Semantička segmentacija slika korištenjem dubokih modela

Ćosić-Dragan, Karla

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 327

SEMANTIČKA SEGMENTACIJA SLIKA KORIŠTENJEM DUBOKIH MODELA

Karla Ćosić-Dragan

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 327

SEMANTIČKA SEGMENTACIJA SLIKA KORIŠTENJEM DUBOKIH MODELA

Karla Ćosić-Dragan

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

DIPLOMSKI ZADATAK br. 327

Pristupnica:	**Karla Ćosić-Dragan (0036516933)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Zoran Kalafatić

Zadatak: **Semantička segmentacija slika korištenjem dubokih modela**

Opis zadatka:

Semantička segmentacija klasificira svaki pojedini piksel u slici na temelju konteksta u kojem se nalazi. U okviru diplomskog rada treba proučiti pristupe za semantičku segmentaciju opisane u literaturi. Odabrati prikladan pristup te programski ostvariti sustav za semantičku segmentaciju prometnih scena. Proučiti javno dostupne skupove podataka za zadatak semantičke segmentacije te pribaviti i pripremiti skup uzoraka za učenje i testiranje oblikovanog sustava. Analizirati dobivene rezultate u pogledu točnosti i računske zahtjevnosti.

Rok za predaju rada: 9. veljače 2024.

Hvala dragom Bogu, dečku i mojoj obitelji na podršci.

SADRŽAJ

1. Uvod

Područje umjetne inteligencije (engl. *artificial intelligence*) predstavlja radnje svojstvene čovjeku koje može obaviti neki računalni sustav. Od preporučivanja sadržaja i pametnog pretraživanja web stranica do prepoznavanja govora i prevođenja teksta, tehnologije umjetne inteligencije nalaze svoju primjenu u svakodnevnom životu prosječnog čovjeka. Jedno od područja primjene tih tehnologija svakako je analiza slika o kojem će biti riječi u ovom radu.

Područje strojnog učenja (engl. *machine learning*) pripada području umjetne inteligencije i istražuje razvoj algoritama koji na temelju dostupnih podataka uče generalizirati neviđene podatke. Metode strojnog učenja mogu biti nadzirane (engl. *supervised*), polunadzirane (engl. *semi-supervised*) i nenadzirane (engl. *unsupervised*). Nadzirane metode koriste označene podatke, odnosno skupove podataka kod kojih se uz ulazne podatke pojavljuju i željeni izlazni podaci (engl. *ground truth*). Polunadzirane metode uz označene podatke koriste neoznačene podatke dok nenadzirane metode koriste isključivo neoznačene podatke. Postoji i pristup strojnom učenju nazvan podržano učenje (engl. *reinforcement learning*) u kojem sustav uči generalizirati podatke metodom pokušaja i pogrešaka. Mnoge pristupe strojnog učenja danas su zamijenile umjetne neuronske mreže.

Analizom slika se specifično bavi područje umjetne inteligencije zvano računalni vid (engl. *computer vision*). Računalni vid na temelju slika i videozapisa pokušava razumjeti i interpretirati vidljivi svijet poput čovjeka. Svoju primjenu nalazi u prepoznavanju lica, generiranju scena unutar virtualne stvarnosti, klasificiranju raznovrsnih slika, lociranju objekata na slikama, semantičkoj segmentaciji slika, itd. U ovom ću se radu konkretno baviti zadatkom semantičke segmentacije slika koji će biti analiziran u kontekstu dubokog učenja koje se kao područje strojnog učenja temelji na umjetnim neuronskim mrežama kojima se uči slijed nelinearnih transformacija skrivenih unutar podataka.

Semantička segmentacija slika područje je dubokog učenja u kojem se svakom pikselu slike dodjeljuje klasa objekta kojem taj piksel pripada. U ovom će se radu

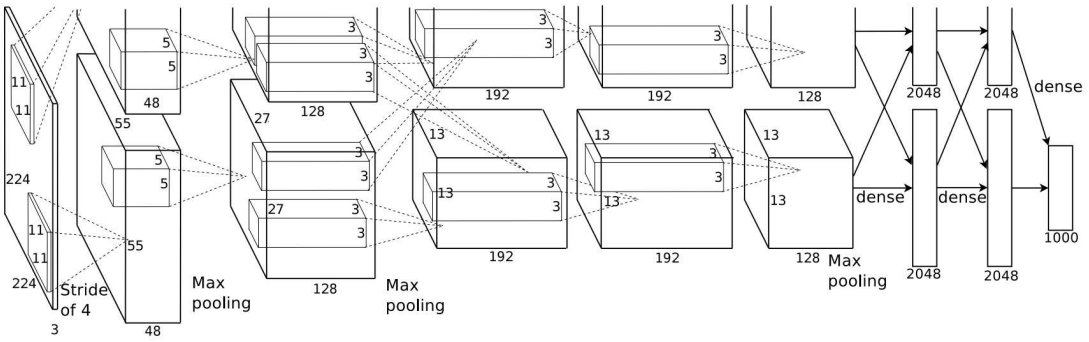
posebno analizirati i segmentirati slike iz prometa. Prepoznavanje i segmentacija objekata iz prometa obavlja se u sustavima za autonomnu vožnju koji su danas uglavnom još u razvoju. Dva su vidno oprečna zahtjeva koja se nameću kod zadatka segmentiranja slika u sklopu autonomne vožnje: preciznost i predikcija u stvarnom vremenu. Naime, složenijim modelima s većom dubinom je moguće obaviti preciznu predikciju, ali im za to treba više vremena od modela s manjom dubinom. U tom je kontekstu potrebano napraviti kompromis između preciznosti i izvođenja u stvarnom vremenu. Osim opsežnog zadatka autonomne vožnje gdje je potrebno prepoznati brojne objekte s kamere, modeli za segmentaciju objekata iz prometa mogu poslužiti jednostavnijem zadatku prepoznavanja prometnih znakova uz pomoć čega bi primjerice bilo moguće prikazati zadnje očitano ograničenje brzine.

2. Standardne arhitekture dubokih modela

U području dubokog učenja klasifikacija slika bio je prvotni zadatak dubokih modela. Klasifikacija slika označava dodijeljivanje klase pojedinoj slici iz skupa klasa C. Na izlazu spomenutih modela nalazi se vektor duljine kardinalnog broja skupa C u kojemu svaka vrijednost označava vjerojatnost pripadanja slike svakoj klasi skupa C nakon čega se klasa s najvećom vjerojatnosti uzima kao izlaz modela. Dakle, svakoj se slici pridaje samo jedna klasa što je značajno lakši zadatak od semantičke segmentacije slike u kojoj svakom pikselu slike valja dodijeliti klasu objekta. U sljedećim poglavljima biti će razmotrene neke od standardnih arhitektura dubokih modela čija je namjena većinom bila klasifikacija slika. Razlog razmatranja spomenutih arhitektura jest korištenje dijelova tih arhitektura kao okosnica (engl. *backbone*) unutar modela za semantičku segmentaciju.

2.1. AlexNet

AlexNet [8] duboki je konvolucijski model naučen klasifikaciji na 1.2 milijuna slika visoke rezolucije skupa podataka ImageNet [3] koji broji 1000 klasa objekata. AlexNet se sastoji od 5 konvolucijskih slojeva unutar kojih se iza prvog, drugog i posljednjeg nalazi po jedan sloj sažimanja maksimumom. Kao nelinearna transformacija koristi se aktivacijska funkcija ReLU (engl. *rectified linear unit*) te se na kraju modela nalaze tri potpuno povezana sloja. Skica AlexNet modela nalazi se na slici 2.1.



Slika 2.1: AlexNet arhitektura. Slika preuzeta iz [8]

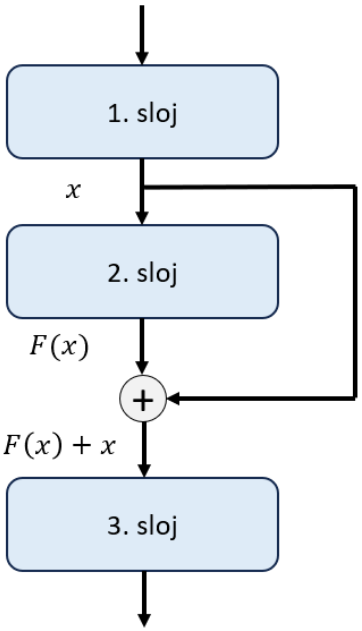
Model je testnom skupu podataka ostvario točnost od 84.6% čime je osigurao pobjedu na natjecanju ILSVRC-2012 [12]. Prvi je duboki model koji je nadmašio algoritme plitkog strojnog učenja (engl. *shallow machine learning*) kod klasifikacije na skupu podataka ImageNet. Algoritmi plitkog strojnog učenja obično sadrže samo jedan sloj transformacije podataka za učenje.

2.2. VGG

VGG (*Visual Geometry Group*) [14] klasična je konvolucijska arhitektura razvijena s ciljem povećanja broja konvolucijskih slojeva što je doprinijelo boljoj točnosti modela na zadatku klasifikacije slika. Sastoji se od grupe modela od kojih je VGG-16 (VGG model s 16 konvolucijskih slojeva) s ukupnom točnosti od 92.7% na skupu podataka ImageNet pobijedio na natjecanju ILSVRC-2014 [12]. Problem prevelikog broja parametara koji je nastao dodavanjem konvolucijskih slojeva ublažen je korištenjem konvolucijske jezgre dimenzija 3×3 za razliku od prethodnih modela kod kojih su korištene veće konvolucijske jezgre.

2.3. ResNet

Arhitektura ResNet (*Residual Network*) pripada skupini arhitektura s preskočnim vezama (engl. *skip connections*). Izlaz iz prethodnog sloja koji se obično dovodi samo na ulaz sljedećeg sloja se u ovom slučaju koristi i kao jedan od ulaza u jedan ili više sljedećih slojeva. Na slici 2.2 je prikazana preskočna veza koja izlaz prvog sloja x dovodi na ulaz drugog i trećeg sloja modela.

