SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2002

Slabo nadzirana semantička segmentacija primjeraka primjenom suparničkog učenja

Katarina Blažić

Zagreb, srpanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 15. ožujka 2019.

DIPLOMSKI ZADATAK br. 2002

Pristupnik: **Katarina Blažić (0036485710)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Slabo nadzirana semantička segmentacija primjeraka primjenom suparničkog učenja**

Opis zadatka:

Semantička segmentacija prirodnih scena je važan zadatak računalnog vida s mnogim zanimljivim primjenama. Ovaj rad proučava varijantu tog zadatka gdje model pored vizualnog razreda u svakoj piknji slike određuje i pripadni primjerak razreda. Fokus rada je na učenju sa slabim nadzorom gdje signal za učenje nije dostupan u svakoj piknji, nego je zadan pravokutnim okvirima primjeraka.
U okviru rada, potrebno je istražiti postojeće pristupe iz literature za ostvarivanje semantičke segmentacije na razini instanci. Posebnu pažnju posvetiti slabo nadziranim pristupima. Izraditi izvedbu programskog sustava za učenje i primjenu slabo nadziranog modela za segmentaciju primjeraka. Prikazati i ocijeniti ostvarene rezultate.
Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 28. lipnja 2019.

Mentor:

Prof. dr. sc. Siniša Šegvić

Predsjednik odbora za diplomski rad profila:

Doc. dr. sc. Marko Čupić

Djelovođa:

Izv. prof. dr. sc. Tomislav Hrkać

Zahvaljujem mentoru prof. dr. sc. Siniši Šegviću na savjetima i pomoći tijekom studija, a posebno tijekom izrade diplomskog rada. Veliko hvala mojoj obitelji na bezuvjetnoj potpori.

SADRŽAJ

1. Uvod

Jedan od najzahtjevnijih zadataka računalnog vida je semantička segmentacija. Zadatak semantičke segmentacije je odrediti klasu svakog piksela u slici što vodi k potpunom razumijevanju slike. Danas se najbolji rezultati na tom području postižu dubokim učenjem. Međutim, duboki modeli zahtijevaju veliku količinu označenih podataka za učenje koje nije jednostavno pribaviti i označiti. Duboki modeli za semantičku segmentaciju se najčešće uče nadzirano s oznakama za učenje na razini piksela. Pribavljanje oznaka na razini piksela zahtijeva mnogo uloženog vremena i resursa koji često nisu dostupni. Skup postupak označavanja podataka je potaknuo razvoj polunadziranih i slabo nadziranih metoda učenja za razne zadatke računalnog vida, pa tako i za semantičku segmentaciju. Polunadzirane metode za semantičku segmentaciju kombiniraju podatke označene na razini piksel s neoznačenima podacima, a slabo nadzirane metode upotrebljavaju oznake koje nisu na razini piksela. Za slabo nadziranu semantičku segmentaciju se najčešće koriste oznake poput pravokutnih okvira i klasifikacijskih oznake koje su prikupljene i korištene na drugim zadacima računalnog vida.

Ovaj rad se prvenstveno bazira na primjeni suparnički učenih generativnih modela na zadatku slabo nadzirane semantičke segmentacije. U radu su opisani suparnički učeni generativni modeli koje karakterizira nestabilan proces učenja te suparnički učen generativni model s gubitkom najmanjih kvadrata koji ublažava problem nestajućih gradijenata. Opisane su metode za slabo nadziranu semantičku segmentaciju koje za učenje koriste različite oblike nadzora. Sve opisane metode se baziraju na dubokom učenju, ali se postupak učenja metoda uvelike razlikuje. Potom su opisani suparnički učeni generativni modeli za nadziranu i polunadziranu semantičku segmentaciju. Na posljetku je opisan i model suparničkog učenja za slabo nadziranu semantičku segmentaciju koji je implementiran u sklopu ovog rada. Dani su detalji vezani za implementaciju modela te rezultati dobiveni na Cityscapes skupu podataka.

2. Semantička segmentacija

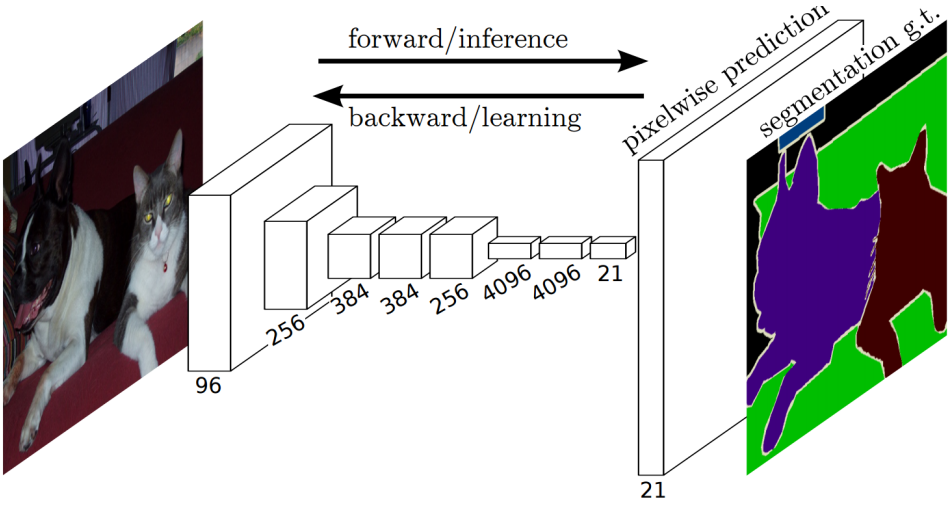
Semantička segmentacija je izazovan zadatak klasifikacije svih piksela u slici. Još izazovniji zadatak je semantička segmentacija na razini instanci koja dodatno grupira sve piksele koji pripadaju jednom primjerku razreda. Primjerice ako u slici imamo tri osobe semantička segmentacija će sve tri osobe segmentirati u istu klasu dok će semantička segmentacija na razini instanci svakoj osobi pridijeliti različitu oznaku. Semantička segmentacija ima primjenu u industriji autonomnih vozila, robotskim sustavima, virtualnoj stvarnosti i na drugim mjestima gdje je važno znati točnu poziciju objekata u prostoru kako bi se zadaci mogli sigurno i točno obaviti. U ovom poglavlju će biti opisani često korišteni duboki modeli za ekstrakciju značajki i semantičku segmentaciju, od kojih će neki biti korišteni i u implementaciji ovoga rada.

2.1. Potpuno konvolucijski model za semantičku segmentaciju

Danas se najbolji rezultati na zadatku semantičke segmentacije postižu dubokim učenjem, točnije dubokim konvolucijskim modelima koji se uče nadzirano s oznakama na razini piksela. Prvi potpuno konvolucijski model za semantičku segmentaciju koji je učen s kraja na kraj je predstavljen u radu [14]. Baza za potpuno konvolucijski model su bili naučeni duboki modeli za klasifikaciju. Autori su zamijenili potpuno povezane slojeve klasifikacijskih modela s konvolucijskim slojevima što im je omogućilo varijabilne dimenzije ulaza i izlaza pogodne za semantičku segmentaciju. Koriste učenje prijenosom (engl. *transfer learning*), to jest koriste naučenu reprezentaciju klasifikacijskih modela te je dodatnim učenjem prilagođavaju za semantičku segmentaciju. Ne koriste nikakvo pretprocesiranje ili postprocesiranje poput uvjetnih slučajnih polja, superpiksela, filtriranja i slično kao što je redovito bio slučaj prije predstavljanja njihovog modela.

Arhitektura korištenog konvolucijskog modela prikazana je na slici 2.1. Svaki po-

datkovni sloj u prikazanom konvolucijskom modelu je dimenzija $h \times w \times d$ gdje su h i w visina i širina sloja, a d njegova dubina. Ulaz modela možemo smatrati prvim podatkovnim slojem pri čemu su visina i širina sloja jednake visini i širini ulazne slike, a dubina broju kanala slike. Sa slike vidimo da s porastom dubine modela prostorne dimenzije konvolucijskih slojeva postaju sve manje, a dubina slojeva sve veća. Da bi izlaz posljednjeg sloja modela dimenzijama odgovarao ulazu potrebno je dobivenu smanjenu reprezentaciju značajki povećati na dimenziju ulaza. U tu svrhu koriste transponiranu konvoluciju. Dodatno su definirali i preskočne veze (engl. *skip connections*) koje povezuju predikcije posljednjeg sloja mreže s plićim podatkovnim slojevima. Zbog toga su konačne predikcije kombinacija grubljih semantičkih informacija iz dubljih slojeva mreže i informacija iz plićih slojeva mreže koji imaju manja receptivna polja te pružaju informacije koje su lokaliziranije i fokusiranije na detalje. Time su znatno poboljšali segmentacijske rezultate.



Slika 2.1: Potpuno konvolucijski model za semantičku segmentaciju [14].

2.2. Mreže Faster R-CNN i Mask R-CNN

Kao što je potpuno konvolucijski model iz [14] svojevrsna prekretnica u arhitekturi dubokih modela za semantičku segmentaciju tako je i arhitektura Faster R-CNN [23] svojevrsna prekretnica u arhitekturi dubokih modele za detekciju objekata. Faster R-CNN je po prvi puta objedinio dvije faze u postupku detekcije objekata u jednu duboku neuronsku mrežu. Prva faza je izdvajanje istaknutih regija u kojima će se tražiti objekti, a druga faza je detekcija objekata unutar tih regija. Prvu fazu obavlja mreža za predlaganje regija, a drugu mreža za detekciju. Dvije mreže dijele konvolucijske značajke te

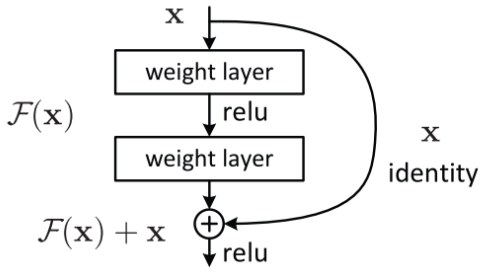
tako tvore jedinstvenu mrežu.

Kao nadogradnja na model Faster R-CNN je nastao model Mask R-CNN [8] za semantičku segmentaciju na razini instanci. Faster R-CNN ima dvije predikcijske grane, jednu za predviđanje klase objekta i drugu za predviđanje koordinata pravokutnog okvira oko objekta. Mask R-CNN dodaje treću granu za predviđanje maske objekta. Kod oba modela se najprije izdvajaju interesne regije dijelom mreže za predlaganje regija. Nakon toga se za svaku regiju ekstrahiraju značajke unaprijed zadanih dimenzija koje se potom procesiraju spomenutim granama. Glavna razlika između modela je u tome da Mask R-CNN uklanja kvantizaciju prilikom ekstrahiranja značajki da bi sačuvao točnije prostorne lokacije koje su jako važne za semantičku segmentaciju. Prva kvantizacija kod Faster R-CNN modela se događa kada se za interesnu regiju koja je zadana koordinatama u ulaznoj slici određuju koordinate pripadnih značajki u mapi značajki koja je na R puta manjoj rezoluciji od ulazne slike. Pri tome se koordinate interesne regije, koja je visine H i širine W, dijele s R i Faster R-CNN zaokružuje koordinate pripadnih značajki koje su visine $H' = \left\lfloor \frac{H}{R} \right\rfloor$ i širine $W' = \left\lfloor \frac{W}{R} \right\rfloor$. Druga kvantizacija se obavlja kada se iz pripadnih značajki ekstrahiraju značajke fiksnih dimenzija $N \times N$. Pri ekstrakciji se pripadne značajke podijele na $N \times N$ regija visine $\left\lfloor \frac{W'}{N} \right\rfloor$ i širine $\left\lfloor \frac{W'}{N} \right\rfloor$ te se iz svake regije uzme maksimalna vrijednost kao vrijednost ekstrahirane značajke. Model Mask R-CNN ekstrahira značajke tako da pripadne značajke koje su visine $H' = \frac{H}{R}$ i širine $W' = \frac{W}{R}$ podijeli na $N \times N$ regija visine $\frac{H'}{N}$ i širine $\frac{W'}{N}$ te unutar svake od regija na 4 lokacije izračuna vrijednost bilinearnom interpolacijom na temelju 4 vrijednosti mape značajki koje su koordinatama najbliže tim lokacijama. Potom za svaku regiju kao vrijednost ekstrahirane značajke odabere najveću ili srednju vrijednost izračunatu na 4 lokacije unutar regije.

2.3. Arhitektura ResNet

Često korištena arhitektura za ekstrakciju značajki je ResNet [7]. Koriste ju i modeli Faster R-CNN i Mask R-CNN. Autori ResNet-a su pokazali da učenje standardnih dubokih modela slabo konvergira za dubine veće od 20 slojeva. Naime, s povećanjem dubine modela se događa da točnost na skupu za učenje dolazi do zasićenja te naglo počinje opadati. U pravilu se ne bi smjelo događati da je točnost dubokih modela manja nego točnost plićih modela koji su njihovi podskupovi, nego bi dublji modeli trebali barem doseći točnost plićih modela. Autori ResNet-a, da bi doskočili tom problemu, predstavljaju gradivne jedinice koje modeliraju rezidualno preslikavanje umjesto standardnog preslikavanja. Neka je $\mathcal{H}(x)$ preslikavanje nekoliko uzastopnih slojeva, a x

ulaz u prvi od tih slojeva. Umjesto da direktno uče preslikavanje $\mathcal{H}(x)$ predlažu učenje rezidualnog preslikavanja $\mathcal{F}(x) := \mathcal{H}(x) - x$. Time funkcija koja se uči postaje $\mathcal{F}(x) + x$. Pretpostavka je da ako je za dublje slojeve preslikavanje identiteta optimalno onda će biti jednostavnije rezidual postaviti na nulu nego postići preslikavanje identiteta nizom uzastopnih nelinearnih preslikavanja. Gradivna jedinica za rezidualno preslikavanje je prikazan na slici 2.2. Vidimo da je formulacija $\mathcal{F}(x) + x$ realizirana preskočnom vezom koja predstavlja preslikavanje identiteta i zbrajanje po elementima. Preskočna veza ne povećava broj parametara modela ni računsku složenost (osim složenosti zbrajanja koja je zanemariva). Ukoliko se dogodi da se dimenzije x i $\mathcal{F}(x)$ ne podudaraju onda je moguće koristiti preskočne veze s nadopunom nulama ili preskočne veze koje obavljaju projekciju ulaza x uz pomoć parametara W_S na dimenzije reziduala $\mathcal{F}(x)$. Projekcijske preskočne veze povećavaju složenost modela i preporučuje ih se koristiti samo za prilagodbu dimenzija. Za dublje modele predlažu se jedinice koje se sastoje od tri sloja umjesto od dva sloja kao što je prikazano na slici 2.2. Redom su to konvolucijski slojevi veličine jezgara 1×1, 3×3 i 1×1. Prvi sloj služi za reduciranje dimenzije ulaza za skuplju operaciju konvolucije u drugom sloju, a treći sloj služi za obnavljanje dimenzije ulaza. U radu su pokazali kako su rezidualne mreže jednostavne za optimizaciju te da poboljšavaju performanse s dubinom mreže više nego što je to do tada bio slučaj.

