

Zahvaljujem se mentoru, prof. dr. sc. Siniši Šegviću na pomoći, strpljenju i znanju kojim mi je pomogao napraviti ovaj rad. Također se zahvaljujem svojoj obitelji i bližnjima na podršci tijekom studiranja.

SADRŽAJ

1. Uvod	1
2. Statistička teorija učenja	3
3. Generalizacija modela	5
3.1. Primjeri izvandistribucijskih podataka	10
4. Adaptacija domene	13
5. Opisi Modela	18
5.1. Model za segmentaciju	18
5.2. Model za prevođenje podataka	19
5.3. Model za adaptaciju domene	21
6. Sustav za semantičku segmentaciju zgrada - DeepSat	22
6.1. Implementacija infrastrukture za učenje	22
6.2. Skup podataka	27
6.3. Eksperimenti i rezultati	28
6.3.1. Eksperimenti s augmentacijama	29
6.3.2. Kovarijatni pomak	32
6.3.3. Ograničenja metode i primjeri rezultata	33
7. Zaključak	36
Literatura	37

1. Uvod

S razvojem metoda dubokog učenja i sa sve većom dostupnošću satelitskih i zračnih snimki otvaraju se nove mogućnosti koje nam omogućuju pametnije iskorištavanje tih podataka. Svrha ovog rada je istraživanje izvedivosti i izrada prototipa sustava za semantičku segmentaciju zgrada u satelitskim i zračnim snimkama te sustava za kontrolu nad procesom učenja takvih modela. Ovakav sustav mogao bi se koristiti u planiranju i kontroli urbanizacije, te bi omogućio prikupljanje vrijednih informacija o stopi urbanizacije odabranog geografskog prostora.

Glavni problem s kojim se susrećemo u ovom radu je generalizacija na izvandistribucijske primjere. Rješenje tog problema je ključno za kvalitetnu izvedbu i praktičnu primjenu ovakvih sustava. Zemlja je geološki, klimatski i vegetacijski vrlo raznoliko mjesto te je generalizacija na temelju ograničenog skupa za učenje na različite lokacije prilično težak zadatak. Nadzirano učenje još uvijek je glavna paradigma u svijetu ML-a te je zbog toga cijena prikupljanja oznaka za ovaj zadatak poprilično velika.

Pristupi kojim nastojimo riješiti navedene probleme je metoda adaptacije i prevođenja domene. U tim tehnikama koristimo neoznačene podatke iz strane domene kako bismo prenijeli karakteristike te domene na označene podatke. Tako možemo iskoristiti dio informacija iz strane domene te na temelju tih podataka bolje generalizirati na izvandistribucijskim podacima. Također, razvijamo i potpuno automatizirani proces obrade podataka i učenja u skladu s MLOps [Visengeriyeva et al. (2021)] praksama pomoću kojih možemo lakše voditi evidenciju o procesu učenja i bolje kontrolirati njegove faze. MLOps principi su standardi kojima je cilj izrada kvalitetnijih ML sustava. Primjenom tih principa nastojimo smanjiti ili ukloniti tehnički dug koji nastaje neodgovornim praksama kod izgradnje bilo kojih tehničkih sustava.



Slika 1.1: Satelitske snimke Novog Zagreba u različitim vremenskim periodima. Ovo je primjer podataka iz prakse na kojima bi model trebao raditi.

2. Statistička teorija učenja

Statistička teorija učenja [Vapnik (1999)] teorijski je temelj koji opisuje metodologiju prema kojoj nastojimo naučiti modele na temelju podataka. Nadzirano učenje sastoji se od četiri komponente:

- Generatora primjera x , koji stvara nezavisno i jednako distribuiran (i.i.d uvjet) skup za učenje $\mathcal{D} = (x_i, y_i)_i^n$ prema nepoznatoj distribuciji $P(x)$.
- Označivača koji za dani vektor x daje oznaku y prema uvjetnoj distribuciji $P(y|x)$.
- Parametriziranog modela koji nastoji naučiti funkciju $f : X \rightarrow Y$ s oznakom $f(x|\theta), \theta \in \Theta$.
- Funkcije gubitka $L(y, f(x|\theta))$ koja predstavlja neku mjeru udaljenosti između predikcija $f(x|\theta)$ i oznake y

Cilj algoritma za učenje je pronaći parametre θ koji minimiziraju očekivanu vrijednost funkcije gubitka $E[L]$. To očekivanje se u literaturi još i naziva *rizik parametra* i označava se s $R(\Theta)$, tj. vrijedi $E[L] = R(\Theta)$. Teorijski rizik parametra dan je s izrazom:

$$R(\Theta) = \int_{(x,y)} L(y, f(x|\theta)) p(x, y) dx dy.$$

Stoga se cilj učenja može zapisati kao:

$$\theta^* = \operatorname{argmin}_\theta R(\theta).$$

Teorijski rizik je u praksi nemoguće izračunati jer ne poznajemo distribuciju $P(x, y)$. Iskorištavamo skup za učenje \mathcal{D} i teorijsku vjerojatnost $P(x, y)$ aproksimiramo empirijskom vjerojatnošću $\hat{P}(x, y)$, koja je jednaka relativnoj frekvenciji primjera u skupu za učenje:

$$\hat{P}(x, y) = \frac{f(x, y)}{\sum_{x,y} f(x, y)}$$

Zbog toga definiramo proceduru *minimizacije empirijskog rizika* (*engl. empirical risk minimization, ERM*) koja minimizira empirijski rizik $R_{emp}(\theta)$. Empirijski rizik nad

skupom primjera veličine l definiramo kao:

$$R_{emp}(\theta) = \frac{1}{l} \sum_i^l L(y, f(x|\theta))$$

U kontekstu klasifikacije kao funkcija gubitka najčešće se koristi gubitak unakrsne entropije (*engl. cross-entropy loss*):

$$L(y, f(x|\theta)) = - \sum_i^C y_i \ln(f_i(x|\theta)).$$

Česta prepostavka koja ne vrijedi u svim slučajevima je prepostavka o nezavisnosti i jednako distribuiranim primjerima skupa za učenje i testiranje (*engl. independent and identically distributed, i.i.d.*). Zbog toga često se dešava da se kod stvaranja skupova podataka, podaci iz različitih domena spajaju. Pod pojmom domene mislimo na podatke koji su generirani u različitim eksperimentalnim postavama, a na istom zadatku. Primjer toga je skup slikanih ručnih potpisa različitih ljudi. Miješanjem tih slika gubimo informaciju o tome da su slike nastale od različitih ljudi. Zbog te prakse gubimo šansu učiti algoritam na jednoj domeni, a testirati na stranim domenama i time usmjeriti algoritam prema učenju invarijantnih značajki. Više o tome u drugom poglavlju.

3. Generalizacija modela

U prošlom poglavlju definirali smo općenitu metodologiju učenja na temelju podataka. U ovom poglavlju definiramo unutardistribucijsku (IID) i izvandistribucijsku (OOD) generalizaciju te navodimo neke mane u statističkoj teoriji učenja.

Definicija (Unutardistribucijska ili IID generalizacija). U okviru statističke teorije učenja kažemo da model ima dobre generalizacijske sposobnosti ako ima niski očekivani gubitak na podacima za testiranje koji su jednako i nezavisno distribuirani (i.i.d podaci) kao i podaci za učenje. Zbog toga tu vrstu generalizacije nazivamo unutardistribucijskom jer generalizaciju modela ocjenjujemo tako da mjerimo performansu modela na jednakim distribuiranim podacima.

Pod pojmom domene mislimo na eksperimentalne uvjete i kontekst u kojem snimamo objekt od interesa. Pritom mislimo na stvari poput npr. scene, svjetline, teksture, okoline, vremenske i lokacijske karakteristike itd. Problem s kojim se u praksi najčešće suočavamo je taj da se većina modela razvija na podacima koji dolaze iz ograničenog broja domena. Nakon što primijenimo modele u stvarnom svijetu, njihova performansa značajno pada. Razlog leži u činjenici da su podaci u stvarnom svijetu rijetko jednako distribuirani kao i podaci na skupu za razvoj. Efekt u kojem se distribucija podataka na skupu za učenje i distribucija podataka na skupu za testiranje razlikuju naziva se **kovarijatni pomak** (*engl. covariate shift*). Gornja definicija može se izraziti formalno kao:

$$p_{train}(\vec{x}) \neq p_{test}(\vec{x}).$$

Zbog toga podatke koji se ne nalaze unutar domena za razvoj nazivamo **izvandistribucijskim podacima**.

U praksi se problem kovarijatnog pomaka najčešće rješava tako da se brzo detektira pojava izvandistribucijskih podataka, zatim se te podatke označi te se nakon toga trenira na tim podacima kako bismo naučili karakteristike nove domene i njenu distribuciju. Problem je taj što generalizacija izvan domene zahtjeva učenje na velikom broju podataka i što je označavanje slika vrlo skup zadatka [Bevandic et al. (2021)].

Da bismo razumjeli zašto kovarijatni pomak uopće nastaje nužno je razumjeti stva-

ranje podataka unutar neke domene. Proces generacije podataka opisujemo strukturnim kauzalnim grafom (*engl. Structural Causal Model, SCM*) ili probabilističkim grafičkim modelom koji opisuje pozadinsku kauzalnu strukturu procesa koja generira podatke [Pearl (2003)]. Podatke koje taj graf stvara nazivamo opservacijskim podacima. U praksi najčešće imamo samo opservacijske podatke, a generativni model podataka je nepoznat. SCM je formalno opisan s četvorkom $(U, V, F, P(u))$, gdje je:

- U je pozadinska varijabla čija je vrijednost uzrokovana procesima van našeg modela.
- V su unutrašnje varijable čije su vrijednosti uzrokovane samo s varijablama unutar našeg modela, tj. s $U \cup V$.
- F je skup funkcija koje određuju vrijednosti unutrašnjih varijabli.
- $P(u)$ je distribucija svih pozadinskih varijabli.