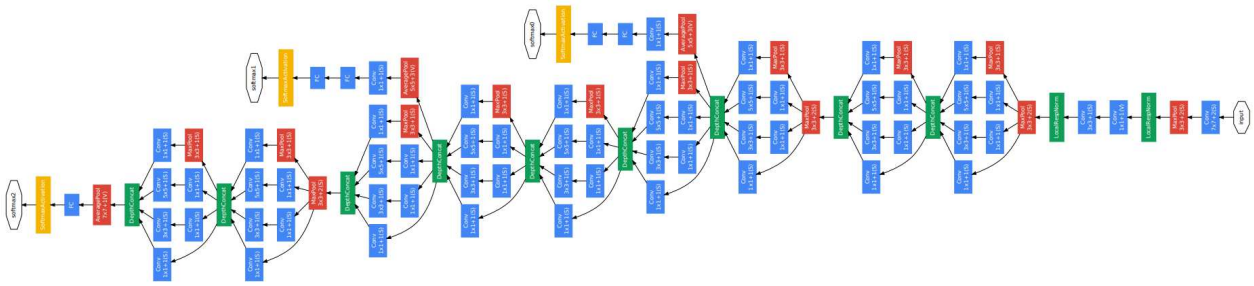


Slika 2.2: Preskočna veza između izlaza iz prvog sloja i ulaza u treći sloj.

U slučaju kada jedan od slojeva pogorša učenje značajki, prethodno naučene značajke će se sačuvati na sljedećem sloju uz pomoć preskočnih veza što će spriječiti pojavu nestajućih ili eksplodirajućih gradijenata. Preskočne veze često se koriste u suvremenim modelima dubokog učenja.

2.4. GoogLeNet

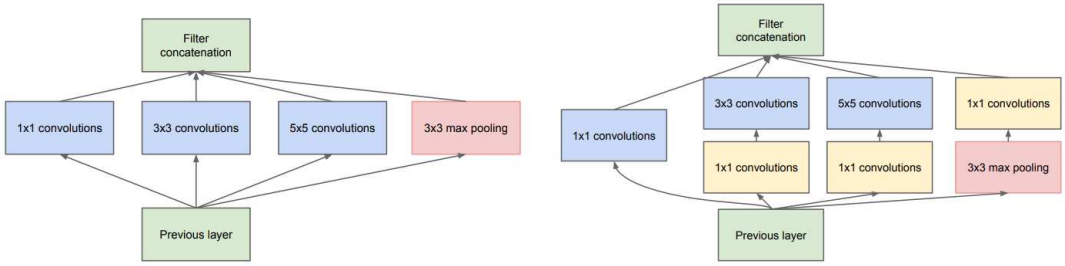
GoogLeNet [15] arhitektura prikazana na slici 2.3 vrsta je arhitektura baziranih na *Inception* modulima. *Inception* moduli omogućavaju korištenje konvolucijskih filtara različitih veličina tijekom istog procesa učenja modela.



Slika 2.3: GoogLeNet arhitektura. Slika preuzeta iz [15]

Arhitekturu su razvili istraživači *Google*-a 2014. godine i svrha joj je klasifikacija

slika i detekcija objekata. Arhitektura je nastala s motivom popravljanja ograničenja prethodnih konvolucijskih arhitektura u smislu dubine i računalne učinkovitosti. Naime, vjerovalo se da dubina modela povoljno utječe na performanse modela, ali problem nestajućih gradijenata kod treniranja modela pokazuje da se performanse modela trebaju poboljšati na neki drugi način. Stoga su istraživači iskoristili *Inception* module prikazane na slici 2.4 uz 1×1 konvolucije uz pomoć kojih se smanjuje broj parametara modela.



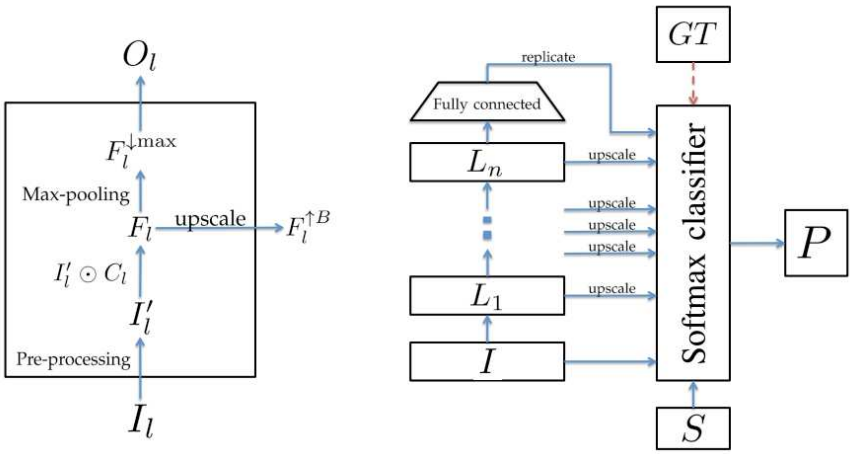
Slika 2.4: Obični *Inception* moduli lijevo i *Inception* moduli s redukcijom dimenzije desno. Slika preuzeta iz [15]

3. Arhitekture modela za semantičku segmentaciju

Semantička segmentacija slike značajno je zahtjevniji zadatak od klasifikacije slika. Kod klasifikacije slika potrebna je samo točna klasa ulazne slike na izlazu modela, dok je kod semantičke segmentacije potreban izlaz veličine ulazne slike koji svakom pikselu dodjeljuje klasu objekta kojemu pripada. Stoga je potrebno unutar modela na neki način održati ili vratiti rezoluciju ulazne slike na izlazu modela.

3.1. Konvolucijske arhitekture

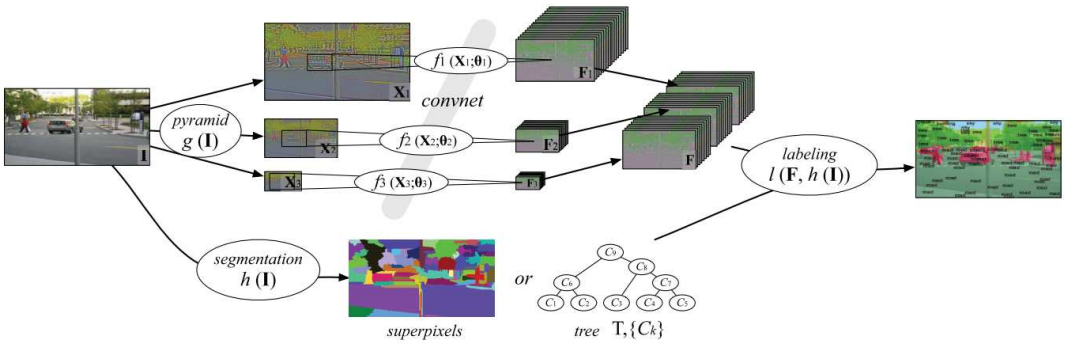
Korištenje modela namijenjenih klasifikaciji objekata za zadatak semantičke segmentacije zahtijeva održavanje rezolucije ulazne slike tijekom treniranja modela. Primjer jednog od takvih modela koji koristi konvolucijske slojeve za sakupljanje semantičkih informacija principom odozgo prema dolje opisan je u članku [5] i prikazan na slici 3.1.



Slika 3.1: Konvolucijski model za sakupljanje segmantičkih informacija principom odozgo prema dolje. Slika preuzeta iz [5].

U spomenutom se modelu ulazna slika dovodi na ulaz modela I nakon čega se propagira do posljednjeg sloja L_n gdje ulazi u potpuno povezani sloj koji se koristi za održavanje globalnog konteksta. Naučene značajke na svakom su sloju po potrebi skalirane bikubičnom interpolacijom na rezoluciju ulazne slike nakon čega su zajedno s izlazom potpuno povezanog sloja povezane u jedan vektor iz kojeg se funkcijom *Softmax* dobiva rezultat. Nakon pete iteracije preko SIFTflow[1] skupa podataka koji sadrži fotografije vanjske okoline s ukupno 33 klase objekta podijeljene u dvije kategorije, prosječna točnost po klasi modela iznosila je 32.1% dok je globalna točnost iznosila 78.7%.

Sličan konvolucijski model predstavljen je u članku [4] i prikazan slikom 3.2. Višeslojna konvolucijska mreža trenirana na slikama u tri različite dimenzije stvara tri skupa mapa značajki od kojih su one manjih dimenzija skalirane tako da im dimenzije odgovaraju onim mapama značajki najveće dimenzije. Mape značajki su na kraju konkatenirane te se na temelju njih uz pomoć različitih metoda dobiva segmentirana slika na izlazu modela. Najveća točnost klasa koju je spomenuti model uz određena poboljšanja na SIFTflow skupu postigao jest 50.8%.



Slika 3.2: Višeslojni konvolucijski model za semantičku segmentaciju. Slika preuzeta iz [4].

Iako opisani modeli daju relativno dobre rezultate, njihova je mogućnost određivanja granica objekata slaba. Razlog tomu leži u načinu skaliranja mapa značajku koji je deterministički. Drugi način dobivanja jednake rezolucije slike na izlazu kao i one na ulazu jest naučeno naduzorkovanje koje je objašnjeno u sljedećem poglavlju u kontestu koder-dekoder arhitektura.

[1] https://www.kaggle.com/datasets/quanbk/sift-flow-dataset/data

3.2. Konvolucijske Koder-dekoder arhitekture

Koder-dekoder arhitektura, kao što sam naziv govori, sastoji se od dva dijela: kodera i dekodera. Koder prima ulazni podatak koji se procesira unutar slojeva kako bi se dobili vektori konteksta na svakom sloju. Ti se vektori konteksta zatim koriste na dekoderu kako bi se generirao izlaz modela. Ovaj se tip arhitekture koristi za generiranje podataka različitog tipa od onog tipa podatka koji se nalazi na ulazu. Primjer namjene jest generiranje teksta na temelju slike ili prevođenje teksta s jednog jezika na drugi i generiranje sažetka teksta.