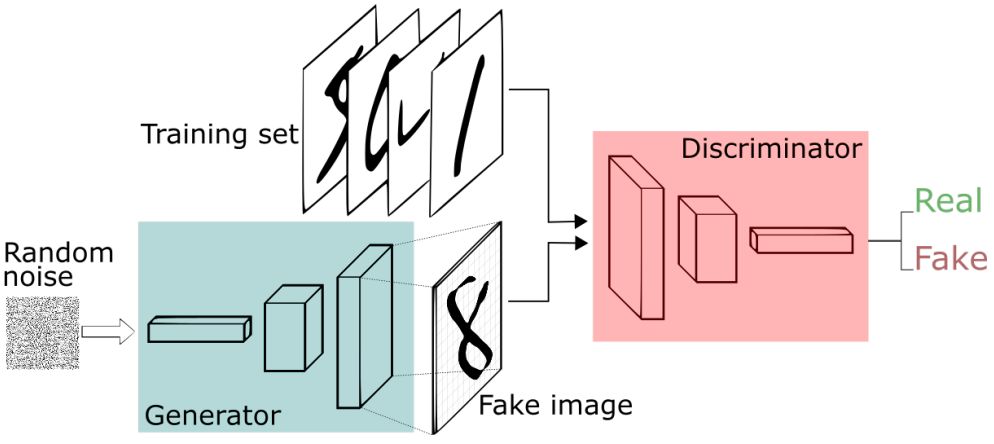


Slika 2.2: Gradivna jedinica za rezidualno preslikavanje [7].

3. Suparnički učeni generativni modeli

Suparnički učeni generativni modeli objedinjuju generativni i diskriminativni model koji se nazivaju generator i diskriminator. Koncept suparnički učenih generativnih modela su predložili Ian Goodfellow i suradnici u radu [6]. Generator koriste za generiranje podataka koji bi trebali biti što sličniji podacima iz skupa za učenje. Generatoru suprotstavljaju diskriminator koji se koristi za klasifikaciju podataka iz skupa za učenje i generiranih podataka. Modeli igraju igru generiranja i klasificiranja podataka kroz koju nastoje približiti distribuciju generiranih podataka distribuciji stvarnih podataka te postići da generirani podaci izgledaju realistično. Generator nema izravan doticaj s podacima iz skupa za učenje nego sve informacije o njima dobiva preko diskriminatora.

Generator i diskriminator su najčešće modelirani kao duboke neuronske mreže. To omogućava učenje modela postupkom propagacije pogreške unatrag i zaobilazi složene postupke optimizacije generativnih modela. Na slici 3.1 je prikazan izgled općenitog suparnički učenog generativnog modela. Kao što je vidljivo sa slike ulaz u



Slika 3.1: Suparnički učen generativni model [26].

generator je vektor šuma z koji je dobiven uzorkovanjem iz normalne ili uniformne distribucije $p_z(z)$. Vektor šuma prolazi kroz generator definiran s parametrima θ_g. Diferencijabilna funkcija generatora $G(z;\theta_g)$ preslikava šum z u prostor podataka i tako

implicitno definira distribuciju generatora p_g. Zadatak generatora je što više približiti distribuciju p_g distribuciji stvarnih podataka iz skupa za učenje p_{data}. Njegov suparnik diskriminator koji na ulazu prima podatak x iz jedne od dviju distribucija definiran je diferencijabilnom funkcijom $D(x; \theta_d)$ s parametrima θ_d. On za svaki x na izlazu daje skalar koji interpretiramo kao vjerojatnost da podatak dolazi is skupa za učenje. Sukladno tome vjerojatnost da podatak dolazi iz distribucije p_g dobijemo kao $1 - D$. Oba modela se uče pomoću sljedeće kriterijske funkcije:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] . \quad (3.1)$$

Kriterijska funkcija definira gubitak binarne unakrsne entropije za diskriminator. Iz kriterijske funkcije je vidljivo da diskriminator želi maksimizirati vjerojatnost D za podatke iz skupa za učenje te istodobno želi da vjerojatnost D za generirane podatke bude što bliža nuli. Generator želi minimizirati vjerojatnost $1 - D$ za generirane podatke, odnosno želi maksimizirati vjerojatnost da dolaze iz skupa za učenje.

Pokazalo se da kriterijska funkcija dana formulom (3.1) ne daje dovoljno snažne gradijente za generator na početku postupka učenja. Razlog tome je činjenica da se na početku učenja generirani primjeri jako razlikuju od stvarnih primjera pa diskriminator s velikom vjerojatnošću detektira generirane primjere. To uzrokuje da izraz $\log(1 - D(G(z)))$ poprimi jako malene vrijednosti te dolazi do zasićenja. Zato se u praksi češće trenira generator da maksimizira izraz $log(D(G(z)))$ umjesto da minimizira izraz $\log(1 - D(G(z)))$. Stoga se za generator koristi sljedeća kriterijska funkcija:

$$\max_G V(G) = \mathbb{E}_{z \sim p_z(z)} [\log D(G(z))] . \quad (3.2)$$

3.1. Postupak učenja suparničkih generativnih modela

Generator i diskriminator se mogu učiti bilo kojim od postupaka temeljenih na gradijentnoj optimizaciji. Postupkom učenja se optimiziraju parametri modela kako bi što bolje zadovoljili kriterijsku funkciju danu formulom (3.1). U model se može ugraditi i neka druga kriterijska funkciju, važno je samo da je diferencijabilana. Uobičajeno je naizmjenice osvježavati parametre dvaju modela. Postupak učenja suparničkih generativnih modela dan je algoritmom 1 u kojem je naveden postupak učenja stohastičkim gradijentnim spustom.

Algoritam 1: Postupak učenja suparničkog generativnog modela stohastičkim gradijentnim spustom uz mini-grupu veličine m. Parametar k je hiperparametar algoritma koji se u praksi često postavi na 1.

za *1 ... ukupan_broj_ iteracija* **čini**
  za *1 ... k* **čini**
    Uzorkuj mini-grupu od m primjera vektora šuma $\left\{z^{(1)}, ..., z^{(m)}\right\}$ iz distribucije $N(z)$
    Uzorkuj mini-grupu od m primjera $\left\{x^{(1)}, ..., x^{(m)}\right\}$ iz distribucije $p_{data(x)}$
    Osvježi parametre diskriminatora zbrajanjem njegovog stohastičkog gradijenta:

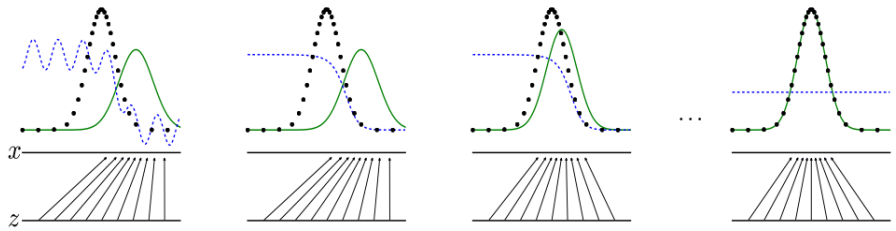
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log(D(x^{(i)})) + \log(1 - D(G(z^{(i)})))\right] \tag{3.3}$$

  kraj
  Uzorkuj mini-grupu od m primjera vektora šuma $\left\{z^{(1)}, ..., z^{(m)}\right\}$ iz distribucije $N(z)$
  Osvježi parametre generatora zbrajanjem njegovog stohastičkog gradijenta:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D(G(z^{(i)}))) \tag{3.4}$$

kraj

Tijek učenja modela u jednodimenzionalnom prostoru je ilustriran četirima skicama na slici 3.2. Prikazane su decizijska granica diskriminatora (plava iscrtkana linija), distribucija generatora (puna zelena linija), distribucija stvarnih podataka iz skupa za učenje (istočkana crna linija) te preslikavanje šuma z u prostor podataka x. Na prvoj po redu skici je prikazano stanje nakon određenog broja iteracija algoritma kada se distribucija generatora počela približavati distribuciji skupa za učenje. Na drugoj skici je prikazano stanje nakon osvježavanja parametara diskriminatora. Os-



Slika 3.2: Tijek učenja suparničkog generativnog modela u jednodimenzionalnom prostoru [6]. Plava iscrtkana linija predstavlja decizijsku granicu diskriminatora, puna zelena iscrtkana linija predstavlja distribuciju generatora te istočkana crna linija predstavlja distribuciju stvarnih podataka iz skupa za učenje.

vježavanje parametara je uzrokovalo da se decizijska granica diskriminatora promijeni i da bolje odijeli dvije distribucije. Treća skica prikazuje stanje nakon osvježavanja parametara generatora koje je uzrokovalo pomak distribucije generatora prema decizijskoj granici, odnosno prema distribuciji podataka iz skupa za učenje. Zadnja po redu skica prikazuje stanje po završetku učenja u kojem je postignut globalni optimuma algoritma. Globalni optimum se postiže kada distribucija generatora p_g postane jednaka distribuciji skupa za učenje p_{data}. Diskriminator tada više ne može razlikovati te dvije distribucije i vjerojatnost $D(x)$ jednaka je jednoj polovini za sve podatke.

Proces učenja suparničkih generativnih modela je jako nestabilan. Možemo ga shvatiti kao igru pronalaženje Nashova ekvilibrija gdje je optimalan ishod igre onaj u kojem niti jedan od igrača nema poticaj promijeniti svoju strategiju nakon razmatranja poteza suparnika. Ekvilibrij se uspostavlja kada oba igrača minimiziraju svoj gubitak. Učenje otežava činjenica da su funkcije cilja suparničkog generativnog modela visokodimenzionalne i da nisu konveksne. Nadalje, promjena parametara generatora koji minimiziraju njegov gubitak može uzrokovati rast gubitka diskriminatora i obrnuto. Stoga je problem traženja Nashove ravnoteže igre težak i modeli često prilikom učenja imaju problema pri konvergenciji i pronalasku zadovoljavajućih rješenja. Često zbog nestabilnosti procesa učenja dolazi do degenerirane generativne distribucije (engl. *mode collapse*). Degenerirana generativna distribucija se očituje u tome što generator za sve vrijednosti na ulazu generira vrlo sličan izlaz jer je uspio naučiti samo dio distribucije stvarnih podataka.

3.2. Globalni optimum modela

Kao što je već spomenuto, globalni optimum igre minimizacije i maksimizacije između generatora i diskriminatora se uspostavlja kada distribucija generatora postane jednaka distribuciji stvarnih podataka iz skupa za učenje. U nastavku ćemo iznijeti dokaz koji to potvrđuje.

Razmotrimo najprije kako dobiti optimalan diskriminator za bilo koji fiksirani generator. Optimizacijski postupak diskriminatora se svodi na maksimizaciju sljedeće kriterijske funkcije:

$$\begin{aligned} V(G,D) &= \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz \\ &= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx. \end{aligned} \tag{3.5}$$

Pomoći će nam ako podintegralnu funkciju kriterijske funkcije promatramo kao sljedeću funkciju:

$$f(y) = a\cdot\log(y) + b\cdot\log(1-y). \tag{3.6}$$

Za tu funkciju vrijedi da ako su a i b realni brojevi iz intervala $[0,1]\setminus\{0,0\}$ onda funkcija $f(y)$ postiže svoj maksimum u točki $\frac{a}{a+b}$. S obzirom na to da su p_{data} i p_g iz intervala $[0,1]$ možemo iskoristi tu činjenicu. Kriterijsku funkciju možemo zapisati na sljedeći način:

$$V(G,D) \leq \int_x \max_y p_{data}(x)\log(y) + p_g(x)\log(1-y))dx. \tag{3.7}$$

Vidimo da kriterijska funkcija doseže svoj maksimum kada podintegralna funkcija poprima maksimalan iznos. To se postiže za $y = D(x) = \frac{p_{data}}{p_{data}+p_g}$.

Sada kada znamo optimalan D možemo ga uvrstiti u kriterijsku funkciju za generator:

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{x\sim p_{data}}\left[\log D_G^*(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D_G^*((G(z)))\right] \\ &= \mathbb{E}_{x\sim p_{data}}\left[logD_G^*(x)\right] + \mathbb{E}_{x\sim p_g}\left[\log(1-D_G^*(x))\right] \\ &= \mathbb{E}_{x\sim p_{data}}\left[\log\frac{p_{data}}{p_{data}+p_g}\right] + \mathbb{E}_{x\sim p_g}\left[\log\frac{p_g}{p_{data}+p_g}\right] \\ &= \int_x p_{data}(x)\log\left(\frac{p_{data}}{p_{data}+p_g}\right) + p_g(x)\log\left(\frac{p_g}{p_{data}+p_g}\right)dx. \end{aligned} \tag{3.8}$$

Cilj optimizacijskog postupka za generator je minimizirati tako definiranu kriterijsku funkciju. Globalni minimum te funkcije iznosi $-\log 4$ i postiže se ako i samo ako vrijedi $p_{data} = p_g$. Pokažimo da je to doista tako.

U jednadžbu (3.8) možemo dodati i oduzeti vrijednost $\log 2$ pomnoženu sa svakom od distribucija bez da se išta promijeni:

$$\begin{aligned} C(G) = &\int_x (\log 2 - \log 2) p_{data}(x) + p_{data}(x) \log\left(\frac{p_{data}}{p_{data} + p_g}\right) + \\ &(\log 2 - \log 2) p_g(x) + p_g(x) \log\left(\frac{p_g}{p_{data} + p_g}\right) dx \\ &= -\log 2 \int_x p_g(x) + p_{data}(x) dx + \int_x p_{data}(x) \left(\log 2 + \log\left(\frac{p_{data}}{p_{data} + p_g}\right)\right) + \\ &p_g(x) \left(\log 2 + \log\left(\frac{p_g}{p_{data} + p_g}\right)\right) dx. \end{aligned} \tag{3.9}$$

Ako iskoristimo svojstvo da je po definiciji integral funkcije gustoće vjerojatnosti jednak jedan dobivamo sljedeće pojednostavljenje:

$$-\log 2 \int_x p_g(x) + p_{data}(x) dx = -log2(1+1) = -2\log 2 = -\log 4. \tag{3.10}$$

Nadalje, koristeći svojstva logaritamske funkcije dobivamo:

$$\log 2 + \log\left(\frac{p_{data}}{p_{data} + p_g}\right) = \log\left(2\frac{p_{data}}{p_{data} + p_g}\right) = \log\left(\frac{p_{data}}{\frac{p_{data}+p_g}{2}}\right). \tag{3.11}$$

Uz iskorištena navedena svojstva kriterijska funkcija sada izgleda ovako:

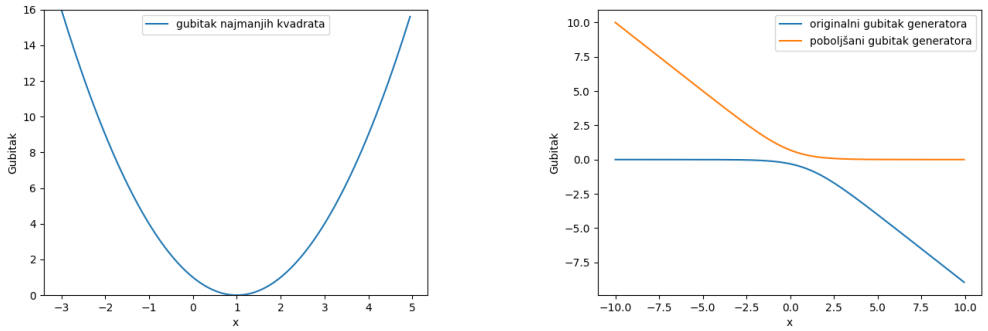
$$\begin{aligned} C(G) &= -\log 4 + \int_x p_{data}(x) \log\left(\frac{p_{data}}{\frac{p_{data}+p_g}{2}}\right) + \int_x p_g(x) \log\left(\frac{p_g}{\frac{p_{data}+p_g}{2}}\right) \\ &= -\log 4 + \mathrm{KL}\left(p_{data} \middle|\middle| \frac{p_{data} + p_g}{2}\right) + \mathrm{KL}\left(p_g \middle|\middle| \frac{p_{data} + p_g}{2}\right) \\ &= -\log 4 + 2 \cdot \mathrm{JSD}\left(p_{data} || p_g\right), \end{aligned} \tag{3.12}$$

gdje KL predstavlja Kullback–Leiblerovu divergenciju, a JSD Jensen–Shannonovu divergenciju. Iznos Jensen-Shannonove divergencije je jednak nula kada su dvije distribucije jednake, a u protivnom je veći od nule. Minimum funkcije iz jednadžbe (3.12) se postiže kada JSD iznosi nula, odnosno kada je distribucija generatora jednaka distribuciji podataka iz skupa za učenje. Time je dokaz gotov.

