji kanala. Potom koristimo standardni gubitak unakrsne entropije. Opisani postupak ilustrira Slika 7.11.

Okosnica	Učenje okosnice	Veličina isječka	Jezična ugrađivanja	mIoU (%) Cityscapes	mIoU (%) GTA5
convnext-base-laiona_320	✗	512	✗	36.35	80.82
convnext-base-laiona_320	✗	512	✓	37.28	81.73

Tablica 7.15: Rezultati semantičke segmentacije korištenjem OpenCLIP *Convnext base* okosnice i jezičnih ugrađivanja. Sve detalje vezane uz postupak učenja opisali smo u odjeljku 7.2.3. Dodatno, vrijednost temperature t postavili smo na 0.07 [61]. Prikazani rezultati dobiveni su vrednovanjem modela iz posljednje epohe na *Cityscapes* podskupu za provjeru te na 600 slika iz izdvojenog podskupa za provjeru skupa podataka *GTA5*.

Rezultat koji smo dobili dodavanjem jezičnih ugrađivanja i usporedbu s rezultatom bez korištenja jezičnih ugrađivanja prikazuje Tablica 7.15. Možemo uočiti kako je dodavanje jezičnih ugrađivanja poboljšalo rezultate u smislu mIoU metrike s obzirom na eksperiment bez jezičnih ugrađivanja za približno 1 mIoU bod. Ipak, ova usporedba nije najpravednija, s obzirom da eksperiment s korištenjem jezičnih ugrađivanja ima dimenziju puta naduzorkovanja 640, dok u eksperimentu bez korištenja jezičnih ugrađivanja dimenzija puta naduzorkovanja iznosi 256. Naknadno ćemo provesti i eksperiment u kojemu su dimenzije puta naduzorkovanja jednake i u slučaju korištenja kao i bez korištenja jezičnih ugrađivanja.



Slika 7.11: Korištenje prednaučene OpenCLIP *Convnext* okosnice te jednorezolucijskog *SwiftNet* [48] puta naduzorkovanja, zajedno s jezičnim ugrađivanjima na izlazu. Nazive razreda, umetnute u jezične predloške, predamo jezičnom koderu i dobijemo jezično ugrađivanje za svaki razred. Ulaznu sliku predamo u okosnicu, a dobivene značajke projiciramo u jezični semantički prostor korištenjem prednaučene projekcije. Nakon toga koristimo put naduzorkovanja jednorezolucijskog *SwiftNeta*, pri čemu je dimenzija puta naduzorkovanja jednaka dimenziji jezičnih ugrađivanja, a za *base* verziju modela iznosi 640. Potom izračunamo kosinusnu sličnost između dobivenih značajki i jezičnih ugrađivanja, dobivene sličnosti bilinearno naduzorkujemo na dimenzije ulazne slike, podijelimo s temperaturom t, a potom nad dobivenim tenzorom izračunamo $softmax$ po dimenziji kanala.

Utjecaj hiperparametara

U ovom odjeljku pogledat ćemo utjecaj različitih hiperparametara na performanse modela u smislu mIoU metrike. Prvi eksperiment razmatranje je utjecaja vrijednosti temperature t kod učenja korištenjem jezičnih ugrađivanja. Rezultate tog eksperimenta prikazuje Tablica 7.16. Najbolje performanse na *Cityscapes* skupu za provjeru dobivamo korištenjem vrijednosti temperature $t = 0.07$, slično kao u radu [61]. U svim eksperimentima u nastavku ovog odjeljka, ako drugačije nije rečeno, koristimo jednake postavke postupka učenja kao što smo opisali u odjeljku 7.2.3.

Okosnica	Temperatura	mIoU (%)	
		GTA5 Val	*Cityscapes* Val
convnext-base-laiona_320	0.70	65.41	35.44
	0.50	67.02	39.67
	0.20	66.30	37.44
	0.09	**74.49**	39.09
	0.07	74.26	**39.36**
	0.04	73.12	36.89

Tablica 7.16: Odabir vrijednosti temperature kojom dijelimo logite nakon izračuna kosinusne sličnosti s jezičnim ugrađivanjima. Veličina isječka iznosila je 512, a početna stopa učenja za put naduzorkovanja je $4e{-}4$. Veličina grupe za učenje iznosi 16. Drugi stupac predstavlja različite vrijednosti temperature kroz eksperimente. Svaki eksperiment pokrenut je kroz 5 epoha na jednoj grafičkoj kartici NVIDIA A100 s 40GiB dostupne memorije.

Okosnica	Veličina grupe	Veličina isječka	Jezična ugrađivanja	mIoU (%)	
				GTA5–val	City–val
convnext-base-laiona_320	8	512	✓	76.45	37.81
			✗	74.56	41.67
		768	✓	78.55	46.17
			✗	77.75	45.96
		1024	✓	79.22	45.17
			✗	78.92	46.51
	16	512	✓	75.67	38.16
			✗	75.73	40.00
		768	✓	78.80	45.23
			✗	78.46	43.44
convnext-large-laion2b_320	8	512	✓	77.19	37.44
			✗	77.50	42.10
		768	✓	79.82	41.24
			✗	79.56	42.78
		1024	✓	80.45	**49.69**
			✗	80.50	44.01
	16	512	✓	77.89	38.63
			✗	77.66	37.28
		768	✓	79.80	46.57
			✗	79.77	45.30

Tablica 7.17: Utjecaj veličine grupe, veličine isječka, verzije okosnice i korištenja jezičnih ugrađivanja za semantičku segmentaciju. Svi eksperimenti provedeni kroz 10 epoha učenja na jednoj grafičkoj kartici NVIDIA A100 s 40GiB dostupne radne memorije. Prikazani rezultati dobiveni su vrednovanjem u posljednjoj epohi učenja.

Okosnica	Veličina grupe	Stopa učenja okosnice	mIoU (%)	
			GTA5 Val	*Cityscapes* Val
convnext-base-in1k_224	8	1e−4	74.11	10.55
		8e−5	73.40	20.26
		6e−5	74.27	16.23
		4e−5	75.90	24.40
		2e−5	76.97	29.12
	16	1e−4	75.51	13.14
		8e−5	76.55	23.38
		6e−5	77.60	22.68
		4e−5	76.97	28.20
		2e−5	76.94	26.77
convnext-base-in22k_384	8	1e−4	77.33	13.14
		8e−5	78.94	17.21
		6e−5	78.42	26.53
		4e−5	78.58	32.15
		2e−5	79.92	38.49
	16	1e−4	78.04	20.07
		8e−5	79.19	23.22
		6e−5	79.80	29.26
		4e−5	78.95	37.34
		2e−5	78.55	44.02
convnext-base-laiona_320	8	1e−4	75.88	10.81
		8e−5	77.46	13.84
		6e−5	79.00	19.90
		4e−5	78.06	33.99
		2e−5	78.84	34.46
	16	1e−4	78.75	16.05
		8e−5	75.04	18.52
		6e−5	78.60	23.00
		4e−5	79.14	32.66
		2e−5	79.24	43.34

Tablica 7.18: Utjecaj skupa podataka za predučenje, veličine grupe i stope učenja okosnice na performanse modela u smislu mIoU metrike. Svaki eksperiment pokrenut kroz 8 epoha učenja, veličina isječka je 512, a početna stopa učenja za put naduzorkovanja u svim eksperimentima iznosi $4\mathrm{e}{-4}$. U svim eksperimentima, osim puta naduzorkovanja, učimo i okosnicu, pri čemu je početna stopa učenja okosnice prikazana u trećem stupcu. Prvi stupac prikazuje skup podataka i rezoluciju na kojoj je okosnica prednaučena. Sve eksperimente pokrenuli smo na jednoj grafičkoj kartici NVIDIA A100 s 40GiB dostupne radne memorije. Prikazane rezultate dobili smo vrednovanjem modela u posljednjoj epohi učenja.

