Zahvaljujem prof. dr. sc. Siniši Šegviću i bacc. ing. Jeleni Bratulić na pomoći i savjetima tijekom izrade rada.

Sadržaj

1. Uvod

Računalni vid je područje računarske znanosti i umjetne inteligencije koje se bavi razumijevanjem i prepoznavanjem sadržaja slika ili videozapisa. Kao i razna druga područja u umjetnoj inteligenciji, računalni vid pokušava pomoću računala riješiti probleme koji su za čovjeka jednostavni, no za računala iznenađujuće kompleksni, s ciljem da računalo imitira ljudske sposobnosti. Postoje razne grane računalnog vida, no u ovom radu fokus je na semantičkoj segmentaciji slike.

Semantička segmentacija je proces podjele slike na više segmenata od kojih svaki ima neku značajnost. U tome procesu svakome pikselu slike dodijeljena je neka predefinirana klasa. Stoga, semantičku segmentaciju može se interpretirati kao poseban slučaj klasifikacije gdje ne klasificiramo cijelu sliku, nego pojedine piksele slike. Prema tome, izlaz modela koji radi semantičku segmentaciju neće biti vjerojatnost pojedine klase već maska slike, gdje svaka različita boja pripada određenoj klasi.

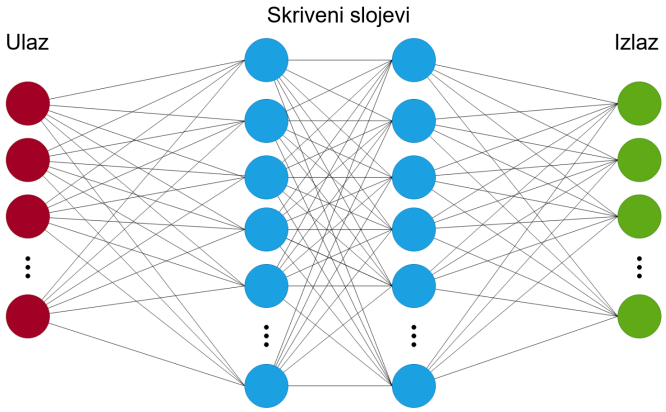
Zanimljiv problem u semantičkoj segmentaciji koji ćemo proučiti u ovome radu jest pomak domene. Pomak domene u semantičkoj segmentaciji je primjena modela treniranog na određenim podatcima, na podatke koji su slični u semantičkom smislu, ali ne dolaze iz istog izvora podataka. Primjer pomaka domene koji ćemo promatrati u sklopu ovog rada koristit će Cityscapes [4] skup podataka kao podatke iz pravog života na kojima mjerimo efekte pomaka domene. Sintetički podatci na kojima ćemo trenirati naš model bit će iz skupa podataka GTA [17]. Cilj nam je uzeti neku arhitekturu za semantičku segmentaciju, npr. SwiftNet [15], trenirati taj model na sintetičkim podatcima te konačno testirati isti model na podatcima iz Cityscapes skupa podatka i prikazati uspješnost modela raznim statističkim mjerama.

2. Duboko učenje

Duboko učenje je grana strojnog učenja koja se temelji na dubokim umjetnim neuronskim mrežama. Pridjev "duboko" dolazi iz toga što neuronske mreže dubokih modela imaju više skrivenih slojeva koji su povezani nelinearnim transformacijama. U riješavanju klasifikacijskih problema u računalnom vidu kao dobro riješenje pokazale su se neuronske mreže temeljene na konvolucijama, odnosno, konvolucijski modeli.

2.1. Umjetna neuronska mreža

Umjetne neuronske mreže [6] su matematički modeli inspirirani biološkim neuronskim mrežama koje formuliraju životinjski mozak. Stoga se elementi mreže nazivaju neuronima, koji su izlazi određenih funkcija. Mreža može imati proizvoljni broj slojeva, a izlaz mreže ovisi o problemu koji rješavamo. Neuronska mreža



Slika 2.1: Shema umjetne neuronske mreže

može, ali ne mora, imati sve neurone iz k-tog sloja povezane sa svim neuronima iz sloja k+1. Takav model nazivamo potpuno povezanim modelom (slika 2.1). Najvažnije svojstvo neuronskih mreža je mogućnost učenja, na čemu se temelji rješavanje problema računalnog vida u domeni klasifikacije i regresije podataka.

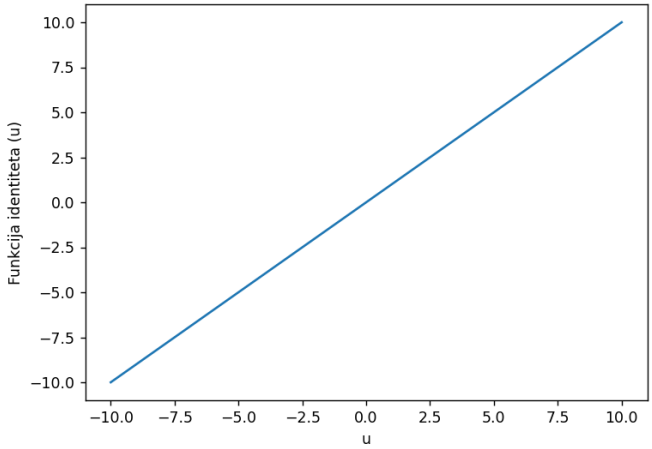
2.1.1. Aktivacijske funkcije

Aktivacijske funkcije [5] u dubokim modelima su nelinearne transformacije sume u pojedinom neuronu koje daju konačni izlaz pojedinog neurona. Aktivacijske funkcije u dubokim modelima moraju biti nelinearne transformacije zato što takve transformacije omogućavaju rješavanje netrivijalnih problema. Linearna kombinacija linearne kombinacije je ponovno linearna kombinacija čime nije moguće riješiti problem klasifikacije linearno-nezavisnih razreda. Neke od najčešće korištenih aktivacijskih funkcija su: funkcija identiteta, sigmoid, softmax, tangens hiperbolni i ReLU.

Funkcija identiteta

Funkcija identiteta je sljedeća:

$$f(u) = u \tag{2.1}$$



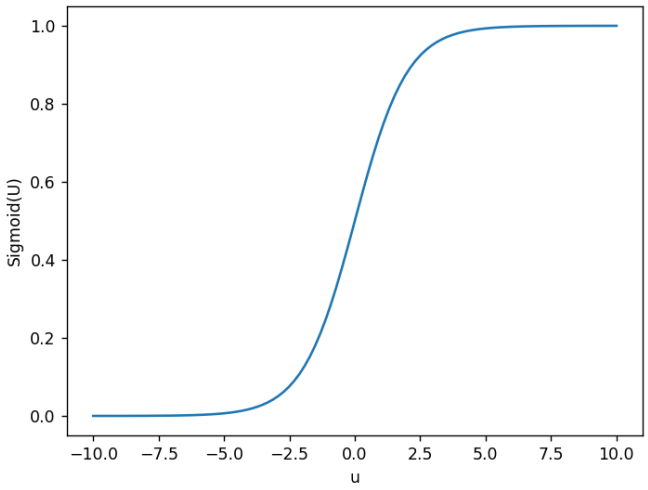
Slika 2.2: Funkcija identiteta

Funkcija identiteta (slika 2.2) je linearno preslikavanje stoga se koristi samo u jednostavnim, plitkim modelima koji rješavaju jednostavne probleme.

Sigmoidna funkcija

Sigmoidna funkcija je sljedeća:

$$f(u) = \frac{1}{1 + e^{-u}} \tag{2.2}$$



Slika 2.3: Sigmoidna funkcija

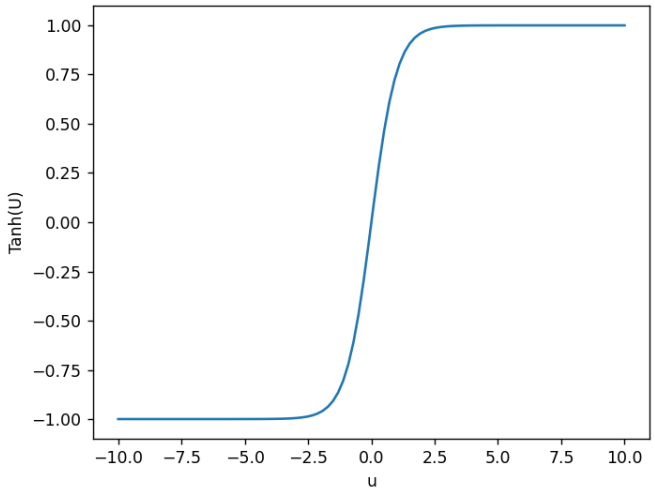
Sigmoidna funkcija (slika 2.3) preslikava brojeve u broj iz intervala $[0, 1]$. Ova funkcija često se koristi u binarnoj klasifikaciji jer njezin rezultat možemo gledati kao vjerojatnost klase. Sigmoidna funkcija je nelinearna transformacija te korištenjem iste omogućavamo modelu uočavanje kompleksnijih, nelinearnih uzoraka u podatcima. No, sigmoidna funkcija ima i svoja ograničenja. Jedan od najvećih nedostataka sigmoidne funkcije jest problem nestajućih gradijenata. Za jako veliku ili jako malu vrijednost u, derivacija sigmoidne funkcije je vrlo mala. Štoviše, maksimalna vrijednost derivacije je samo 0.25. Tijekom prolaska unatrag koristimo pravilo ulančavanja te će svaki faktor našeg umnoška biti broj manji ili jednak 0.25 pa konačni umnožak može biti ekstremno malen. Sve to čini optimizaciju naše mreže vrlo teškom jer iščezavanjem gradijenta naš pomak postaje neprimjetan te mreža može skroz prestati učiti, ili općenito, konvergencija postaje previše spora. Drugi važni nedostatak sigmoidne funkcije je taj što izlazi iz nje nisu centrirani oko nule. Stoga će gradijenti za neki neuron biti svi pozitivni ili svi negativni što rezultira puno sporijom konvergencijom.