3.3. Suparnički generativni model s gubitkom najmanjih kvadrata

S obzirom na to da su uočili kako originalna kriterijska funkcija iz jednadžbe (3.1) može dovesti do problema nestajućih gradijenata prilikom treniranja generatora, autori suparnički učenih generativnih modela predlažu korištenje poboljšane kriterijske funkcije iz jednadžbe (3.2). Autori u radovima [16] i [17] upozoravaju kako i poboljšana kriterijska funkcija može dovesti do problema nestajućih gradijenata za vrijeme treniranja te predlažu umjesto gubitka sigmoidalne unakrsne entropije koristiti gubitak najmanjih kvadrata.

Prilikom korištenja poboljšane kriterijske funkcije dolazi do pojave nestajućih gradijenata za generirane podatke koji se nalaze na ispravnoj strani decizijske granice, ali se nalaze daleko od same granice, a samim time daleko i od stvarnih podataka. Za te podatke gubitak sigmoidalne unakrsne entropije će ući u zasićenje dok će gubitak najmanjih kvadrata biti velik i tjerat će podatke bliže decizijskoj granici. Veći gubitak će rezultirati i većim gradijentima prilikom optimizacije generatora što će ublažiti problem nestajućih gradijenata. Usporedbe radi na slici 3.3 su prikazani gubitak sigmoidalne unakrsne entropije za dvije varijante kriterijske funkcije generatora i gubitak najmanjih kvadrata. Gubitak najmanjih kvadrata ulazi u zasićenje u samo jednoj točki (kada je x jednak nuli) dok gubitak sigmoidalne unakrsne entropije ulazi u zasićenje kada je x relativno malen za originalnu kriterijsku funkciju te kada je x relativno velik za poboljšanu kriterijsku funkciju.



(a) Gubitak najmanjih kvadrata **(b)** Gubitak sigmoidalne unakrsne entropije

Slika 3.3: Gubici generatora

Ideja iza korištenja gubitka najmanjih kvadrata je jako jednostavna; taj gubitak bi trebao tjerati generirane podatke koji se nalaze daleko od decizijske granice, bilo da nalaze s ispravne ili s pogrešne strane, bliže samoj granici. Kriterijske funkcije

suparnički učenog generativnog modela definirane s gubitkom najmanjih kvadrata su sljedeće:

$$\begin{aligned}\min_D V_{LSGAN}(D) &= \frac{1}{2}\mathbb{E}_{x\sim p_{data}}\left[(D(x)-b)^2\right] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[(D(G(z))-a)^2\right] \\ \max_G V_{LSGAN}(G) &= \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[(D(G(z))-c)^2\right].\end{aligned} \tag{3.13}$$

Pri tome a i b predstavljaju oznake pridijeljene generiranim, odnosno stvarnim podacima prilikom treniranja diskriminatora, dok c predstavlja oznaku koju generator želi nametnuti diskriminatoru za generirane podatke.

Može se pokazati da optimizacija kriterijske funkcije iz jednadžbe (3.13) minimizira Pearsonovu χ^2 divergenciju između distribucija $p_{data} + p_g$ i $2p_g$ ako parametri a, b i c zadovoljavaju uvijete da $b - c = 1$ i $b - a = 2$. Taj dokaz je sličan onom za globalni optimum suparnički učenih generativnih modela koji pokazuje da optimizacija originalne kriterijske funkcije minimizira Jensen–Shannonovu divergenciju između distribucija p_g i p_{data}. U radu [10] je pokazano kako se optimizacija funkcije iz jednadžbe (3.12) ponaša slično kao i optimizacija Kullback-Leiblerove divergencije između distribucija p_g i p_{data}:

$$\mathrm{KL}(p_g||p_{data}) = -\int_x p_g(x) ln\left(\frac{p_{data}(x)}{p_g(x)}\right) dx. \tag{3.14}$$

KL divergenciju odlikuje svojstvo forsiranja nula (engl. *zero-forcing property*) [17]. Svojstvo forsiranja nula se očituje u tome KL divergencija tjera distribuciju p_g da bude blizu nule na mjestima gdje je distribucija p_{data} blizu nule. To je posljedica činjenice da bi iznos KL divergencije, prema formuli (3.14), bio beskonačan u slučaju da je $p_{data} = 0$ i $p_g > 0$. Nasuprot KL divergenciji Pearsonova divergencije nema svojstvo forsiranja nula što ublažava problem degenerirane generativne distribucije prilikom učenja suparničkog generativnog modela. Štoviše Pearsonova divergencija je sposobna definirati distribuciju na mjestima gdje druga distribucija nije definirana. Pearsonova divergencija koja se minimizira gubitkom najmanjih kvadrata dana je sljedećom formulom:

$$\chi^2_{Pearson}(p_{data} + p_g||2p_g) = \int_x \frac{(2p_g(x) - (p_{data}(x) + p_g(x))^2}{p_{data}(x) + p_g(x)}. \tag{3.15}$$

χ^2 divergencija poprima beskonačan iznos kada su zadovoljeni uvjeti: $p_{data} + p_g = 0$ i

$p_g - p_{data} > 0$. Oba uvjeta nije moguće zadovoljiti jer iz definicije distribucija vrijedi $p_g \geq 0$ i $p_{data} \geq 0$, stoga ne dolazi do degradacije distribucija.

Preporučene vrijednosti oznaka prilikom treniranja su: $a = -1, b = 1, c = 0$. Te vrijednosti zadovoljavaju definirane uvijete za minimizaciju χ^2 divergencije optimizacijom kriterijske funkcije. Eksperimentima se pokazalo da se mogu koristiti i vrijednosti binarnog koda $a = 0, b = 1, c = 1$ koje ne zadovoljavaju uvijete. Međutim, korištenje preporučenih vrijednosti dovodi do brže konvergencije.

4. Slabo nadzirana semantička segmentacija

Jedna od mana dubokih konvolucijskih modela je ta da trebaju veliku količinu podataka za treniranje kako bi postigli dobre rezultate. Taj nedostatak se da ublažiti učenjem s prijenosom znanja, odnosno s uporabom modela koji su predtrenirani na velikim skupovima podataka, primjerice na skupu ImageNet [19] za klasifikaciju koji sadrži 10^6 slika. Pretrenirane modele je potrebno dodatno trenirati za željenu primjenu i na ciljanim podacima, ali mogu davati bolje rezultate u manjem broju iteracija i s manjom količinom podataka. Ipak na tisuće označenih podataka su potrebne da se predtrenirani modeli uspješno prilagode za željenu domenu [11].

Zadaci poput semantičke segmentacije i semantičke segmentacije na razini instanci najbolje rezultate postižu nadziranim učenjem koje zahtijeva oznake za učenje na razini piksela. Semantička segmentacija zahtijeva da se svakom pikselu pridruži oznaka klase dok segmentacija na razini instanci zahtijeva da svi pikseli koji pripadaju jednoj instanci klase budu posebno označeni. Međutim, takva vrsta označavanja podataka zahtijeva mnogo uloženog vremena i resursa. Primjerice za označavanje COCO skupa podataka [13], koji sadrži 328,000 slika, klasifikacijske oznake za 91 razed i segmentacijske oznake za instance iz 80 razreda, bilo je utrošeno preko 70,000 radnih sati. Od toga je približno 20,000 sati utrošeno na označavanje razreda zastupljenih u slici, pri čemu je oznaka razreda stavljana preko jedne instance razreda u slici. Potom je oko 10,000 sati utrošeno na stavljanje oznake razreda preko preostalih instanci razreda u slici, ako je takvih bilo. Preostalo vrijeme je utrošeno na segmentaciju instanci iz 80 razreda. Vidimo da oznake za učenje semantičke segmentacije oduzimaju daleko najviše vremena i samim time zahtijeva najviše uloženih resursa.

Zato se razmatraju polunadzirane i slabo nadzirane metode učenja. Polunadzirane metode kombiniraju potpuno označene podatke s neoznačenima podacima dok slabo nadzirane metode upotrebljavaju oznake koje nisu na razini piksela i koje je moguće brže označiti.

4.1. Oblici slabog nadzora za semantičku segmentaciju

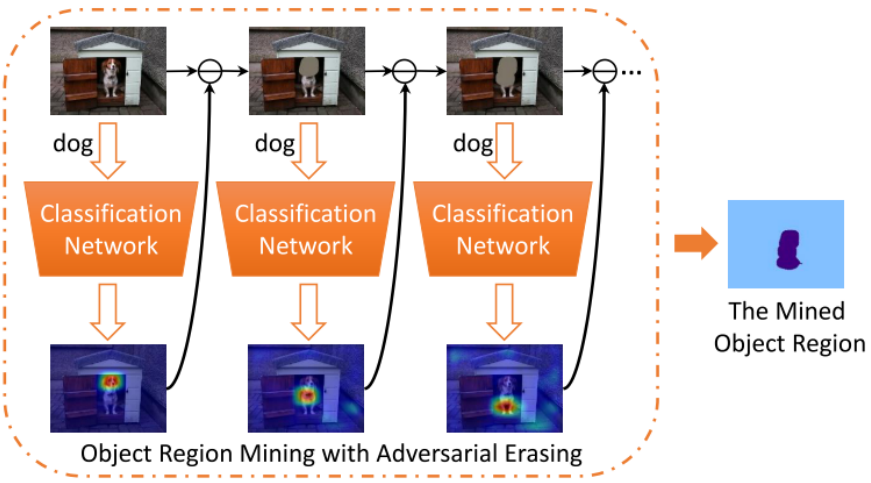
Da bi postupak prikupljanja oznaka za semantičku segmentaciju bio što brži i jeftiniji predložene su razne metode koje prilikom učenja ne koriste oznake na razini piksela nego jednostavnije oblike nadzora koje nazivamo slabi nadzor. Kao slabi nadzor se mogu koristiti klasifikacijske oznake na razini slike, pravokutni okviri oko objekata, točke te šare. U nastavku ovog poglavlja ćemo navesti primjere i načine učenja dubokih modela koji koriste navedene oblike nadzora za semantičku segmentaciju.

4.1.1. Učenje s klasifikacijskim oznakama na razini slike

Najslabiji oblik nadzora među navedenima su svakako klasifikacijske oznake na razini slike. Njih se može najbrže pribaviti, ali pružaju najmanje informacija. Svi drugi oblici slabog nadzora pružaju barem približne informacije o lokaciji objekta kojeg je potrebno segmentirati dok ovaj oblik ne pruža takvu vrstu informacija.

Razmotrit ćemo pristup predstavljen u radu [28] koji koristi ovaj oblik nadzora, a bazira se na dubokom klasifikacijskom modelu. Klasifikacijski modeli se najčešće fokusiraju na malene i raštrkane diskriminativne regije objekata koje su dovoljne za razlikovanje jedne klase objekata od drugih, ali za semantičku segmentaciju je važna lokalizacija cjelovitih regija. Zato su autori da bi premostili jaz između klasifikacije i semantičke segmentacije osmislili pristup u kojem se regije objekta postupno lokaliziraju i proširuju. Uvode proces takozvanog suparničkog brisanja (engl. *adversarial erasing*). Najprije uče duboku klasifikacijsku mrežu s dostupnim klasifikacijskim oznakama. Zatim tu klasifikaciju mrežu koriste da bi lokalizirali najdiskriminativnije regije objekata koje zatim brišu iz ulazne slike. Nakon brisanja performansne klasifikacijske mreže opadaju i mreža mora naučiti nove diskriminativne regije kako bi ispravno klasificirala slike. Postupak se ponavlja više puta i na kraju postupka se regije koje su obrisane tijekom postupka spajaju i tvore segmentacijske maske. Postupak je ilustriran na slici 4.1. Ilustrirana su tri koraka postupka te su za svaki korak prikazani ulaz, klasifikacijska mreža i klasifikacijska aktivacijska mapa dobivena postupkom opisanim u [29]. Klasifikacijske aktivacijske mape su zaslužne za lokalizaciju diskriminativnih regija.

Nadalje, da bi uklonili šum iz segmentacijskih maski i postigli točnije maske na rubovima objekata koriste klasifikacijske aktivacijske mape koje su dobivane tijekom postupka. Pomoću njih i predviđenih klasifikacijskih pouzdanosti formiraju pomoćne segmentacijske mape koje služe za identifikaciju područja s malom klasifikacijskom



Slika 4.1: Formiranje segmentacijske maske postupkom suparničkog prisanja [28].

pouzdanosti. Ta područja su potom uklonjena iz segmentacijskih maski dobivenih postupkom suparničkog brisanja što dokazano uklanja šum u maskama. Segmentacijske maske dobivene ovim postupkom se koriste za nadzirano učenje dubokog segmentacijskog modela.

4.1.2. Učenje s označenim točkama

Prvi korak prema snažnijem nadzoru nakon klasifikacijskih oznaka na razini slike su oznake na razini točaka. Oznake mogu biti po jedna točka za svaku klasu koja se nalazi na slici ili po jedna točka za svaku instancu klase na slici. U radu [25] su pokazali kako je trošak označavanja točaka u odnosu na klasifikacijske oznake na razini slike zanemariv ali dovodi do poboljšanja rezultata semantičke segmentacije. Razmotrit ćemo njihov pristup. Oni su iskoristili gubitak modela za slabo nadziranu semantičku segmentaciju koji koristi klasifikacijske oznake na razini slike [20] i proširili ga gubitkom za označene točke. Gubitak modela [20] omogućava višestruko učenje instanci, odnosno istodobnu segmentaciju više instanci u slici i glasi:

$$\mathcal{L}(S, L) = -\frac{1}{|L|}\sum_{c \in L} \log(S_{m_c c}), \quad \text{uz } m_c = \arg\max_{i \in \mathcal{I}} S_{ic}. \tag{4.1}$$

Pri tome je $L \subseteq \{1, ..., N\}$ skup klasa prisutnih u slici, $\mathcal{I}$ skup piksela prisutnih u slici, S_{ic} softmax izlaz dubokog konvolucijskog segmentacijskog modela za piksel i i klasu

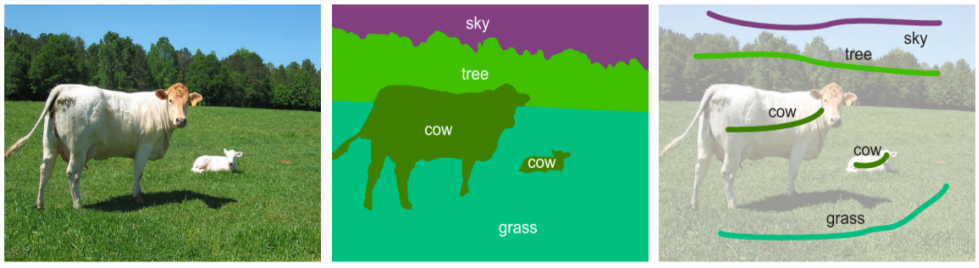
c. Taj gubitak potiče svaku označenu klasu c iz L da ima visoku vjerojatnost za barem jedan piksel m_c u slici. Analogno tome gubitak modela koji koristi oznake na razini točaka glasi:

$$\mathcal{L}(S, L, L') = -\frac{1}{|L|}\sum_{c\in L}\log(S_{m_c c}) - \frac{1}{|L'|}\sum_{c\in L'}\log(1 - S_{m_c c}) - \sum_{i\in\mathcal{I}_S}\log(S_{iG_i}). \quad (4.2)$$

Gdje $L' \subseteq \{1, ..., N\}$ predstavlja skup klasa koje nisu prisutne u slici, $\mathcal{I}_S$ skup označenih piksela gdje je klasa piksela i dana s G_i. Drugi dio gubitka odgovara činjenici da niti jedan piksel ne bi trebao imati visoku vjerojatnost za klase koje nisu prisutne u slici. Posljednji dio gubitka se odnosi na označene piksele i u stvari je to gubitak unakrsne entropije. Rezultati dobiveni učenjem segmentacijskog modela s gubitkom danim formulom (4.2) su dodatno poboljšani u kombinaciji s gubitkom koji se oslanja na potencijal da piksel pripada objektu, a ne pozadini.