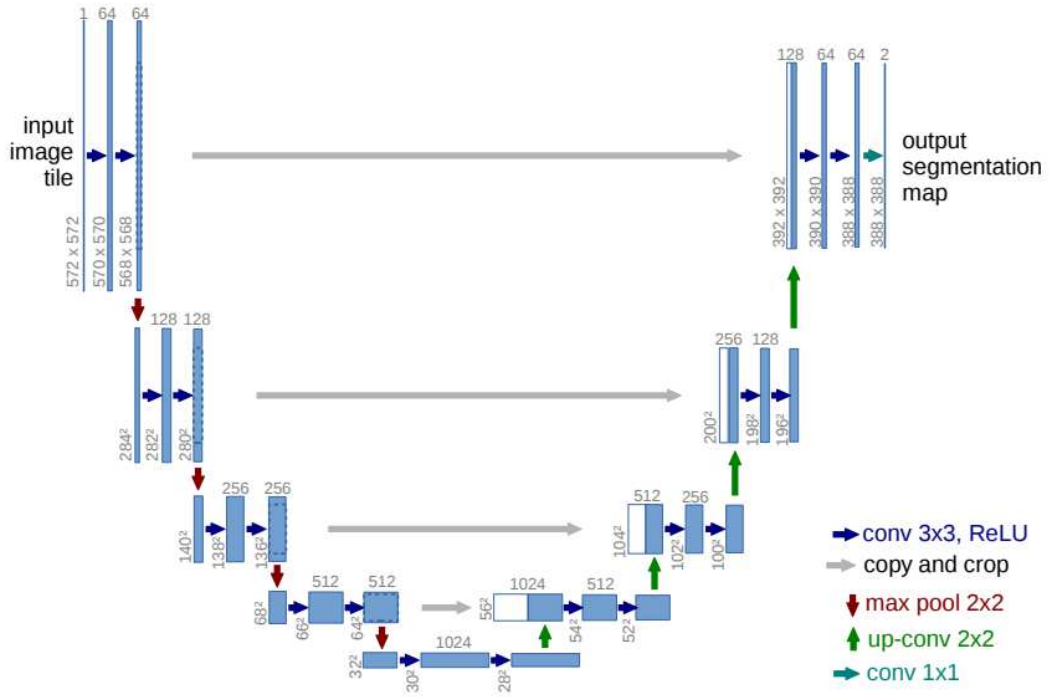
Osim spomenutih primjena, u novije se vrijeme koder-dekoder arhitektura koristi za semantičku segmentaciju slika te predstavlja najsuvremeniji (engl. *state-of-the-art*) pristup tom zadatku. Koder na temelju slike generira mape značajki manje rezolucije čije se dekodiranje na veću rezoluciju zatim uči tijekom procesiranja slike. Neki od primjera ovakvih modela opisani su u nastavku.

3.2.1. Potpuno konvolucijske mreže

Potpuno konvolucijske mreže (engl. *Fully Convolutional Networks*, FCN) [9] unutar kodera kao okosnicu koriste prije prihvaćene i dokazane arhitekture za ekstrakciju značajki. Konkretno se koriste AlexNet, VGG i GoogLeNet arhitekture.

3.2.2. U-Net

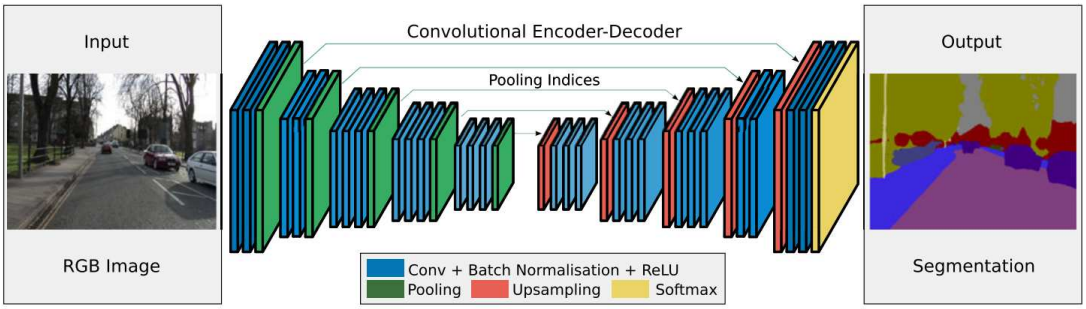
U-Net [11] arhitektura simetrična je koder-dekoder arhitektura korištena za segmentaciju biomedicinskih slika u izvornoj svrsi. Lijeva sažimajuća strana sastoji se od slijeda 3×3 konvolucija gdje nakon svake od konvolucija slijedi nelinearna transformacija ReLU te sažimanje maksimumom 2×2. Desna proširujuća strana sastoji se od niza 2×2 obrnutih konvolucija između kojih se nalaze 3×3 konvolucije uz sloj ReLU iza svake. Posljednja 1×1 konvolucija korištena je za mapiranje svakog 64-dimenzionalnog vektora u vektor s dimenzijom veličine željenog broja klasa. Kod ove arhitekture unutar slojeva dekodera svaki modul za naduzorkovanje kao ulaz uzima izlaz prethodnog modula i izlaz odgovarajućeg modula unutar sažimajuće (koder) strane koji se konkatenira na blok podataka koji se procesira u tom modulu. Opisani U-Net model nalazi se na slici 3.3.



Slika 3.3: U-Net arhitektura. Slika preuzeta iz [11].

3.2.3. SegNet

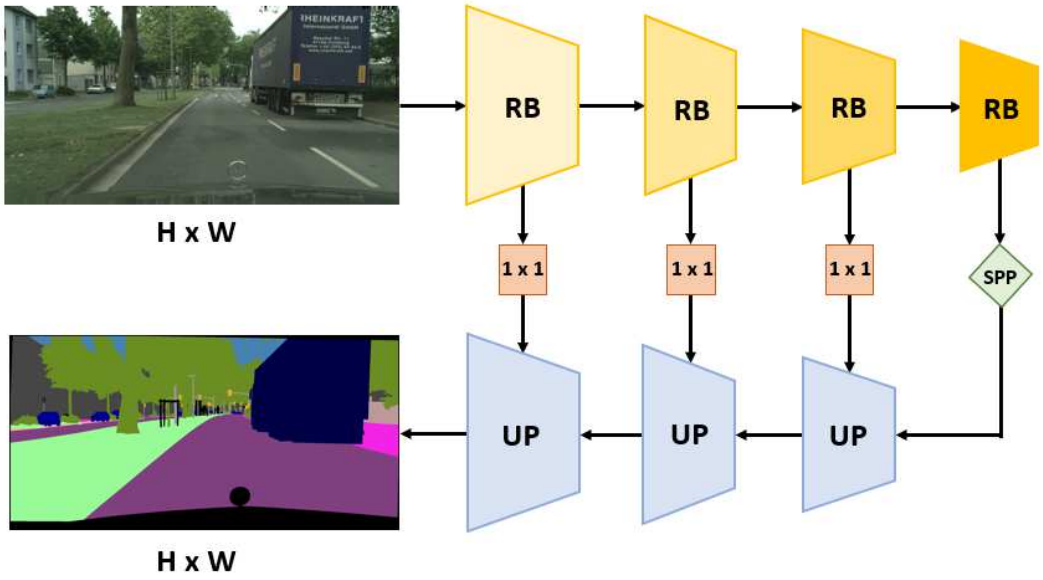
SegNet [1] arhitektura još je jedna od simetričnih koder-dekoder arhitektura korištenih za semantičku segmentaciju. Koder se sastoji od pet slojeva od po dvije ili tri konvolucije popraćene normalizacijom nad grupom, slojem ReLU i sažimanjem maksimumom dok se u dekoderu isti konvolucijski slojevi pojavljuju zrcalno uz sloj naduzorkovanja na početku. Na kraju modela nalazi se *Softmax* klasifikator. Skica modela nalazi se na slici 3.4.



Slika 3.4: SegNet arhitektura. Slika preuzeta iz [1].

3.2.4. SwiftNet

SwiftNet arhitektura prikazana na slici 3.5 i opisana u članku [10] sastoji se od četiri modula u koderu i tri modula u dekoderu uz SPP (engl. *spatial pyramid pooling*) sloj između kodera i dekodera. Svaki se modul unutar kodera sastoji od niza blokova koji imaju dva konvolucijska sloja s jezgrom veličine 3×3 između kojih se nalazi jedan sloj ReLU. Iza svakog konvolucijskog sloja nalazi se sloj normaliziranja po grupi (engl. *batch normalization*). Moduli unutar dekodera sadrže samo jedan konvolucijski sloj što znači da je ova koder-dekoder arhitektura asimetrična. SPP sloj služi za povećanje receptivnog polja. Slojevi uskog grla (engl. *bottleneck*) koji se nalaze između odgovarajućih modula kodera i dekodera služe za prilagođavanje dimenzije iz modula u koderu u modul u dekoderu uz pomoć 1×1 konvolucije.



Slika 3.5: SwiftNet arhitektura. Slika izrađena po uzoru na [10].

4. Funkcije gubitka i metrike

4.1. Funkcije gubitka

Modeli dubokog učenja za optimizaciju parametara koriste gradijentni spust pri čemu se iterativno računa gradijent funkcije pogreške uz ažuriranje parametara u skladu s tim. Optimizacija parametara se pri tome provodi tako da se za svaki parametar w računa derivacija funkcije gubitka po tom parametru $\frac{dL}{dw}$. Kako bi to bilo moguće, potrebno je da funkcija gubitka bude derivabilna funkcija. Funkcije gubitka za binarnu i kategoričku klasifikaciju i njihova ugađanja opisana su u nastavku.

Unakrsna entropija jest mjera različitosti dviju distribucija vjerojatnosti. U strojnom se učenju obično koristi za optimizaciju klasifikacijskih modela. Pogreška binarne unakrsne entropije (engl. *binary cross-entropy error*) izvodi se iz Bernoullijeve distribucije i računa se sljedećom formulom:

$$E_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \log \hat{y}_i))$$

gdje y_i označava ispravnu binarnu klasifikaciju u iznosu od 0 ili 1, a $\hat{y}_i$ vjerojatnost dobivanja prve, odnosno druge klase. Funkcija koja se nalazi unutar sume u formuli pogreške unakrsne entropije naziva se gubitak binarne unakrsne entropije (engl. *binary cross-entropy loss*) i prikazana je sljedećom formulom:

$$L_{BCE} = -y \log(\hat{y}) - (1 - y)\log(1 - \log \hat{y})$$

Funkcija gubitka unakrsne entropije označava vrijednost pogreške na jednom primjeru, dok je funkcija pogreške zapravo srednja vrijednost funkcije gubitka na skupu primjera.