Na slici 3.2 možemo vidjeti graf i opis varijabli tog procesa. Zbog toga što je pozadinska varijabla c pokretač procesa stvaranja podatka x , postojat će asocijacija između domene d i cilja y . To znači da će domena i ciljna varijabla zavisiti jedna o drugoj, tj. $p(y, d) \neq p(y)p(d)$. To je problematično zato što se iz značajki domene može predvidjeti varijabla cilja y . Kažemo da postoji asocijacija između varijabli d i y koja nije kauzalna, tj. ta **asocijacija je lažna** (*engl. spurious correlation*) [Ilse et al. (2020)] jer ne postoji direktna kauzalna veza između varijabli. U slučaju kada su dvije varijable zavisne, postoji 3 mogućnosti kako bi te dvije varijable mogle biti povezane. Prve dvije mogućnosti su da jedna od njih uzrokuje direktno ili indirektno drugu varijablu. Posljednja mogućnost je ta da postoji pozadinska zajednička varijabla koja uzrokuje obadvije varijable. Zbog toga će opservacijski podaci uvijek sadržavati neistinite asocijacije koje će model iskoristiti da bi poboljšao performanse. To nije dobro jer model uči loše značajke koje neće imati dobru prediktivnu moć na izvandistribucijskim podacima.

Način kako bismo mogli pomoći modelu da nauči kvalitetnije značajke jest da kontroliramo varijable domene h_d ili d . Na taj način zatvaramo put između varijabli domene i cilja i činimo ih nezavisnima. U literaturi se takva procedura naziva intervencija i formalno se označava s do operatorom [Pearl (2003)]. Korištenjem intervencije na domenu d želimo uspostaviti svojstvo $P(y|do(d)) = P(y)$ i natjerati model da ne obraća pažnju na domenu nego samo na cilj. Intervencija može označavati postavljanje varijable na određenu vrijednost ili dodavanje nasumičnog šuma uzorkovanoj vrijednosti te varijable. U praksi se takva procedura može implementirati modeliranjem domene i ciljnog objekta u nekom grafičkom alatu te kontroliranjem značajki domene i ciljnog

objekta. Nakon toga možemo vrlo lagano generirati opservacijske podatke iz takvog modeliranog svijeta te ih koristiti za učenje kvalitetnijih značajki.

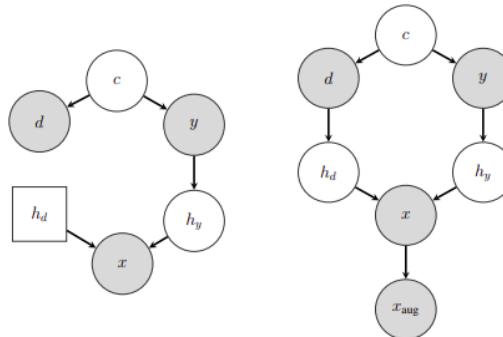
U većini slučajeva ne poznajemo generativni proces podataka, nego posjedujemo samo opservacijske podatke. Zbog toga što imamo pristup samo realizacijama varijable X dodajemo još jedan sloj obrade podataka s namjerom da simuliramo intervenciju nad domenom u generativnom procesu. Za to koristimo **augmentacije** ili nasumične transformacije nad podacima tijekom pribavljanja podataka. Simulacija intervencije uz pomoć augmentacije može se vidjeti na slici 3.1. Augmentacije i intervencije moraju zadovoljavati uvjet ekvivariantnosti (*engl. intervention-augmentation equivariant*) [Ilse et al. (2020)]. Kažemo da je funkcija f ekvivariantna s obzirom na transformaciju g ako vrijedi:

$$f(g(x)) = h(f(x)) \quad (3.1)$$

Drugačije rečeno, transformacija g na ulaznim podacima x se prenosi na reprezentaciju $f(x)$ pomoću funkcije h . U našem slučaju mora vrijediti slična relacija, ali su u njoj augmentacija i *do* operator međusobno zamjenjivi:

$$\text{aug}(f_x(h_d, h_y)) = f_x(\text{do}(h_d), h_y) \quad (3.2)$$

To jest biramo augmentacije koje će davati isti efekt kao da radimo intervenciju na značajkama domene.

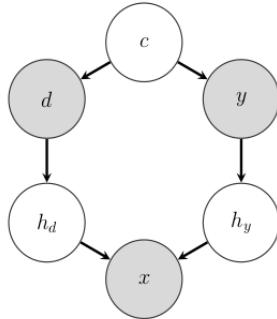


Slika 3.1: Simulacija intervencije pomoću augmentacije. Na lijevoj slici prikazan je generativni model u kojem je napravljena intervencija na značajkama domene, dok je na desnoj slici prikazan graf kojem je dodana još jedna varijabla x_{aug} koja predstavlja proces augmentacije podataka.

Sada možemo definirati svojstvo izvandistribucijske generalizacije.

Definicija (Izvandistribucijska ili OOD generalizacija). Imamo na raspolaganju N grupa podataka G , tj.

$$\{G^{d=i}\}_{i=1}^N, G^{d=i} = \{(x, y)_{i=1}^M\}.$$



Slika 3.2: Strukturni kauzalni graf procesa koji stvara podatke \mathbf{x} . Kružići predstavljaju slučajne varijable koje podliježu nekoj distribuciji. Proces započinje općenitom pozadinskom varijablu c čija realizacija vodi k realizaciji varijable d koja predstavlja domenu skupa podataka i y koja predstavlja cilj koji želimo klasificirati, detektirati, segmentirati, itd. Varijable h_d i h_y predstavljaju značajke prethodnih varijabli. Spajanjem tih dviju varijabli dobivamo konični primjer \mathbf{x} . U tom grafu varijabla c je pozadinska varijabla, dok su sve ostale unutrašnje varijable. Sivom bojom su označene opservacijske varijable, a bijelom skrivene. Varijable se nazivaju opservacijskim jer možemo vidjeti njihove realizacije.