4.1.3. Učenje s označenim šarama

Treći, također dosta jednostavan način označavanja, je označavanje šarama. Na slici 4.2 je prikazana usporedba oznaka na razini piksela i šara. Vidno je jednostavnije šarama označiti objekte jer se ne mora voditi računa o njihovim rubovima. To pogotovo vrijedi za objekte koji imaju nejasne granice i nemaju dobro definiran oblik. U usporedbi s klasifikacijskim oznakama na razini slike ovakav način označavanja nudi informacije o lokaciji objekta, dok u usporedbi s pravokutnim okvirima ne daje nikakve informacije o granicama objekta.



Slika 4.2: Usporedba potpunog označavanja i označavanja šarama [12].

U radu [12] je predstavljen model za semantičku segmentaciju nadziran sa šarama. Model se sastoji od grafičkog modela i dubokog konvolucijskog segmentacijskog modela. Grafički model služi za propagaciju oznaka danih šarama na neoznačene piksele, čime se formiraju oznake za nadzirano učenje segmentacijskog modela. Oznake

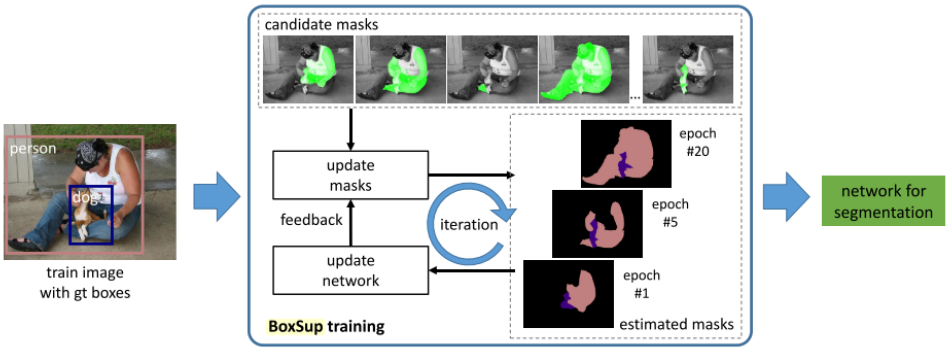
se propagiraju tako da se ulaznu slika dijeli u superpiksele. Superpikselima koji su obuhvaćeni šarama se dodijeli oznaka klase šara, a ostalim superpikselima se dodijeli oznaka na temelju sličnosti s označenim superpikselima i na temelju predikcija segmentacijskog modela. Iz toga vidimo da su dva modela međusobno zavisna jer predikcije segmentacijskog modela imaju utjecaj prilikom propagacije oznaka grafičkog modela i jer grafički model određuje oznake kojima će se učiti segmentacijski model. Optimizacija modela se odvija naizmjenice, to jest prilikom optimizacije parametara jednog modela se fiksiraju parametri drugog modela.

4.1.4. Učenje uz pomoć pravokutnih okvira

Pravokutni okviri su jedna od najboljih zamjena za oznake na razini piksela jer daju informacije o klasi objekta, njegovoj lokaciji te površini koju objekt zauzima u slici. Mogu se označiti otprilike deset puta brže od segmentacijskih maksi što pruža značajnu uštedu. Opisat ćemo dva pristupa koja koriste pravokutne okvire za semantičku segmentaciju. Zajedničko im je da na temelju pravokutnih okvira generiraju segmentacijske maske pomoću kojih nadzirano uče duboke konvolucijske segmentacijske modele.

BoxSup [4] je metoda za semantičku segmentaciju koja za učenje koristi pravokutne okvire, međutim razmatra potpuno nadzirano učenje dubokog segmentacijskog modela za koje su potrebne oznake na razini piksela. Da bi priskrbili takve oznake koriste nenadziranu metodu koja generira segmente iz pravokutnih okvira. Metoda koja se koristi za generiranje segmenata je MCG [21]. Ta metoda je izabrana zato što se njome može dobiti mnogo raznovrsnih segmenata za jedan pravokutni okvir. Postupak učenja BoxSup metode se sastoji od dva koraka koji se opetovano izvode jedan nakon drugoga. U prvom koraku se svakom od pravokutnih okvira iz skupa za učenje pridijeli po jedan generirani segment, a u drugom koraku se uz pomoć tih segmenata potpuno nadzirano uči segmentacijski model. Svakom od generiranih segmenata je pridijeljena oznaka klase koja se mijenja i osvježava za vrijeme učenja. Klasa pridružena segmentu može biti jedna od klasa iz skupa za učenje ili pozadinska klasa koja označava da segment ne predstavlja objekt nego da pripada pozadini. Prilikom pridjeljivanja segmenta pravokutnom okviru kandidati za odabir su segmenti čija klasa se podudara s klasom okvira. Oko svakog kandidacijskog segmenta se formira obuhvaćajući pravokutni okvir te se odabire onaj segment čiji pravokutni okvir ima najveći presjek povrh unije s pravokutnim okvirom iz skupa za učenje. Da se izbjegne lokalni optimum prilikom optimizacije modela može se umjesto segmenta s najvećim presjekom povrh unije

nasumično odabrati jedan od k najbolje rangiranih segmenata. Nakon dodijele segmenata slijedi učenje dubokog segmentacijskog modela. Konkretnije, koristi se duboka konvolucijska segmentacijska mreža zajedno s uvjetnim slučajnim poljima koja dodatno procesiraju izlaz mreže. Parametri koji se optimiziraju tijekom učenja su oznake klasa pridijeljene segmentima te parametri segmentacijskog modela. Prilikom osvježavanja parametara segmentacijskog modela klase pridijeljene segmentima su fiksirane kao što su i prilikom osvježavanja klasa segmenata parametri segmentacijskog modela fiksirani. Prikaz postupka učenja je dan na slici 4.3.



Slika 4.3: Prikaz postupka učenja metode BoxSup [4].

Drugi primjer slabo nadzirane semantičke segmentacije zasnovane na pravokutnim okvirima [11] svoj zadatak postavlja kao problem uklanjanja šuma u oznakama. Koristi duboki segmentacijski model koji uči potpuno nadzirano, kao što je slučaj i kod BoxSup metode. Najprije se formiraju inicijalne semantičke oznake za potpuno nadzirano učenje. Nakon toga se jednu epohu uči segmentacijski model s inicijalnim oznakama. Nakon prve epohe učenja predikcije segmentacijskog modela postaju oznake za nadzirano učenje u drugoj epohi. Metoda učenja je rekurzivna, to jest nakon svake epohe učenja predikcije modela se koriste kao oznake za učenje u sljedećoj epohi. Ideja je da će segmentacijski model u svakoj epohi ukloniti šum iz predikcija dobivenih nakon prethodne epohe te postupno poboljšavati oznake.

Predložili su više načina za formiranje inicijalnih oznaka na temelju pravokutnih okvira. Svi načini su ilustrirani na slici 4.4. Zajedničko im je to da piksele koji nisu zahvaćeni niti jedim pravokutnim okvirom označavaju pozadinskom klasom. Malo detaljnije ćemo objasniti neke od načina:

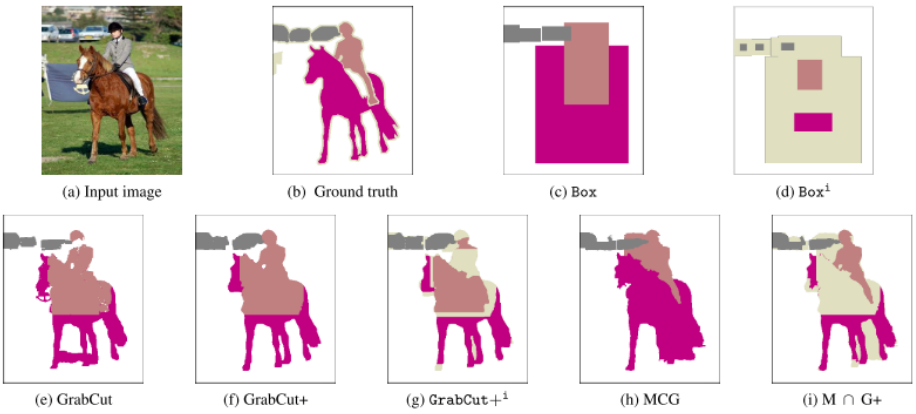
– Jedan od najjednostavnijih načina označavanja je prikazan na slici 4.4c. Svi pikseli koje se nalaze unutar pravokutnog okvira označavaju se oznakom njegove klase. Ako se dva pravokutna okvira preklapaju pretpostavlja se da manji dolazi ispred.

– Na tragu prethodnog načina je i onaj prikazan na slici 4.4d. Samo 20% unutrašnjosti pravokutnog okvira se označava oznakom njegove klase, dok preostali dio unutrašnjosti pravokutnog okvira predstavlja regije koje se ignoriraju za vrijeme učenja (engl. *ignore regions*). Ignorirati znači ne računati gubitak za te regije jer njihova klasa nije točno određena u datom trenutku. Povod takvom označavanju je pretpostavka da 20% unutrašnjosti okvira ima veću šansu da se preklapa s objektom unutar okvira čime bi se trebao smanjiti šum u oznakama. Odziv ovakvih oznaka bi trebao biti manji, a preciznost veća nego li kod prethodnog načina označavanja.

– Segmenti objekta unutar pravokutnog okvira se mogu generirati nekom od varijanti GrabCut algoritma [24] kao što je prikazano na slici 4.4e-g.

– Također se mogu generirati upotrebom MCG metode, koju koristi i BoxSup metoda, što je prikazano slikom 4.4h.

– Posljednji način je generiranje oznaka kombinacijom varijante GrabCut algoritma i MCG metode. Tim načinom su dobiveni najbolji rezultati prilikom učenja modela. Primjer tih oznaka je dan na slici 4.4i.



Slika 4.4: Primjeri segmentacijskih oznaka dobivenih iz pravokutnih okvira [11].

Predikcije modela dobivene nakon pojedinih epoha se dodatno procesiraju kroz u tri faze prije nego li se iskoriste kao oznake za učenje. U prvoj fazi se svi pikseli van pravokutnih okvira ponovno postavljaju na oznaku pozadinske klase. U drugoj fazi se predikcije čija je površina puno manja od površine pripadnih pravokutnih okvira zamjenjuju s inicijalnim oznakama koje su korištene u prvoj epohi. Posljednja faza služi za filtriranje oznaka gustim uvjetnim slučajnim poljima kako bi se dobile preciznije granice objekata.

5. Semantička segmentacija primjenom suparničkog učenja

Suparnički učeni generativni modeli imaju široku primjenu. Neke od najpoznatijih primjena su: generiranje realističnih slika, generiranje slika na temelju tekstualnog opisa, strojno prevođenje slika (primjerice pretvaranje slika snimljenih danju u slike snimljene noću), sinteza videa te stvaranje slika visoke rezolucije iz slika niske rezolucije. Mi ćemo detaljnije istražiti njihovu primjenu i doprinos na problemu semantičke segmentacije.

5.1. Modeli za nadziranu i polunadziranu semantičku segmentaciju

Modeli suparničkog učenja za semantičku segmentaciju koja će biti opisani u ovom odijeljku imaju zajedničko to što za učenje koriste označene podatke na razini piksela. Svi minimiziraju gubitak unakrsne entropije za označene podatke i kombiniraju ga sa suparničkim gubitkom. Neki od njih dodatno iskorištavaju neoznačene i slabo označene podatke kako bi poboljšali segmentacijske rezultate. Objasnit ćemo ulogu generativnog i diskriminativnog modela te pretpostavke na kojima se modeli temelje.

Prva primjena suparnički učenih generativnih modela za problem semantičke segmentacije zabilježena je u radu [15]. Generator je duboka konvolucijska mreža koja generira segmentacijske mape. Zadatak drugog modela, diskriminatora, je da klasificira segmentacijske mape koje dolaze iz skupa za učenje i segmentacijske mape generirane od strane generatora. Diskriminator bi trebao moći detektirati nekonzistentnosti višeg reda između te dvije skupine segmentacijskih mapa te potaknuti generator da generira segmentacijske mape statistički bliže onima iz skupa za učenje. To jest zadatak diskriminatora je da potiče prostornu bliskost segmentacijskih oznaka. Prostorna bliskost do neke mjere može biti potaknuta i uporabom uvjetnih slučajnih polja, međutim

ona povećavaju složenost modela za vrijeme testiranja.

Gubitak modela je težinska kombinacija višeklasne unakrsne entropije i gubitka suparničkog generativnog modela, odnosno binarne unakrsne entropije. Model prilikom učenja koristi samo označene podatke, odnosno za svaku ulaznu sliku x ima i pripadnu segmentacijsku oznaku y. S obzirom na to da se koriste segmentacijske oznake na razini piksela radi se o nadziranom učenju. Gubitak na skupu za učenje, koji sadrži ukupno N parova slika i oznaka, dan je sljedećom formulom:

$$\mathcal{L}(G,D) = \sum_{i=1}^{N} \mathcal{L}_{MCE}(G(x_i), y_i) - \lambda \left[\mathcal{L}_{BCE}(D(x_i, y_i), 1) + \mathcal{L}_{BCE}(D(x_i, G(x_i)), 0)\right]. \tag{5.1}$$

Pri tome su $\mathcal{L}_{MCE}$ i $\mathcal{L}_{BCE}$ gubici višeklasne i binarne unakrsne entropije definirani za jednojedinične oznake y i predikcije $\hat{y}$ dimenzija $H \times W \times C$:

$$\begin{aligned} \mathcal{L}_{MCE}(\hat{y}, y) &= -\sum_{i=1}^{H\times W}\sum_{c=1}^{C} y_{ic}\, ln\, \hat{y}_{ic} \\ \mathcal{L}_{BCE}(\hat{z}, z) &= -\left[z\, ln\, \hat{z} + (1-z) ln(1-\hat{z})\right] \end{aligned} \tag{5.2}$$

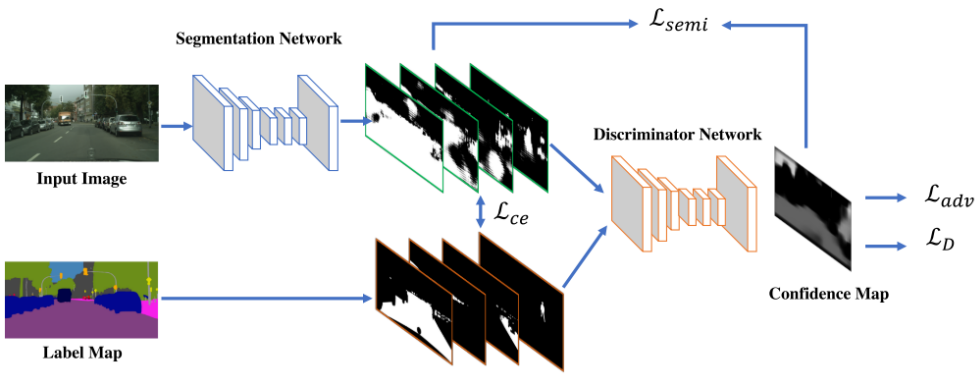
Gubitak višeklasne unakrsne entropije potiče podudaranje na razini piksela dok suparnički gubitak ima veće vidno polje i potiče podudaranja višeg reda. Provedeni su usporedni eksperimenti za gubitak višeklasne unakrsne entropije i težinski gubitak dan formulom (5.1) koji su pokazali da suparnički gubitak dovodi do poboljšanja rezultata semantičke segmentacije.