Opisane formule vrijede u slučaju klasifikacije na dvije klase. Za računanje pogreške i gubitka klasifikacije na K klasa te je formule potrebno poopćiti. Pritom će klasifikacija primjera biti prikazana kao vektor indikatorskih varijabli (engl. *one-hot vector*) koji na indeksu točne klase među indeksima u rasponu od $(0, K - 1)$ sadržava

vrijednost 1 dok su ostale vrijednosti u vektoru 0. Pogreška kategoričke unakrsne entropije (engl. *categorical cross-entropy error*) izvedena iz Multinoullijeve distribucije prema tome iznosi:

$$E_{CCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)} \ln \hat{y}_k^{(i)}$$

dok gubitak kategoričke unakrsne entropije (engl. *categorical cross-entropy loss*) iznosi:

$$L_{CCE} = -\sum_{k=1}^{K} y_k \ln \hat{y}_k$$

Problem koji se često pojavljuje u sustavima klasifikacije ili semantičke segmentacije jest nejednakost pojave instanci klasa unutar skupa podataka. Neke se klase stalno pojavljuju, dok se druge pojavljuju puno manji broj puta. Opisana neravnoteža rezultirat će lošom predikcijom modela na manjim klasama. Kako bi model bio više osjetljiv na klase u manjini, funkcija gubitka unakrsne entropije može se ugoditi tako da se pozitivni primjeri pomnože s parametrom β pri čemu će formula gubitka težinske binarne unakrsne entropije (engl. *weighted binary cross-entropy loss*) [6] glasiti:

$$L_{WBCE} = -\beta y \log(\hat{y}) - (1-y)\log(1-\log\hat{y})$$

Osim što se pozitivnim primjerima može dodati težina β, isto se može učiniti i s negativnim primjerima. Takvo se ugađanje modelira gubitkom uravnotežene binarne unakrsne entropije (engl. *balanced binary cross-entropy loss*) [6]:

$$L_{BBCE} = -\beta y \log(\hat{y}) - (1-\beta)(1-y)\log(1-\log\hat{y})$$

4.2. Osnovne mjere vrednovanja klasifikacije

Prije opisa osnovnih mjera vrednovanja klasifikacije potrebno je objasniti označavanje klasa i nazive slučajeva klasifikacije u kontekstu binarne klasifikacije. Označimo prvu klasu binarne klasifikacije s 1, a drugu s 0. Klasu označenu s 1 nazovimo pozitivna klasa, a klasu označenu s 0 negativna klasa. Prilikom predikcije modela binarnog klasifikatora, model može pozitivni ili negativni primjer svrstati u pozitivnu ili negativnu klasu te se primjeri mogu razložiti u sljedeća četiri slučaja:

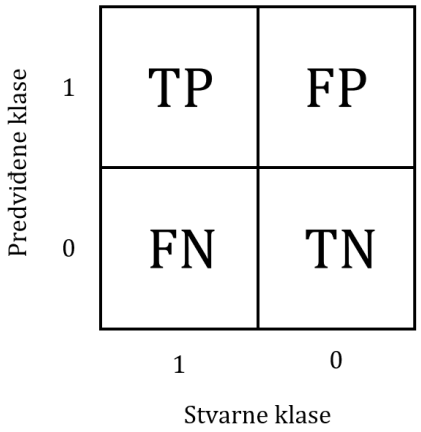
1. stvarno pozitivni primeri (engl. *true positives*, TP) - pozitivni primjeri koji su klasificirani kao pozitivni

2. lažno pozitivni primjeri (engl. *false positives*, FP) - negativni primjeri koji su klasificirani kao pozitivni

3. stvarno negativni primjeri (engl. *true negatives*, TN) - negativni primjeri koji klasificirani kao negativni

4. lažno negativni primjeri (engl. *false positives*, FP) - pozitivni primjeri koji su klasificirani kao negativni

Osnovne mjere vrednovanja klasifikacije provode se nad matricom zabune ili kontigencije (engl. *confusion (contingency) matrix*). Matrica zabune za binarnu klasifikaciju tablično prikazuje broj poklapanja ili nepoklapanja primjera s klasama i prikazana je slikom 4.1.



Slika 4.1: Matrica zabune ili kontigencije za dvije klase

Osnovne mjere vrednovanja koje se dobivaju iz matrice zabune su točnost (engl. *accuracy*), preciznost (engl. *precision*), odziv (engl. *recall*), ispadanje (engl. *fall-out*) i specifičnost (engl. *specificity*). Točnost označava postotak točno klasificiranih primjera i računa se formulom:

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

Preciznošću se mjeri udio stvarno pozitivnih primjera unutar svih primjera koji su klasificirani kao pozitivni. Formula za preciznost glasi:

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

Odziv jest udio točno klasificiranih pozitivnih primjera unutar svih primjera koji su stvarno pozitivni, bili klasificirani pozitivno ili negativno i računa se formulom:

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

Ispadanje ili stopa lažnih pozitiva (engl. *False Positive Rate*, FPR) označava udio krivo klasificiranih pozitivnih primjera unutar svih stvarno negativnih primjera kako god bili klasificirani i glasi:

$$FO = \frac{FP}{FP + TN}$$

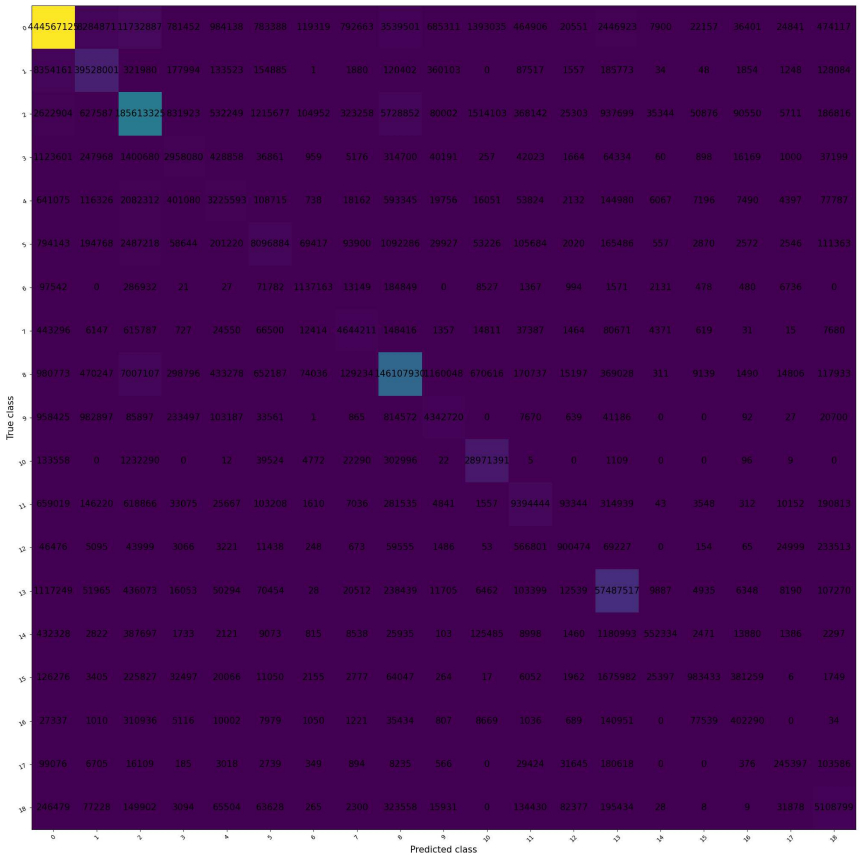
Specifičnost naposljetku mjeri udio točno klasificiranih negativnih primjera unutar svih primjera koji su klasificirani negativno i računa se kao:

$$Spec = \frac{TN}{TN + FP}$$

Dodatna mjera vrednovanja klasifikacije jest mjera F_1 (engl. F_1 *score*) koja je određena kao harmonijska sredina odziva i preciznosti i računa se sljedećom formulom:

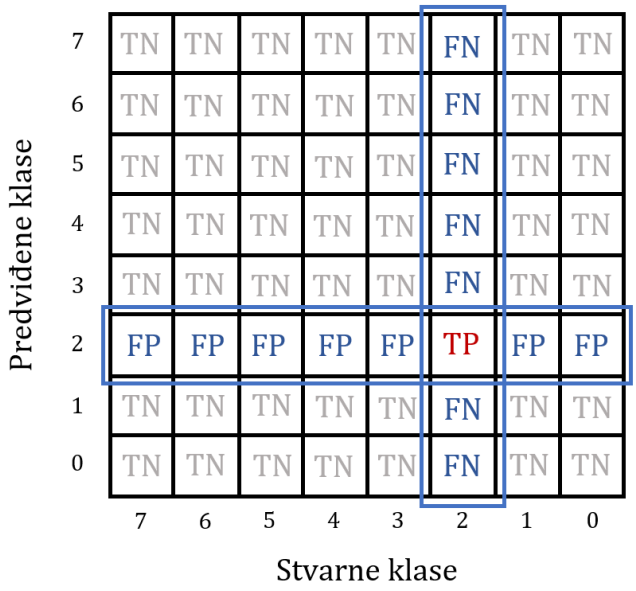
$$F_1 = \frac{2}{\frac{1}{Pr} + \frac{1}{Rec}} = \frac{2 \cdot Pr \cdot Rec}{Pr + Rec}$$

Osim binarne klasifikacije, potrebno je moći navedene mjere vrednovanja izračunati i u slučaju višeklasne klasifikacije. Za to je potrebna višeklasna matrica zabune. Primjer takve matrice nalazi se na slici 4.2.



Slika 4.2: Primjer višeklasne matrice zabune. Slika generirana prilikom implementacije sustava za semantičku segmentaciju prometnih scena.

Poopćenje spomenutih mjera vrednovanja na više klasa provodi se tako što se svaka mjera računa posebno za svaku klasu. Ako stupci matrice zabune označavaju stvarne klase, a retci klase predviđene klasifikacijom, tada je za pojedinu klasu broj lažnih negativa jednak zbroju elemenata stupca rednog broja te klase osim dijagonalnog elementa, a broj lažnih pozitiva jest zbroj elemenata retka rednog broja te klase bez dijagonalnog elementa. Broj stvarnih pozitiva odgovara broju na dijagonali matrice zabune na indeksu broja klase. Zbroj ostalih elemenata matrice za tu klasu odgovara broju stvarnih negativa. Mjere vrednovanja se zatim računaju na isti način kao i kod binarne klasifikacije. Opisano je računanje stvarnih i lažnih pozitiva i negativa za klasu 2 prikazano slikom 4.3.



Slika 4.3: Matrica zabune s označenim poljima za predikciju klase 2.

4.3. Mjere vrednovanja semantičke segmentacije

Budući da je semantička segmentacija zapravo klasifikacija slike na razini piksela, sve se navedene mjere vrednovanja mogu koristiti i u tom području. Ipak, mjera vrednovanja koja se najviše koristi u semantičkoj segmentaciji jest omjer presjeka i unije (engl. *intersection over union*) prema formuli:

$$IoU = \frac{|A \cap B|}{|A| \cup |B|}$$

Omjer presjeka i unije u kontekstu stvarnih i lažnih pozitiva i negativa na razini piksela računa se na sljedeći način:

$$IoU = \frac{TP}{TP + FN + FP}$$

Ovaj se *IoU* računa posebno za svaku pojedinu klasu. Problem koji se očituje kod ove mjere ponovno je nejednakost instanci klasa. Na primjer, kod semantičke segmentacije prometnih scena puno piksela na slici pripada klasi ceste ili neba, dok mali broj piksela pripada klasi pješaka ili biciklista. Model će tada prilikom učenja lakše prepoznati *veće* klase i one će imati veći *IoU*. Stoga se često mjera omjera presjeka i unije skalira tako da se računa ukupni srednji omjer presjeka i unije kao prosjek svih *IoU* svake klase (engl. *Mean intersection over union*, *MIoU*).