Tangens hiperbolni

Funkcija tangens hiperbolni je sljedeća:

$$f(u) = \frac{e^{2u} - 1}{e^{2u} + 1} \tag{2.3}$$

Tangens hiperbolni (slika 2.4) je modificirana sigmoidna funkcija koja može



Slika 2.4: Tangens hiperbolni

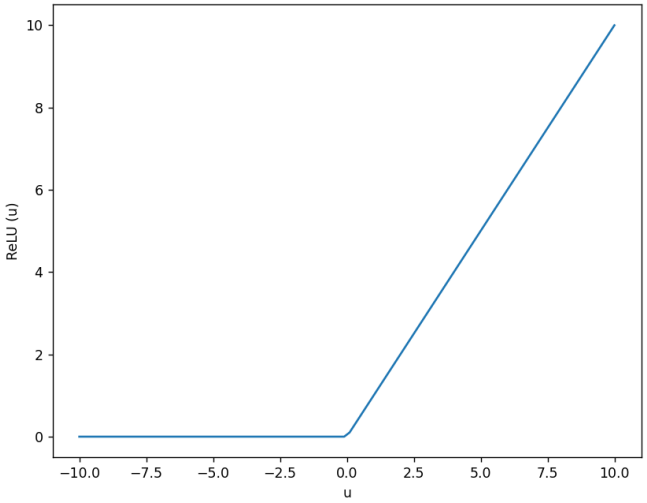
poprimati vrijednosti iz intervala $[-1, 1]$. Zbog tog svojstva tangens hiperbolni rješava problem sigmoidne funkcije koji se tiče centriranosti izlaza, zato što su oni sada centrirani oko nule. No, problem nestajućeg gradijenta je i dalje prisutan.

Zglobnica (ReLU)

Funkcija zglobnica je sljedeća:

$$f(u) = \max(0, u) \tag{2.4}$$

Zglobnica (slika 2.5) danas je jedna od popularnijih aktivacijskih funkcija. Ona djeluje tako da na izlazu vraća 0 ako je ulazna vrijednost negativna, ili samo ulaznu vrijednost ako je ona pozitivna. Glavna prednost zglobnice je ta što je računalno nezahtjevna jer za njen izračun nisu potrebne kompleksne matematičke operacije. Zglobnica je također nelinearna kao i prijašnje navedene funkcije no ona nema problem nestajućeg gradijenta. Ipak, zglobnica nije savršena, a njen nedostatak je problem umiruće zglobnice. Taj problem nastaje kada negativne vrijednosti prolaze kroz zglobnicu zato što se one preslikavaju u 0. Stoga, neki



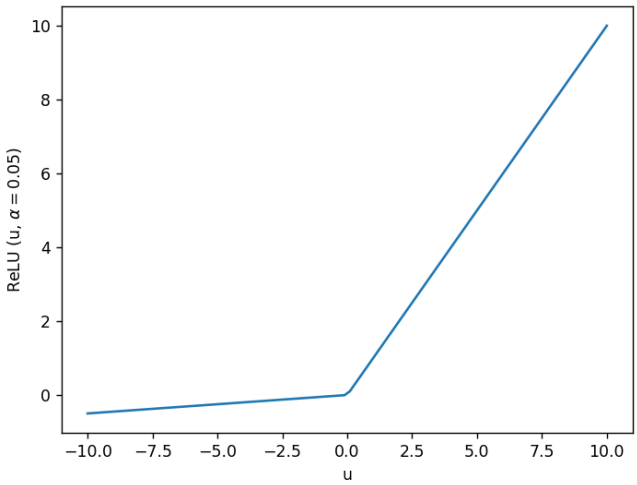
Slika 2.5: Funkcija zglobnica

neuroni postanu takozvani "umrli neuroni" tijekom prolaska unatrag čime onemogućavaju korekciju određenih parametara. Najbolja primjena zglobnice je u skrivenim slojevima zato što je puno jeftinija za računat od sigmoidne funkcije i tangensa hiperbolnog. No, ublažavanje tog problema postižemo takozvanom propusnom zglobnicom. [5]

Propusna zglobnica (Propusni ReLU)

Funkcija propusne zglobnice je sljedeća:

$$f(u) = \max(u, \alpha u) \tag{2.5}$$



Slika 2.6: Funkcija propusne zglobnice

gdje je α neka mala konstanta (često 0.01) (slika 2.6). Dodanom konstantom propusna zglobnica rješava problem saturacije gradijenta tako da vrijednostima manjim od nule daje neku malu vrijednost, kako bi se provelo učenje prolaskom unatrag. Međutim, odabir same konstante α nije jednostavno te je to glavni nedostatak ove funkcije.

2.2. Učenje neuronske mreže

Učenje neuronskih mreža [6] se dijeli na nadzirano učenje, nenadzirano učenje te podržano učenje, od kojih je nadzirano učenje ono koje je korišteno u ovome radu.

Nadzirano učenje je postupak učenja u kojem podatci za treniranje imaju svoje oznake koje naš model pokušava predvidjeti. Za svaki podatak iz skupa za treniranje model pokušava predvidjeti spomenutu oznaku te se zatim ta predikcija uspoređuje s istinitom oznakom. Nakon usporedbe radi se prilagodba parametara modela te se postupak ponavlja dok ne zadovoljimo neki prag točnosti koji smo definirali nekom metrikom i funkcijom gubitka. Također, tijekom treniranja naši podatci su podijeljeni u 3 skupa podataka: skup za treniranje, skup za validaciju i skup za testiranje. Skup za treniranje je dosta veći od ostala dva zato što nam ona služe za provjeru performansi našeg modela te kao osigurač od pretreniranja. Treniranje se odvija kroz epohe, a u svakoj epohi se jednom prolazi kroz cijeli skup za treniranje te se nakon toga mjere performanse na skupu za validaciju. Skup za treniranje se koristi nakon što je treniranje završeno kao skup na kojem mjerimo konačne performanse našeg modela kako bi ga mogli uspoređivati s drugim modelima.

2.2.1. Funkcija gubitka

Funkcija gubitka [6] u učenju modela služi upravo za usporedbu predviđene i istinite oznake. Postoje razne funkcije gubitka s raznim prednostima i nedostatcima, no za ovaj rad je najbitnija takozvana unakrsna entropija.

Unakrsna entropija

Unakrsna entropija je funkcija gubitka koja se najčešće koristi u zadatcima klasifikacije, pa tako i semantičke segmentacije. Funkcija unakrsne entropije je sljedeća:

$$L_{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.6}$$

gdje je y_i istinita oznaka, $\hat{y}_i$ predviđena oznaka dok je C ukupan broj klasa. [10]

Proces treniranja se u konačnici svodi na optimizaciju odabrane funkcije gubitka. U slučaju unakrsne entropije, optimizacija se svodi na traženje minimuma.

2.2.2. Optimizacijski algoritmi

Kako bi pronašli dovoljno dobar minimum, na raspolaganju su nam razni algoritmi optimizacije [6]. Najzastupljeniji od njih je stohastički gradijentni spust (*SGD*) [6] i njegove varijante koji se svodi na računanje gradijenta prema svakom parametru modela. Pošto nam gradijent pokazuje smjer u kojem funkcija najbrže raste, cilj nam je uzeti smjer suprotan gradijentu te pomicati vrijednosti svih parametara modela u iznosu njihovih gradijenta pomnoženih s nekom konstantnom koju nazivamo stopom učenja. Stopa učenja je uglavnom neki broj između 0 i 1, pošto prevelika stopa dovodi do nemogućnosti konvergencije. Najpopularnija varijanta stohastičkog gradijentnog spusta u dubokom učenju je *Adam* [6] inačica, koja je korištena u ovom radu.

Gradijentni spust

Gradijentni spust [3] je iterativni optimizacijski algoritam za pronalazak minimuma funkcije. Gradijent sam po sebi je vektor koji pokazuje u smjeru najbržeg rasta funkcije iz koordinate na kojoj se nalazimo. Stoga, postupak gradijentnog spusta se svodi na računanje gradijenta funkcije te zatim kretanja u smjeru suprotnom od izračunatog gradijenta. Postupak je iterativan zato što se krećemo manjim koracima do minimuma pošto je gradijent u svakoj koordinati drugačiji. Gradijent računamo tako da funkciju parcijalno deriviramo po svim njenim parametrima. Označimo našu funkciju s $\mathbf{f}(\mathbf{x})$, a njen pripadajući gradijent kao $\nabla\mathbf{f}(\mathbf{x})$. Ako funkciju $\mathbf{f}$ aproksimiramo Taylorovim razvojem prvog reda, dobivamo aproksimaciju:

$$\mathbf{f}(\mathbf{x} + \triangle\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \nabla\mathbf{f}(\mathbf{x})^\intercal \triangle\mathbf{x} \tag{2.7}$$

Sada u $\triangle\mathbf{x}$ uvrstimo pomak u smjeru negativnog gradijenta; $\triangle\mathbf{x} = -\epsilon\nabla\mathbf{f}(\mathbf{x})$. Pod pretpostavkom da Taylorova aproksimacija vrijedi, kretanjem u tom smjeru vrijednost funkcije $\mathbf{f}(\mathbf{x})$ trebala bi padati:

$$\mathbf{f}(\mathbf{x} - \epsilon\nabla\mathbf{f}(\mathbf{x})) \approx \mathbf{f}(\mathbf{x}) - \epsilon\nabla\mathbf{f}(\mathbf{x})^\intercal\nabla\mathbf{f}(\mathbf{x}) \tag{2.8}$$

Očigledno je da ćemo kretanjem u smjeru suprotnom od gradijenta doći do lokalnog minimuma. Stopu učenja nam predstavlja hiperparametar ϵ.

2.2.3. Prolaz unatrag

Prolaz unatrag [6] glavni je dio učenja neuronske mreže, odnosno, to je korak u kojem mijenjamo vrijednosti parametara. Postupak se svodi na računanje gradi-

jenta funkcije gubitka za svaki parametar mreže. Kako bi to postigli koristimo se pravilom ulančavanja kod derivacije funkcije:

$$\frac{df(g(x))}{x} = \frac{df(g(x))}{g(x)}\frac{dg(x)}{x} \tag{2.9}$$

U svakoj iteraciji treniranja računamo parcijalne derivacije za sve parametre te zatim dodamo rezultate tih derivacija, pomnožene s $-\epsilon$, svim parametrima mreže:

$$w_i' = w_i - \epsilon\frac{\partial f}{\partial w_i} \tag{2.10}$$

$$b_j' = b_j - \epsilon\frac{\partial f}{\partial b_j} \tag{2.11}$$

gdje je f naša funkcija gubitka, w_i i-ta težina i b_i i-ti parametar pristranosti. No, kako ne bi računali gradijent za svaki podataka pojedinačno, računat ćemo ga za grupe podataka (eng. *batch*) veličine m $(x_1, x_2, x_3, ..., x_m)$. Takav postupak se naziva stohastički gradijentni spust:

$$w_i' = w_i - \frac{\epsilon}{m}\sum_{k=1}^{m}\frac{\partial f_{x_k}}{\partial w_i} \tag{2.12}$$

$$b_j' = b_j - \frac{\epsilon}{m}\sum_{k=1}^{m}\frac{\partial f_{x_k}}{\partial b_j} \tag{2.13}$$

2.2.4. Metrike

Za validaciju i testiranje klasifikacijskih modela koriste se određene statističke metrike. Najbitnija od njih su preciznost [13] i matrica zabune [13].