Zanimljivo je još promotriti arhitekturu modela koji je korišten kao diskriminator. Jednadžba (5.1) pokazuje da diskriminator prima dva ulaza, sliku i pripadnu segmentacijsku mapu. U toj varijanti svaki od ulaza se procesira zasebnom granom neuronske mreže te se procesirani podaci potom konkateniraju prije prosljeđivanje dalje u mrežu. Postoji i druga varijanta u kojoj diskriminator prima samo jedan ulaz. Predložene su tri varijante jedinstvenog ulaza:

- Najjednostavniji varijanta koristi samo segmentacijske mape. Segmentacijske mape iz skupa za učenje se predaju u jednojediničnom kodu (engl. *one-hot encoding*), a generirane segmentacijske mape u nepromijenjenom obliku.
- Druga varijanta spaja sliku i pripadnu segmentacijsku mapu operacijom množenja. Segmentacijske mape koje imaju C kanala se pomnože sa svakim od tri kanala RGB slika tvoreći tako ulaz veličine $3C$ kanala.

– Treća varijanta je slična prvoj, a razlika je u tome što se segmentacijske mape iz skupa za učenje ne predaju u jednojediničnom kodu. Pretpostavka je da bi diskriminator mogao naučiti razlikovati segmentacijske mape iz skupa za učenje koje se sastoje samo od nula i jedinica od generiranih koje sadržavaju vrijednosti između nule i jedan. Stoga se segmentacijske mape iz skupa za učenje transformiraju iz jednojediničnog koda u distribuciju koja na mjestu jedinice ima minimalno vrijednost τ, a na mjestima gdje su bile nule vrijednosti takve da distribucija bude što sličnija pripadnoj distribuciji generiranih segmentacijskih mapa $G(x)$ prema kriteriju Kullback–Leiblerovu divergencije. Formalno na poziciju jedinice koju označavamo s l dolazi vrijednost $y_{il} = max(\tau, G(x)_{il})$. Dok na ostale pozicije koje označavamo s c dolaze vrijednosti: $y_{ic} = G(x)_{ic}(1 - y_{il})/(1 - G_{il})$.

Sličan, ali dosta proširen, način korištenja modela suparničkog učenja za semantičku segmentaciju predstavljen je u radu [9]. Ponovno generator ima ulogu segmentacijskog modela, a diskriminator klasificira između generiranih segmentacijskih mapa i onih iz skupa za učenje. Glavna razlika u odnosu na [15] je ta da diskriminator na izlazu klasificira, odnosno daje vjerojatnost za svaki piksel segmentacijske mape umjesto jedne vjerojatnosti za cijelu segmentacijsku mapu. Također diskriminator na ulazu ne prima sliku nego mu je ulaz istovjetan prethodno objašnjenoj najjednostavnijoj varijanti spomenutog modela. Izlaze iz diskriminatora nazivamo vjerojatnosne mape. One se dodatno mogu koristiti za procjenu pouzdanosti, odnosno nesigurnosti segmentacijskog modela ukazujući na područja za koja je model siguran da ih je dobro segmentirao. Izgled modela je prikazan na slici 5.1.



Slika 5.1: Model suparničkog učenja za polunadziranu semantičku segmentaciju s naznačenim gubicima [9].

Za treniranje modela se koriste označeni podaci na razini piksela u kombinaciji

s neoznačenim podacima te stoga kažemo da je model treniran polunadzirano. Diskriminator se trenira samo na označenim podacima dok se generator trenira na svim podacima. Razmotrit ćemo gubitke modela da bismo shvatili način učenja modela. Definirajmo najprije oznake i njihove dimenzije. Ulaz u generator je RGB slika x dimenzija $H \times W \times 3$. Generirane segmentacijske mape koje su izlaz generatora G su dimenzija $H \times W \times C$ gdje C označava broj mogućih klasa. Segmentacijske oznake iz skupa za učenje y su također tih dimenzija. Vjerojatnosne mape koje su izlaz diskriminatora D su dimenzija $H \times W \times 1$. Sukladno uvedenim oznakama gubitak diskriminatora je definiran za sve pozicije (h, w) izlaza:

$$\mathcal{L}(D) = -\sum_{h,w} Y_i \log(D(y_i))^{(h,w)}) + (1 - Y_i) \log(1 - D(G(x_i))^{(h,w)}), \tag{5.3}$$

pri čemu je $Y_i = 1$ za segmentacijske mape iz skupa za učenje i $Y_i = 0$ za segmentacijske mape iz generatora. Vidimo da gubitak diskriminatora nije niša drugo nego suparnički gubitak binarne unakrsne entropije definiran za svaki piksel izlaza. Gubitak generatora je dosta složeniji i možemo ga promotriti kao nadogradnju težinskog gubitka definiranog formulom (5.1) kojem je nadodana još jedna komponenta $\mathcal{L}_{SEMI}$:

$$\mathcal{L}(G) = \mathcal{L}_{CE} + \lambda_{ADV}\mathcal{L}_{ADV} + \lambda_{SEMI}\mathcal{L}_{SEMI}. \tag{5.4}$$

Komponente gubitka $\mathcal{L}_{CE}$ i $\mathcal{L}_{ADV}$ odgovaraju komponentama unakrsne entropije i suparničkog gubitka binarne unakrsne entropije u formuli (5.1). Komponenta unakrsne entropije je zadužena samo za označene podatke, komponenta $\mathcal{L}_{SEMI}$ za neoznačene podatke, dok je suparnička komponenta zadužena za sve podatke. Prilikom korištenja suparničke komponente na neoznačenim podacima potrebno je koristi manju vrijednost težine λ_{ADV} nego za označene podatke.

Za treniranje s neoznačenim podacima se koriste vjerojatnosne mape iz diskriminatora da bi se odredila pouzdanost generiranih segmentacijskim mapi. Za sve piksele čija je pouzdanost veća od zadanog praga T_{SEMI} pretpostavlja se da su generirane oznake točne te se koriste za osvježavanje diskriminatora gubitkom unakrsne entropije. Prema tome gubitak $\mathcal{L}_{SEMI}$ možemo shvatiti kao maskirani gubitak unakrsne entropije definiran samo za ona područja generalnih segmentacijskih maski koja diskriminator proglasi pouzdanima:

$$\mathcal{L}_{SEMI} = -\sum_{h,w}\sum_{c \in C} I(D(G(x_i))^{(h,w)} > T_{SEMI})\hat{Y}_i^{(h,w,c)} \log(D(x_i)^{(h,w,c)}). \tag{5.5}$$

I predstavlja indikatorsku funkciju, a $\hat{Y}_i$ generiranu segmentacijsku mapu u jednojediničnom kodu gdje je jedinica na poziciji $c^* = \arg\max_c G(x_i)^{(h,w,c)}$. U eksperimentima su pokazali da suparnički gubitak i gubitak $\mathcal{L}_{SEMI}$ poboljšavaju rezultate semantičke segmentacije te da neoznačeni podaci mogu biti koristan izvor informacija. Model postiže bolje rezultate u usporedbi s [15] na skupu podataka Pascal VOC 2012 [5]. Srednji Jaccardov indeks za oba modela na Pascal VOC 2012 skupu za validaciju je dan u tablici 5.1. Naveden je srednji Jaccardov indeks za modele koji su trenirani s gubitkom unakrsne entropije i s težinskom kombinacijom suparničkog gubitka i gubitka unakrsne entropije.

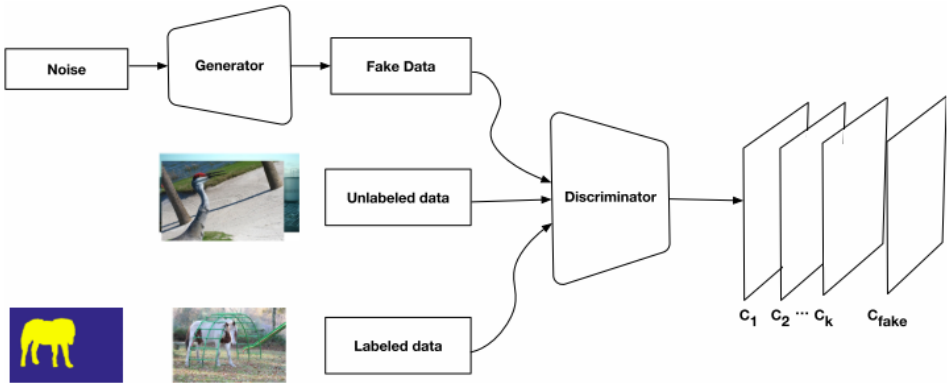
Tablica 5.1: Srednji Jaccardov indeks na Pascal VOC 2012 skupu za validaciju. Usporedba modela koji koriste gubitak unakrsne entropije samostalno i u kombinaciji sa suparničkim gubitkom.

Model	bez suparničkog gubitka	sa suparničkim gubitkom
[15]	0.718	0.720
[27]	0.736	0.749

Podosta drugačija upotreba suparnički učenih generativnih modela za semantičku segmentaciju, u smislu uloga generatora i diskriminatora, opisana je u radu [27]. Generator se koristi za generiranje dodatnih primjera za učenje dok se diskriminator koristi kao segmentacijska mreža. Diskriminator možemo promatrati i kao klasifikacijsku mrežu koja radi klasifikaciju na razini piksela. Moguće semantičke klase su K klasa iz skupa za učenje te dodatna $K + 1$ klasa za generirane primjere. Pristup se bazira na ideji da će primjeri iz generatora potpomoći da stvarni primjeri budu bliže u prostoru značajki i da naučene značajke budu informativnije što će omogućiti točniju klasifikaciju piksela. Pretpostavlja da bi primjeri koji su blizu u ulaznom prostoru značajki trebali pripadati istoj klasi. Za učenje želi iskoristi veliku količinu neoznačenih i slabo označenih podataka koje kombinira s manjim skupom označenih podataka koji mu pružaju potpuni nadzor. Neoznačeni podaci doprinose regularizaciji te učenju skrivenih značajki za semantičku segmentaciju. Ne kombinira neoznačene i slabo označene podatke nego predlaže dva različita modela.

Prvi model razmatra kombinaciju podataka označenih na razini piksela i neoznačenih podataka. Skica modela je prikazana na slici 5.2. Vidimo da diskriminator prima označene, neoznačene i generirane podatke. Za svaku skupinu podataka ima različite ciljeve:

– Za označene podatke želi minimizirati gubitak unakrsne entropije $\mathcal{L}_{CE}$ između predikcija i oznaka y.



Slika 5.2: Model suparničkog učenja korišten za polunadziranu semantičku segmentaciju. Generator generira dodatne primjere za učenje, a diskriminator ima ulogu segmentacijskog modela [27].

– Za neoznačene podatke želi minimizirati vjerojatnost da pripadaju klasi $K+1$ odnosno želi maksimizirati vjerojatnost da podaci stvarni.

– Generirane podatke želi odijeliti od stvarnih i klasificirati ih u klasu $K+1$.

Sukladno tome gubitak diskriminatora je sljedeći:

$$\begin{aligned}\max_D \mathcal{L}(D) = \mathbb{E}_{x\sim p_{data}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right] \\ -\gamma\mathbb{E}_{x,y\sim p_{(y,x)}}\left[\mathcal{L}_{CE}(y, P(y|x,D)\right],\end{aligned} \tag{5.6}$$

pri čemu je:

$$D(x) = \left[1 - P(y = K+1|x)\right]. \tag{5.7}$$

Istodobno generator transformira ulazni šum z u prostor podataka želeći natjerati diskriminator da klasificira piksele generiranih podataka u jednu od K klasa stvarnih primjera:

$$\min_G \mathcal{L}(G) = \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right]. \tag{5.8}$$

Vidimo da gubici nalikuju na originalni gubitak suparničkog generativnog modela dan formulom (3.1), a glavna razlika je u tome da diskriminator više nije binarni klasifikator na razini slike nego višeklasni klasifikator na razini piksela.

Drugi model kombinira označene i slabo označene podatke. Slabo označeni u ovom slučaju podrazumijeva da su dostupne klasifikacijske oznake na razini slike, odnosno da su poznate klase koje su zastupljene na slici ali ne i njihova distribucija. Da bi iskoristili dodatne oznake prilikom učenja generatora i diskriminatora koriste uvjetovane suparničke generativne modele [18]. Kod takvih modela se generatoru i diskriminatoru pružaju dodatne informacije koje služe za navođenje generatora pri stvaranju

podataka. U ovom slučaju dodane informacije su oznake klasa zastupljenih u slici. Gubitak općenitog uvjetovanog suparničkog generativnog modela je sljedeći:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x,l\sim p_{data}(x,l)} \left[\log D(x,l)\right] + \mathbb{E}_{z\sim p_z(z,l),l\sim p_l(l)} \left[\log(1 - D(G(z,l),l))\right]. \tag{5.9}$$

Gdje je $p_l(l)$ distribucija oznaka l, $D(x,l)$ združena distribucija primjera i oznaka, $G(z,l)$ združena distribucija šuma i oznaka koja ukazuje da oznake kontroliraju uvjetovanu distribuciju $p_z(z|l)$ generatora. Vidimo da se modeli sada predstavljaju združene distribucije primjera i oznaka.

Dodatne oznake klasa tjeraju generator da nauči vezu između oznaka i izlaza te da generira čim bolje slike. To ponovno tjera diskriminator da nauči što informativnije značajke i pravi odnos između oznaka. Gubitak generatora i diskriminatora uvjetovanog modela za semantičku segmentaciju je kombinacija gubitaka prvog modela i gubitka općenitog uvjetovanog suparničkog generativnog modela. Generator ovoga puta transformira vektor šuma z na koji je konkateniran jednojedinični vektor klasa l u prostor podataka. Zadatak mu je maksimizirati vjerojatnost da su klase dane vektorom l zastupljene u slici. Diskriminator ponovno prima tri vrste podataka, ali u odnosu na prvi model ne prima neoznačene podatke nego slabo označene podatke. Ciljevi diskriminatora za označene i generirane podatke su ostali isti dok za slabo označene podatke ne nastoji minimizirati vjerojatnost da pripadaju $K + 1$ klasi nego maksimizirati vjerojatnost da pripadaju nekoj od klasa l zastupljenih u slici. Pokazano je kako velika količina neoznačenih ili slabo označenih podataka u kombinaciji sa suparničkim gubitkom poboljšava rezultate semantičke segmentacije.

5.2. Model za slabo nadziranu semantičku segmentaciju

U radu [22] je predstavljen model suparničkog učenja za semantičku segmentaciju na razini instanci koji ćemo opisati u ovom odjeljku. Za učenje modela se koriste pravokutni okviri i stoga je pristup slabo nadzirani. Pravokutni okviri mogu biti dostupni iz skupa za učenje, ali ako to nije slučaj može ih se dobiti dubokom detekcijskom mrežom. Konkretno koristi se detekcijska mreža Faster R-CNN. Generator služi za generiranje segmentacijskih maski, a igra između generatora i diskriminatora je igra izrezivanja i lijepljenja. Objekt se izreže iz ulazne slike uz pomoć pravokutnog okvira i generirane maske i zalijepi se na novu pozadinu koja je također izrezana iz ulazne slike. Te slike dobivene izrezivanjem i lijepljenjem nazivamo lažne slike. Diskriminator nastoji klasificirati originalne slike objekata i lažne slike te daje generatoru signale za učenje. Ideja na kojoj se bazira ovaj model je da se objekti mogu gibati neovisno od pozadine. Točnije ako izrežemo objekt s neke pozicije u slici i zalijepimo ga na drugu poziciju on će dalje izgledati realistično. Sama ideja je jednostavna i intuitivna.