5. Skupovi podataka

5.1. Cityscapes

Skup podataka Cityscapes [2] jedan je od najčešće korištenih skupova podataka za izgradnju modela semantičke segmentacije prometnih scena. Sastoji se od 5000 fotografija s pripadajućim slikama na kojima su objekti koji se pojavljuju na prometnoj sceni precizno označeni. Fotografije su dobivene iz videozapisa snimljenih na ulicama unutar 50 njemačkih gradova. Ukupno je 30 klasa objekata podijeljenih u 9 kategorija kao što je prikazano na tablici 5.1. Modeli se najčešće treniraju na podskupu od 19 klasa objekata koje su podcrtane u tablici 5.1.

plosnata površina	cesta, pločnik, parkiralište, željeznička pruga
osoba	pješak, vozač
vozilo	automobil, kamion, autobus, vlak, motocikl, bicikl, karavan, prikolica
konstrukcija	zgrada, zid, ograda, zaštitna ograda, most, tunel
objekt	prometni stup, prometni stupovi, prometni znak, semafor
priroda	vegetacija, teren
nebo	nebo
prazan prostor	tlo, statički prostor, dinamički prostor

Tablica 5.1: Klase prometnih objekata.

Primjer fotografije iz skupa Cityscapes s pripadajućim labelama nalazi se na slici 5.1.

Slika 5.1: Fotografija s pripadajućim labelama iz skupa Cityscapes.

5.2. ACDC

Skup podataka ACDC [13] (*Adverse Conditions Dataset with Correspondences*) jest skup fotografija prometnih scena u nepovoljnim uvjetima. Sastoji se od 4006 fotografija koje su pravilno raspoređene unutar četiri skupine koje označavaju nepovoljne uvjete na cesti: kiša, magla, snijeg i noć. Svaka fotografija dolazi u paru s fotografijom iste scene u normalnim uvjetima uz sliku s precizno označenim objektima te scene i binarnu masku za razlikovanje područja fotografije s jasnim semantičkim sadržajem od onih područja s nejasnim sadržajem. Klase objekata koje se pojavljuju u ovom skupu podataka su iste one koje se pojavljuju i u skupu podataka Cityscapes. Primjer fotografije iz prometa u snijegu i njoj pripadajuće slike labela i fotografije u normalnim uvjetima nalazi se na slici 5.2.

Slika 5.2: Fotografija prometne scene u snijegu s pripadajućim labelama i pripadajućom fotografijom u normalnim uvjetima iz skupa ACDC.

6. Implementacija i eksperimenti

U sklopu zadatka semantičke segmentacije korišten je model SwiftNet s ResNet18 okosnicom i skupovi podataka Cityscapes i ACDC opisani prethodno. Za implementaciju je korišten programski paket Pytorch i programski okvir Pytorch Lightning.

6.1. Programski paket Pytorch

PyTorch[1] jest programski paket za implementaciju modela dubokog učenja razvijen od strane Facebook-a i objavljen 2016. godine. Napisan je u programskom jeziku Python[2]. Paket nudi tip podataka `Tensor` za čuvanje vrijednosti vektora i matrica te skup operacija koje se mogu provoditi nad tenzorima. Kako bi se omogućila efikasnost prilikom treniranja, Pytorch omogućava ubrzano računanje nad tenzorima koristeći GPU (grafička procesna jedinica, engl. *graphics processing unit*) što predstavlja značajno ubrzanje računanja u usporedbi s računanjem uz korištenje CPU (centralna procesorska jedinica, engl. *central processing unit*). Osim toga, u sklopu Pytorch-a omogućene su numeričke optimizacije nad općim matematičkim izrazima korisne za proces treniranja modela dubokog učenja.

6.2. Programski okvir Pytorch Lightning

Pytorch Lightning[3] programski je okvir za implementaciju postupka treniranja modela uz praćenje gubitka i metrika. Klasa u kojoj se nalazi programski kod modela treba nasljeđivati klasu `LightningModule` i implementirati metodu `forward` kojom se obavlja unaprijedni prolazak kroz slojeve modela. Unutar metode `configure_optimizers` u istoj klasi moguće je podesiti način na koji će se model trenirati, npr. hoće li za treniranje koristiti stohastički gradijentni spust ili algoritam Adam. Zatim se

[1] https://pytorch.org/

[2] https://www.python.org/

[3] https://lightning.ai/docs/pytorch/stable/

unutar `training_step` metode treba implementirati logika treniranja modela, odnosno uzimanja podataka iz skupa podataka za treniranje i prosljeđivanja tih podataka u model i računanje funkcije gubitka. Ukoliko se usporedno s treniranjem provodi postupak validacije modela, potrebno je implementirati metodu `validation_step`. Primjer korištenja okvira Pytorch Lightning jest sljedeći:

```
import pytorch_lightning as pl
...

class SemsegModel(pl.LightningModule):
    def __init__(...):
        ...

    def forward(self, batch, image_size=None):
        features, additional = self.backbone(batch['image'])
        logits = self.logits.forward(features)
        if (not self.training) or self.upsample_logits:
            logits = upsample(logits, image_size)
        additional['logits'] = logits
        additional['model'] = self
        additional = {**additional, **batch}
        return logits, additional

    def loss(self, batch):
        labels = batch['labels'].cuda()
        logits, additional = self.forward(batch, ...)
        return self.criterion(logits, labels, batch=batch, ...)

    def training_step(self, batch, batch_idx):
        loss = self.loss(batch)
        self.log('Train loss', loss)
        return loss

    def validation_step(self, batch, batch_idx):
        if self.trainer.current_epoch % 5 == 0:
            loss = self.loss(batch)
            self.log('Validation loss', loss)
            return loss
```

```
def configure_optimizers(self):
    lr = 4e-4
    fine_tune_factor = 4
    weight_decay = 1e-4
    optim_params = [
        {
            'params': self.random_init_params(),
            'lr': lr,
            'weight_decay': weight_decay
        },
        {
            'params': self.fine_tune_params(),
            'lr': lr / fine_tune_factor,
            'weight_decay': weight_decay / fine_tune_factor
        },
    ]
    optimizer = optim.Adam(optim_params, betas=(0.9, 0.99))

    epochs = 250
    lr_min = 1e-6
    lr_scheduler = optim.lr_scheduler.CosineAnnealingLR(
        optimizer,
        epochs,
        lr_min
    )

    return [optimizer], [lr_scheduler]
```

Unutar okvira Pytorch Lightning moguće je implemetirati i modul za učitavanje podataka iz skupa podataka uz pomoć klase `LightningDataModule` koju je potrebno u programskom kodu naslijediti i implemenirati metode `train_dataloader` i `val_dataloader` za učitavanje podataka iz skupova za treniranje i validaciju. Osim toga, mogu se implementirati i metode `prepare_data` za pripremu podataka (metoda se izvršava na jednoj grafičkoj jedinici) i `setup` za postavljanje podataka (metoda se izvršava na više grafičkih jedinica). Primjer modula za učitavanje podataka je sljedeći:

```
import pytorch_lightning as pl
```

```
...

class CityscapesDataModule(pl.LightningDataModule):

    def __init__(...):
        ...

    def train_dataloader(self):
        return DataLoader(
            dataset=self.train_dataset,
            batch_size=self.batch_size,
            shuffle=True,
            num_workers=self.num_workers,
            collate_fn=self.collate_fn
        )

    def val_dataloader(self):
        return DataLoader(
            dataset=self.val_dataset,
            batch_size=1,
            num_workers=self.num_workers,
            collate_fn=self.collate_fn
        )
```

Naposljetku, potrebno je instancirati model i modul za učitavanje podataka i pokrenuti učenje modela. Uz to u slijedećem je primjeru definirana je instanca klase `ModelCheckpoint` koja služi kao *callback* funkcija i prati zadanu metriku tijekom treniranja i na temelju te metrike sprema zadnjih nekoliko najboljih modela. U primjeru se prati *MIoU* metrika i sprema se zadnji najbolji model na temelju te metrike uz model dobiven u zadnjoj epohi.

```
model = SemsegModel(resnet, num_classes)
data_module = CityscapesDataModule()

checkpoint_callback = ModelCheckpoint(
    monitor="mean_iou",
    dirpath="logs/checkpoints",
    filename="model-{epoch:03d}-{mean_iou:.2f}",
    save_top_k=1,
```

```
    mode="max",
    save_last=True,
)

trainer = pl.Trainer(
    accelerator='gpu',
    max_epochs=250,
    logger=csv_logger,
    callbacks=[checkpoint_callback]
)

trainer.fit(model, data_module)
```

Prethodni su programski kodovi implementirani u sklopu zadatka ovog diplomskog rada. Još je puno dodatnih mogućnosti koje Pytorch Lightning okvir nudi.

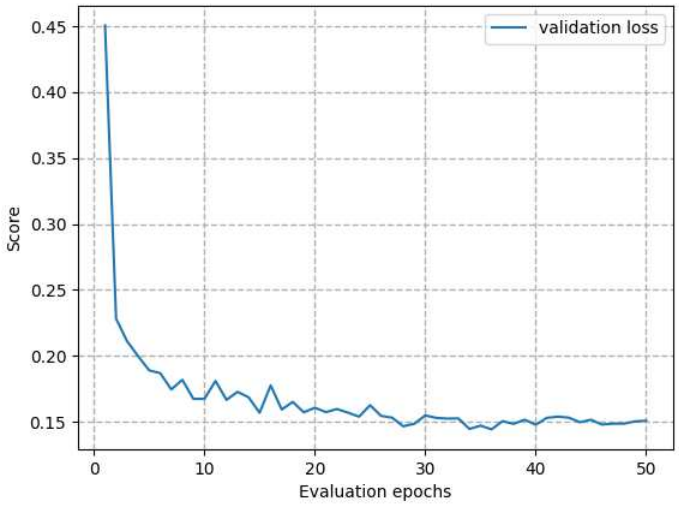
6.3. Provedba eksperimenata

Model je učen na 250 epoha na oba skupa podataka pri čemu je u svakoj petoj epohi provedena evaluacija gubitka i metrika. Za računanje gubitka korištena je funkcija gubitka unakrsne entropije. Kao algoritam optimizacije modela korišten je algoritam Adam [7]. Veličina grupe (engl. *batch size*) slika korištena za učenje modela je 14. Model je treniran uz pomoć grafičke kartice NVIDIA GeForce GTX 1080 Ti. Osim metrika i gubitka, nad naučenim je modelom izmjeren i broj slika koje model može segmentirati u jednoj sekundi (engl. *Frames per Second*, FPS) kako bi se predočila računalna složenost SwiftNet modela na navedenoj grafičkoj kartici. Programski kod za računanje FPS metrike je sljedeći:

```
n = len(val_dataloader.dataset)
t0 = perf_counter()
for step, batch in tqdm(enumerate(val_dataloader), total=len(val_dataloader)):
    batch['image'] = batch['image'].to('cuda')
    logits, _ = model.forward(batch, batch['image'].shape[1:3])
    pred = torch.argmax(logits.data, dim=1).cpu().numpy().astype(np.uint32)
    torch.cuda.synchronize()
t1 = perf_counter()
fps = n / (t1 - t0)
```