Ponovljeni eksperimenti - originalna podjela GTA5

U ovom odjeljku proveli smo eksperiment s originalnom podjelom[18] skupa podataka *GTA5*. Broj podataka u svakom od podskupova možemo vidjeti u Tablici 7.19.

	Podskup za učenje	Podskup za provjeru	Podskup za ispitivanje
Broj podataka	12402	6347	6155

Tablica 7.19: Broj podataka u svakom podskupu (za učenje, provjeru i ispitivanje) kod originalne podjele skupa podataka *GTA5*.

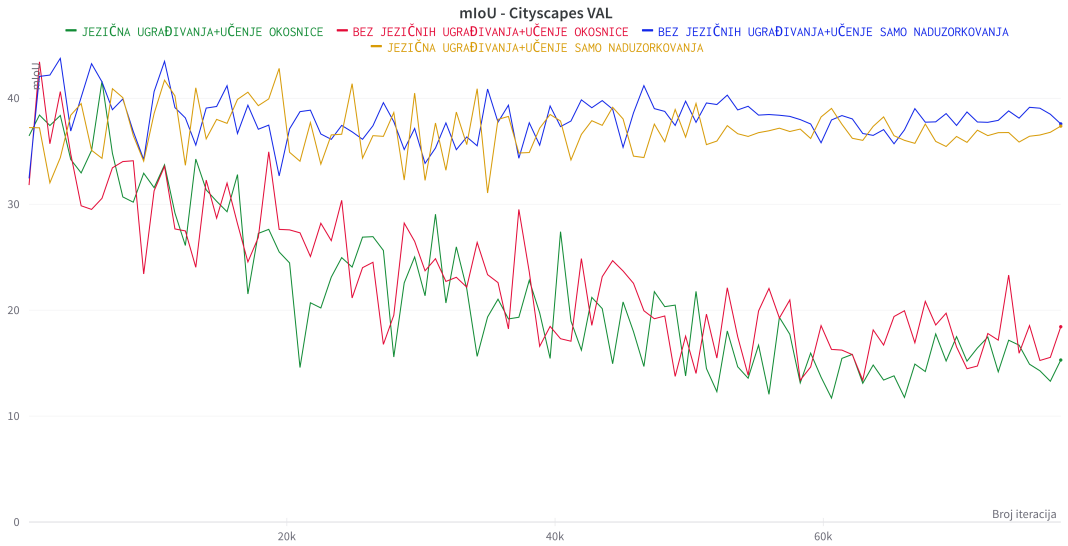
Pokrenuli smo učenje modela 4 puta, a promjene između pokretanja su u tomu koristimo li jezična ugrađivanja, odnosno učimo li okosnicu. Ako učimo okosnicu, za stopu učenja okosnice odabrali smo vrijednost $2\mathrm{e}-5$ (Tablica 7.18). Modele smo učili 100 epoha, a veličina grupe iznosila je 16. Zbog veličine skupa za provjeru, modele smo vrednovali svakih 10 epoha. Naposljetku smo za svako pokretanje odabrali spremljene težine iz epoha gdje smo dobili najbolje performanse na *GTA5* skupu za provjeru, a potom smo te modele vrednovali na *GTA5* skupu za ispitivanje. Prikazali smo i vrijednost mIoU metrike koju tako dobiveni modeli postižu na *Cityscapes* skupu za provjeru. Sve ostale detalje vezane uz postupak učenja opisali smo ranije (odjeljak 7.2.3). Važno je napomenuti kako je u ovim eksperimentima dimenzija puta naduzorkovanja jednaka i u slučaju korištenja, kao i bez korištenja jezičnih ugrađivanja, a u slučaju *Convnext base* okosnice ta dimenzija iznosi 640. Rezultate ovog eksperimenta prikazuje Tablica 7.20.

Okosnica	Učenje okosnice	Veličina isječka	Jezična ugrađivanja	Val mIoU (%) Cityscapes	Test mIoU (%) GTA5
convnext-base-laiona_320	✗	512	✗	37.57	75.44
convnext-base-laiona_320	✓	512	✗	18.43	76.56
convnext-base-laiona_320	✗	512	✓	37.37	75.64
convnext-base-laiona_320	✓	512	✓	15.29	77.95

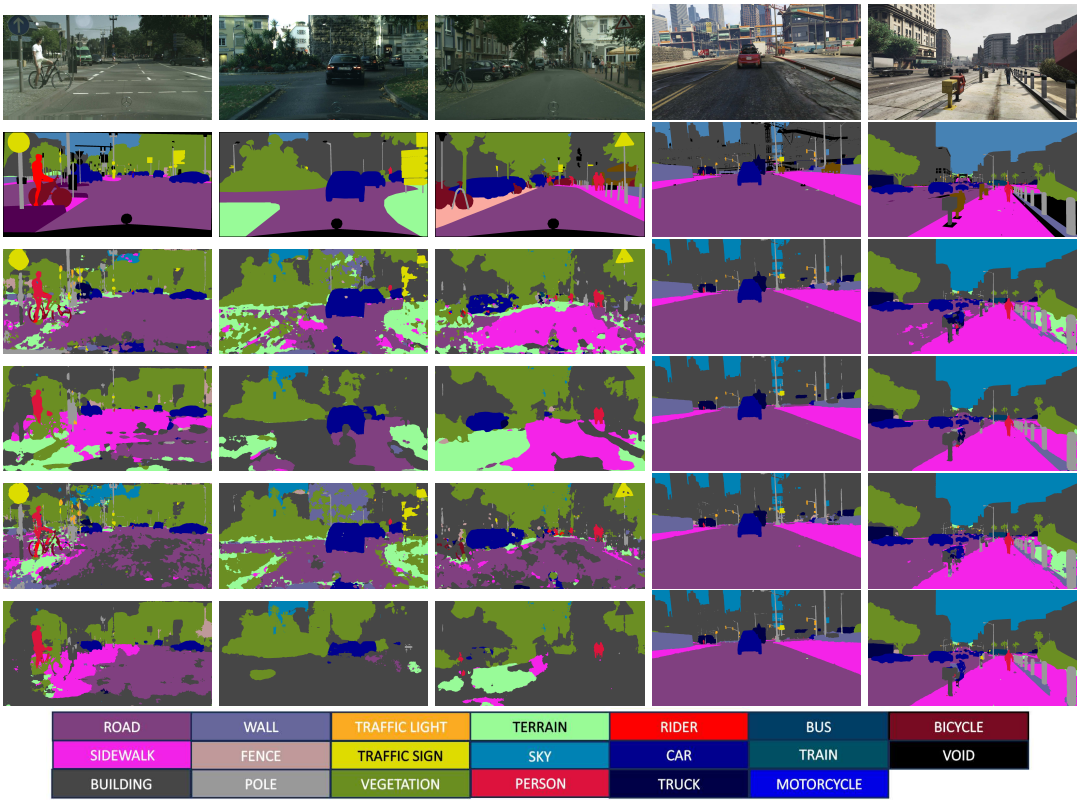
Tablica 7.20: Rezultati semantičke segmentacije korištenjem OpenCLIP *Convnext base* okosnice. Svi detalji vezani uz postupak učenja opisani su u odjeljku 7.2.3. Prikazani rezultati dobiveni su vrednovanjem modela iz najbolje epohe (*GTA5* Val mIoU) na *Cityscapes* podskupu za provjeru te na ispitnom podskupu skupa podataka *GTA5*. Sve eksperimente pokrenuli smo na jednoj grafičkoj kartici NVIDIA A100 s 40 GiB dostupne radne memorije. Veličina grupe u svim eksperimentima jest 16, dok je dimenzija puta naduzorkovanja u svim pokretanjima jednaka dimenziji jezičnih ugrađivanja te iznosi 640.