Preciznost

Preciznost je definirana kao omjer broja točno klasificiranih podataka i ukupnog broja podataka. Koristi se za evaluaciju i testiranje modela te je matematički definirana na sljedeći način:

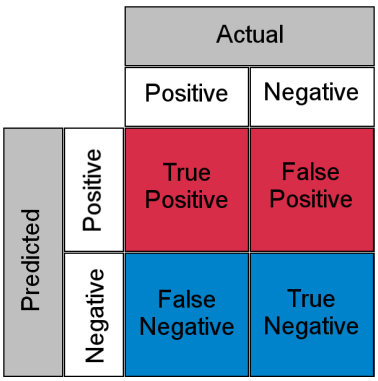
$$A_{cc} = \frac{TP + TN}{FP + FN + TP + TN} \tag{2.14}$$

gdje su:

- TP (eng. *true positive*) - točno pozitivno klasificirani podatci
- TN (eng. *true negative*) - točno negativno klasificirani podatci
- FP (eng. *false positive*) - krivo pozitivno klasificirani podatci
- FN (eng. *false negative*) - krivo negativno klasificirani podatci

Matrica zabune

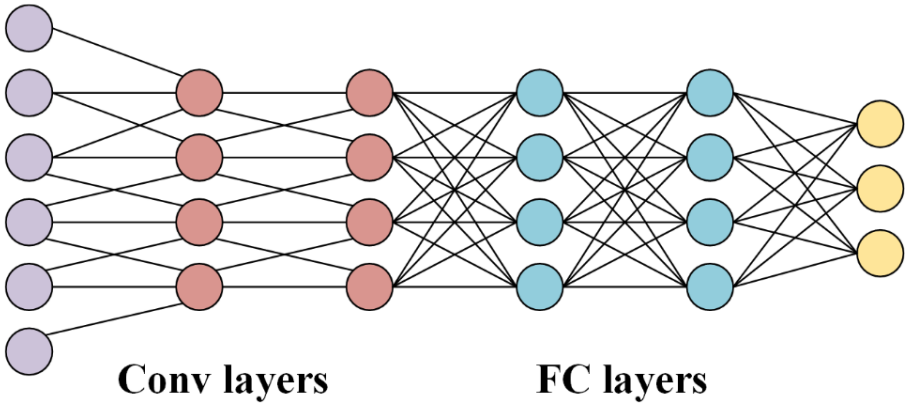
Matrica zabune (slika 2.7) je tablični prikaz za vizualizaciju performansi našeg algoritma. To je kvadratna matrica čiji redovi označavaju istinite klase dok stupci označavaju predviđene klase. Ona nam daje uvid u kako model percipira svaku klasu te postoje li neke sklonosti prema određenim klasama.



Slika 2.7: Matrica zabune za binarnu klasifikaciju[22]

2.3. Konvolucijski modeli

U današnje vrijeme, konvolucijski modeli [11] su temelj računalnog vida. Konvolucijski modeli su neuronske mreže koje se temelje na operaciji konvolucije, koja se provodi nad cijelom slikom. Konvolucijski modeli imaju sposobnost uočavanja raznih značajki na slikama te su stoga najuspješniji modeli u raznim područjima računalnog vida kao što klasifikacija, detekcija objekata i segmentacija. Takvi modeli se sastoje od slojeva konvolucije, sažimanja, aktivacijskih funkcija te potpuno povezanog sloja koji služi za konačnu klasifikaciju. Arhitektura konvolucij-



Slika 2.8: Arhitektura konvolucijske mreže[11]

ske mreže (slika 2.8) je analogna obrascu povezanosti neurona u ljudskom mozgu te je inspirirana organizacijom u vidnom korteksu. Zasebni neuroni reagiraju na impulse u jako ograničenoj regiji poznatoj kao receptivno polje te skup navedenih polja pokriva cijelo vidno područje.

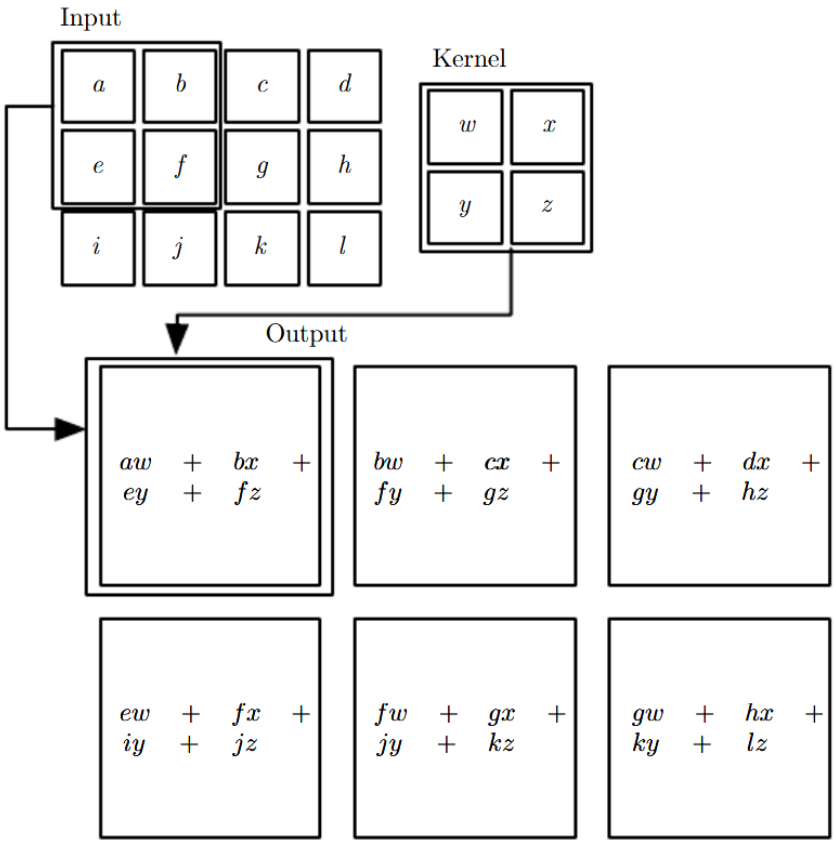
2.3.1. Konvolucija

U dubokim modelima za računanje konvolucije koristi se Frobeniusov unutarnji produkt (slika 2.9) koji je definiran slično kao skalarni produkt, samo što djeluje na matrice istih dimenzija te se označava s $\langle \mathbf{A}, \mathbf{B} \rangle_{\mathbf{F}}$. Formula za računanje Frobeniusovog unutarnjeg produkta je sljedeća:

$$\langle \mathbf{A}, \mathbf{B} \rangle_{\mathbf{F}} = \sum_{i,j} \overline{A_{i,j}} B_{i,j} \tag{2.15}$$

Frobeniusov produkt radimo na ulaznom podatku (na početku je to slika dimenzija visina x širina x broj kanala) i određenoj jezgri (eng. *kernel*) koja

je matrica sa svojim odabranim dimenzijama. Svaka jezgra služi za izvlačenje određene značajke slike koju provlačimo kroz mrežu. Jezgra djeluje na sliku tako da krene s gornjeg lijevog ruba i računa produkt između sebe i matrice piksela istih dimenzija na kojima se trenutno nalazi. Rezultat produkta je skalar i on se upisuje kao element takozvane mape značajki trenutne jezgre. Također, Frobeniusov produkt se računa posebno za svaki kanal slike te se rezultati zbrajaju u konačan rezultat. Nakon toga, naša jezgra se pomiče određeni broj mjesta udesno i ponavlja istu operaciju te tako gradi svoju mapu značajki. Parametri pomaka i dubine samoe jezgre određujemo pri definiciji iste, ovisi što nam ta jezgra predstavlja.



Slika 2.9: Procedura 2-D konvolucije[6]

2.3.2. Sažimanje

Sloj sažimanja služi za smanjivanje dimenzionalnosti konvolucijskih značajki. Time smanjujemo računalnu moć potrebnu za računanje daljnjih značajki te možemo ukloniti šum u podatcima odnosno slučajne i nebitne aktivacije. Najčešći

algoritam sažimanja je maksimalno sažimanje (eng. *max pooling*). Maksimalno sažimanje je postupak u kojem iz mape značajki uzimamo samo najveće vrijednosti iz određenih dijelova mape. Sloj sažimanja funkcionira tako da mapu značajki podijelimo na određeni broj jednakih područja te iz svakog područja izvlačimo maksimalnu vrijednost. Uz smanjivanje kompleksnosti računa, sažimanje nam također pomaže u smanjivanju vjerojatnosti pretreniranja.

2.4. Semantička segmentacija

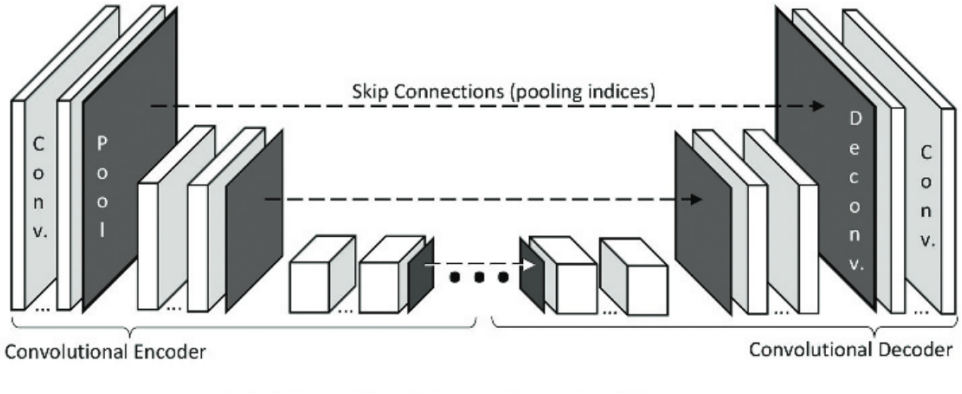
Semantička segmentacija [21] je postupak razdvajanja slike na određeni broj segmenata, gdje svaki segment predstavlja neku klasu. Kolokvijalno, semantičku segmentaciju možemo gledati kao klasificiranje pojedinih piksela, dok su segmenti grupe piksela koji pripadaju istoj klasi. Klase u slučaju semantičke segmentacije su određeni objekti u pravom svijetu, npr. automobil, pješak, cesta itd. Izlaz iz modela koji radi semantičku segmentaciju je takozvana segmentacijska maska koja nam govori o prisutnosti i poziciji objekata neke klase (slika 2.10).

Slika 2.10: Slika na ulazu i njena segmentacijska maska

Semantička segmentacija ima mnoštvo primjena poput prepoznavanje dijelova lica, prepoznavanje raznih objekata u prometu te prepoznavanje i lociranje određenih anomalija na CT ili MRI skenovima [12].