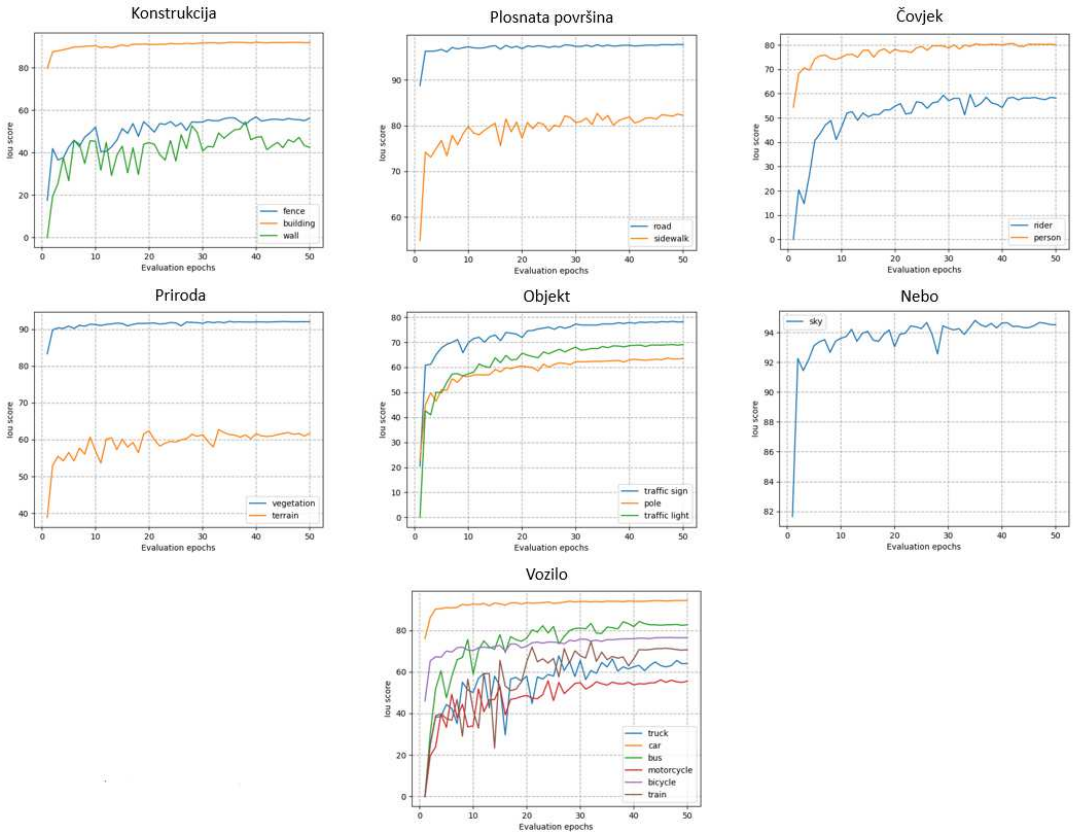
6.3.1. Rezultati na skupu podataka Cityscapes

Na slici 6.1 prikazan je gubitak kroz evaluacijske epohe na skupu podataka za validaciju. Gubitak se naglo smanjivao u prvih 50 epoha, a potom se postupno smanjivao i stabilizirao na iznosu od 0.15.



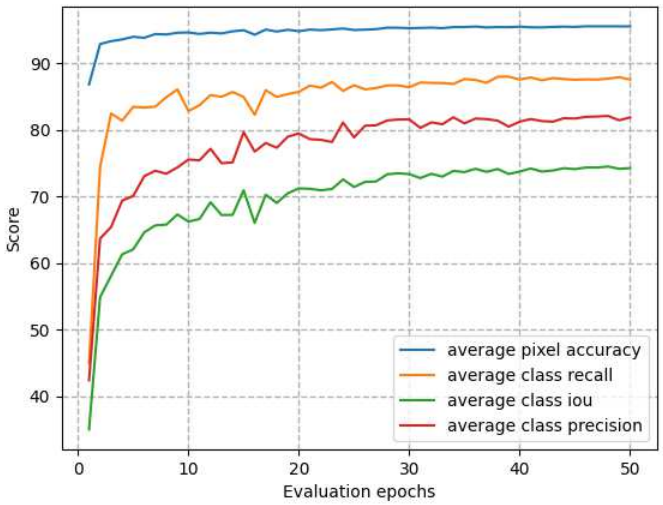
Slika 6.1: Graf gubitka na skupu podataka za validaciju.

Na slici 6.2 prikazani su grafovi omjera presjeka i unije za svaku klasu kroz evaluacijske epohe podijeljeni po kategorijama klasa. Vidljivo je da nad klasama koje se više pojavljuju u skupu podataka i zauzimaju više prostora na slici poput neba i ceste postignut veći omjer presjeka i unije u odnosu na klase koje zauzimaju manje prostora na slici poput vozača i prometnih stupova i klase koje se manje pojavljuju u skupu podataka poput motocikala i kamiona.



Slika 6.2: Grafovi omjera presjeka i unije tijekom treniranja za svaku klasi podijeljeni po kategorijama klasa.

Graf prosječne točnosti, odziva, preciznosti i prosjeka nad unijom po klasama kroz evaluacijske epohe prikazan je na slici 6.3.

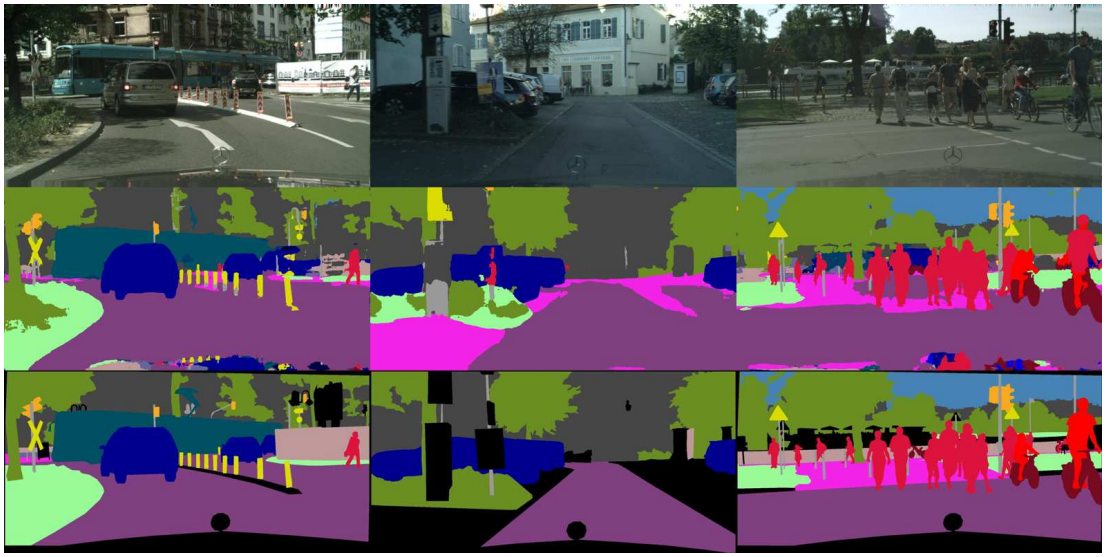


Slika 6.3: Graf prosječnog omjera presjeka i unije, točnosti, odziva i preciznosti po klasama.

Model je najbolji prosječni omjer presjeka i unije ostvario u 235. epohi u iznosu od 74.52%. na skupu Cityscapes. U tablici 6.1 navedeni su prosjeci nad unijom za svaku od 19 klasa u 235. epohi. Na slici 6.4 nalaze se primjeri predikcija na validacijskom skupu podataka. U prvom retku su fotografije scena iz prometa, u drugom retku labele predviđene modelom, a u trećem retku stvarne labele, odnosno tzv. *ground truth*. Regije u trećem retku obojane crno označavaju klase koje su se prilikom treniranja i evaluiranja metrika ignorirale i stoga su predikcije za te regije u drugom retku nasumične. Najmanji *IoU* ostvarila je klasa motocikl dok je najveći *IoU* ostvaren za klasu cesta.

Kategorija	**Klasa**	**Omjer presjeka i unije (Cityscapes)**
plosnata površina	cesta	97.66%
	pločnik	82.06%
osoba	pješak	80.19%
	vozač	57.49%
vozilo	automobil	94.37%
	kamion	65.51%
	autobus	82.82%
	vlak	70.72%
	motocikl	55.20%
	bicikl	76.40%
konstrukcija	zgrada	91.91%
	zid	47.17%
	ograda	55.47%
objekt	prometni stup	63.30%
	prometni znak	78.31%
	semafor	69.05%
priroda	vegetacija	92.04%
	teren	61.64%
nebo	nebo	94.62%

Tablica 6.1: Mjera *IoU* izračunata za svaku klasu u epohi u kojoj je model ostvario najveći *MIoU*.



Slika 6.4: Fotografije prometnih scena, predikcije labela SwiftNet modelom i stvarne labele za skup podataka Cityscapes.

Treniranim se modelom može procesirati približno 24 fotografije u sekundi što predstavlja zadovoljavajući rezultat budući da je standardni FPS korišten u filmovima, pametnim telefonima i video stream-ovima upravo 24.