Pokušat ćemo rekonstruirati tijek nastanka modela. Pretpostavimo da je definiran segmentacijski model G koji na temelju slike X i dostupnog pravokutnog okvira B generira segmentacijsku masku M, odnosno $M = G(X, B)$. S obzirom na to da nisu dostupne segmentacijske oznake iz skupa za učenje autori su morali osmisliti način na koji će izračunati gubitak modela i procijeniti kvalitetu generiranih maski. Tu su im u pomoć priskočili suparnički generativni modeli. Diskriminator koji je dio modela se može interpretirati kao promjenjiva funkcija gubitka, u smislu da je funkcija gubitka generativnog modela definirana pomoćnim parametrima koji ne sačinjavaju generativni model. [15]. U kontekstu suparničkog generativnog modela generatoru je dodijeljena uloga segmentacijskog modela dok je diskriminator postao zadužen za ocjenu kvalitete segmentacijskih maski. Diskriminator po definiciji radi na principu da su mu predočeni stvarni i generirani podaci koje nastoji klasificirati. Stvarni podaci su podaci iz skupa za učenje, primjerice slike ili segmentacijske oznake. Pošto su iz skupa za učenje dostupne samo slike i pravokutni okviri autori nisu imali puno izbora nego kao stvarne podatke koje će vidjeti diskriminator izabrati objekte izrezane iz slika uz pomoć pravokutnih okvira. Međutim, time je nastao veliki jaz između stvarnih podataka i generiranih segmentacijskih maski. Da bi generirani podaci nalikovali na stvarne sintetizirali su lažne slike. Lažna slika F se dobije tako da se iz slike X i pravokutnog okvira B (skraćeno X_B) izreže objekt na temelju generalne maske M

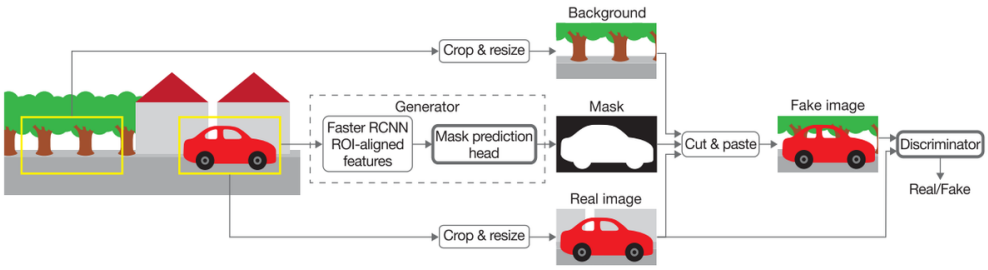
te se potom izrezani objekt zalijepi na pozadinsku sliku koja je izrezana iz slike X s lokacije B' tvoreći tako lažnu sliku:

$$F = MX_B + (1 - M)X_{B'} \tag{5.10}$$

Nakon što smo definirali sve ulaze i izlaze modela možemo definirati zajedničku funkciju cilja:

$$\min_G \max_D \mathcal{L}_{CPGAN}(D, G) = \mathbb{E}_{(X,B,B')\sim p_{data}} \log D(X_B) + \log(1 - D(F)). \tag{5.11}$$

Diskriminator nastoji maksimizirati dok generator nastoji minimizirati funkciju cilja. Diskriminator nastoji ispravno klasificirati stvarne i lažne slike pružajući pritom generatoru signale za učenje koji mu omogućavaju da stvara sve bolje i bolje segmentacijske maske. Generator nastoji generiranim segmentacijskim maskama poboljšati kvalitetu lažnih slika te zbuniti diskriminator pri klasifikaciji. Sažeti prikaz generiranja lažnih slika i načina na koji model funkcionira dan je na slici 5.3.



Slika 5.3: Prikaz suparničkog generativnog modela za semantičku segmentaciju koji formira lažne slike izrezivanjem i lijepljenjem segmentiranih objekata na novu pozadinu [22].

Spomenuli smo da se izrezani objekti lijepe na pozadinsku sliku ali nismo naveli kako odrediti njenu lokaciju B'. Naime objekti nisu toliko neovisni o pozadini kao što to na prvi pogled izgleda. Primjerice na slikama nije uobičajeno vidjeti automobile na nebu ili na vrhu drveća. Zbog toga su predložene dvije jednostavne heuristike za pronalaženje pozadine na koju će se objekti lijepiti. Prva heuristika predlaže lijepljenje objekta bilo gdje u slici pazeći pritom da se objekt ne preklopi s drugim objektima iste klase. Druga heuristika predlaže lijepljenje objekta uzimajući u obzir dubinu slike te zadržava omjer između veličine objekata i njegove udaljenosti od kamere.

Moguće je više slučajeva degradacije postupka učenja ovakvoga modela. Razmotrit ćemo dva slučaja degradacije i njihove posljedice. Prvi slučaj je kada generator generira maske koje sve piksele proglase pikselima pozadine te niti jedan piksel ne proglasi pikselom objekta. Tada će lažna slika biti jednaka pozadinskoj slici jer je

$M = 0$ pa iz formule (5.10) slijedi: $F = 0 \cdot X_B + (1 - 0)X_{B'} = X_{B'}$. Lažna slika će biti realistična ali neće sadržavati objekt koji želimo segmentirati te bi ju diskriminator trebao klasificirati kao lažnu sliku navodeći na taj način generator da promijeni svoju odluku. Autori navode kako bi taj problem trebao automatski biti riješen, ali dodaju kako nije na odmet uvesti još jedan klasifikacijski gubitak za generator:

$$\min_G \mathcal{L}_{CLS}(G) = \mathbb{E}_{(X,B')\sim p_{data}} \log(1 - D_{CLS}(F)). \tag{5.12}$$

To bi značilo uvođenje još jednog diskriminatora koji bi trebao klasificirati slike koje sadržavaju i ne sadržavaju objekte. Taj diskriminator se ne bi učio istodobno s generatorom i diskriminatorom koji klasificira stvarne i lažne slike nego bi bio naučen prije i ne bi se osvježavao za vrijeme njihova učenja.

Drugi slučaj degradacije je kada generator sve piksele proglasi pikselima objekta. U tom slučaju $M = 1$ i $F = X_B$, odnosno lažne slike će biti jednake stvarnima i diskriminator neće pružati nikakve korisne informacije za učenje generatora. Da bi se to izbjeglo generatoru se daje šire vidno polje na način da se generirane maske nadopune s nulama sa svih strana. Lijevo i desno se dopune za 10% širine, a gore i dolje za 10% visine maske. S takvim maskama će rubovi lažnih slika uvijek dolaziti od pozadinskih slika te će diskriminator vidjeti širi kontekst oko samih objekata. Ako generator sve piksele proglasi pikselima objekta diskriminator će detektirati naglu promjenu pozadine na rubovima i vrlo lagano će prepoznati lažne slike.

6. Implementacija

U sklopu ovog rada je implementiran suparnički učen generativni model za slabo nadziranu semantičku segmentaciju primjeraka opisan u odjeljku 5.2. Model pretpostavlja da su svi objekti iste klase te je stoga za svaku klasu objekata potrebno učiti zaseban model. U nastavku će biti opisana arhitektura modela i implementacijski detalji te navedene biblioteke korištene pri implementaciji.

6.1. Arhitektura modela

Tri su glavna modula ovoga modela: generator, diskriminator i modul koji sintetizira lažnu sliku. Detaljna arhitektura generatora i diskriminatora dana je u tablici 6.1. U generatoru se koristi aktivacijska funkcija zglobnice (engl. *rectified linear unit, ReLU*), a u diskriminatoru funkcija LeakyReLU s parametrom $\alpha = 0.2$. Model je implementiran s gubitkom najmanjih kvadrata te se stoga ne koristi nikakva aktivacijska funkcija na izlazu diskriminatora što omogućava korištenje različitih vrijednosti parametara a, b i c.

Tablica 6.1: Arhitektura suparnički učenog generativnog modela za slabo nadziranu semantičku segmentaciju. Konv je kratica za konvoluciju nakon koje je navedena visina, širina i dubina konvolucijskih jezgri, S označava korak konvolucije, a P nadopunu nulama na rubovima.

Generator		**Diskriminator**	
Dimenzije izlaza	Sloj	Dimenzije izlaza	Sloj
$7 \times 7 \times 2048$	Poravnate značajke iz Faster R-CNN	$34 \times 34 \times 3$	Ulazna slika
$7 \times 7 \times 256$	Konv $1 \times 1 \times 256, S = 1, P = 1$	$32 \times 32 \times 64$	Konv $3 \times 3 \times 64, S = 1, P = 0$
$7 \times 7 \times 256$	Konv $3 \times 3 \times 256, S = 1, P = 1$	$15 \times 15 \times 128$	Konv $3 \times 3 \times 128, S = 2, P = 0$
$14 \times 14 \times 256$	Bilinearna interpolacija	$7 \times 7 \times 256$	Konv $3 \times 3 \times 256, S = 2, P = 0$
$14 \times 14 \times 256$	Konv $3 \times 3 \times 256, S = 1, P = 1$	$3 \times 3 \times 512$	Konv $3 \times 3 \times 512, S = 2, P = 0$
$28 \times 28 \times 256$	Bilinarna interpolacija	4608	Izravnavanje u vektor
$28 \times 28 \times 256$	Konv $3 \times 3 \times 256, S = 1, P = 1$	1	Potpuno povezani sloj
$28 \times 28 \times 1$	Konv $3 \times 3 \times 1, S = 1, P = 1$		
$28 \times 28 \times 1$	Sigmoidalna funkcija		

U sve slojeve osim u izlazni sloj generatora i ulazni sloj diskriminatora smo dodali

normalizaciju po grupi. Bez normalizacije po grupi se tijekom učenja modela događalo da sve vrijednosti generiranih maski budu nula ili jako mali brojevi blizu nule, odnosno svi pikseli ulaznih slika su bivali klasificirani kao pozadina. Normalizacija po grupi je ublažila tu pojavu. Na ulaz diskriminatora smo dodali Gaussov filter identičan onome u radu [2]. To je uvelike povećalo stabilnost učenja modela. Bez filtera bi diskriminator već nakon prve epohe uspio u potpunosti nadjačati generator te bi kvaliteta generiranih maski naglo opala nakon prve epohe učenja. S filterom je diskriminatoru otežan zadatak razlikovanja lažnih i stvarnih slika jer su mu detalji u ulaznim slikama manje izraženi.

Generator se uvelike oslanja na mrežu Faster R-CNN te koristi njene ekstrahirane značajke koje su poravnate na temelju pravokutnog okvira oko objekta. Uz pomoć značajki generira maske fiksne veličine $28 \times 28 \times 1$ piksel. Da bi se spriječio slučaj degradacije u kojem generator sve piksele označava pikselima objekta, generirane maske su nadopunjene s tri nule, odnosno za 10% visine i 10% širine na svaku stranu čineći tako maske dimenzija $34 \times 34 \times 1$ koje su po visini i širini jednake ulazu diskriminatora. Modul koji sintetizira lažne slike reže objekte iz pravokutnih okvira uz pomoć proširenih maski i lijepi ih na novu pozadinu. Međutim, da bi proširene maske odgovarale slikama objekata potrebno je slike objekata i pozadinske slike prije izrezivanja i lijepljenja proširiti u ulaznoj slici za 10% njihove visine i 10% njihove širine. Potom je potrebno bilinearnom interpolacijom promijeniti veličinu proširenih slika na $34 \times 34 \times 3$ piksela tako da se visinom i širinom podudaraju s visinom i širinom maski.

Izrezani objekti će se nalaziti u sredini lažnih slika jer će rubovi lažnih slika odgovarati pozadinskim slikama. Stoga je potrebno i stvarne slike koje se predaju na ulaz diskriminatora proširiti u ulaznoj slici za 10% njihove širine i 10% njihove visine kako bi se objekti na njima nalazili u sredini slike i veličinom odgovarali objektima na lažnim slikama. Potom je proširene stvarne slike potrebno bilinearnom interpolacijom svesti na veličinu ulaza diskriminatora.

6.2. Odabir lokacije za lijepljenje

Implementirali smo dva načina za određivanje lokacije pozadine na koju će se lijepiti objekti. Oba načina uzimaju u obzir veličinu objekta i njegovu udaljenost od kamare te stoga pozadinu određuju duž horizontalne linije na kojoj se nalazi objekt. Na slici 6.1 je prikazana usporedba lijepljenja nekoliko objekata (bijelog automobila, plavog automobila i pješaka) na različitim lokacijama. Na lijevoj slici su objekti zalijepljeni duž horizontalne linije u odnosu na svoju originalnu lokaciju, a na desnoj slici su zalijep-

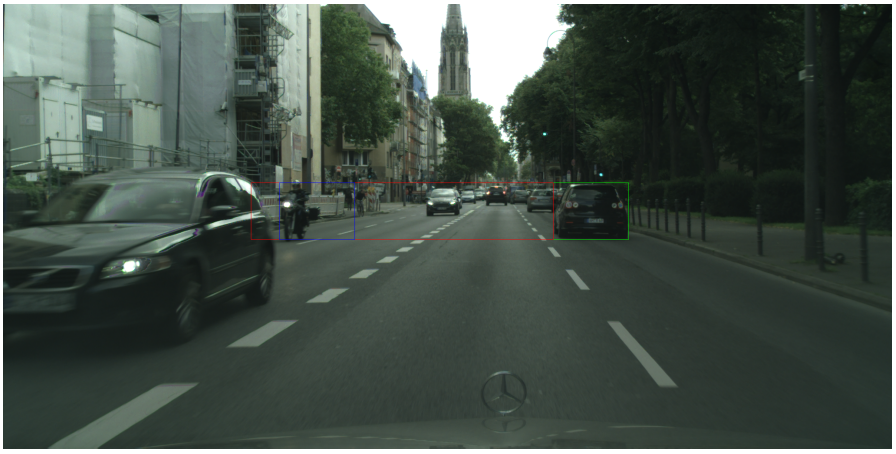
ljeni na nasumično odabranim lokacijama u slici. Objekti mnogo realističnije izgledaju na lijevoj slici i zato ćemo lokaciju pozadine smjestiti na horizontalnu liniju na kojoj se nalazi objekt kojega lijepimo.

Slika 6.1: Usporedba lijepljenja objekata na horizontalnoj liniji u odnosu na originalnu poziciju (slika lijevo) i na nasumično odabranim pozicijama (slika desno) [22].

Kod prvog načina lokaciju pozadine određujemo uz pomoć normalne razdiobe. Ne dozvoljavamo preklapanje pozadine s pravokutnim okvirom oko objekta kojeg lijepimo i zato to područje eliminiramo prilikom odabira lokacije. Potom gledamo je li veća površina horizontalne linije lijevo ili desno od eliminiranog područja te odabiramo lokaciju pozadine iz veće površine. Za lokaciju je potrebno odrediti samo pomak od lijevog ruba slike, dok su pomak od gornjeg ruba slike, visina i širina identične onima od pravokutnog okvira oko objekta. Pomak od lijevog ruba slike generiramo nasumično iz normalne razdiobe pri čemu je očekivanje normalne razdiobe pomaknuto za $2W$ (W je širina pravokutnog okvira) u odnosu na lijevi rub pravokutnog okvira u smjeru veće površine, a standardna devijacija je jednaka W.

Kod drugog načina lokaciju pozadine određujemo na slučajan način uz pomoć uniformne razdiobe pazeći pri tom da se ne preklapa s pravokutnim okvirom niti jednog od objekata u slici koji je iste klase kao i objekt kojega lijepimo. Najprije eliminiramo sva područja unutar horizontalne linije koja se preklapaju s pravokutnim okvirima te dobijemo dva ili više mogućih područja za odabir lokacije. Ako ne postoji niti jedno područje šire od W onda uzmemo najšire područje i proširimo ga do W formirajući tako lokaciju pozadine. Ako postoji više područja koja su šira od W onda slučajnim odabirom odaberemo jedno od tih područja te unutar njega ponovno na slučajan način odredimo lokaciju pozadine. Ako postoji samo jedno područje šire od W onda unutar njega na slučajan način odredimo lokaciju pozadine. Na slici 6.2 je prikazana usporedba određivanja lokacije pozadine na dva objašnjena načina pri čemu su prikazani pravokutni okvir oko objekta (prikazan zelenom bojom), dozvoljena područja za odabir pozadine (prikazana crvenom bojom) te odabrana lokacija (prikazana plavom bojom). Prikazano dozvoljeno područje za odabir pozadine uz pomoć normalne razdiobe se širi tri standardne devijacije od očekivanja normalne razdiobe jer je to područje

najizglednije za odabir lokacije.