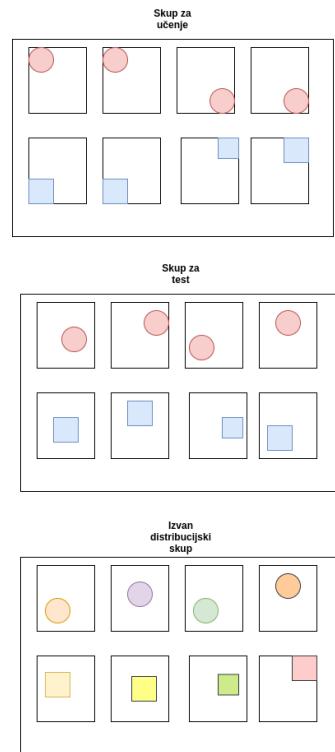
U svakoj grupi podataka primjeri su različito distribuirani. Algoritam koji posjeduje sposobnost izvandistribucijske generalizacije učit će na nekoj grupi d_i , pod distribucijom $p(x)^{d_i}$ i radit će podjednako dobro na ostalim neviđenim grupama koje su druge distribuirane. Da bi se takva situacija uopće mogla desiti model mora naučiti **invarijantne** značajke iako grupe na kojima uči imaju neistinite asocijacije u podacima. Sinonim za svojstvo invarijantnosti je stabilnost, a formalna definicija je:

$$f(g(x)) = f(x) \quad (3.3)$$

Kažemo da je funkcija f invarijantna na transformaciju g . Uporaba transformacije g je suvišna jer je f otporna na tu transformaciju. Svojstvo invarijantnosti je poseban slučaj ekvivarijantnosti u kojem je funkcija h iz jednadžbe 3.2 zamijenjena funkcijom identiteta.

Primjerice, model koji uči prepoznavati krave samo iz slika planinskih pašnjaka puno će lošije raditi ako testiramo taj algoritam u pustinjskim predjelima zbog toga što model nije naučio značajke koje su invarijantne na to gdje se krave nalaze. Da bi model posjedovao svojstvo izvandistribucijske generalizacije mora naučiti značajke koje su invarijantne na domenu na kojoj je model učio. Da bi se to postiglo potrebno je razbiti povezanost značajki cilja i domene augmentacijama, te pomoći modelu da nauči visoko apstraktne značajke (*engl. invariant high-level features*) invarijante na domenu u kojoj su naučene. Primjer problema izvandistribucijske generalizacije može

se vidjeti na 3.3.



Slika 3.3: Primjer skupa podataka u kojem postoji opasnost da model nauči lažnu asocijaciju. Učimo model na podacima koji sadrže slike plavih kvadrata i crvenih krugova, zadataka modela je predvidjeti klasu objekta na slici. Nažalost posjedujemo samo skup u kojem su kružići u gornjem lijevom i donjem desnom kutu, a kvadrati u suprotnim kutovima. Postoji opasnost da model napravi vezu između lokacije i klase objekta koja u stvarnom životu ne postoji. Zbog toga će performanse modela na skupu za testiranje bitni znatno lošije. U izvandistribucijskim podacima stvar je još komplikiranija jer su objekti različitih boja.

3.1. Primjeri izvandistribucijskih podataka

U ovom dijelu prikazujemo primjere izvandistribucijskih podatka s kojima bi sustav za segmentaciju zgrada trebao raditi. Radi se o satelitskim snimkama grada Zagreba u različitim vremenskim trenucima. Neke karakteristike tih podataka je relativno jednostavno naučiti, poput osnovnih geometrijskih transformacija, dok su neke transformacije komplikiranije i potrebne su sofisticirane metode za uspostavljenje svojstva invarijantnosti na te transformacije.



Slika 3.4: Skaliranje. Različiti sateliti imaju različite prostorne rezolucije piksela. Prostorna rezolucija piksela je površina Zemlje koju on prikazuje.



Slika 3.5: Rotacija. Performanse modela ne bi trebale pasti za rotacije između [-180, 180].



Slika 3.6: Svjetlina, kontrast i zasićenje. Svaki satelit ima drugačiju senzorsku opremu koja daje drugačiju boju slike.



Slika 3.7: Sjene. Najčešće se radi o sjenama koje nastaju zbog obližnjih oblaka.



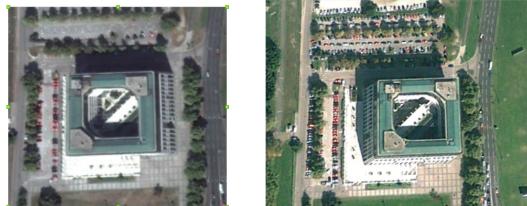
Slika 3.8: Gustoća naselja. Za gusto naseljene gradove postoji opasnost da model ne uspije razlučiti zgrade jednu od druge zbog njihove blizine.