6.3.2. Rezultati na skupu podataka ACDC

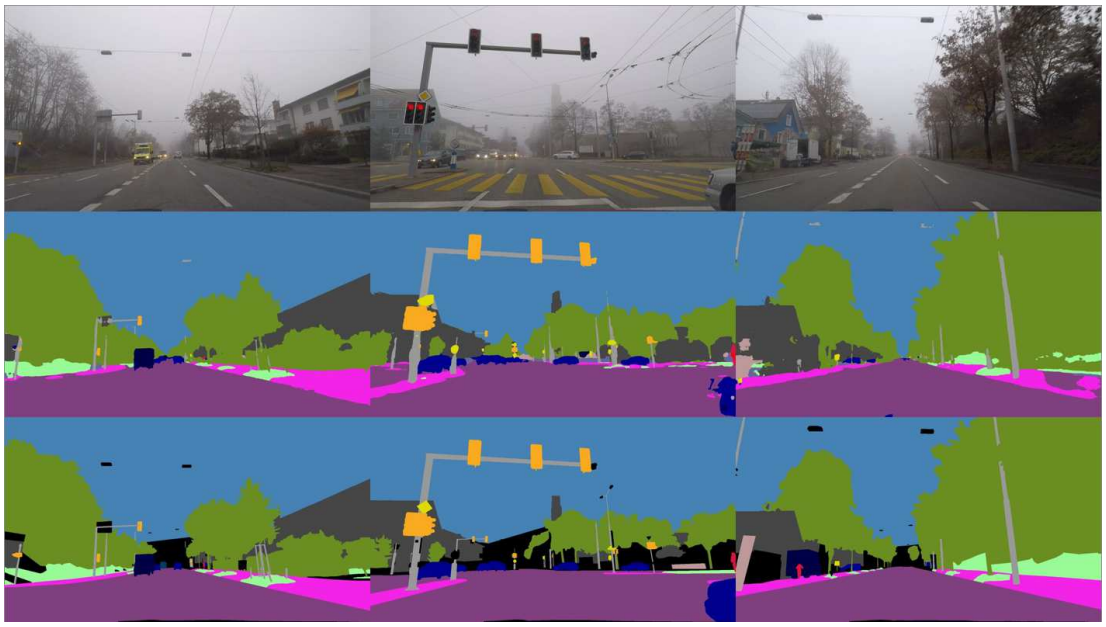
Na skupu ACDC model je ostvario najbolji *IoU* u 240. epohi u iznosu od 63.72% što je očekivano manje od postignuća na Cityscapes-u budući da se radi o prikazu prometa u nepovoljnim uvjetima. Na tim su fotografijama granice među klasama nejasne, pogotovo u noćnim uvjetima. Gubitak unakrsne entropije se u ovom slučaju stabilizirao na iznosu od 0.24. U tablici 6.2 navedeni su prosjeci nad unijom za svaku klasu u 240. epohi. Klase pješak i vozač ostvarile su značajno manji omjer pesjeka i unije u odnosu na skup podataka Cityscapes.

Kategorija	Klasa	Omjer presjeka i unije (ACDC)
plosnata površina	cesta	93.51%
	pločnik	74.10%
osoba	pješak	54.23%
	vozač	12.36%
vozilo	automobil	85.16%
	kamion	46.60%
	autobus	86.78%
	vlak	83.26%
	motocikl	30.85%
	bicikl	41.88%
konstrukcija	zgrada	85.13%
	zid	51.25%
	ograda	43.08%
objekt	prometni stup	60.12%
	prometni znak	61.78%
	semafor	71.39%
priroda	vegetacija	86.43%
	teren	47.06%
nebo	nebo	95.80%

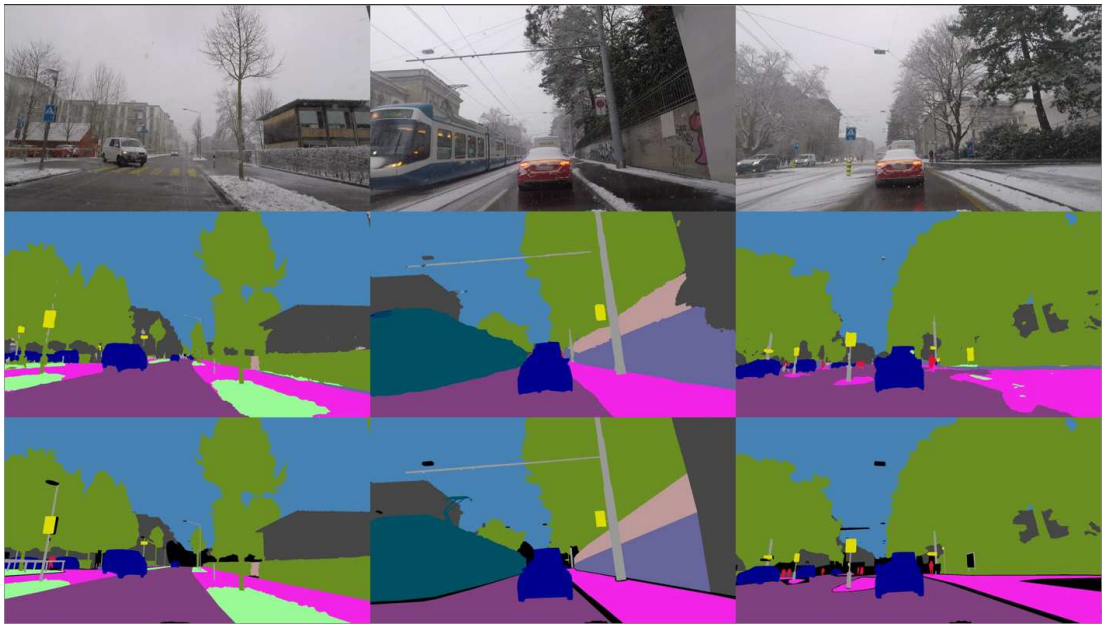
Tablica 6.2: Mjera *IoU* izračunata za svaku klasu u epohi u kojoj je model ostvario najveći *MIoU*.

Na slikama 6.5, 6.6, 6.7, 6.8 nalaze se predikcije na validacijskom skupu podataka redom u noćnim, maglovitim, snježnim i kišnim uvjetima. U noćnim i snježnim uvjetima granice između nekih klasa na *ground truth* slikama obojane su crno budući da s fotografija prometnih scena nije najjasnije kojoj klasi pripadaju granični pikseli te se predikcije nad tim pikselima prilikom računanja gubitka modela zanemaruju.

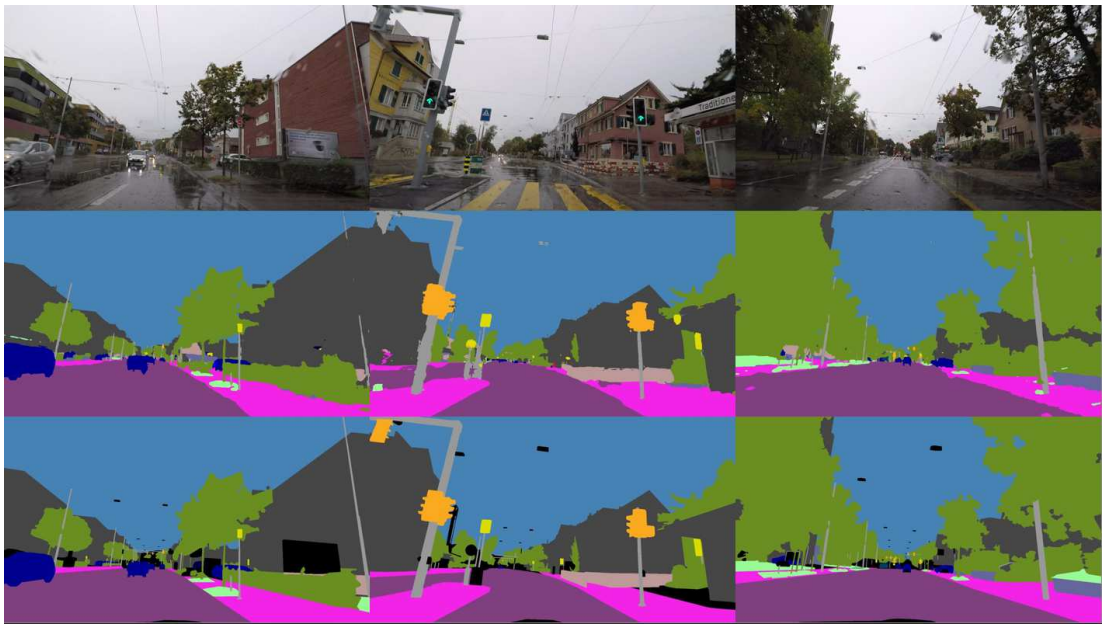
Slika 6.5: Fotografije prometnih scena, predikcije labela SwiftNet modelom i stvarne labele za skup podataka ACDC u noćnim uvjetima.



Slika 6.6: Fotografije prometnih scena, predikcije labela SwiftNet modelom i stvarne labele za skup podataka ACDC u maglovitim uvjetima.



Slika 6.7: Fotografije prometnih scena, predikcije labela SwiftNet modelom i stvarne labele za skup podataka ACDC u snježnim uvjetima.



Slika 6.8: Fotografije prometnih scena, predikcije labela SwiftNet modelom i stvarne labele za skup podataka ACDC u kišnim uvjetima.

Model naučen na cjelokupnom ACDC skupu podataka evaluiran je na validacijskim skupovima svakog pojedinog podskupa tog skupa. Rezultati evaluacije prikazani su u tablici 6.3. Najmanji *MIoU* u iznosu od 48.23% dobiven je za podskup fotografija snimljenih u noćnim uvjetima što je očekivanu budući da se u tim uvjetima granice

između klasa ne mogu lako razaznati. Najveći *MIoU* postignut je pak na fotografijama u maglovitim uvjetima u iznosu od 64.97%.

Nepovoljna prilika	*MIoU*	Odziv	Preciznost	Točnost
noć	48.23%	63.4%	57.27%	87.06%
magla	69.97%	86.22%	77.1%	95.14%
kiša	62.14%	75.48%	70.54%	94.75%
snijeg	66.52%	81.32%	74.15%	94.37%

Tablica 6.3: Mjere izračunate nad svim validacijskim podskupovima u ovisnosti o vremenskim prilikama.

Model treniran na ACDC skupu podataka isto kao i model treniran na Cityscapes-u ostvaruje FPS u iznosu od približno 24 fotografije u sekundi na grafičkoj kartici NVIDIA GeForce GTX 1080 Ti.