2.4.1. Arhitektura

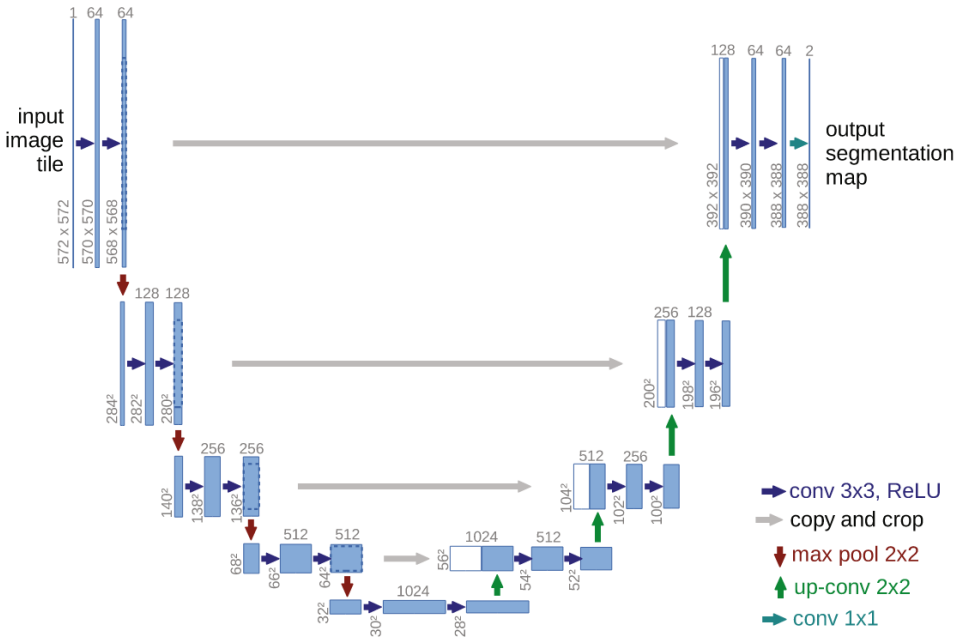
Arhitektura modela za semantičku segmentaciju [23] sastoji se od dva dijela; enkodera i dekodera (slika 2.11). Enkoder služi za izvlačenje značajki iz slike pomoću različitih jezgri dok je dekoder odgovoran za generiranje konačne segmentacijske maske pomoću dobivenih značajki. Enkoder se uglavnom sastoji od neke konvolucijske arhitekture kao npr. ResNet [8]. Najpopularniji pristup za dekodere je takozvana potpuno konvolucijska mreža. Potpuno konvolucijske mreže su inačica na obične konvolucijske mreže gdje zadnji potpuno povezani sloj zamijenimo s 1 x 1 konvolucijama. Zatim nadodamo takozvane dekonvolucije, koje su konvolucije koje povećavaju postojeće dimenzije te se pomoću njih kreira konačna segmentacijska maska. Jedna od najpoznatijih arhitektura za semantičku segmentaciju koja je također temeljena na potpunim konvolucijskim mrežama jest U-Net [18]. U sljedećem dijelu proći ćemo kroz par arhitektura za semantičku segmentaciju.



Slika 2.11: Enkoder-dekoder arhitektura[23]

U-Net

U-Net [18] je konvolucijska neuronska mreža koja je dizajnirana za segmentaciju slike u domeni biomedicine. Ona se sastoji od slojeva kontrakcije i slojeva ekspanzije, stoga poprima svoj prepoznatljivi "U" oblik (slika 2.12).



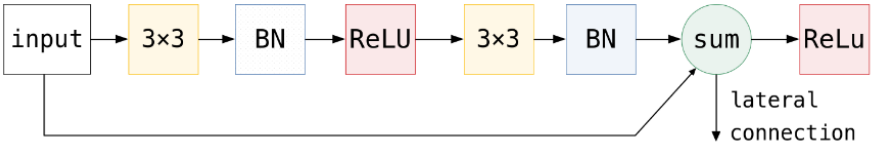
Slika 2.12: U-Net arhitektura[18]

Slojevi kontrakcije su tipični slojevi konvolucijske mreže koji se sastoje od niza 3 x 3 konvolucija za koje se koristi aktivacijska funkcija ReLU te maksimalnih sažimanja reda 2. Tijekom kontrakcije, prostorne informacije su smanjene dok su informacije o značajkama povećane. Slojevi ekspanzije potom kombiniraju

informacije o značajkama i prostorne informacije kroz nekoliko dekonvolucija i konkatenacija sa značajkama viših rezolucija iz slojeva kontrakcije.

SwiftNet

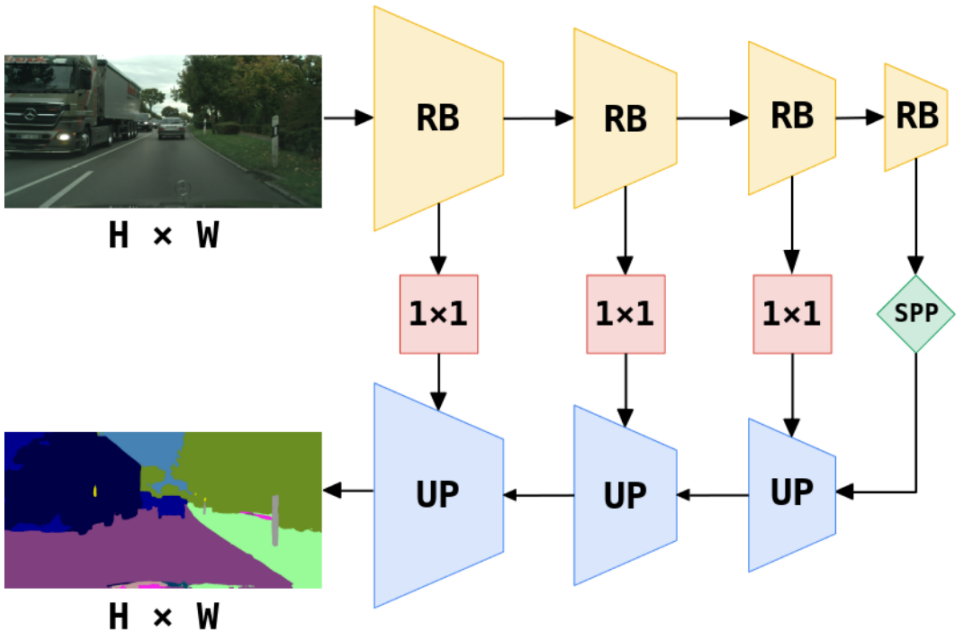
SwiftNet [15] je arhitektura razvijena na FER-u koja je specijalizirana za semantičku segmentaciju u stvarnom vremenu. Ona se sastoji od tri glavna djela: enkoder, dekoder i modul za povećavanje receptivnog polja. Enkoder je uglavnom ResNet-18 [8], pred treniran na ImageNet-u, dok postoji opcija i MobileNet V2, no u ovom radu nije korištena. Dekoder se sastoji od niza modula za naduzorkovanje s lateralnim vezama. Ti moduli imaju dva ulaza: značajke u maloj rezoluciji i lateralne značajke iz prijašnjeg sloja enkodera. Značajke manje rezolucije su prvo naduzorkovanje bilinearnom interpolacijom do rezolucije lateralnih značajki. Zatim se obje značajke miješaju sumom po svim elementima i 3×3 konvolucijom (slika 2.13).



Slika 2.13: Strukturni dijagram zadnje rezidualne jedinice unutar konvolucijskog bloka. Izlaz iz ReLU čvora se dalje propagira u sljedeći rezidualni blok. [15]

Što se tiče modula za povećanje receptivnog polja, postoje dvije opcije: prostorno piramidalno sažimanje i piramidalna fuzija. Prvo ćemo razmotriti jednorazinsku inačicu SwiftNet-a (slika 2.14).

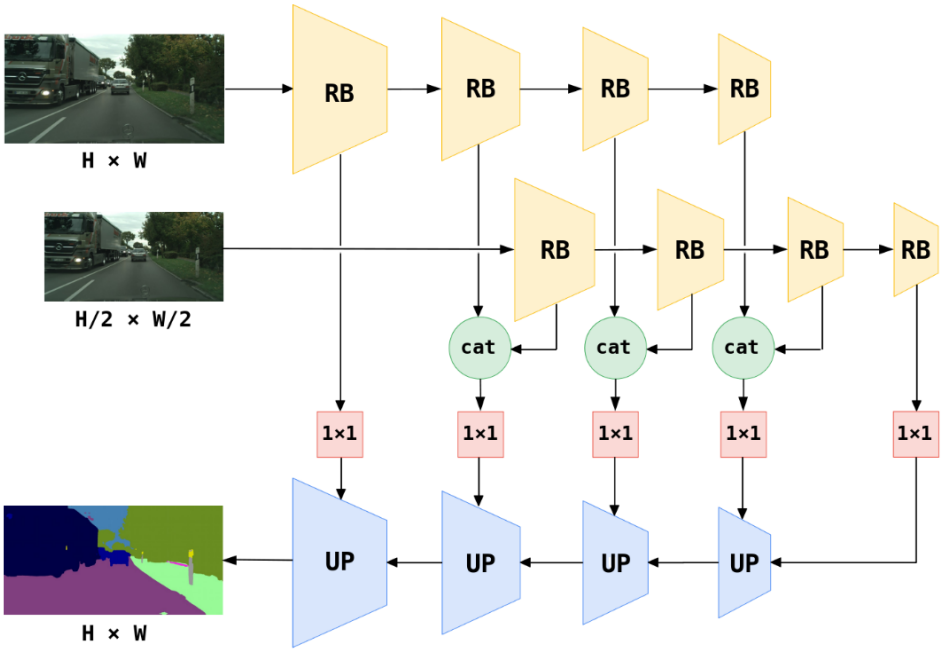
Prva grupa konvolucija proizvodi značajke na 4 puta manjoj rezoluciji od rezolucije ulazne slike. Sve ostale konvolucijske grupe smanjuju rezoluciju za faktor 2 te u konačnici na kraju enkodera značajke su dimenzija $H/32 \times W/32$. Značajke se zatim šalju u sloj s prostornim piramidalnim sažimanjem kako bi se uvećalo receptivno polje. Potom se rezultirajući tenzor šalje u dekoder koji naduzorkovanjem proizvodi segmentacijsku masku. Zbog različite dimenzionalnosti značajki u enkoderu i dekoderu, koristimo $1x1$ konvolucije na lateralnim vezama. Razlika u dimenzionalnosti proizlazi iz toga što svaka grupa konvolucija enkodera ima više konvolucija dok moduli za naduzorkovanje u dekoderu imaju samo jednu konvoluciju. Dekoderovi moduli za naduzorkovanje funkcioniraju u tri koraka: bilinearno naduzorkovanje značajki, zbrajanje rezultata s lateralnim vezama te



Slika 2.14: Arhitektura jednorazinske inačice SwiftNet-a s prostornim piramidalnim sažimanjem. Žuti trapezi predstavljaju konvolucijske grupe enkodera. Zelenim dijamantom je označeno prostorno piramidalno sažimanje dok su crveni kvadrati takozvani *bottleneck* slojevi odnosno konvolucije na lateralnim vezama. Plavi trapezi su slojevi za naduzorkovanje [15].

primjena 3×3 konvolucije.