Slika 3.9: Arhitektura i stil gradnje. Dijelovi nekog urbanog grada posjeduju različite arhitekturne stilove. Zadatak je naučiti stabilne karakteristike koje posjeduju različiti stilovi.



Slika 3.10: Godišnja doba. Zbog različitih godišnjih doba dešava se promjena boja te je zbog toga potrebno predvidjeti i prilagoditi rad sustava svakom dobu.



Slika 3.11: Različiti kutevi snimanja (*engl. off-nadir*). Sateliti najčešće ne snimaju površinu Zemlje koja se nalazi direktno ispod njih nego pod nekim kutem. Taj kut se u različitim prolažima nad istom lokacijom razlikuje jedan od drugoga.



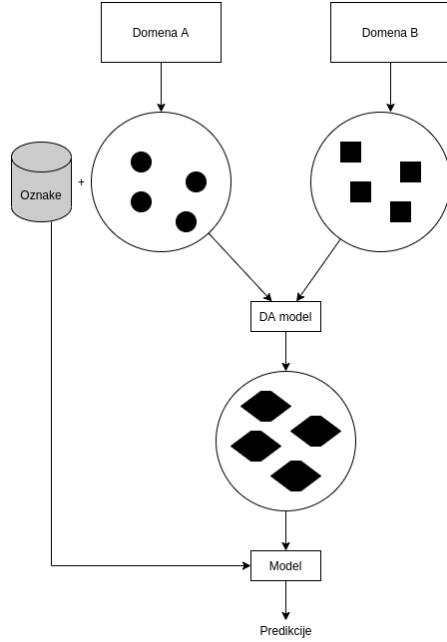
Slika 3.12: Atmosferske prilike. Često se zna desiti situacija u kojoj su pojedini objekti zaklonjeni i nije ih moguće vidjeti zbog različitih atmosferskih prilika. Takve situacije se najčešće rješavaju izradom nezavisnog sustava koji služi za segmentaciju atmosferskih prilika i ignoriranjem zaklonjenih regija.

4. Adaptacija domene

Adaptacija domene (*engl. domain adaptation*) je tehnika u kojoj je cilj prijenos karakteristika s neoznačenog skupa podataka na označene skupove podataka pri čemu želimo zadržavati sadržaj varijable cilja tih podataka. Ovu tehniku možemo iskoristiti onda kada posjedujemo podatke iz dviju različitih domena, gdje je prva domena označena, a druga nije ili je podskup njezinih podataka označen. S njom možemo jako brzo proširiti vlastiti skup podataka i korištenjem tih podataka poboljšati generalizacijske sposobnosti modela te smanjiti cijenu označivanja podataka. Općenita shema koja opisuje tu tehniku može se vidjeti na slici 4.1. Kako bismo mogli iskoristiti ovu metodu moramo posjedovati podatke iz različitih domena. Domenu koja posjeduje oznake zovemo izvornom, a ona koja nije u potpunosti označena naziva se ciljna domena.

U literaturi [Wilson i Cook (2018)] se adaptacije domene dijeli na nenadziranu i polu-nadziranu metodu. U prvom slučaju ciljna domena uopće ne sadrži oznake, dok je u drugom slučaju označen samo njezin podskup. Još jedna podjela [Wang i Deng (2018)] dolazi na temelju toga gdje radimo adaptaciju podataka. Zbog toga postoji dvije kategorije algoritma: adaptacija domene pomoću primjera (*engl. instance-based DA*) i adaptacija domene pomoću značajki (*engl. feature-based DA*). U prvoj inaćici adaptiramo originalne podatke, a u drugoj inaćici adaptiramo ekstrahirane značajke između domena. U ovom radu odabran je pristup u kojem radimo nenadziranu adaptaciju domene na originalnim primjerima ili slikama.

Trenutne metoda adaptacije domene najčešće se temelje na uporabi nekoliko GAN-ova, tj. neuronskih mreži koje igraju suparničku igru (*engl. adversarial game*) kako bi poravnali različite distribucije pri čemu je sadržaj ciljnih slika netaknut. GAN-ovi [Goodfellow et al. (2014)] su nenadzirana metoda u kojoj postoje dvije konvolucijske ili neuronske mreže: diskriminator i generator. Diskriminator je mreža kojoj je cilj predvidjeti da li ulazni podatak dolazi iz skupa za učenje, tj. uči distribuciju $P(\vec{x} \in \mathcal{D})$. Generator je pak mreža koja na ulazu prima vektor šuma i pokušava generirati što vjerniji sintetički primjerak iz skupa za učenje \mathcal{D} . Arhitektura te mreže prikazana je na slici 4.2. Te dvije mreže imaju suprotne ciljeve: generator želi da diskriminator



Slika 4.1: Dijagram koji opisuje metodu domenske adaptacije. Domena A generira podatke koje sadrže koncept kruga, dok domena B stvara podatke koje sadrže koncept kvadrata. Po-sjedujemo oznake za podatke A te stoga možemo lako naučiti koncept kruga, ali ne možemo naučiti koncept kvadrata jer nemamo oznake za te podatke. Zbog toga koristimo tehniku domenske adaptacije kako bismo prenijeli koncept kvadrata na podatke iz domene A. Nakon adaptacije dobivamo podatke koji u sebi sadrže karakteristike iz domene B, npr. koncept kuta koji se ne može naučiti u domeni A. Ti će podaci pomoći modelu u usvajaju dijelova znanja koje se nalaze u domeni B.