[18] `https://download.visinf.tu-darmstadt.de/data/from_games/`

Možemo zaključiti kako performanse (u vidu mIoU metrike) na *Cityscapes* skupu za provjeru značajno degradiraju u slučaju učenja okosnice kroz veliki broj epoha. Također, vidimo kako su performanse modela slične s korištenjem ili bez korištenja jezičnih ugrađivanja. Dodatno, uočavamo da učenje okosnice donosi do najviše 2 mIoU boda na ispitnom skupu *GTA5*, a pritom značajno degradira performanse na *Cityscapesu*. Zanimljivo je pogledati kako performanse modela kod kojih učimo okosnicu na *Cityscapes* skupu podataka padaju kroz epohe (Slika 7.12). Učimo li samo put naduzorkovanja, optimiramo približno 13M parametara, dok u slučaju odmrzavanja okosnice učimo dodatnih 88.6M parametara. Možemo zaključiti kako korištenjem jezičnih ugrađivanja nismo degradirali performanse modela, a dobili smo mogućnost učenja bez unaprijed definiranog broja razreda. Želimo li dodati još neku semantičku kategoriju, dovoljno je dodati taj razred u skup naziva semantičkih razreda i ugraditi ga u jezični koder kako bismo dobili konkretno jezično ugrađivanje za dodani razred. Nekoliko kvalitativnih primjera semantičke segmentacije za 4 modela iz Tablice 7.20 prikazuje Slika 7.13.



Slika 7.12: Kretanje mIoU po epohama za eksperimente iz Tablice 7.20. Sve detalje vezane uz postupak učenja opisali smo u odjeljku 7.2.3. Plava krivulja pokazuje kretanje metrike mIoU kroz epohe za model koji ne koristi jezična ugrađivanja i ne uči okosnicu, dok narančasta krivulja prikazuje kretanje metrike mIoU za model koji koristi jezična ugrađivanja bez učenja okosnice. Crvena krivulja predstavlja rezultate bez jezičnih ugrađivanja i s učenjem okosnice, dok zelena krivulja prikazuje dobivene mIoU vrijednosti kroz epohe s korištenjem jezičnih ugrađivanja te s učenjem okosnice.



Slika 7.13: Vizualizacija rezultata semantičke segmentacije s 4 različite konfiguracije modela koje smo naveli u Tablici 7.20. Prvi redak predstavlja ulazne slike, dok drugi redak predstavlja stvarne oznake (engl. *ground truth*). Prva tri stupca slike su iz *Cityscapesa*, a posljednja dva stupca slike su iz *GTA5*. Treći su redak predikcije modela bez učenja okosnice i bez korištenja jezičnih ugrađivanja, potom slijede predikcije modela s učenjem okosnice i bez korištenja jezičnih ugrađivanja. Peti redak predstavlja predikcije modela s korištenjem jezičnih ugrađivanja i bez učenja okosnice, dok se u posljednjem retku uči okosnica te se koriste jezična ugrađivanja.

7.2.4. Dodavanje perturbacija i promjena gubitka

Za razliku od prethodnih eksperimenata u kojima smo koristili jezična ugrađivanja i OpenCLIP *Convnext* okosnicu, u ovom odjeljku koristit ćemo gubitak svjestan granica (odjeljak 3.2.2) i perturbacije (engl. *augmentation*) ulazne slike. Perturbacije koje smo koristili su: rastresanje skale (engl. *scale jittering*), vodoravno zrcaljenje (engl. *random horizontal flip*) te rastresanje bojom (engl. *color jittering*). Ovoga puta modele smo učili 10 epoha na cijelom skupu podataka *GTA5*, a performanse smo vrednovali na *Cityscapes* skupu za provjeru. Koristili smo veličinu grupe 16, početna stopa učenja za put naduzorkovanja u svakom eksperimentu iznosi $4\mathrm{e}-4$, a u slučaju učenja okosnice početna stopa učenja iznosi $2\mathrm{e}{-5}$. Kao i prije (odjeljak 7.2.3) koristimo kosinusno

kaljenje i optimizator Adam [34]. Jezična ugrađivanja za svaki razred dobili smo umetanjem naziva razreda u 85 tekstualnih predložaka i uprosječivanjem izlaza jezičnog kodera. U ovom odjeljku koristili smo i višerezolucijsku inačicu *SwiftNeta* [48] (odjeljak 3.2.2). U svim eksperimentima koristili smo rastresanje skale (slučajan broj iz intervala $[0.5, 2]$) i vodoravno zrcaljenje ($p = 0.5$). U eksperimentima u kojima koristimo rastresanje boje sve parametre odabiremo slučajno iz intervala $[0, 1]$. U svim eksperimentima koristimo OpenCLIP *Convnext* [43] okosnicu, a eksperimentirali smo s *base* i *large* verzijama modela. Rezultate provedenih eksperimenata prikazuje Tablica 7.21.

Okosnica	Učenje okosnice	Put naduzorkovanja	Jezična ugrađivanja	Rastresanje boje	Uniformno uzorkovanje	Preciznost operacija	Val mIoU(%) Cityscapes
laiona-base	✗	*SwiftNet*-ss-640	✗	✗	✗	16	46.07
laiona-base	✗	*SwiftNet*-ss-640	✗	✗	✗	32	48.76
laiona-base	✗	*SwiftNet*-ss-640	✓	✗	✗	16	49.19
laiona-base	✗	*SwiftNet*-ss-640	✓	✗	✗	32	50.55
laiona-base	✗	*SwiftNet*-pyr-640	✗	✗	✗	16	47.66
laiona-base	✗	*SwiftNet*-pyr-640	✗	✗	✗	32	47.13
laiona-base	✗	*SwiftNet*-pyr-640	✓	✗	✗	16	48.44
laiona-base	✗	*SwiftNet*-pyr-640	✓	✗	✗	32	47.54
laiona-base	✓	*SwiftNet*-ss-640	✓	✗	✗	16	38.44
laiona-base	✗	*SwiftNet*-ss-640	✓	✗	✓	16	51.55
laiona-base	✗	*SwiftNet*-ss-640	✓	✓	✗	16	50.29
laiona-base	✗	*SwiftNet*-ss-640	✓	✓	✓	16	51.22
laiona-base	✗	*SwiftNet*-ss-640	✓	✓	✓	32	52.10
laion2b-large	✗	*SwiftNet*-ss-768	✓	✗	✗	16	52.06
laion2b-large	✗	*SwiftNet*-ss-768	✓	✗	✗	32	52.61
laion2b-large	✓	*SwiftNet*-ss-768	✓	✗	✗	16	46.52
laion2b-large	✗	*SwiftNet*-ss-768	✓	✓	✓	16	**54.11**
laion2b-large	✗	*SwiftNet*-ss-768	✓	✓	✓	32	53.15