Za povećanje receptivnog polja osim prostorno piramidalnog sažimanja možemo koristiti piramidalnu fuziju. Njena glavna ideja je primjenjivanje svih gore navedenih procesnih blokova na više istih slika odjednom, no različitih rezolucija. Primjena enkoderskih blokova je identična kao u prošlom primjeru, jedino što se mijenja su blokovi za naduzorkovanje (slika 2.15). U ovom modelu blokovima za naduzorkovanje dodajemo sve mape značajki iz enkoderskih blokova koji imaju odgovarajuću dimenziju.



Slika 2.15: Arhitektura inačice SwiftNet-a s piramidalnom fuzijom. Ovdje su enkoderski blokovi dijeljeni na svim piramidalnim razinama. Značajke istih rezolucija su konkatenirane [15].

2.4.2. Metrika

Budući da semantička segmentacija klasificira pojedine piksele, korištenje obične preciznosti kao metrike nije idealno pošto možemo dobiti vrlo velike brojke ako naš model dobro klasificira velike pozadinske klase ili ako je općenito zastupljenost klasa neravnomjerna. Stoga se za semantičku segmentaciju uvodi nova metrika poznata kao Jaccardov indeks [2]. Definicija Jaccardovog indeksa je sljedeća:

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

gdje nam A i B predstavljaju točnu i predviđenu segmentacijsku masku. Jaccardov indeks koristimo pojedinačno za sve klase te zatim računamo srednju vrijednost svih Jaccardovih indeksa kao konačnu metriku.

3. Pomak domene

Neuronske mreže zahtijevaju veliku količinu podataka za treniranje. Ručno označavanje podataka stoga je težak i skup posao. Također, neuronska mreža ima dobre performanse samo ako su testni podatci iz iste distribucije kao i oni za treniranje. Na primjer, skup podataka kreiran sa slikama s mobitela ima znatno drugačiju distribuciju od slika s DSLR kamere. Na takvim zadatcima tradicionalne metode prenošenja znanja podbacuju. Promjenu distribucije testnih podataka nazivamo pomak domene [9].

Pomak domene koji istražujemo u ovome radu jest treniranje modela za semantičku segmentaciju na sintetičkim podatcima te testiranje tog modela na podatcima iz stvarnog svijeta s jednakim klasama.

3.1. Prijenosno učenje

Prijenosno učenje [16] je tehnika strojnog učenja kojom model stvoren i treniran za određeni problem koristimo kao početni model za rješavanje nekog drugog problema. Koristeći prijenosno učenje, naš model bi u teoriji trebao puno brže i bolje konvergirati [14]. Razlog tome je taj što duboki modeli u početnim slojevima uče prepoznavati osnovne oblike i uzorke na slikama koji su često prisutni u raznim drugim domenama. No kako bi prijenosno učenje bilo uspješno, naš prostor značajki između podataka dvaju modela bi u teoriji trebao biti sličan, odnosno, distribucije podataka bi trebale biti slične. Na primjer, model koji je treniran za prepoznavanje automobila bi trebao biti dobar početni model nekog drugog modela kojemu je cilj prepoznavati kamione. Prijenosno učenje je od posebne koristi kada imamo nedovoljan broj podataka kao što je to česti slučaj u semantičkoj segmentaciji.

3.1.1. Poopćavanje domene

Poopćavanje domene [7] pokušava iskoristiti znanje iz jedne ili više domena kako bi naš model mogao generalizirati neviđene domene. Općenito, kada treniramo model na jednom skupu podataka i primjenjujemo ga na drugi, pretpostavljamo da je distribucija oba skupa jednaka, no u stvarnosti je to rijetko slučaj. Na primjer, u klasifikaciji slika slike iz različitih izvora mogu biti prikupljene pri različitim uvjetima (svjetlina, perspektiva, pozadina itd.). Klasifikatori trenirani samo na jednoj domeni imaju jako loše performanse na neviđenim domenama. Taj problem se naziva problemom poopćavanja domene. U svrhu rješavanja ovog problema razvili su se razni algoritmi prilagođavanja domene.

U ovom radu razmatramo poopćavanje domene kada model treniramo na sintetičkim podatcima koji imaju isti broj klasa kao i testni podatci iz stvarnog svijeta. U eksperimentima ćemo vidjeti kako se model treniran isključivo na sintetičkim podatcima ponaša na tim stvarnim podatcima.

3.1.2. Prilagođavanje domene

Prilagođavanje domene [9] je oblik prijenosnog učenja u kojem nastojimo trenirani model koristiti na podatcima čija je domena slična domeni na kojoj je model treniran. Domene moraju imati identičan prostor značajki, no različite distribucije. Konvencionalni algoritmi za strojno učenje loše se prilagođavaju tom pomaku distribucije podataka.

Prilagođavanje domene dijeli se na tri vrste: nenadzirano, polu-nadzirano i nadzirano prilagođavanje domene. U ovom radu fokus je na nadziranom prilagođavanju domene, odnosno, svi primjeri iz obje domene imaju svoje pripadajuće oznake. Postoje razni algoritmi za smanjivanje sraza između dvaju domena, kao na primjer *DAL* [9] (*eng. Domain Adversarial Learning.*) To je skup tehnika koje pokušavaju naučiti domensko neovisne reprezentacije podataka kako bi model mogao dobro djelovati na više sličnih domena s pomakom distribucije podataka. Jedan od takvih algoritama je *DANN* [9] (*eng. Domain Adversarial Neural Network*) koji integrira igru dva igrača u prilagođavanje domene. Prvi igrač je diskriminator domene D treniran za razlikovanje značajki izvorne domene od značajki ciljne domene, a drugi je igrač generator značajki ψ koji je istovremeno treniran da zbuni diskriminator domene.

3.2. Sintetički podatci

Veliki dio uspjeha dubokih konvolucijskih mreža se svodi na postojanje dostatne količine označenih podataka. Dok je označavanje podataka za neke zadatke poput klasifikacije lagano i jednostavno, ono može biti podosta skupo i zahtjevno za zadatke poput semantičke segmentacije. U semantičkoj segmentaciji potrebno je raditi označavanje za sve piksele slike te vrijeme koje je potrebno za označavanje jedne slike iz npr. Cityscapes skupa podataka je otprilike sat vremena. Još jedna prepreka u prikupljanju podataka u nekim domenama kao što su medicinske slike je ta što za točne oznake trebamo mišljenje stručnjaka što može biti vrlo skupo.

Jedan od obećavajućih pristupa rješavanju ovih problema su upravo sintetički podatci [19][17]. No, kao što je napomenuto u prošlom djelu, modeli trenirani na takvim podatcima imaju poteškoća s podatcima iz stvarnog svijeta. U ovome radu istražit ćemo performanse modela treniranih na sintetičkim podatcima te efekte dodatnog treniranja na stvarnim podatcima.

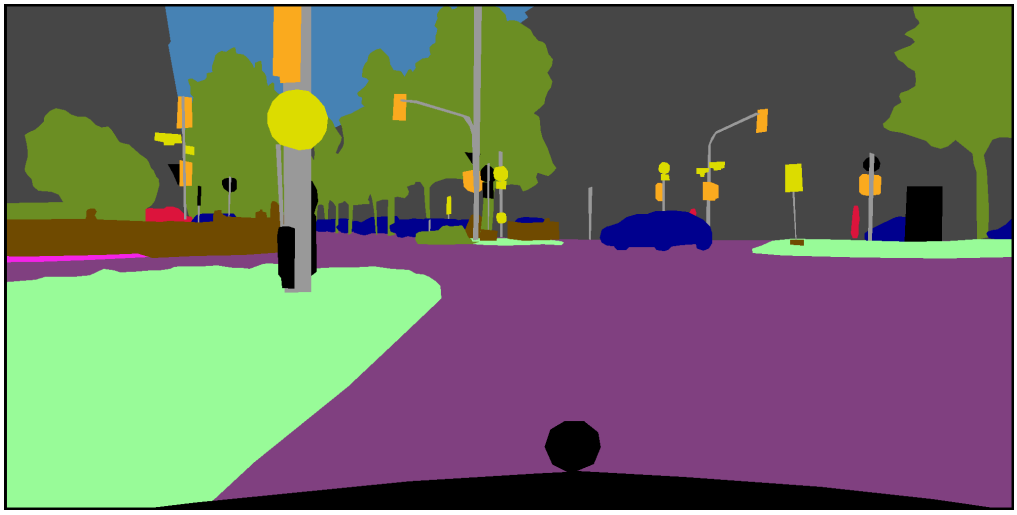
4. Skupovi podataka

U ovome poglavlju navest ću skupove podataka koji su korišteni u sklopu rada. Odlučio sam se posvetiti problemu pomaka domene kod autonomnih automobila te sam za skup podataka iz stvarnog svijeta odabrao Cityscapes [4], a GTA [17] za sintetički skup podataka.

4.1. Cityscapes

Cityscapes je skup podataka specijaliziran za semantičko razumijevanje urbanih scenarija. Sastoji se od 5000 označenih podataka te 20000 ugrubo označenih podataka. U sklopu rada koristio sam isključivo prvi podskup podataka. Cityscapes također ima 30 klasa, od kojih je 11 za ovaj rad bilo nepotrebno pa su bile izbačene iz samih podataka.