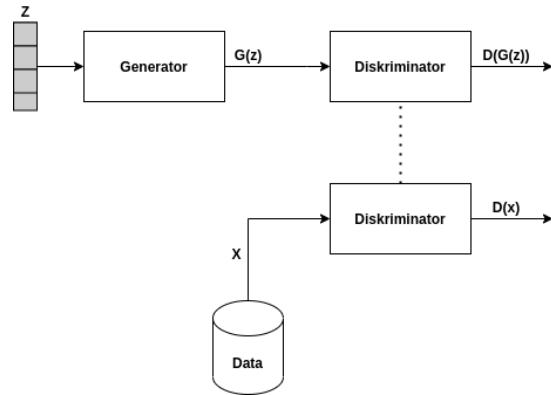
za njegove sintetičke primjere daje visoke vjerojatnosti, dok diskriminatore želi točno klasificirati primjere iz skupa podataka. Formuliramo ta dva različita interesa kroz optimizaciju minimax gubitka:

$$(\theta_G^*, \theta_D^*) = \operatorname{argmin}_{\theta_G} \max_{\theta_D} V(\theta_D, \theta_G) = \operatorname{argmin}_{\theta_G} \max_{\theta_D} \left\{ \mathbb{E}_x [\log D(x)] + \mathbb{E}_z [\log (1 - D(G(z)))] \right\} \quad (4.1)$$

Ovdje varijabla $x \sim p_{data}(x)$ predstavlja uzorke iz empirijske distribucije podataka, imamo varijablu $z \sim p_z(z)$ koja predstavlja šum, te imamo diskriminatore $D(x; \theta_d)$ i generator $G(z; \theta_g)$. U gornjoj jednakosti imamo dva gubitka, tj. dvije očekivane vrijednosti. Prvi izraz je očekivana izglednost da je primjer prav, a drugi izraz je očekivana izglednost da je primjer lažan. Zbog toga što za minmax optimizacijsku proceduru vrijedi:

$$\min_X \max_Y V(x, y) = \max_Y \min_X V(x, y) \quad (4.2)$$

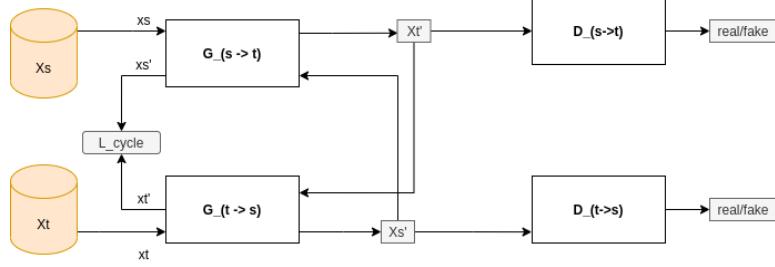
može se birati redoslijed evaluacije minimax izraza.



Slika 4.2: Osnovna arhitektura GAN modela. Generator generira lažne primjere, dok diskriminatore klasificira primjere u stvarne ili lažne.

Sad kada znamo osnove GAN arhitekture, opisujemo tehniku prevođenja domene (*engl. domain translation*). Za razliku od adaptacije domene modeli za prevođenje domene nisu dizajnirani s ciljem da zadrže semantiku ciljnog objekta, npr. zgrada, već da samo prenesu stil iz jedne domene na drugu. Zbog toga što će se stil iz ishodišne domene također prenijeti i na piksele ciljnog objekta u izvornoj domeni to će za posljedicu oslabiti povezanost piksela ciljnog objekta i maski. Postoji nekoliko različitih načina kako se prevođenje domene može implementirati, ali trenutno su najpoznatije implementacije koje koriste suparničke mreže. CycleGAN [Zhu et al. (2017)] je upravo primjer takve arhitekture, a poseban je po tome što ne zahtijeva uparene primjere između domena. U toj arhitekturi, za mapiranje domene koristimo uvjetne GAN-ove (*engl. conditional GANs*) [Mirza i Osindero (2014)], koji se razlikuju od običnih GAN-ova po tome što generatori stvaraju sintetičke slike uvjetovane drugim primjerom, a ne vektorom šuma. Treniranjem uvjetnih GAN-ova snimke s ulaza prevodimo u željenu domenu, a to radimo uz pomoć CycleGAN arhitekture [Zhu et al. (2017)] koja se nalazi na slici 4.3.