(a) Određivanje lokacije uz pomoć normalne razdiobe.

(b) Određivanje lokacije uz pomoć uniformne razdiobe.

Slika 6.2: Određivanje lokacije pozadine. Zelenom bojom je prikazan pravokutni okvir oko objekta koji se reže i lijepi na novu lokaciju, crvenom bojom su prikazana dozvoljena područja te plavom bojom odabrana lokacija.

6.3. Korištene biblioteke

Za implementaciju je korišten programski jezik Python verzije 3.7.3. Glavne korištene biblioteke su: PyTorch za implementaciju dubokog modela, NumPy za razne manipulacije s podacima, PIL za pripremu slika za učenje te Matplotlib za vizualizaciju podataka, gubitaka i gradijenata za vrijeme učenja.

7. Rezultati

7.1. Podatkovni skup

Za učenje i validaciju implementiranog modela koristimo Cityscapes skup podataka [3]. To je skup slika uličnih scena koje su snimljene iz kamera postavljenih na automobilu u 50 različitih gradova. Sadrži 5,000 označenih slika s oznakama na razini piksela za semantičku segmentaciju. Te slike su podijeljene u tri skupa: skup za učenje koji sadrži 2,975 slika, skup za validaciju koji sadrži 500 slika i skup za testiranje koji sadrži 1,525 slika. Za učenje implementiranog modela ćemo koristi pravokutne okvire iz skupa za učenje, a za ocjenu kvalitete modela ćemo koristiti oznake na razini piksela iz skupa za validaciju pošto oznake iz skupa za testiranje nisu javno dostupne.

Cityscapes skup podataka sadrži i 5,000 video isječaka s po 30 slika. Označene slike su 20. po redu u video isječcima dok preostale slike iz video isječaka nisu označene. U radu [22] su za učenje slabo nadziranog modela korišteni pravokutni okviri iz skupa za učenje i pravokutni okviri dobiveni Faster R-CNN modelom na svim slikama iz video isječaka koji su povezani sa skupom za učenje. To je ukupno 2,975 slika iz skupa za učenje i 89,250 slika iz video isječaka. Na tragu toga ćemo i mi za učenje, osim pravokutnih okvira iz skupa za učenje, koristiti pravokutne okvire dobivene Faster R-CNN modelom na 1., 8., 13., 25. i 30. po redu slici iz video isječaka povezanih sa skupom za učenje te ćemo vidjeti kako dodatni podaci utječu na točnost modela.

U skupu su dostupne oznake za semantičku segmentaciju na razini instanci za 8 razreda: čovjek, vozač, automobil, kamion, autobus, vozilo na tračnicama i motocikl. Mi smo odabrali implementirani model učiti s primjercima iz klase automobil jer je ta klasa najbrojnija u skupovima za učenje i validaciju. Međutim, klasa automobil je jako zahtjevna za semantičku segmentaciju na razini instanci jer se u prosjeku 9 instanci automobila preklapa u slici. U radu [8] tvrde da je upravo preklapanje instanci iste klase najveći izazov za semantičku segmentaciju na razini instanci.

7.2. Metrika

Za ocjenu kvalitete modela koristit ćemo standardnu mjeru za semantičku segmentaciju, a to je Jaccardov indeks (engl. *Intersection over Union, IoU*). Prilikom evaluacije modela za semantičku segmentaciju na razini instanci piksele koji pripadaju promatranoj instanci označimo klasom objekta, a preostale piksele (pozadinu, druge instance iste klase, instance drugih klasa, ...) označimo klasom pozadine. Jaccardov indeks računamo prema sljedećoj formuli:

$$IoU = \frac{TP}{TP + FP + FN}, \tag{7.1}$$

gdje je TP broj piksela instance koje je model ispravno označio, FP broj piksla pozadine koje je model označio kao piksele instance i FN broj piksela instance koje je model označio kao piksele pozadine.

7.3. Eksperimentalno vrednovanje

7.3.1. Validacija rezolucije ulaznih slika

Iz skupa za učenje smo odbacili sve instance čiji se pravokutni okviri nisu mogli proširiti u ulaznoj slici. Za ekstrakciju poravnatih značajki koristimo mrežu Faster R-CNN koja se bazira na ResNet arhitekturi s 50 konvolucijskih slojeva i koja je trenirana na COCO skupu podataka. Za vrijeme učenja suparničkog generativnog modela ne osvježavamo slojeve Faster R-CNN mreže. Treniranu mrežu smo preuzeli s [1]. Mreža je trenirana s ulazom dimenzija $800 \times 1333 \times 3$, a koristimo ju na Cityscapes skupu podataka koji se sastoji od slika dimenzija $1024 \times 2048 \times 3$. Stoga nam se nametnulo pitanje trebamo li ostaviti slike iz Cityscapes skupa podataka u originalnoj veličini ili ih smanjiti na veličinu ulaza na kojem je mreža učena. Da bi odgovorili na to pitanje evaluirali smo treniranu mrežu na automobilima iz Cityscapes skupa za validaciju s dvije navedene dimenzije ulaza. Za evaluaciju je korištena metrika prosječna preciznost (engl. *average precision, AP*), a rezultati evaluacije su dani u tablici 7.1. U indeksu metrike je naveden minimalni Jaccardov index između detekcije i pravokutnog okvira iz skupa za validaciju koji određuje pozitivnu detekciju. Metrika $AP_{IoU=0.50:0.95}$ odgovara aritmetičkoj sredini prosječnih preciznosti s Jaccardovim indeksima u rasponu od 0.50 do 0.95 s korakom 0.05. U posljednja tri retka tablice dana je prosječna preciznost nad malim, srednjim i velikim objektima. Prema definiciji iz COCO skupa podataka mali objekti su oni koji imaju manje od 32^2 piksela u segmentacijskoj maski, srednji

oni koji imaju između 32^2 i 96^2 piksela te veliki oni koji imaju više 96^2 od piksela u segmentacijskom maski. Evaluacijski rezultati pokazuju da je prosječna preciznost na skupu za validaciju generalno bolja za dimenzije ulaza $1024 \times 2048 \times 3$. Zato smo tijekom ekstrakcije poravnatih značajki slike ostavili u originalnoj veličini.

Tablica 7.1: Prosječna preciznost Faster R-CNN mreže na automobilima iz Cityscapes skupa za validaciju. Prosječna preciznost je definirana prema standardu COCO skupa podataka.

	Dimenzije ulaza	
Metrika	$800 \times 1333 \times 3$	$1024 \times 2048 \times 3$
$AP_{IoU=0.50:0.95}$	0.370	**0.434**
$AP_{IoU=0.50}$	0.632	**0.703**
$AP_{IoU=0.75}$	0.375	**0.447**
$AP_{IoU=0.50:0.95,mali}$	0.079	**0.161**
$AP_{IoU=0.50:0.95,srednji}$	0.415	**0.501**
$AP_{IoU=0.50:0.95,veliki}$	**0.742**	0.732

7.3.2. Utjecaj veličine objekata na konvergenciju učenja

Evaluacijski rezultati iz tablice 7.1 nam također omogućuju da uočimo kako detektor Faster R-CNN ima dosta lošije performanse na malim objektima. Razlog tome može biti u tome da se predikcije rade na ekstrahiranim značajkama koje su na 32 puta manjoj rezoluciji od ulaznih slika te zato za male objekte značajke nisu dovoljno informativne. To nam daje naslutiti da ekstrahirane značajke za male objekte također neće biti dovoljno informativne na zadatku semantičke segmentacije jer je taj zadatak još kompleksniji od zadatka detekcije objekata. Proveli smo identičnu evaluaciju i s Faster R-CNN detektorom koji ekstrahira značajke na različitim rezolucijama i bazira na arhitekturi ResNet s 50 konvolucijskih slojeva. Rezultati su bili malo bolji od rezultata dobivenim s detektorom koji ekstrahira značajke na jednoj rezoluciji, ali smo odlučili koristiti detektor koji ekstrahira značajke na jednoj rezoluciji da bi rezultati bili usporedivi onima iz [22].

Najprije smo učili suparnički generativni model na automobilima iz skupa za učenje. Željeli smo vidjeti kako će se model ponašati na malim automobilima koji su predstavljali problem prilikom detekcije mrežom Faster R-CNN. Primijetili smo da model ne uspijeva segmentirati male automobile te da za njih generira prazne maske. Zato smo se odlučili fokusirati na velike automobile, odnosno na automobile čiji su pravokutni okviri veći od 100 piksela po visini i po širini, po uzoru na [22]. Model je za takve automobile prestao generirati prazne maske.

Za usporedbu performansi našeg slabo naziranog modela, kojeg ćemo oslovljavati s GAN, koristit ćemo dvije metode. Prva je jednostavna metoda koju ćemo oslovljavati s Box, a koja sve piksele unutar pravokutnog okvira označava pikselima objekta. S obzirom na to da su pravokutni okviri jako tijesno postavljeni oko automobila iz Cityscapes skupa podataka onda ima smisla usporedba s tom metodom. Druga metoda, koju ćemo oslovljavati s Nadzirana, je nadzirana varijanta našeg modela, odnosno model koji je identičan generatoru našega modela ali koji smo učili s oznakama na razini piksela i gubitkom unakrsne entropije. Nadzirani model smo učili s identičnim parametrima kao i slabo nadzirani model te također za vrijeme učenja modela nismo osvježavati slojeve preuzete iz detektora Faster R-CNN. Jaccardov indeks za sve tri metode na automobilima iz Ciyscapes skupa za validaciju dan je u tablici 7.2. Metode koje su učene samo na velikim automobilima su također evaluirane samo na velikim automobilima. Rezultati pokazuju da polunadzirani model ne uspijeva postići zadovoljavajuće rezultate nad svim automobilima jer je lošiji od naivne Box metode. Za velike automobile model postiže bolje rezultate od Box metode te doseže 80% performansi nadziranoga modela.

Tablica 7.2: Jaccardov indeks na automobilima iz Cityscapes skupa za validaciju. Metoda Box sve piksele proglašava pikselima objekta, GAN je slabo nadzirani suparnički generativni model, a Nadzirana metoda je generator slabo nadziranog modela učen s oznakama na razini piksela.

Metoda	Svi automobili	Veliki automobili
Box	0.5570	0.6437
GAN	0.4876	0.6831
Nadzirana	0.7160	0.8548

7.3.3. Utjecaj skupa za treniranje modela

S obzirom na to da model ne generira dobre segmentacijske maske za malene automobile sljedeće ćemo eksperimente provoditi isključivo na velikim automobilima, to jest modele ćemo učiti i validirati na velikim automobilima. Provjerit ćemo koristi li slabo nadzirani suparnički generativni model od dodatnih podataka za učenje. Model ćemo učiti s označenim pravokutnim okvirima iz skupa za učenje i s predikcijama dobivenim Faster R-CNN detektorom na po pet slika iz video isječaka povezanih sa skupom za učenje. U tablici 7.3 je dan Jaccardov indeks za model učen samo s označenim pravokutnim okvirima iz skupa za učenje i za model učen na predikcijama i označenim pravokutnim okvirima iz skupa za učenje. Jaccardov indeks modela učenog

s dodatnim podacima je značajno manji, a razlog tome mogu biti nedovoljno precizne detekcije Faster R-CNN detektora.

Tablica 7.3: Usporedba Jaccardovog indeksa na velikim automobilima iz Cityscapes skupa za validaciju za model učen s oznakama iz skupa za učenja i predikcijama dobivenim Faster R-CNN detektorom na po pet slika iz video isječaka povezanih sa skupom za učenje.

Skup za učenje	GAN
oznake	0.6831
oznake + predikcije na 5 slika iz videa	0.6084

7.3.4. Utjecaj razdiobe za odabir pozadine

Prethodni eksperimenti su provedeni sa slabo nadziranim suparničkim generativnim modelom koji određuje lokaciju pozadine uz pomoć normalne razdiobe. U tablici 7.4 je naveden Jaccardov indeks za slabo nadzirani suparnički generativni model koji lokaciju pozadine određuje uz pomoć uniformne razdiobe i pazi da se u pozadini ne nalaze drugi automobili. Takav model ćemo oslovljavati s GAN$_{uniform}$. Jaccardov indeks pokazuje da su rezultati dobiveni s oba modela usporedivi te da je odabir lokacije pozadine na oba načina prikladan za učenje modela. S obzirom na broj preklapanja instanci automobila u Cityscapes skupu podataka možda nije potrebno paziti da se u pozadini lažnih slika ne nalaze drugi automobili jer su automobili često prisutni u pozadini stvarnih slika.

Tablica 7.4: Jaccardov indeks na velikim automobilima iz skupa za validaciju. GAN model lokaciju pozadine određuje uz pomoć normalne razdiobe, a GAN$_{uniform}$ model uz pomoć uniformne razdiobe.

Model	
GAN	0.6831
GAN$_{uniform}$	0.6829

7.3.5. Usporedba s rezultatima iz originalnog rada

Slabo nadzirani suparnički generativni model iz originalnog rada [22] je učen s označenim pravokutnim okvirima iz skupa za učenje i s predikcijama dobivenim Faster R-CNN detektorom na svim slikama iz video isječaka povezanih sa skupom za učenje. Mi smo najbolje rezultate dobili učeći model samo na oznakama iz skupa za učenje te

nam se učenje s predikcijama nije pokazalo korisnim. U tablici 7.5 je dan Jaccardov indeks modela iz [22] i našeg modela učenog na različitim skupovima za učenje. Sudeći prema Jaccardovu indeksu možemo zaključiti da su rezultati našeg modela učenog samo s oznakama iz skupa za učenje usporedivi s rezultatima iz originalnog rada.

Tablica 7.5: Usporedba Jaccardovog indeksa na velikim automobilima iz Cityscapes skupa za validaciju za naš model učen s oznakama iz skupa za učenja i predikcijama dobivenim Faster R-CNN detektorom na po pet slika iz video isječaka te za model iz [22] učen s oznakama iz skupa za učenje i predikcijama na svim slikama iz video isječaka.

Skup za učenje	GAN	[22]
oznake	0.6831	-
oznake + predikcije na 5 slika iz videa	0.6084	-
oznake + predikcije na svim slikama iz videa	-	0.67

7.3.6. Utjecaj točnosti detekcijskih okvira

S ciljem da bi poboljšali rezultate prethodnih eksperimenta odlučili smo prilagoditi predtrenirani Faster R-CNN detektor Cityscapes skupu podataka, odnosno dodatno ga trenirati na instancama iz označenog Cityscapes skupa za učenje. Usporedba prosječne preciznosti detektora treniranog na COCO te potom na Cityscapes skupu podataka i detektora treniranog samo na COCO skupu podataka dana je u tablici 7.6. Rezultati evaluacije na automobilima iz Cityscapes skupa za validaciju pokazuju da je prosječna preciznost detektora prilagođenog Cityscapes skupu podataka značajno veća.

Tablica 7.6: Prosječna preciznost Faster R-CNN mreže na automobilima iz Cityscapes skupa za validaciju. Model označen s COCO + Cityscapes je predtreniran na COCO skupu podataka te učenjem s prijenosom prilagođen Cityscapes skupu podataka, dok je model označen s COCO treniran samo na COCO skupu podataka.