Slika 4.1: Primjer slike iz Cityscapes skupa podataka

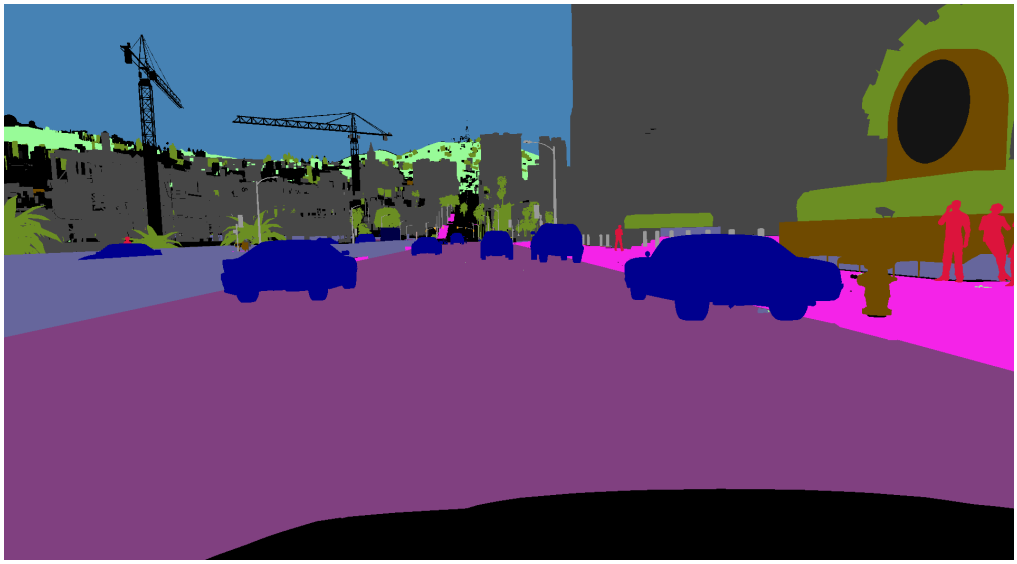


Slika 4.2: Primjer segmentacijske maske iz Cityscapes skupa podataka

4.2. GTA

GTA je sintetički skup podataka specijaliziran za semantičko razumijevanje urbanih scenarija. Njegovi podatci su izvučeni iz komercijalne video igrice *Grand Theft Auto V* koja zbog svoje realistične grafike i velike količine različitih objekata daje odličnu podlogu za kreiranje sintetičkih podataka. Glavna prednost korištenja GTA podataka je ta što ih ima jako puno te njihovo označavanje je neusporedivo brže od označavanja podataka iz stvarnog svijeta kao što su Cityscapes podatci. Za usporedbu, označavanje 25000 GTA podataka je trajalo oko 49 sati, dok je označavanje samo jednog podatka iz Cityscapes skupa zadatak koji traje više od sat vremena. Način na koji je postignuta ta brzina je taj da se umetne omotač između video igrice i operacijskog sustava koji nam omogućava snimanje, modificiranje i reprodukciju prikaznih komandi. Raspršenim adresiranjem određenih izvora za prikaz poput geometrije, tekstura i shadera koje igra šalje grafičkom sklopu, možemo generirati potpise objekata koji ustraju kroz razne scene i razne igračke sjednice [17]. U konačnici, imamo brzi i efikasni način za stvaranje arbitrarne količine podataka bez previše ljudskog napora, a opće je poznata stvar da je glavni problem u zadatcima semantičke segmentacija nestašica podataka. Također pokazalo se da modeli koji su trenirani na ovom skupu podataka te na samo 1/3 skupa CamVid imaju bolje performanse od onih modela koji su trenirani isključivo na cijelom skupu CamVid. Odnosno, treniranje na sintetičkim podatcima je vrlo učinkovito ako je taj skup podataka velik i raznolik.

Slika 4.3: Primjer slike iz GTA skupa podataka



Slika 4.4: Primjer segmentacijske maske iz GTA skupa podataka

5. Implementacija

U radu je korišten programski jezik Python 3 koji ima razne biblioteke za područje dubokog učenja te semantičke segmentacije. Treniranje i evaluacija odrađeni su preko Kaggle[1] sustava koji omogućuje pokretanje Python koda na dostatnim grafičkim karticama. Odabrani GPU za ovaj rad je NVIDIA T4 x2. Odabrana arhitektura je prijašnje opisani *SwiftNet* s ResNet-18 enkoderom. Korištene biblioteke su Numpy[2] i PyTorch[3].

5.1. Numpy

Numpy je fundamentalna biblioteka za računarsku znanost u Pythonu. Ona pruža objekte višedimenzionalnih polja, razne izvedene objekte i puno funkcionalnosti za brze operacije nad poljima. Neke od tih operacija su: matematičke i logičke manipulacije oblicima, sortiranje, označavanje, diskretne Fourierove transformacije, osnovna linearna algebra, osnovne statističke operacije i još mnoštvo drugih korisnih alata [1]. Numpy-eva nevjerojatna brzina je postignuta korištenjem prevedenog C-a kao izvršnog jezika.

5.2. PyTorch

PyTorch je biblioteka za Python razvijena od strane *Meta-e* (prijašnji *Facebook*) koja pruža fleksibilnost u dubokom učenju. Radni tok u PyTorch-u je najbliži mogući do istog u NumPy-u. Razlog za svoju veliku popularnost stekao je zbog sljedećih razloga: lako iskoristiv API, vrlo sličan Numpy-u te dinamičko računanje grafova. PyTorch također pruža potporu za rad s više GPU-eva, vlastite učitavače podataka (eng. *DataLoaders*) i pojednostavljeno pretprocesiranje. Od

[1] https://www.kaggle.com/
[2] https://numpy.org/
[3] https://pytorch.org/

svoje pojave na početku 2016. godine, postao je vrlo popularan u znanstvenim krugovima kao *go-to* biblioteka za duboko učenje. Kao glavne objekte PyTorch koristi takozvane PyTorch tenzore koji jako nalikuju na Numpy višedimenzionalna polja. Jedna od glavnih prednosti PyTorch-a je njegova mogućnost automatske diferencijacije kod računanja gradijenata i prolaska unatrag, što ga čini vrlo pristupačnim okvirom za treniranje i razvoj modela [20]. U sklopu računalnog vida, PyTorch pruža razne funkcionalnosti za baratanje slikama.

6. Eksperimenti

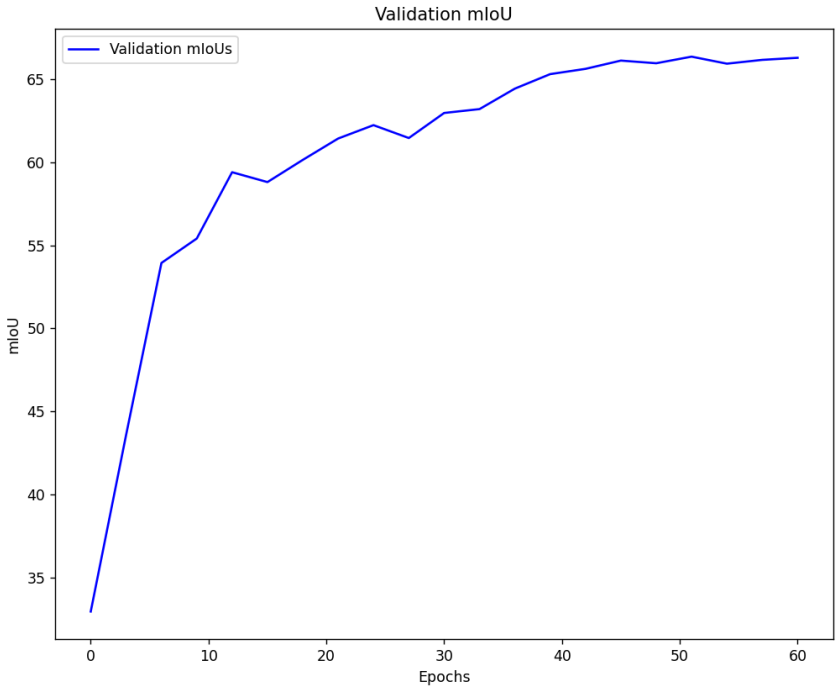
Kako bi prikazao utjecaj pomaka domene prvo ću pokazati performanse modela na sintetičkoj domeni na kojoj je i treniran (GTA),a zatim na pravoj domeni odnosno na Cityscapes podatcima. Nakon toga pokazat ću performanse tog istog modela nakon dodatnog treniranja na Cityscapes podatcima.

6.1. Hiperparametri

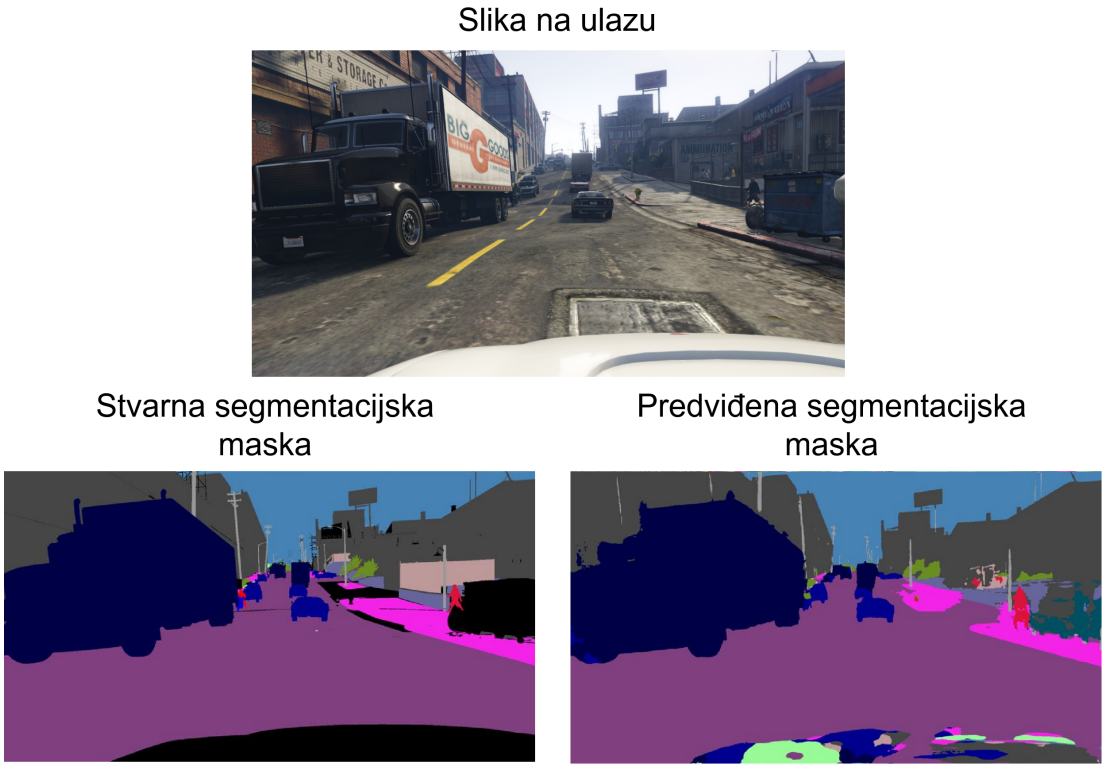
Treniranje se radilo na transformiranim slikama tako da se radilo nasumično obrezivanje slike i njene maske, veličine $768x768$. Broj podataka za treniranje je bio 2500, broj podataka za validaciju 500 te broj podataka za testiranje isto 500. Treniranje je trajalo 60 epoha, a veličina grupe je bila 14 parova slika i maski. Početna stopa učenja je bila 0.0004, a minimalna stopa učenja 0.000001. Validacija se radila nakon svake 3. epohe, uključujući prvu. Korištena funkcija gubitka je unakrsna entropija, a optimizacijski postupak Adam.