Tablica 7.21: Eksperimenti s dodanim perturbacijama ulazne slike i s dodavanjem gubitka svjesnog granica (odjeljak 3.2.2) pri rješavanju zadatka semantičke segmentacije. U svakom eksperimentu koristimo OpenCLIP *Convnext* okosnicu, a prvi stupac predstavlja verziju korištene okosnice. Sve modele učili smo s veličinom grupe 16 kroz 10 epoha. Sve eksperimente pokrenuli smo na dvije grafičke kartice NVIDIA A100 s 40GiB dostupne radne memorije. U svim eksperimentima koristimo rastresanje skale (slučajan broj iz intervala $[0.5, 2]$) te vodoravno zrcaljenje. Dobiveni rezultati u smislu mIoU metrike dobiveni su vrednovanjem modela iz posljednje epohe učenja. Oznaka *SwiftNet-ss-640* označava korištenje jednorezolucijskog *SwiftNeta* [48] s dimenzijom puta naduzorkovanja 640, dok oznaka *SwiftNet-pyr-640* označava korištenje višerezolucijskog *SwiftNeta* s piramidom značajki, pri čemu je dimenzija puta naduzorkovanja jednaka 640. U eksperimentima u kojima koristimo rastresanje boje sve parametre odabiremo slučajno iz intervala $[0, 1]$. U svim eksperientima dimenzija isječaka na kojima učimo naše modele iznosi 1024. Modele učimo na cijelom skupu podataka *GTA5* [63].

Iz Tablice 7.21 zanimljivo je uočiti kako jednorezolucijski *SwiftNet* [48] ostvaruje bolje performanse od višerezolucijskog *SwiftNeta* s piramidom značajki. Osim toga, možemo vidjeti kako učenje s jezičnim ugrađivanjima doprinosi rezultatima u svakom od pokretanja. Dodatno, možemo zaključiti kako dodavanje uniformnog uzorkovanja razreda prilikom učenja poboljšava performanse u vidu mIoU metrike. Dodavanje rastresanja boje nije doprinijelo performansama modela. Shodno prethodnim eksperimentima možemo zaključiti kako učenje okosnice, čak i s blagom stopom učenja $(2\mathrm{e}-5)$, značajno pogoršava performanse na *Cityscapes* skupu za provjeru. Konačno, uočavamo kako smo korištenjem *large* verzije *Convnext* okosnice dobili na performansama. Najbolji rezultat postigli smo korištenjem *large* okosnice, jezičnih ugrađivanja, uniformnog uzorkovanja i rastresanja boje. Pritom su parametri okosnice bili smrznuti.

Dodatno, proveli smo eksperiment u kojemu koristimo jezična ugrađivanja pri rješavanju zadatka semantičke segmentacije u smislu generalizacije domene *Cityscapes→ACDC*. Model učimo na skupu stvarnih scena vožnje iz *Cityscapesa*, a vrednujemo na stvarnim scenama vožnje u teškim vremenskim uvjetima (*ACDC*). Proveli smo eksperiment s korištenjem i bez korištenja jezičnih ugrađivanja, a rezultate provedenih eksperimenata prikazuje Tablica 7.22. Uočavamo kako nismo dobili poboljšanje korištenjem jezičnih ugrađivanja.

Okosnica	Učenje okosnice	Put naduzorkovanja	Jezična ugrađivanja	Preciznost operacija	Val mIoU(%) ACDC
laiona-base	✗	*SwiftNet*-pyr-640	✗	32	53.96
laiona-base	✗	*SwiftNet*-pyr-640	✓	32	53.84

Tablica 7.22: Eksperimenti s dodanim perturbacijama ulazne slike i s dodavanjem gubitka svjesnog granica (odjeljak 3.2.2) pri rješavanju zadatka semantičke segmentacije. U oba eksperimenta koristimo OpenCLIP *Convnext base* okosnicu. Oba modela učili smo s veličinom grupe 16 kroz 50 epoha. Oba eksperimenta pokrenuli smo na dvije grafičke kartice NVIDIA V100 s 32GiB dostupne radne memorije. U svim eksperimentima koristimo rastresanje skale (slučajan broj iz intervala $[0.5, 2]$) te vodoravno zrcaljenje. Dobiveni rezultati u smislu mIoU metrike dobiveni su vrednovanjem modela iz posljednje epohe učenja. Oznaka *SwiftNet-pyr-640* označava korištenje višerezolucijskog *SwiftNeta* [48] s piramidom značajki, pri čemu je dimenzija puta naduzorkovanja jednaka 640. U oba eksperimenta dimenzija isječaka na kojima učimo naše modele iznosi 1024. Modele učimo na cijelom skupu podataka *Cityscapes*, a vrednujemo na *ACDC* [70] skupu za provjeru.

7.3. Nenadzirana prilagodba domene

U ovom odjeljku bavimo se reprodukcijom stanja tehnike u području prilagodbe domene. Konkretno, radi se o metodama *HRDA* [27] i *MIC* [29]. Obje spomenute metode nastale su na temelju metode *DAFormer* [28]. Ove metode svoju efikasnost temelje na korištenju okosnice *SegFormer* [84], a u okviru ovoga odjeljka proveli smo eksperimente koristeći *Convnext* [43] i *Convnext v2* [81] okosnice te smo prikazali usporedbu rezultata. Svi dobiveni rezultati odnose se na prilagodbu domene s umjetnih na stvarne slike iz vožnje *GTA5→Cityscapes*.

7.3.1. *HRDA* - reprodukcija rezultata

U ovom odjeljku bavimo se reprodukcijom rezultata *HRDA* [27] metode. Prvo smo reproducirali eksperiment prilagodbe domene koristeći koder *SegFormera* [84] pod nazivom *MiT-B5* kao okosnicu *HRDA* pristupa. Osim toga, pokrenuli smo eksperimente gdje smo kao okosnicu koristili *Convnext* [43] modele prednaučene na različitim skupovima podataka. Tablica 7.23 prikazuje rezultate provedenih eksperimenata.