Uvjetni generator $G_{S \rightarrow T}$ i diskriminatore $D_{S \rightarrow T}$ igraju suparničku igru u kojoj u kočnici generator uči empirijsku distribuciju $p_{data}(x_t)$ ciljne domene T , a diskriminatore više ne prepoznaje $p_{data}(x_t)$ i $p(x'_t)$. Ista stvar se događa i u suprotnom smjeru. Iako su oba generatora naučili distribucije $p_{data}(x_s)$ i $p_{data}(x_t)$ to ne jamči da će sintetički podaci x'_t i x'_s biti interpretabilni i semantički razumljivi jer postoji široka klasa para-



Slika 4.3: Arhitektura CycleGAN [Zhu et al. (2017)] modela za prevođenje domene. Arhitektura se sastoji od dva uvjetna GAN-a, podataka iz izvorne domene S , podataka iz ciljne domene T i kružnog gubitka L_{cycle} . Taj gubitak je definiran kao L1 udaljenost između izvorne slike i njene rekonstrukcije. Rekonstrukcija se dobiva prevođenjem izvorne slike u stanu domenu te ponovnim vraćanjem sintetičke slike u izvornu domenu.

metara generatora koji će dati istu distribuciju podataka, a samo nekolicina čuva semantiku ulaznih primjera. Zato uvodimo još jedan uvjet koji sintetički primjeri moraju zadovoljavati, a riječ je o **kružnoj konzistentnosti** (*engl. cycle-consistency*) podataka. Taj uvjet kaže da se mapirani primjeri x'_t i x'_s moraju moći ponovno preslikati u izvornu domenu suprotnim generatorom i da moraju biti što sličniji izvornim primjerima x_t i x_s . Tu rečenicu možemo jednostavnije shvatiti njenom matematičkom formulacijom:

$$L_{cycle}(\theta_{s \rightarrow t}, \theta_{t \rightarrow s}) = \mathbb{E}_{x_s \sim p_{data}(x_s)}[|G_{s \rightarrow t}(G_{t \rightarrow s}(x_s)) - x_s|_1] + \mathbb{E}_{x_t \sim p_{data}(x_t)}[|G_{t \rightarrow s}(G_{s \rightarrow t}(x_t)) - x_t|_1] \quad (4.3)$$

U konačnici ukupni gubitak glasi:

$$\begin{aligned} L(G_{s \rightarrow t}, G_{t \rightarrow s}, D_{t \rightarrow s}, D_{s \rightarrow t}) &= L_{GAN}(G_{s \rightarrow t}, D_{t \rightarrow s}, X_s, X_t) + \\ &\quad L_{GAN}(G_{t \rightarrow s}, D_{s \rightarrow t}, X_t, X_s) + \\ &\quad \lambda L_{cycle}(G_{t \rightarrow s}, G_{s \rightarrow t}) \end{aligned} \quad (4.4)$$

S tim gubitkom rješavamo:

$$\theta_{s \rightarrow t}, \theta_{t \rightarrow s} = \operatorname{argmin}_{G_{s \rightarrow t}, G_{t \rightarrow s}} \max_{D_{s \rightarrow t}, D_{t \rightarrow s}} L(G_{s \rightarrow t}, G_{t \rightarrow s}, D_{t \rightarrow s}, D_{s \rightarrow t})$$

Model za adaptaciju s kojim radimo eksperimente inspiriran je CyCada (*engl. Cycle-Consistent Adversarial Domain Adaptation*) modelom [Hoffman et al. (2017)]. Problem nerazlikovanja sadržaja objekta i okoline rješavamo dodavanjem još jednog člana u ukupni gubitak – gubitak semantičke konzistentnosti (*engl. semantic-consistency loss*). Taj gubitak kažnjava mijenjanje piksela koji predstavljaju objekt od interesa. Za razliku od prevođenja podataka u novu domenu, u adaptaciji domene koristimo oznaće iz izvorne domene Y_s . Kod računanja gubitka semantičke konzistentnosti također

koristimo model f_s koji je naučen na podacima (X_s, Y_s) . Gubitak semantičke konzistentnosti zapravo je empirijski rizik prethodno naučenog nadziranog modela f_s nad mapiranim podacima X'_t i Y_s . Empirijski rizik najčešće iskazujemo pomoću unakrsne entropije. Svrha semantičke konzistentnosti je ta da usporedi predikcije naučenog modela f_s nad adaptiranim podacima s maskama i da kažnjava prevelika odstupanja.

Prepostavljamo da je model f_s invarijantan na značajke okoline i da je naučen prepoznavati značajke sadržaja u danim podacima. U našem slučaju taj model je smrznut i ne uči se. Koristeći prethodnu prepostavku možemo iskoristiti taj model za kažnjavanje generatora adaptacija koji je znatno promijenio sadržaj podataka i time izračunati građijente težina generatora koji će smanjiti empirijski rizik modela f_s .

Gubitak semantičke konzistentnosti računamo za oba generatora $G_{s \rightarrow t}$, $G_{t \rightarrow s}$ i zbrajamо ta dva gubitka. Kod računanja gubitka semantičke konzistentnosti generatora $G_{t \rightarrow s}$ imamo problem jer nemamo označene podatke pa zbog toga koristimo predikcije modela f_s kao zamjenu za oznake. Takva metoda se naziva pseudooznačavanje [Lee (2013)] jer za označavanje neoznačenih primjera koristimo model f_s . Zbog efekta ko-vrijatnog pomaka i pogreška u predviđanju, te oznake imat će šum. Postupak koji smo opisali naziva se označavanje sa šumom (*engl. noisy labelling*).