6.2. Rezultati na sintetičkoj domeni

Nakon 60 epoha, model na sintetičkim podatcima postiže mIoU veličine 66.36% (slika 6.1) što je vrlo zadovoljavajuće. S većim brojem epoha lako bi se dobili bolji rezultati, no treniranje na 60 epoha je trajalo 9 sati te zbog Kaggle-ovog ograničenja nije bilo moguće trenirati za znatno više epoha.



Slika 6.1: Kretanje mIoU kroz epohe



Slika 6.2: Predikcija maske na skupu podataka GTA. Model je Swiftnet, treniran na skupu podataka GTA.

Iz tablice (slika 6.1) možemo iščitati da model za klase koje predstavljaju veće objekte ima veću preciznost, a razlog tome je što je lakše dobiti veći Jaccardov indeks za jednostavnije i veće grupacije piksela (poput neba, ceste itd.). Za dobro segmentiranje objekata koji su manjih i kompleksnih oblika poput ograda ili bicikla potrebno je duže treniranje pošto su to već manji detalji slika i njihove maske zahtijevaju veću preciznost (slika 6.2).

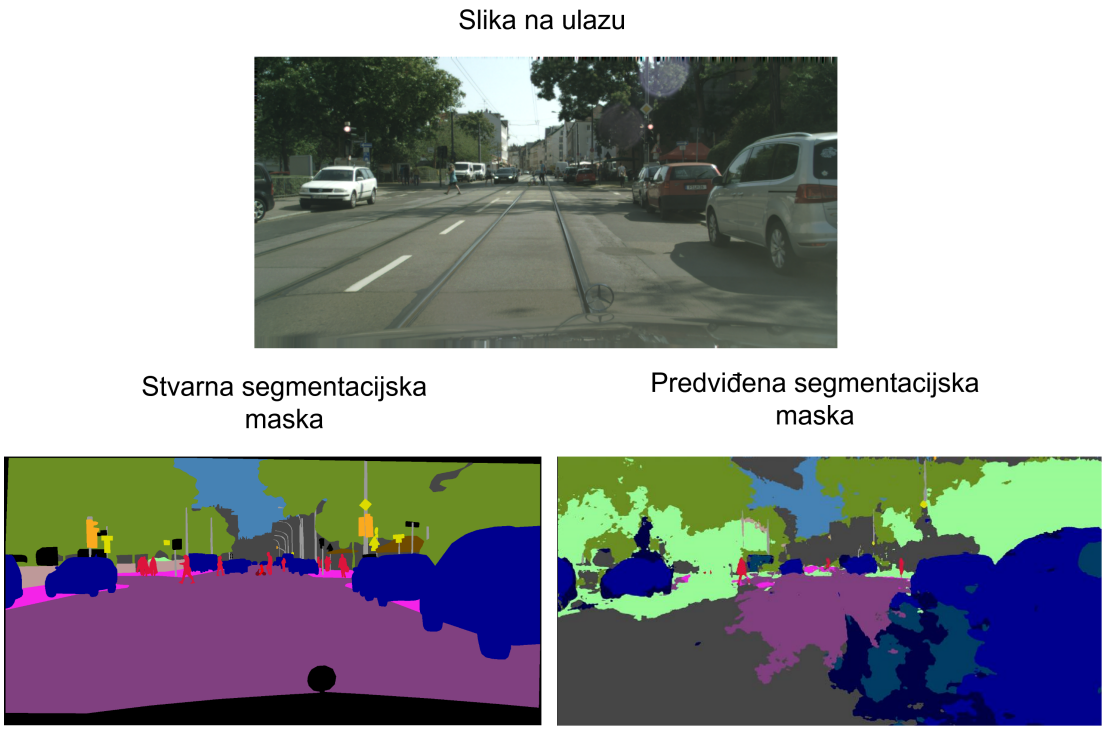
Tablica 6.1: IoU vrijednosti na skupu podataka GTA

Klasa	**IoU (%)**	**mIoU (%)**
Kolnik (eng. *Road*)	98.78	
Pločnik (eng. *Sidewalk*)	84.91	
Građevina (eng. *Building*)	88.43	
Zid (eng. *Wall*)	58.62	
Ograda (eng. *Fence*)	37.74	
Stup (eng. *Pole*)	60.30	
Semafor (eng. *Traffic light*)	59.71	
Prometni znak (eng. *Traffic sign*)	54.68	
Vegetacija (eng. *Vegetation*)	85.31	**66.36**
Prirodni teren (eng. *Terrain*)	77.00	
Nebo (eng. *Sky*)	97.30	
Osoba (eng. *Person*)	58.12	
Vozač (eng. *Rider*)	48.96	
Auto (eng. *Car*)	89.73	
Kamion (eng. *Truck*)	87.70	
Bus (eng. *Bus*)	91.88	
Motocikl (eng. *Motorcycle*)	68.79	
Bicikl (eng. *Bicycle*)	12.79	

6.3. Rezultati na stvarnoj domeni

6.3.1. Direktna evaluacija na podatcima iz stvarnog svijeta

Za sljedeće rezultate uzeo sam model treniran na sintetičkim podatcima (GTA) i direktno ga evaluirao na podatcima iz stvarnog svijeta (Cityscapes) bez ikakvog postupka prilagodbe novoj domeni. Stoga, rezultati su za trećinu lošiji od onih za domenu na kojoj je model treniran (slika 6.3), iako su semantički te dvije domene identične. Postignuti mIoU na Cityscapes podatcima je samo 16.98%.



Slika 6.3: Predikcija maske na skupu podataka Cityscapes. Model je Swiftnet, treniran na skupu podataka GTA.

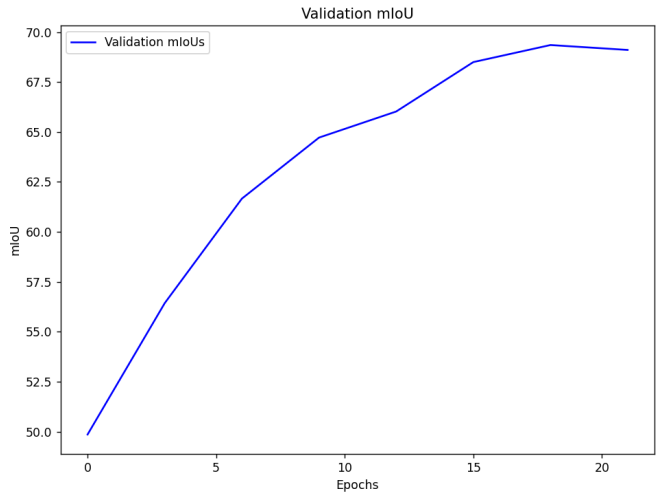
Tablica 6.2: IoU vrijednosti na skupu podataka Cityscapes

Klasa	**IoU (%)**	**mIoU (%)**
Kolnik (eng. *Road*)	20.78	
Pločnik (eng. *Sidewalk*)	2.75	
Građevina (eng. *Building*)	54.48	
Zid (eng. *Wall*)	0.77	
Ograda (eng. *Fence*)	2.71	
Stup (eng. *Pole*)	16.30	
Semafor (eng. *Traffic light*)	19.57	
Prometni znak (eng. *Traffic sign*)	10.57	
Vegetacija (eng. *Vegetation*)	59.96	**16.98**
Prirodni teren (eng. *Terrain*)	2.33	
Nebo (eng. *Sky*)	28.04	
Osoba (eng. *Person*)	40.57	
Vozač (eng. *Rider*)	4.08	
Auto (eng. *Car*)	49.78	
Kamion (eng. *Truck*)	3.10	
Bus (eng. *Bus*)	6.17	
Vlak (eng. *Train*)	0.00	
Motocikl (eng. *Motorcycle*)	0.61	
Bicikl (eng. *Bicycle*)	0.11	

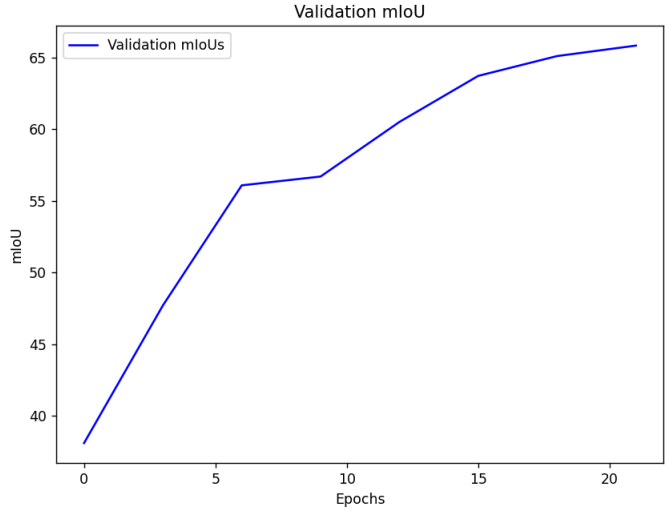
6.3.2. Ugađanje (eng. *fine-tuning*) modela na podatcima iz stvarnog svijeta

U prošlom odlomku smo vidjeli da su rezultati direktne evaluacije poprilično loši. No, u dijelu u kojem sam opisivao sintetičke podatke i GTA skup podataka, naveo sam da modeli pred trenirani na sintetičkim podatcima puno brže i bolje konvergiraju kada ih se dodatno trenira na podatcima iz stvarnog svijeta, u usporedbi s modelima koji su trenirani isključivo na stvarnim podatcima. U ovom odlomku prikazat ću rezultate nakon dodatnog treniranja na Cityscapes skupu podataka. Hiperparametri ostaju nepromijenjeni osim broja epoha, koji je u ovom slučaju postavljen na 20. U nastavku su graf kretanja mIoU metrike tijekom ugađanja na skupu Cityscapes (slika 6.4), graf kretanja mIoU metrike tijekom treniranja modela na skupu Cityscapes (slika 6.5), primjer predikcije segmentacijske maske