Implementacijski detalji

Za slike ciljne domene koristimo skup podataka *Cityscapes* [11], a slike izvorne domene slike su iz igrice GTA5 [63]. Arhitektura se temelji na *DAFormeru* [28]. Osim okosnice *MiT-B5* prednaučene na *ImageNet-1k* skupu podataka koja se koristi u orignalnom radu, isprobali smo i *Convnext* okosnice prednaučene na *ImageNet-1k*, *ImageNet-22k* te *LAION-5B* [72] skupovima podataka. Kao dekoder u svim eksperimentima koristimo lagani dekoder svjestan konteksta (engl. *lightweight context-aware decoder*) [28], a dimenzija ugrađivanja dekodera iznosi 256. Konfiguracija učenja preuzeta je iz članka [27]. Koristimo samonadzirani pristup nenadziranoj prilagodbi domene [28]. Kao optimizator koristimo AdamW [46] s početnom stopom učenja $6\mathrm{e}{-5}$ za koder te $6\mathrm{e}{-4}$ za dekoder, a veličina grupe je 2. Koristi se zagrijavanje stope učenja kao što je opisano u 5.3, $\alpha = 0.999$ (5.5) te $\tau = 0.968$ (5.3). Za perturbacije ulaznih slika pratili smo metodu *DACS* [78]. Dimenzija isječka konteksta nakon poduzorkovanja jednake su dimneziji detaljnog isječka i iznose 512×512, a faktor poduzorkovanja jest 2.

Analiza rezultata

	Okosnica	Prednaučeno	Broj iteracija	Udaljenost značajki	Cityscapes mIoU (%)
1	*MiT-B5* [84]	*ImageNet-1k*	40000	✓	73.79
2	*MiT-B5* [84]	*ImageNet-1k*	40000	✓	73.81
3	*Convnext base*	*ImageNet-1k*	40000	✓	68.57
4	*Convnext base*	*ImageNet-1k*	120000	✓	71.60
5	*Convnext base*	*ImageNet-22k*	40000	✓	73.65
6	*Convnext base*	*LAION-A*	40000	✗	63.22
7	*Convnext base*	*LAION-A*	40000	✓	65.32
8	*Convnext base*	*LAION-A*	120000	✓	64.72
9	*Convnext base*	*LAION-A* *ImageNet-12k*	40000	✓	67.14

Tablica 7.23: Reprodukcija rezultata *HRDA* [27] metode. Prvi redak prikazuje rezultate preuzete iz originalnog članka [27], a drugi redak prikazuje reprodukciju tog rezultata korištenjem istih parametara kao u originalnom članku. Drugi stupac prikazuje skupove podataka korištene u fazi predučenja. Četvrti stupac govori koristimo li udaljenost značajki od prednaučenog modela pri računanju gubitka (5.3). Svi implementacijski detalji vezani uz postupak učenja jednaki su onima iz odjeljka 7.3.1. *ImageNet-1k* prednaučeni modeli prednaučeni su na rezoluciji 224, *ImageNet-22k* na rezoluciji 384, dok su *LAION* modeli prednaučeni na rezoluciji 320. Sve eksperimente pokretali smo na jednoj grafičkoj kartici NVIDIA RTX 3090 s 24GiB dostupne radne memorije. Izvorna je domena *GTA5*, a ciljna *Cityscapes*.

Prvi zaključak iz Tablice 7.23 je da smo uspješno reproducirali rezultate originalnog članka [27]. Zanimljivo je kako promjenom korištene okosnice iz *MiT-B5* u *Convnext base*, a zadržavanjem skupa za predučenje *ImageNet-1k* performanse padaju za više od 5 mIoU bodova. Taj jaz u rezultatima uspjeli smo smanjiti na 2 mIoU boda povećavanjem broja iteracija učenja s 40000 na 120000. Korištenjem skupa *ImageNet-22k* gotovo smo se izjednačili s rezultatima iz članka [27], što nam govori da predučenje s većom rezolucijom i na većem broju *ImageNet* slika može poboljšati performanse. Iznenađujuće je da korištenje *LAION* [72] skupa podataka kao skupa za predučenje nije doprinijelo rezultatima, kao što možemo vidjeti u retcima 6 i 7 Tablice 7.23. Ipak, isključivanjem korištenja udaljenosti značajki od prednaučenog modela performanse su degradirale za 2 mIoU boda, što znači da značajke modela prednaučenog na *LAION* skupu podataka nose znanje korisno za zadatak prilagodbe domene. Nadalje, povećanje broja iteracija učenja modela prednaučenog na *LAION* skupu podataka nije poboljšalo rezultate.

7.3.2. *MIC* - reprodukcija rezultata

Osim reprodukcije rezultata za metodu *HRDA*, rezultate smo reproducirali i za metodu *MIC* [29] koju smo opisali u odjeljku 5.5. Osim reprodukcije, slično kao i prije, proveli smo eksperimente s *Convnext* okosnicom. Za razliku od prethodnog odjeljka, eksperimente smo proveli i s *Convnext v2* okosnicom, ali i s *large* verzijama modela.

Implementacijski detalji

Za ciljnu domenu ponovno koristimo slika *Cityscapesa* [11], dok izvornu domenu predstavljaju slike iz *GTA5* [63]. S obzirom da se *MIC* nastavlja na metodu *HRDA*, svi detalji vezani uz učenje jednaki su opisanom u odjeljku 7.3.1. Dodatno, vjerojatnost kojom smo maskirali isječke slike ciljne domene iznosila je 0.7, dok je veličina maskiranih regija iznosila 64×64, a te vrijednosti preuzeli smo iz originalnog rada [29].

Analiza rezultata

Iz Tablice 7.24 prvo zaključujemo kako smo uspješno reproducirali rezultat iz originalnog članka [29]. Usporedimo li redove 2 i 4 Tablice 7.24 vidimo kako je inicijalizacija okosnice modelom koji je prednaučen na *ImageNetu* ključna, s obzirom da smo, bez promjene drugih parametara, inicijalizacijom prednaučenim težinama dobili 40 mIoU bodova. Dodatno, usporedimo li redove 2 i 3 Tablice 7.24, možemo zaključiti da smo povećanjem broja iteracija 3 puta dobili na performansama u vidu 15 mIoU bodova. Nadalje, uočavamo kako korištenje *Convnext base* i *Convnext base v2* okosnica nije uspjelo nadmašiti originalne rezultate, neovisno o inicijalizaciji. Zanimljivo je da *Convnext v2 large* okosnica daje značajno lošije razultate, čak i od *Convnext v2 base* okosnice. Iz redaka 5 i 6 odnosno 9 i 10 Tablice 7.24 možemo uočiti kako korištenje preklapajućih prozora pri generiranju pseudooznaka i vrednovanju modela nije presudno.