Metrika	COCO + Cityscapes	COCO
$AP_{IoU=0.50:0.95}$	**0.562**	0.434
$AP_{IoU=0.50}$	**0.797**	0.703
$AP_{IoU=0.75}$	**0.595**	0.447
$AP_{IoU=0.50:0.95,mali}$	**0.260**	0.161
$AP_{IoU=0.50:0.95,srednji}$	**0.658**	0.501
$AP_{IoU=0.50:0.95,veliki}$	**0.855**	0.732

Ponovili neke od eksperimenata sa slabo nadziranim suparničkim generativnim modelom koristeći ekstrahirane značajke iz Faster R-CNN detektora prilagođenog Cityscapes skupu podataka. Rezultati ponovljenih eksperimenata su dani u tablici 7.7, a mo-

del s novim značajkama je označen kao $\text{GAN}_{Cityscapes}$. Predikcije ze učenje modela $\text{GAN}_{Cityscapes}$ su dobivene s Faster R-CNN treniranim na Cityscapes skupu podataka. Model $\text{GAN}_{Cityscapes}$ ima veći Jaccardov indeks od GAN modela koji koristi značajke detektora treniranog na COCO skupu podataka. Međutim, Jaccardov indeks je značajno veći samo za model koji je treniran na oznakama iz skupa za učenje i predikcijama na slikama iz video sekvenci. Vjerojatno su točnije predikcije Faster R-CNN modela treniranog na Cityscapes skupu podataka doprinijele tome.

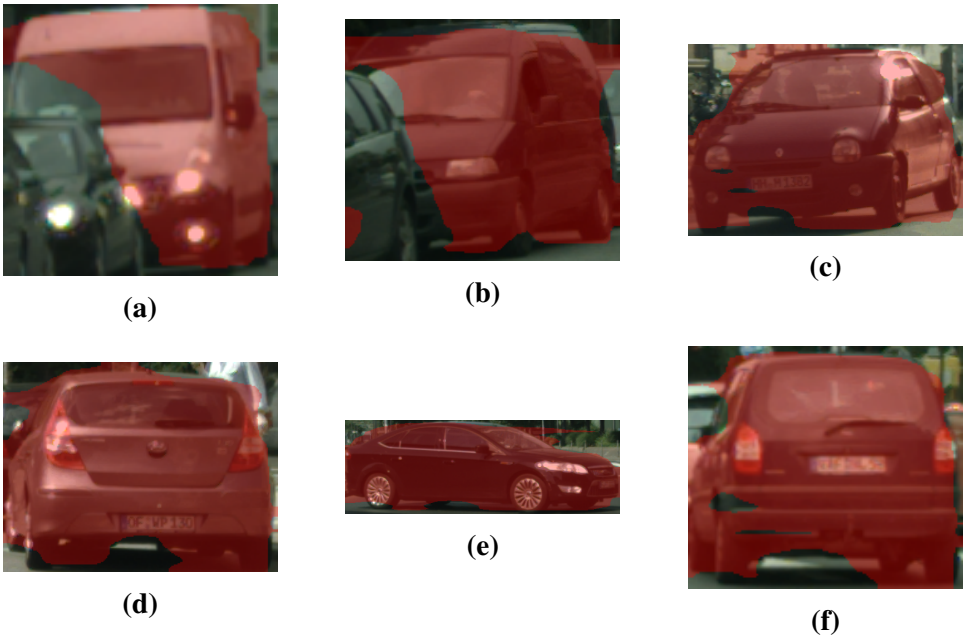
Tablica 7.7: Usporedba Jaccardovog indeksa na automobilima iz Cityscapes skupa za validaciju za model $\text{GAN}_{Cityscapes}$ koji koristi ekstrahirane značajke iz Faster R-CNN detektora treniranog na Cityscapes skupu podataka i za model GAN koji koristi ekstrahirane značajke iz Faster R-CNN detektora treniranog na COCO skupu podataka.

	$\text{GAN}_{Cityscapes}$	GAN
Veliki automobili		
oznake	**0.6924**	0.6831
oznake + predikcije na 5 slika iz videa	**0.6653**	0.6084
Svi automobili		
oznake	**0.5178**	0.4876

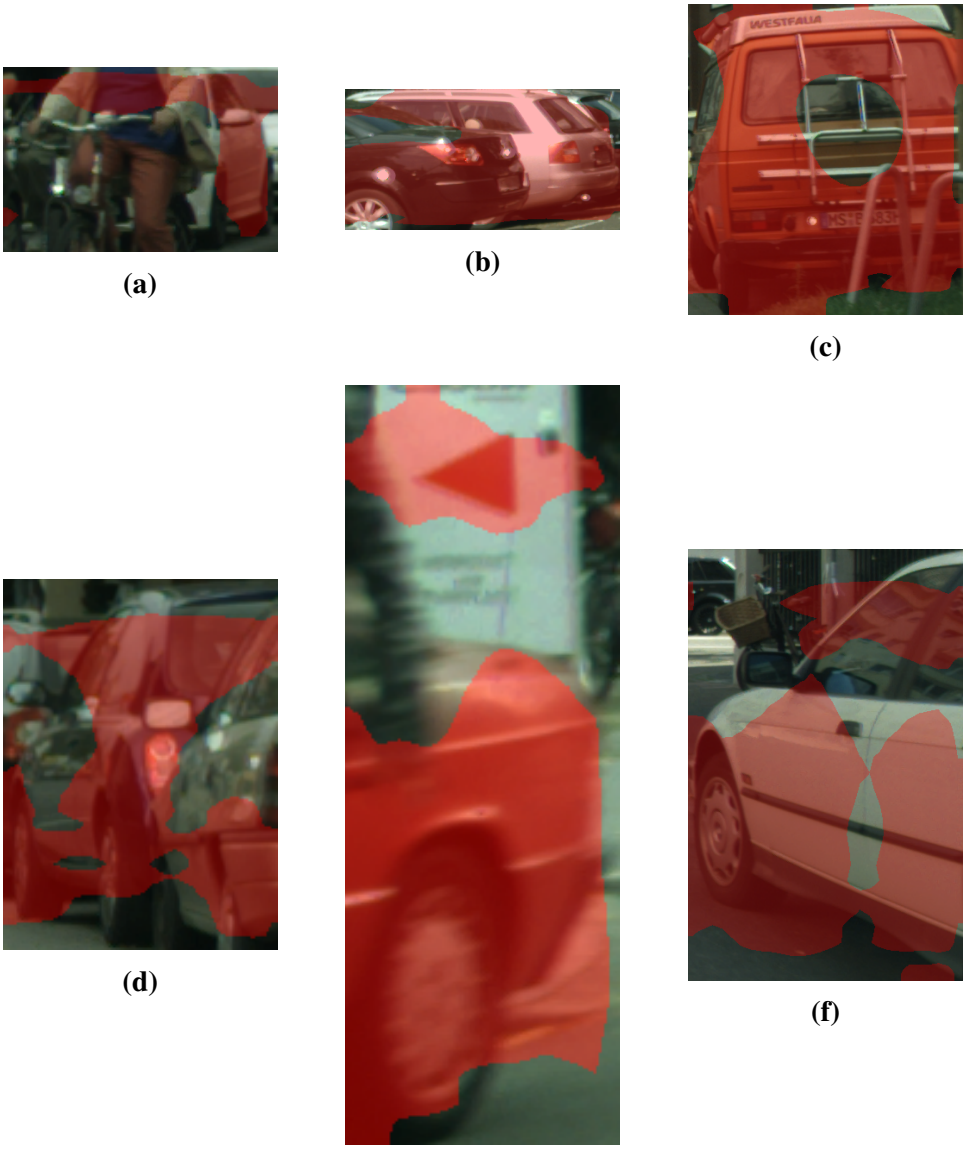
7.3.7. Kvalitativni prikaz rezultata

Primjeri stvarnih i lažnih slika te segmentacijskih maski dobivenih za vrijeme učenja $\text{GAN}_{Cityscapes}$ modela na velikim automobilima iz Cityscapes skupa za učenje su prikazani na slici 7.1. Vidimo da segmentacijske maske nisu jako precizne, ali da lažne slike svejedno izgledaju realistično te nalikuju na stvarne slike. Također vidimo da je česta pojava da se u slikama nalazi više instanci automobila te da za model nije jednostavno odrediti koju od instanci segmentirati. Primjeri segmentacijskih maski dobivenih istim modelom i preklopljenih preko automobila iz Cityscapes skupa za validaciju su prikazani na slikama 7.2 i 7.3. Na slici 7.2 su prikazane neke od boljih, a na slici 7.3 su prikazane neke od lošijih segmentacija. Model najčešće griješi kada se u slici nalazi više preklopljenih instanci te kada automobili nisu u fokusu slike nego su zaklonjeni instancama drugih klasa kao što je to slučaj na slici 7.3a.

Slika 7.1: Primjer stvarnih i lažnih slika za vrijeme učenja suparničkog generativnog modela. U prvom redu su prikazane proširene stvarne slike, u drugom redu proširene segmentacijske maske i u trećem redu sintetizirane lažne slike.



Slika 7.2: Primjer boljih segmentacijskih maki dobivenih slabo nadziranim suparničkim generativnim modelom na slikama iz Cityscapes skupa za validaciju.



Slika 7.3: Primjer lošijih segmentacijskih maki dobivenih slabo nadziranim suparničkim generativnim modelom na slikama iz Cityscapes skupa za validaciju.

8. Zaključak

U ovom radu su opisani neki od pristupa za slabo nadziranu semantičku segmentaciju koji se baziraju na dubokom učenju. Opisani pristupi ne koriste oznake za učenje na razini piksela nego koriste označene pravokutne okvire, klasifikacijske oznake, točke i šare. Opisani su i suparnički učeni generativni modeli koji se koriste za semantičku segmentaciju. U sklopu rada je implementiran model suparničkog učenja za slabo nadziranu semantičku segmentaciju zasnovan na pravokutnim okvirima i ideji da se objekti mogu gibati neovisno od pozadine. Generator suparničkog generativnog modela generira segmentacijske maske na temelju kojih se objekti izrezuju iz slika i lijepe na novu pozadinu tvoreći lažne slike. Diskriminator služi za ocjenu kvalitete lažnih slika i pruža signale za učenje generatora.

Generator implementiranog slabo nadziranog suparničkog generativnog modela je specifičan po tome što koristi značajke iz Faster R-CNN modela treniranog za detekciju objekata te na temelju tih ekstrahiranih značajki gradi segmentacijske maske. Time je pokazano kako se model naučen za detekciju objekata može prijenosom znanja upotrijebiti za semantičku segmentaciju bez potrebe za dodatnim oznakama za učenje.

Eksperimentima na automobilima iz Cityscapes skupa podataka smo pokazali kako se model s malo označenih pravokutnih okvira može naučiti za semantičku segmentaciju. Posebno zahtjevna se pokazala segmentacija instanci koje se preklapaju s drugim instancama istog razreda. Model nije pokazao dobre performanse za malene automobile iz Cityscapes skupa podataka jer značajke korištenog Faster R-CNN detektora nisu prilagođene malenim objektima. Za velike automobile se usporedivi rezultati postižu s detektorom koji je treniran samo na COCO skupu podataka i s detektorom dodatno treniranim na Cityscapes skupu podataka. Za malene automobile smo poboljšali rezultate nakon što smo detektor dodatno učili na Cityscapes skupu podataka, međutim model i daje ne daje zadovoljavajuće rezultate za malene automobile. U budućnosti bi se trebalo više vremena utrošiti na segmentaciju malenih objekata te koristiti detektor prilagođen manjim objektima.

LITERATURA

[1] Naučeni modeli faster r-cnn i mask r-cnn. URL https://github.com/facebookresearch/maskrcnn-benchmark/blob/master/MODEL_ZOO.md.

[2] R. Arandjelović and A. Zisserman. Object discovery with a copy-pasting gan. 2019. URL https://arxiv.org/abs/1905.11369.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. URL http://arxiv.org/abs/1604.01685.

[4] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. 2015. URL http://arxiv.org/abs/1503.01640.

[5] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010. URL https://doi.org/10.1007/s11263-009-0275-4.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. 2014. URL https://arxiv.org/abs/1406.2661.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[8] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[9] W. Hung, Y. Tsai, Y. Liou, Y. Lin, and M. Yang. Adversarial learning for semi-supervised semantic segmentation. *CoRR*, abs/1802.07934, 2018. URL http://arxiv.org/abs/1802.07934.

[10] F. Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? 2015. URL hhttps://arxiv.org/abs/1511.05101.

[11] A. Khoreva, R. Benenson, J. H. Hosang, M. Hein, and B. Schiele. Weakly supervised semantic labelling and instance segmentation. *CoRR*, abs/1603.07485, 2016. URL http://arxiv.org/abs/1603.07485.

[12] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. *CoRR*, abs/1604.05144, 2016. URL http://arxiv.org/abs/1604.05144.

[13] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[15] P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks. *CoRR*, abs/1611.08408, 2016. URL http://arxiv.org/abs/1611.08408.

[16] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016. URL http://arxiv.org/abs/1611.04076.

[17] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. On the effectiveness of least squares generative adversarial networks. *CoRR*, abs/1712.06391, 2017. URL http://arxiv.org/abs/1712.06391.

[18] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. URL http://arxiv.org/abs/1411.1784.

[19] O. R. Olga, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual

recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, 2015. ISSN 0920-5691. URL http://dx.doi.org/10.1007/s11263-015-0816-y.

[20] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. *CoRR*, abs/1412.7144, 2014.

[21] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marqués, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *CoRR*, abs/1503.00848, 2015. URL http://arxiv.org/abs/1503.00848.

[22] T. Remez, J. Huang, and M. Brown. Learning to segment via cut-and-paste. *CoRR*, abs/1803.06414, 2018. URL http://arxiv.org/abs/1803.06414.

[23] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[24] C. Rother, V. Kolmogorov, and A. Blake. Grabcut -interactive foreground extraction using iterated graph cuts. 2004.

[25] O. Russakovsky, A. L. Bearman, V. Ferrari, and F. Li. What's the point: Semantic segmentation with point supervision. *CoRR*, abs/1506.02106, 2015. URL http://arxiv.org/abs/1506.02106.

[26] T. Silva. An intuitive introduction to generative adversarial networks (gans), 2018. URL https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans Accessed: 2019-06-10.

[27] N. Souly, C. Spampinato, and M. Shah. Semi and weakly supervised semantic segmentation using generative adversarial network. *CoRR*, abs/1703.09695, 2017. URL http://arxiv.org/abs/1703.09695.

[28] Y. Wei, J. Feng, X. Liang, M. Cheng, Y. Zhao, and S. Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. *CoRR*, abs/1703.08448, 2017. URL http://arxiv.org/abs/1703.08448.

[29] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016.

Slabo nadzirana semantička segmentacija primjeraka primjenom suparničkog učenja

Sažetak

Semantička segmentacija prirodnih scena je važan zadatak računalnog vida. Skup postupak pribavljanja oznaka na razini piksela potrebnih za nadzirani pristup semantičkoj segmentaciji je potaknuo razvoj slabo nadziranih pristupa. U ovom radu su opisani neki od slabo nadziranih pristupa za semantičku segmentaciju koji se baziraju na dubokom učenju kao i primjene suparničkog učenja za semantičku segmentaciju. U sklopu rada je implementiran model suparničkog učenja za slabo naziranu semantičku segmentaciju primjeraka zasnovan na pravokutnim okvirima. Generator koristi značajke naučenog Faster R-CNN detektora uz pomoć kojih gradi segmentacijske maske, a diskriminator pruža signale za učenje. Za učenje i validaciju modela su korišteni automobili iz Cityscapes skupa podataka te su prikazani dobiveni rezultati.

Ključne riječi: suparničko učenje, semantička segmentacija, segmentacija primjeraka, slabo nadzirano učenje, duboko učenje

Weakly supervised instance segmentation using adversarial learning

Abstract

Semantic segmentation of natural scenes is an important task of computer vision. Due to expensive pixel-wise annotations required for fully supervised semantic segmentation various weakly supervised approaches have been proposed. In this thesis, several weakly supervised semantic segmentation approaches based on deep learning have been described as well as applications of adversarial learning to semantic segmentation. Adversarial learning model for object instance segmentation based on bounding boxes is implemented. Generator uses learned Faster R-CNN features and constructs a segmentation mask while discriminator provides learning signals. Model was trained and evaluated on cars form Cityscapes dataset and the results are shown.

Keywords: adversarial learning, semantic segmentation, instance segmentation, weakly-supervised learning, deep learning