6.3.3. Evaluacija SwiftNet modela učenog na Cityscapes-u na ACDC skupu podataka i obratno

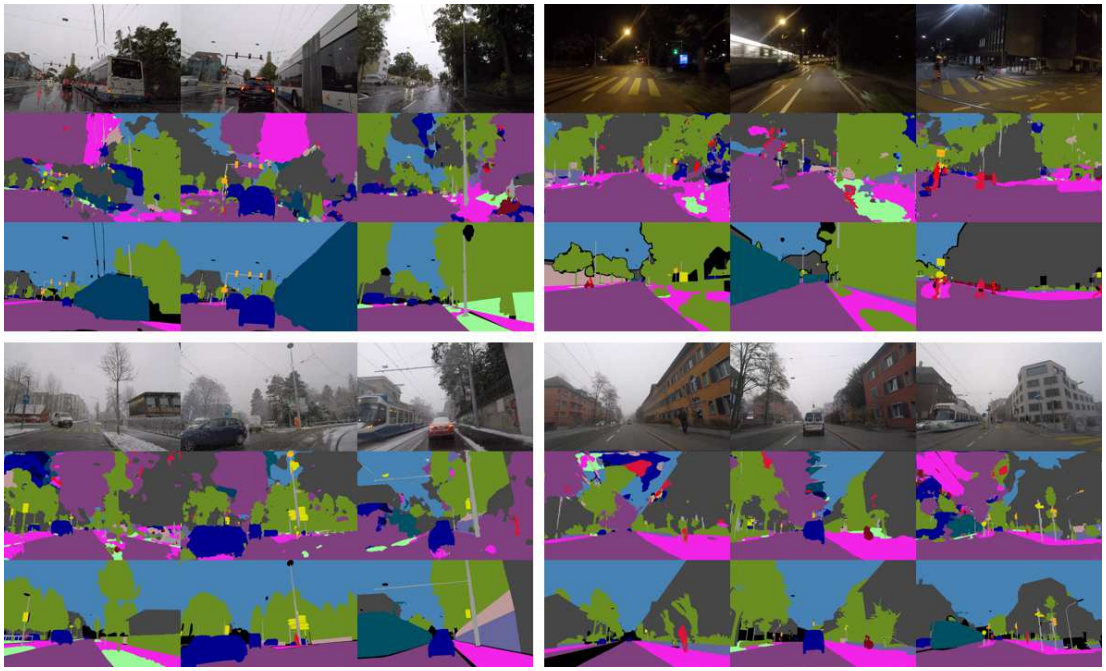
Osim spomenutih evaluacija modela na validacijskim skupovima istog skupa podataka na kojem je model bio treniran, zanimljivo je bilo provjeriti kako se model treniran na jednom skupu podataka ponaša na drugom sličnom skupu. Eksperiment pokazuje da model treniran na Cityscapes-u na validacijskom skupu ACDC skupa ostvaruje *MIoU* u iznosu od 24.61%. U tablici 6.4 prikazani su rezultati evaluacije na svakom validacijskom podskupu ACDC skupa. Očekivano se najmanji *MIoU* dobiva za fotografije u noćnim uvjetima, a najveći *MIoU* za fotografije u maglovitim uvjetima. Ostale su metrike po iznosu očekivano manje nego u tablici 6.3.

Nepovoljna prilika	*MIoU*	Odziv	Preciznost	Točnost
noć	8.65%	37.4%	17.62%	40.61%
magla	37.36%	68.39%	47.87%	86.21%
kiša	30.06%	53.11%	39.75%	81.27%
snijeg	28.21%	58.11%	37.1%	75.23%

Tablica 6.4: Mjere modela učenog na Cityscapes-u izračunate nad svim validacijskim podskupovima u ovisnosti o vremenskim prilikama.

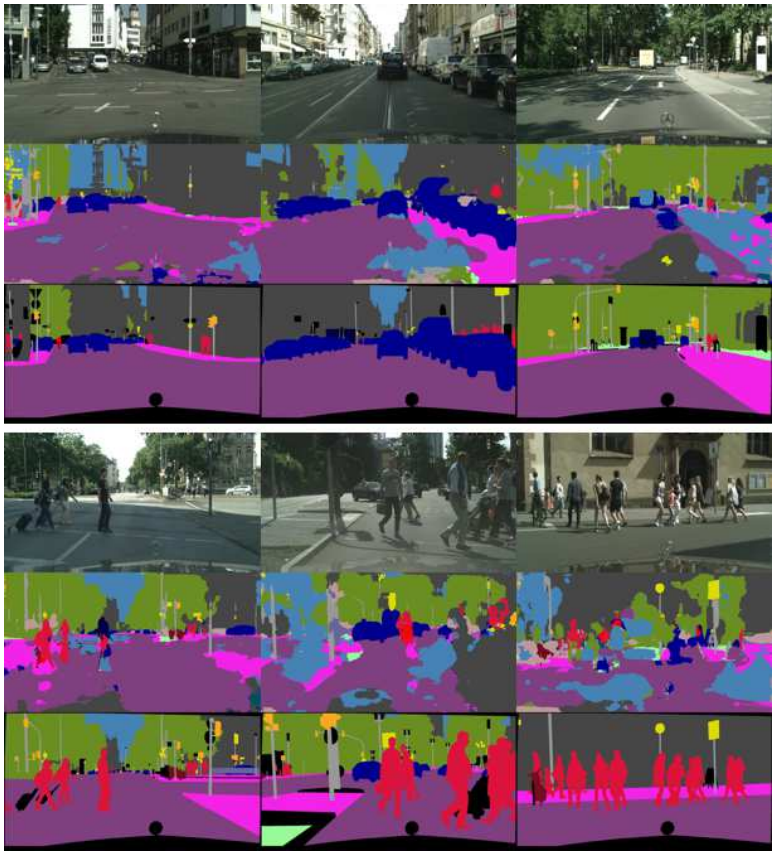
Na slici 6.9 uz fotografije i *ground truth* labele iz skupa podataka ACDC nalaze se

predikcije na ACDC skupu modela učenog na Cityscapes-u. Predikcije nisu zadovoljavajuće zbog kapljica na kameri u kišnim uvjetima, nejasnih granica među klasama u noćnom uvjetima te boje neba u svim uvjetima koja nikada nije svjetloplava zbog čega su predikcije neba uglavnom nasumične. Osim toga, ostale su klase isto zbog vremenskih neprilika obojane drugačije nego na Cityscapes-u zbog čega ih je uz pomoću SwiftNet modela treniranog na Cityscapes-u teško prepoznati.



Slika 6.9: Fotografije i *ground truth* labele skupa podataka ACDC uz predikcije modela učenog na Cityscapes-u.

U obratnom slučaju gdje je model koji je treniran na ACDC skupu evaluiran na Cityscapes-u dobiven je *MIoU* u iznosu od 39.97%. Predikcije tog modela na fotografijama iz Cityscapes skupa nalaze se na slici 6.10. U ovom su slučaju većinom predikcije ceste nasumične.



Slika 6.10: Fotografije i *ground truth* labele skupa podataka Ciytscapes uz predikcije modela učenog na ACDC-u.

6.3.4. Evaluacija SwiftNet modela na fotografijama iz okoline

Evaluacija SwiftNet modela na fotografijama iz okoline pokazuje zadovoljavajuće rezultate. Na slici 6.11 prikazane su predikcije objekata na fotografijama prometnih scena u povoljnim uvjetima evaluirane uz pomoć modela treniranog na Cityscapes skupu podataka, a na slici 6.12 su prikazane predikcije objekana na fotografijama u noćnim uvjetima evaluirane uz pomoć modela treniranog na ACDC skupu podataka.

Slika 6.11: Fotografije iz okoline i predikcije u normalnim vremenskim uvjetima.

Slika 6.12: Fotografije iz okoline i predikcije u noćnim uvjetima.

7. Zaključak

Nadzirano učenje modela na zadatku semantičke segmentacije zahtjevan je zadatak u području dubokog učenja za koji je potrebna odgovarajuća koder-dekoder arhitektura. Odabrana SwiftNet arhitektura uz ResNet okosnicu ostvaruje dobre rezultate na fotografijama prometnih scena iz korištenih skupova podataka. Uz implementaciju uz pomoć programskog okvira Pytorch Lightning proces treniranja djelomično je automatiziran. Model je na skupovima podataka Cityscapes i ACDC ostvario zadovoljavajuće rezultate. Na Cityscapes-u je postignut prosječni omjer presjeka i unije u iznosu od 74.52%, a na skupu ACDC u iznosu od 63.72%.

Daljnji eksperimenti koji bi mogli biti provedeni uključivali bi korištenje drugih okosnica na SwiftNet arhitekturi i dodatne skupove podataka poput skupova Mapillary Vistas[1], Berkeley DeepDrive[2], India Driving Dataset[3], CamVid[4], KITTI[5], GTA5[6] i mnogih drugih.

[1] https://www.mapillary.com/dataset/vistas
[2] https://bdd-data.berkeley.edu/
[3] https://idd.insaan.iiit.ac.in/
[4] https://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/
[5] https://www.cvlibs.net/datasets/kitti/
[6] https://www.v7labs.com/open-datasets/gta5

LITERATURA

[1] Vijay Badrinarayanan, Alex Kendall, i Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, i Bernt Schiele. The cityscapes dataset. U *CVPR Workshop on the Future of Datasets in Vision*, svezak 2. sn, 2015.

[3] 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.

[4] 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, 2012.

[5] Carlo Gatta, Adriana Romero, i Joost van de Veijer. Unrolling loopy top-down semantic feedback in convolutional deep networks. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[6] Shruti Jadon. A survey of loss functions for semantic segmentation. U *2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB)*, stranice 1–7. IEEE, 2020.

[7] Diederik P Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] 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.

[9] 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.

[10] Marin Orsic, Ivan Kreso, Petra Bevandic, i Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, stranice 12607–12616, 2019.

[11] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. U *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, stranice 234–241. Springer, 2015.

[12] 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. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[13] 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)*, October 2021.

[14] Karen Simonyan i Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, i Andrew Rabinovich. Going deeper with convolutions. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 1–9, 2015.

Semantička segmentacija slika korištenjem dubokih modela

Sažetak

Semantička segmentacija u kontekstu računalnog vida provodi dodjeljivanje klase objekta svakom pikselu slike u ovisnosti kojem objektu taj piksel pripada. Za taj se zadatak većinom koriste koder-dekoder arhitekture koje kao okosnicu obično koriste neke od standardnih arhitektura dubokog učenja. U ovom je radu predstavljen pregled nekih od standardnih arhitektura dubokog učenja, zatim pregled koder-dekoder arhitektura i metrika korištenih za semantičku segmentaciju. Za eksperimente je odabrana SwiftNet arhitektura uz programski okvir Pytorch Lightning i skupove podataka Cityscapes i ACDC. Postignuti su očekivani rezultati i provedene su predikcije na validacijskim skupovima.

Ključne riječi: duboko učenje, semantička segmentacija, koder-dekoder arhitekture, Pytorch Lightning, skup podataka Cityscapes

Semantic segmentation of images using deep models

Abstract

Semantic segmentation in the context of computer vision assigns an object class to each pixel on the image depending on which object that pixel belongs to. The encoder-decoder architectures are mostly used for this task along with some of the standard deep learning architectures used as a backbone. This master thesis presents an overview of some standard deep architectures, followed by an overview of encoder-decoder architectures and metrics used for semantic segmentation. SwiftNet architecture with Pytorch Lightning framework were used for the experiments along with Cityscapes and ACDC datasets. Expected results were achieved and predictions were made on validation sets.

Keywords: deep learning, semantic segmentation, encoder-decoder architectures, Pytorch Lightning, Cityscapes dataset