	Okosnica	Prednaučeno	Broj iteracija	Pomični prozor	Cityscapes mIoU (%)
1	*MiT-B5* [84]	*ImageNet-1k*	40000	✓	75.93
2	*MiT-B5* [84]	✗	40000	✓	35.15
3	*MiT-B5* [84]	✗	120000	✓	49.66
4	*MiT-B5* [84]	*ImageNet-1k*	40000	✓	75.78
5	*Convnext base*	*ImageNet-22k*	40000	✓	72.14
6	*Convnext base*	*ImageNet-22k*	40000	✗	73.29
7	*Convnext v2 base*	*ImageNet-22k*	40000	✗	67.38
8	*Convnext base*	*LAION-A ImageNet-12k*	40000	✗	68.55
9	*Convnext large*	*ImageNet-22k*	40000	✓	74.32
10	*Convnext large*	*ImageNet-22k*	40000	✗	74.05
11	*Convnext v2 large*	*ImageNet-22k*	40000	✗	58.16
12	*Convnext large*	*LAION-2B ImageNet-12k*	40000	✗	75.83

Tablica 7.24: Reprodukcija rezultata *MIC* [29] metode. Prvi redak prikazuje rezultate preuzete iz originalnog članka [29], a četvrti redak prikazuje reprodukciju tog rezultata korištenjem istih parametara kao u originalnom članku. Drugi stupac prikazuje skupove podataka korištene u fazi predučenja. Četvrti stupac govori koristimo li preklapajući pomični prozor (engl. *sliding window*) pri generiranju pseudooznaka. Svi implementacijski detalji vezani uz postupak učenja jednaki su onima iz odjeljka 7.3.2. *ImageNet-1k* prednaučeni modeli prednaučeni su na rezoluciji 224, *ImageNet-22k* na rezoluciji 384, dok su *LAION* modeli prednaučeni na rezoluciji 320. Sve eksperimente pokretali smo na jednoj grafičkoj kartici NVIDIA RTX A6000 s 49GiB dostupne radne memorije. Izvorna je domena *GTA5*, a ciljna *Cityscapes*.

7.3.3. Generalizacija domene

Za razliku od zadatka prilagodbe domene, kod generalizacije domene za vrijeme učenja nemamo pristup slikama ciljne domene. Cilj je naučiti model na izvornoj domeni, a vrednovati na slikama ciljne domene te tako dobiti uvid u sposobnost generalizacije modela. Pokrenuli smo eksperimente s okosnicama *MiT-B5* [84], *Convnext base* [43] te *Convnext base v2* [81], a dobivene rezultate prikazuje Tablica 7.25. Što se tiče implementacijskih detalja, jednaki su kao u odjeljku 7.3.2, ali pritom valja naglasiti da više ne koristimo slike ciljne domene, odnosno ne generiramo pseudooznake. Važno je naglasiti kako i dalje koristimo maskiranje regija, ali ovoga puta maskiramo slike izvorne, a ne ciljne domene. Iz Tablice 7.25 možemo zaključiti kako su performanse značajno lošije u usporedbi s eksperimentima prilagodbe domene koje smo prikazali u Tablicama 7.23 i 7.24. Takav rezultat ima smisla s obzirom da model uči samo

na slikama izvorne domene, bez učenja na pseudooznakama ciljne domene. Također je zanimljivo kako, za razliku od prilagodbe domene, *Convnext* okosnice nadmašuju rezultate dobivene koderom *MiT-B5*.

	Okosnica	Prednaučeno	Broj iteracija	Pomični prozor	Cityscapes mIoU (%)
1	*MiT-B5* [84]	*ImageNet-1k*	40000	✓	48.71
2	*Convnext base*	*ImageNet-1k*	40000	✗	52.18
2	*Convnext base*	*ImageNet-22k*	40000	✗	55.11
3	*Convnext v2 base*	*ImageNet-22k*	40000	✗	50.24
4	*Convnext base*	*LAION-A ImageNet-12k*	40000	✗	42.19

Tablica 7.25: Rezultati metode *MIC* na zadatku generalizacije domene *GTA5→Cityscapes*. Drugi stupac prikazuje skupove podataka korištene u fazi predučenja. Četvrti stupac govori koristimo li preklapajući pomični prozor (engl. *sliding window*) pri generiranju pseudooznaka. Svi implementacijski detalji vezani uz postupak učenja jednaki su onima iz odjeljka 7.3.2, ali više ne koristimo slike ciljne domene. *ImageNet-1k* prednaučeni modeli prednaučeni su na rezoluciji 224, *ImageNet-22k* na rezoluciji 384, dok su *LAION* modeli prednaučeni na rezoluciji 320. Sve eksperimente pokretali smo na jednoj grafičkoj kartici NVIDIA RTX A6000 s 49GiB dostupne radne memorije. Izvorna je domena *GTA5*, a ciljna *Cityscapes*.

8. Zaključak

Rad se sastoji od 3 glavna dijela, a poveznica između ta tri dijela rješavanje je zadatka semantičke segmentacije scena vožnje. Prvi dio rada bavi se razvojem modela za obavljanje semantičke segmentacije u teškim vremenskim uvjetima, a motivacija iza ovog dijela rada bilo je sudjelovanje na natjecanju semantičke percepcije u teškim vremenskim uvjetima pod nazivom ACDC2023. Metoda se temelji na učenju modela na velikom skupu označenih i pseudooznačenih slika. Konačni model ostvario je drugi rezultat na službenom poretku ACDC2023 natjecanja. Budući rad u ovom dijelu uključivao bi učenje još jedne iteracije modela s poboljšanim pseudooznakama, kao i eksperimentiranje s novijim arhitekturama koje se temelje na klasifikaciji maski [7, 32].

U drugom dijelu rada bavimo se primjenom latentnih jezičnih reprezentacija za rješavanje zadatka semantičke segmentacije. Cilj ovog dijela rada usporediti je modele prednaučene kontrastno na parovima slika i odgovarajućih opisa s modelima prednaučenima na *ImageNetu*. Modele smo učili na skupu slika dobivenima iz računalne igrice, a vrednovali na stvarnim slikama iz vožnje. Primijetili smo kako povećanje dimenzija isječaka na kojima učimo modele utječe na sposobnsot generalizacije. Dodatno, najbolje rezultate vrednovanjem na stvarnim slikama ostvario je model koji je koristio jezična ugrađivanja i velike dimenzije slučajnih isječaka za učenje. Budući rad u ovom dijelu uključivao bi dodavanje pseudooznaka stvarnih slika, odnosno ciljne domene, u postupak učenja modela. Zanimljiva smjernica za budući rad bila bi uključivanje uzorkovanja rijetkih razreda u postupak učenja, kao i uključivanje zagrijavanja stope učenja [28].

Treći dio rada bavi se reprodukcijom rezultata stanja tehnike [27, 29] iz područja nenadzirane prilagodbe domene za zadatak semantičke segmentacije. Prilagodba domene odnosi se pritom na učenje modela na slikama iz računalne igrice GTA5 [63], a vrednovanje na stvarnim slikama scena vožnje [11]. Osim same reprodukcije, eksperimentirali smo s konvolucijskim okosnicama umjesto originalno korištenih okosnica

koje se temelje na *Transformeru*. Rezultate [27, 29] uspješno smo reproducirali, a zanimljivo je da umetanjem konvolucijskih okosnica nismo uspjeli nadmašiti originalne rezultate, čak i korištenjem okosnica prednaučenih na značajno većim skupovima podataka. Budući rad u ovom dijelu uključuje ugrađivanje latentnih jezičnih reprezentacija, za koje se nadamo da mogu pomoći robusnosti, u trenutno stanje tehnike [29] u području nenadzirane prilagodbe domene za zadatak semantičke segmentacije scena vožnje.

LITERATURA

[1] Ethem Alpaydin. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 3 izdanju, 2014. ISBN 978-0-262-02818-9.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, i Geoffrey E. Hinton. Layer normalization, 2016.