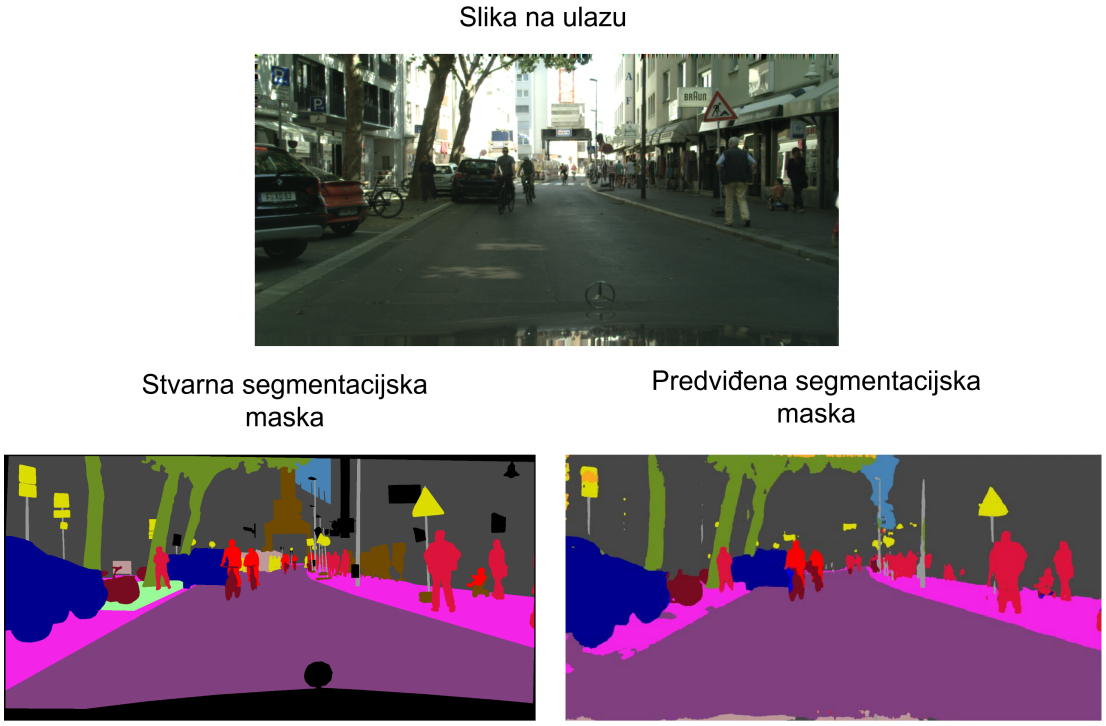
(slika 6.6) te tablica IoU vrijednosti za pojedine klase (slika 6.3).



Slika 6.4: Kretanje mIoU kroz epohe tijekom ugađanja modela na Cityscapes skupu podataka



Slika 6.5: Kretanje mIoU kroz epohe tijekom treniranja modela na Cityscapes skupu podataka



Slika 6.6: Predikcija maske na skupu podataka Cityscapes. Model je Swiftnet, treniran na skupu podataka GTA te ugađan na skupu podataka Cityscapes.

Tablica 6.3: IoU vrijednosti na skupu podataka Cityscapes nakon ugađanja

Klasa	**IoU (%)**	**mIoU (%)**
Kolnik (eng. *Road*)	97.08	
Pločnik (eng. *Sidewalk*)	79.03	
Građevina (eng. *Building*)	90.43	
Zid (eng. *Wall*)	42.47	
Ograda (eng. *Fence*)	49.12	
Stup (eng. *Pole*)	55.96	
Semafor (eng. *Traffic light*)	58.08	
Prometni znak (eng. *Traffic sign*)	69.40	
Vegetacija (eng. *Vegetation*)	91.47	**69.35**
Prirodni teren (eng. *Terrain*)	60.15	
Nebo (eng. *Sky*)	94.06	
Osoba (eng. *Person*)	76.09	
Vozač (eng. *Rider*)	46.65	
Auto (eng. *Car*)	92.54	
Kamion (eng. *Truck*)	64.36	
Bus (eng. *Bus*)	75.50	
Vlak (eng. *Train*)	62.40	
Motocikl (eng. *Motorcycle*)	42.02	
Bicikl (eng. *Bicycle*)	70.81	

Iz rezultata vidimo da je performansa ovog modela prestigla čak i model koji je treniran i evaluiran samo na sintetičkim podatcima. Također, vidimo iz grafa kretanja mIoU metrike (slike 6.4, 6.5) da je konvergencija znatno brža i bolja ako je naš model pred treniran na sintetičkim podatcima slične domene.

7. Zaključak

U ovome radu glavna tematika je bila pomak domene u semantičkoj segmentaciji. Definirali smo što je to neuronska mreža, kako se ona evaluira i optimira. Spomenuli smo funkcije gubitka i optimizacijske postupke kojima te funkcije minimiziramo kao što je *SGD*. Napravili smo kratki pregled konvolucijskih mreža kao aktualnih arhitektura za rješavanje problema u računalnom vidu. Zatim smo objasnili semantičku segmentaciju i ukratko opisali najbitnije arhitekture u kontekstu ovog rada.

Nakon definiranja gore navedenih osnovnih pojmova, definirali smo što je to zapravo pomak domene i neke scenarije u kojima se pojavljuje. Zatim smo definirali specifični pomak domene kojim se bavimo u ovome radu, a to su sintetički podatci. Definirali smo skupove podataka kao izvorne i ciljne domene (GTA i Cityscapes), opisali smo implementacijske detalje te smo se bacili na obradu rezultata.

U eksperimentima smo vidjeli kako modeli trenirani na sintetičkim podatcima, bez prilagođavanja pravoj domeni, imaju ogroman pad performansi na podatcima iz stvarnog svijeta. Razlog za ovakav pad performansi je upravo pomak distribucije podataka, koji je i dalje neriješen problem u semantičkoj segmentaciji. Danas se razvijaju sve bolji algoritmi koji bi smanjili sraz između dvije, semantički identične, ali distribucijski različite domene.

Kako bi pokazali da su sintetički podatci i dalje od velike koristi, koristili smo model pred treniran na sintetičkim podatcima kao početni model za učenje Cityscapes podataka. Koristeći taj model, učenje je trajalo znatno kraće te nakon samo 20 epoha model je pokazao zadovoljavajuće performanse.

Literatura

[1] *What is NumPy?* URL https://numpy.org/doc/stable/user/whatisnumpy.html.

[2] Jeroen Bertels, Tom Eelbode, Maxim Berman, Dirk Vandermeulen, Frederik Maes, Raf Bisschops, i Matthew B. Blaschko. Optimizing the dice score and jaccard index for medical image segmentation: Theory and practice. U Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, i Ali Khan, urednici, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, stranice 92–100, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32245-8.

[3] Petra Bevandić, Marin Oršić, Ivan Grubišić, Josip Šarić, i Siniša Šegvić. Nulta laboratorijska vježba iz dubokog učenja. 2023. URL http://www.zemris.fer.hr/~ssegvic/du/lab0.shtml.

[4] 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. *CoRR*, abs/1604.01685, 2016. URL http://arxiv.org/abs/1604.01685.

[5] Bin Ding, Huimin Qian, i Jun Zhou. Activation functions and their characteristics in deep neural networks. U *2018 Chinese control and decision conference (CCDC)*, stranice 1836–1841. IEEE, 2018.

[6] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Ishaan Gulrajani i David Lopez-Paz. In search of lost domain generalization, 2020.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition, 2015.

[9] Junguang Jiang, Yang Shu, Jianmin Wang, i Mingsheng Long. Transferability in deep learning: A survey, 2022.

[10] Kiprono Elijah Koech. Cross-entropy loss function. 2020. URL `https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e`.

[11] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, i Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022. doi: 10.1109/TNNLS.2021.3084827.

[12] Michael. 2021. URL `https://keymakr.com/blog/semantic-segmentation-uses-and-applications/`.

[13] Sarang Narkhede. Understanding confusion matrix. 2018. URL `https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62`.

[14] Maxime Oquab, Leon Bottou, Ivan Laptev, i Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 1717–1724, 2014.

[15] Marin Oršić, Ivan Krešo, Petra Bevandić, i Siniša Šegvić. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images, 2019.

[16] Sinno Jialin Pan i Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[17] Stephan R. Richter, Vibhav Vineet, Stefan Roth, i Vladlen Koltun. Playing for data: Ground truth from computer games. U Bastian Leibe, Jiri Matas, Nicu Sebe, i Max Welling, urednici, *European Conference on Computer Vision (ECCV)*, svezak 9906 od *LNCS*, stranice 102–118. Springer International Publishing, 2016.

[18] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. U Nassir Navab, Joachim Hornegger, William M. Wells, i Alejandro F. Frangi, urednici, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, stranice 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[19] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, i Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[20] JalFaizy Shaikh. An introduction to pytorch – a simple yet powerful deep learning library. 2018. URL `https://www.analyticsvidhya.com/blog/2018/02/pytorch-tutorial/`.

[21] Martin Thoma. A survey of semantic segmentation, 2016.

[22] Threshold Tom, 2019. URL `https://commons.wikimedia.org/wiki/File:ConfusionMatrixRedBlue.png`.

[23] Irem Ulku i Erdem Akagündüz. A survey on deep learning-based architectures for semantic segmentation on 2d images. *Applied Artificial Intelligence*, 36(1):2032924, 2022. doi: 10.1080/08839514.2022.2032924. URL `https://doi.org/10.1080/08839514.2022.2032924`.

Sažetak

Današnji modeli za semantičku segmentaciju postižu vrlo dobre rezultate na određenim podatcima, no pomak domene je i dalje neriješen problem u računalnom vidu. U ovome radu proučavamo posljedice pomaka domene u kontekstu sintetičkih podatka te koje su koristi i mane u korištenju sintetičkih podataka. U radu su opisani glavni pojmovi u području dubokog učenja relevantnih za ovu tematiku poput neuronskih mreža, konvolucijskih i semantičkih modela. Također je opisano poopćenje domene, prilagođavanje domene i pojam prijenosnog učenja kao bitne pojmove u kontekstu pomaka domene. Korišteni podatci su GTA i Cityscapes, a arhitektura je SwiftNet s ResNet-18 okosnicom. Kod je napisan u programskom jeziku Python 3 koristeći biblioteke NumPy i PyTorch. Cijeli kod je pokrenut preko platforme Kaggle koja omogućava pokretanje koda na GPU jedinicama.

Ključne riječi: Duboko učenje, računalni vid, pomak domene, poopćenje domene, prijenosno učenje, prilagođavanje domene, sintetički podatci, semantička segmentacija.

Sensitivity of semantic segmentation to domain shift

Abstract

State of the art semantic segmentation models perform very well on specific datasets, but domain shift is still an unresolved problem in computer vision. In this paper we observe the effects of domain shift in a specific case of synthetic data and we discuss both the pros and cons of using said data for training semantic segmentation models. The paper goes over the basics of deep learning relevant to the context, such as neural networks, convolutional and semantic segmentation models. Furthermore, the paper also goes over domain generalization, domain adaptation and transfer learning as important concepts in relation to domain shift. Datasets used in this paper are GTA and Cityscapes, while the used architecture is SwiftNet with a ResNet-18 backbone. The code was written in Python 3 using NumPy and PyTorch libraries. The whole project was run on the Kaggle platform, which allows users to run their Python code on powerful GPUs.

Keywords: Deep learning, computer vision, domain shift, domain generalization, transfer learning, domain adaptation, synthetic data, semantic segmentation