[3] Vijay Badrinarayanan, Alex Kendall, i Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, i Yoshua Bengio. Neural machine translation by jointly learning to align and translate. U Yoshua Bengio i Yann LeCun, urednici, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`.

[5] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, i Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather, 2020.

[6] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, i Alan L. Yuille. Attention to scale: Scale-aware semantic image segmentation. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, i Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 1290–1299, June 2022.

[8] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, i Jenia Jitsev.

Reproducible scaling laws for contrastive language-image learning. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 2818–2829, June 2023.

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, i Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, i Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, i Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, i Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. U H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, i H. Lin, urednici, *Advances in Neural Information Processing Systems*, svezak 33, stranice 18613–18624. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, i Li Fei-Fei. Imagenet: A large-scale hierarchical image database. U *2009 IEEE Conference on Computer Vision and Pattern Recognition*, stranice 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, i Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. U *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, stranice 4171–4186, Minneapolis, Minnesota, Lipanj 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, i Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. U *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[16] Clement Farabet, Camille Couprie, Laurent Najman, i Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013. doi: 10.1109/TPAMI.2012.231.

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, i Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

[18] Alex Graves. Generating sequences with recurrent neural networks, 2014.

[19] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, i Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. U *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=lL3lnMbR4WU.

[20] Zellig S. Harris. Distributional structure, WORD,10:2-3, 146.1652, 1954. URL https://doi.org/10.1080/00437956.1954.11659520.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. U *Computer Vision – ECCV 2014*, stranice 346–361. Springer International Publishing, 2014. doi: 10.1007/978-3-319-10578-9_23. URL https://doi.org/10.1007%2F978-3-319-10578-9_23.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Identity mappings in deep residual networks, 2016.

[24] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, i Ross Girshick. Masked autoencoders are scalable vision learners. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 16000–16009, June 2022.

[25] Dan Hendrycks i Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[26] Sepp Hochreiter i Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[27] Lukas Hoyer, Dengxin Dai, i Luc Van Gool. Hrda: Context-aware high-resolution domain-adaptive semantic segmentation, 2022.

[28] Lukas Hoyer, Dengxin Dai, i Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 9924–9935, June 2022.

[29] Lukas Hoyer, Dengxin Dai, Haoran Wang, i Luc Van Gool. Mic: Masked image consistency for context-enhanced domain adaptation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 11721–11732, June 2023.

[30] Xiaowei Hu, Chi-Wing Fu, Lei Zhu, i Pheng-Ann Heng. Depth-attentional features for single-image rain removal. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[31] Sergey Ioffe i Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. U Francis Bach i David Blei, urednici, *Proceedings of the 32nd International Conference on Machine Learning*, svezak 37 od *Proceedings of Machine Learning Research*, stranice 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/ioffe15.html`.

[32] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, i Humphrey Shi. Oneformer: One transformer to rule universal image segmentation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 2989–2998, June 2023.

[33] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, i Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. U *International Conference on Machine Learning*, 2021.

[34] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization. U Yoshua Bengio i Yann LeCun, urednici, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[35] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, i Piotr Dollar. Panoptic segmentation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[36] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, i Ross Girshick. Segment anything, 2023.

[37] Ivan Krešo, Josip Krapac, i Siniša Šegvić. Efficient ladder-style densenets for semantic segmentation of large images, 2019.

[38] Alex Krizhevsky, Ilya Sutskever, i Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. U F. Pereira, C.J. Burges, L. Bottou, i K.Q. Weinberger, urednici, *Advances in Neural Information Processing Systems*, svezak 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[39] Y. Lecun, L. Bottou, Y. Bengio, i P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[40] Boyi Li, Kilian Q. Weinberger, Serge Belongie, Vladlen Koltun, i René Ranftl. Language-driven semantic segmentation, 2022.

[41] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, i Piotr Dollar. Focal loss for dense object detection. U *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[42] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, i Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. U *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, stranice 10012–10022, October 2021.

[43] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, i Saining Xie. A convnet for the 2020s. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 11976–11986, June 2022.

[44] Jonathan Long, Evan Shelhamer, i Trevor Darrell. Fully convolutional networks for semantic segmentation. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[45] Ilya Loshchilov i Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. U *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Skq89Scxx`.

[46] Ilya Loshchilov i Frank Hutter. Decoupled weight decay regularization, 2019.

[47] Timo Lüddecke i Alexander Ecker. Image segmentation using text and image prompts. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 7086–7096, June 2022.

[48] Marin Oršić and Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition*, 110:107611, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107611. URL `https://www.sciencedirect.com/science/article/pii/S0031320320304143`.

[49] Tomás Mikolov, Kai Chen, Greg Corrado, i Jeffrey Dean. Efficient estimation of word representations in vector space. U Yoshua Bengio i Yann LeCun, urednici, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL `http://arxiv.org/abs/1301.3781`.

[50] Vinod Nair i Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. U *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, stranica 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[51] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, i Ming-Hsuan Yang. Intriguing properties of vision transformers. U M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, i J. Wortman Vaughan, urednici, *Advances in Neural Information Processing Systems*, svezak 34, stranice 23296–23308. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/c404a5adbf90e09631678b13b05d9d7a-Paper.pdf.

[52] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, i Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. U *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[53] Lukás Neumann, Michelle Karg, Shanshan Zhang, Christian Scharfenberger, Eric Piegert, Sarah Mistr, Olga Prokofyeva, Robert Thiel, Andrea Vedaldi, Andrew Zisserman, i Bernt Schiele. Nightowls: A pedestrians at night dataset. U *Asian Conference on Computer Vision*, 2018.

[54] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, i Lennart Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. U *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, stranice 1369–1378, January 2021.

[55] Sayak Paul i Pin-Yu Chen. Vision transformers are robust learners, 2021.

[56] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, Mingxing Tan, i Quoc V. Le. Combined scaling for zero-shot transfer learning, 2023.

[57] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, i Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40 (4-5):681–690, dec 2020. doi: 10.1177/0278364920979368. URL https://doi.org/10.1177%2F0278364920979368.

[58] Alec Radford i Karthik Narasimhan. Improving language understanding by generative pre-training, 2018.

[59] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, i Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[60] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, i Ilya Sutskever. Learning transferable visual models from natural language supervision. U Marina Meila i Tong Zhang, urednici, *Proceedings of the 38th International Conference on Machine Learning*, svezak 139 od *Proceedings of Machine Learning Research*, stranice 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

[61] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, i Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[62] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, i Tapani Raiko. Semi-supervised learning with ladder networks, 2015.

[63] Stephan R. Richter, Vibhav Vineet, Stefan Roth, i Vladlen Koltun. Playing for data: Ground truth from computer games, 2016.

[64] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[65] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, i Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[66] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, i Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, dec 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

[67] Christos Sakaridis, Dengxin Dai, i Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, 126 (9):973–992, mar 2018. doi: 10.1007/s11263-018-1072-8. URL https://doi.org/10.1007%2Fs11263-018-1072-8.

[68] Christos Sakaridis, Dengxin Dai, Simon Hecker, i Luc Van Gool. Model adaptation with synthetic and real data for semantic dense foggy scene understanding. U *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[69] Christos Sakaridis, Dengxin Dai, i Luc Van Gool. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. U *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[70] Christos Sakaridis, Dengxin Dai, i Luc Van Gool. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. U *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, stranice 10765–10775, October 2021.

[71] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, i Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs, 2021.

[72] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, i Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. U *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=M3Y74vmsMcY`.

[73] Ilya Sutskever, Oriol Vinyals, i Quoc V. Le. Sequence to sequence learning with neural networks, 2014. URL `https://arxiv.org/abs/1409.3215`.

[74] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, i Zbigniew Wojna. Rethinking the inception architecture for computer vision. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[75] Xin Tan, Ke Xu, Ying Cao, Yiheng Zhang, Lizhuang Ma, i Rynson W. H. Lau. Night-time scene parsing with a large real dataset. *IEEE Transactions on Image Processing*, 30:9085–9098, 2021. doi: 10.1109/tip.2021.3122004. URL `https://doi.org/10.1109%2Ftip.2021.3122004`.

[76] Antti Tarvainen i Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. U I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, i R. Garnett, urednici, *Advances in Neural Information Processing Systems*, svezak 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/68053af2923e00204c3ca7c6a3150cf7-Paper.pdf.

[77] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, i Herve Jegou. Training data-efficient image transformers & distillation through attention. U Marina Meila i Tong Zhang, urednici, *Proceedings of the 38th International Conference on Machine Learning*, svezak 139 od *Proceedings of Machine Learning Research*, stranice 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

[78] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, i Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. U *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, stranice 1379–1389, January 2021.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, i Illia Polosukhin. Attention is all you need. U I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, i R. Garnett, urednici, *Advances in Neural Information Processing Systems*, svezak 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[80] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, i Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, 2021.

[81] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, i Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 16133–16142, June 2023.

[82] Chenyun Wu, Zhe Lin, Scott Cohen, Trung Bui, i Subhransu Maji. Phrasecut: Language-based image segmentation in the wild. U *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[83] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, i Jian Sun. Unified perceptual parsing for scene understanding. U *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[84] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, i Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. U M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, i J. Wortman Vaughan, urednici, *Advances in Neural Information Processing Systems*, svezak 34, stranice 12077–12090. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/64f1f27bf1b4ec22924fd0acb550c235-Paper.pdf`.

[85] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, i Kaiming He. Aggregated residual transformations for deep neural networks. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[86] Wei Yin, Yifan Liu, Chunhua Shen, Anton van den Hengel, i Baichuan Sun. The devil is in the labels: Semantic segmentation from sentences, 2022.

[87] Fisher Yu i Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. U Yoshua Bengio i Yann LeCun, urednici, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.07122`.

[88] Fisher Yu, Vladlen Koltun, i Thomas Funkhouser. Dilated residual networks. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[89] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, i Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.

[90] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, i Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with

localizable features. U *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[91] Oliver Zendel, Matthias Schörghuber, Bernhard Rainer, Markus Murschitz, i Csaba Beleznai. Unifying panoptic segmentation for autonomous driving. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 21351–21360, June 2022.

[92] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, i Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 18123–18133, June 2022.

[93] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, i David Lopez-Paz. mixup: Beyond empirical risk minimization. U *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1Ddp1-Rb`.

[94] Mingmin Zhen, Jinglu Wang, Lei Zhou, Tian Fang, i Long Quan. Learning fully dense neural networks for image semantic segmentation, 2019.

[95] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, i Yi Yang. Random erasing data augmentation, 2017.

[96] Chong Zhou, Chen Change Loy, i Bo Dai. Extract free dense labels from clip, 2022.

[97] Ziqin Zhou, Yinjie Lei, Bowen Zhang, Lingqiao Liu, i Yifan Liu. Zegclip: Towards adapting clip for zero-shot semantic segmentation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 11175–11185, June 2023.

[98] Čala, Rino and Martinović, Ivan. Implementacija konverzacijskih modela s različitim crtama osobnosti, 2022.

[99] Siniša Šegvić. Predavanja iz kolegija Duboko učenje 1, UniZG, FER, 2023. URL `http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf`.

Primjena jezičnih ugrađivanja za semantičku segmentaciju

Sažetak

Semantička segmentacija zadatak je iz područja računalnoga vida koji za cilj ima odrediti pripadnost svakog piksela slike odgovarajućem semantičkom razredu. Rad se sastoji od 3 dijela, a poveznica između ta tri dijela rješavanje je zadatka semantičke segmentacije scena vožnje. U prvom dijelu rada bavimo se semantičkom segmentacijom scena vožnje u teškim vremenskim uvjetima. Korištenjem skupa podataka za učenje koji se sastojao od slika iz više dostupnih skupova podataka te pseudooznačavanjem neoznačenih slika naučili smo model koji je ostvario drugi rezultat na ACDC2023 natjecanju. Drugi dio rada bavi se semantičkom segmentacijom primjenom jezičnih ugrađivanja, a cilj ovog dijela vidjeti je pomaže li korištenje jezičnih ugrađivanja generalizaciji. Modele smo učili na umjetnom skupu slika vožnje koje su dobivene iz računalne igrice GTA5, a vrednovali na stvarnim slikama vožnje. Dobiveni rezultati ukazuju da korištenje jezičnih ugrađivanja može poboljšati performanse modela na slikama iz stvarne domene. Konačno, treći dio rada bavi se semantičkom segmentacijom scena vožnje u okviru nenadziranog prilagođavanja domene. Reproducirali smo rezultate koji predstavljaju stanje tehnike te eksperimentirali s različitim verzijama konvolucijskih okosnica. Korištenje konvolucijskih okosnica nije uspjelo nadmašiti originalno korištenje arhitekture *Transformer* na zadatku prilagođavanja domene, iako su korištene konvolucijske okosnice prednaučene na puno većim skupovima podataka.

Ključne riječi: duboko učenje, računalni vid, semantička segmentacija, jezična ugrađivanja u računalnom vidu, adaptacija domene, generalizacija domene, natjecanje ACDC

Application of Language Embeddings for Semantic Segmentation

Abstract

Semantic segmentation is a computer vision task that aims to assign a class label to every pixel in an input image. This thesis is divided into three main parts, all connected by the goal of solving semantic segmentation for driving scenes. In the first part, we address the task of semantic segmentation for driving scenes under adverse weather conditions. We collected images from diverse datasets and applied pseudo-labeling to unlabeled images. The model trained on this dataset achieved the second-best results in the ACDC2023 challenge. In the second part of the thesis, the investigation revolves around the application of language embeddings for semantic segmentation. The primary goal is to explore whether incorporating language embeddings facilitates domain generalization and contributes to the development of more robust models. Synthetic images extracted from the GTA5 video game were utilized to train the models, while the evaluation was conducted on real-world driving scene images. The findings reveal that the integration of language embeddings has the potential to enhance domain generalization capabilities. Lastly, the thesis delves into solving semantic segmentation for driving scenes through unsupervised domain adaptation. We reproduced state-of-the-art results and conducted experiments using various convolutional architectures as backbones. However, the application of convolutional backbones did not yield improved results compared to using the *Transformer* architecture, despite the convolutional backbones being pretrained on much larger datasets.

Keywords: deep learning, computer vision, semantic segmentation, language embeddings in computer vision, domain adaptation, unsupervised domain generalization, ACDC challenge