Izraz za ukupni gubitak semantičke konzistentnosti iskazan je ovim izrazom:

$$L_{sem}(G_{s \rightarrow t}, G_{t \rightarrow s}, X_s, X_t, f_s) = L_{task}(f_s, G_{t \rightarrow s}(X_t), Y'_t) + L_{task}(f_s, G_{s \rightarrow t}(X_s), Y_s)$$

gdje su oznake sa šumom iskazane kao $Y'_t = argmax(f_s(X_t))$.

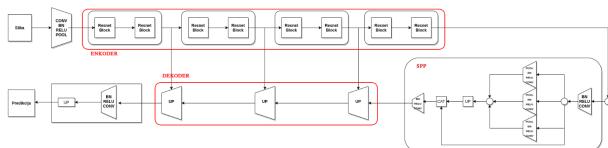
5. Opisi Modela

U ovom poglavlju slijede opisi modela koji su korišteni za segmentaciju kuća te za adaptaciju i translaciju podataka. Također predstavljamo implementaciju modela u programskom okviru PyTorch.

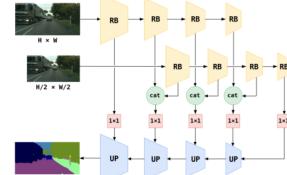
5.1. Model za segmentaciju

Koristimo arhitekturu za semantičku segmentaciju razvijenu na FER-u naziva SwiftNet [Orsic et al. (2019)]. SwiftNet je arhitektura namijenjena za semantičku segmentaciju prometnih slika koja je napravljena s ciljem predviđanja razreda objekta u stvarnom vremenu. Dizajn SwiftNeta je vođen s ciljem da se može izvoditi na mobilnim i ugradbenim uređajima ograničenih performansi. Arhitektura je predstavljena u dvije inačice, u prvoj inačici korištena je prostorno sažimajuća piramida (engl. spatial pyramid pooling-SPP), a u drugoj inačici korištena je metoda rezolucijskih piramida (engl. resolution pyramid). Cilj obje inačice je povećanje receptivnog polja modela što doprinosi boljoj detekciji velikih objekta na slici i širenju konteksta kojeg model vidi na slici. Tip arhitektura modela je koder-dekoder arhitektura s lateralnim vezama između dekodera i kodera. U odnosu na U-Net modele [Ronneberger et al. (2015)], dekoder je asimetričan u odnosu na koder. Koder kao izlaz daje sažetu i semantički bogatiju verziju ulazne slike, te zbog toga razloga pikseli manjih objekta imaju nižu vjerojatnost prepoznavanja na izlazu. Rješenje tog problema jest kombiniranje aktivacija pličih slojeva i dubljih slojeva. U SwiftNet implementaciji koder je izведен pomoću rezidualnih blokova (*engl. resnet*) [He et al. (2015)].

Arhitekture SwiftNeta sa SPP modulom (Single Scale SwiftNet) i piramidalnog SwiftNeta prikazane su na slikama 5.1 i 5.2.



Slika 5.1: Single Scale SwiftNet, crvenom bojom označeni su moduli kodera i dekodera. Između kodera i dekodera nalazi se SPP modul, a u kraju modela nalazi se slojevi za fino podešavanje modela na zadani skup podataka.



Slika 5.2: SwiftNet s 2-razinskom piramidom. K-razinska piramida interpolira slike na k različitim rezolucijama što rezultira povećanjem receptivnog polja aktivacija. Parametri kodera su dijeljeni. Značajke istih rezolucija se spajaju lateralnim vezama i kombiniraju sa semantički bogatijim značajkama.

5.2. Model za prevodenje podataka

Osnovna arhitektura CycleGAN [Zhu et al. (2017)] modela može se vidjeti na slici 4.3. Za diskriminatore preuzimamo PatchGAN arhitekturu, koja se spominje u [Isola et al. (2016)]. Za razliku od običnog diskriminatora koji klasificira cijelu sliku u kategorije *real/fake*, PatchGAN klasificira $N \times N$ adreske ulazne slike i te izlaze na kraju uprosjećuje. Za generator koristimo Resnet arhitekturu s 9 blokova. Autori CycleGANa [Zhu et al. (2017)] preuzimaju taj generator iz modela koji se koristi za super-rezoluciju [Johnson et al. (2016)].

Slijedi prikaz koda za učenje CycleGan modela:

```
1 netG_A, netG_B, netD_A, netD_B = build_models()
2 optimizer_G = torch.optim.Adam(itertools.chain(netG_A.parameters(),
       netG_B.parameters()), lr=0.0002, betas=(0.5, 0.999))
3 optimizer_D = torch.optim.Adam(itertools.chain(netD_A.parameters(),
       netD_B.parameters()), lr=0.0002, betas=(0.5, 0.999))
4 dataset = build_dataset()
5 cycle_lambda = 100
6 idt_lambda = 10
7 bce_logit_loss = nn.BCEWithLogitsLoss()
8 real_label, fake_label = 1.0, 0.0
9
10 for epoch in range(latest_epoch, train_opt["epochs"]):
11     model.train_state()
12     for i in range(len(dataset)):
13         real_A, real_B, labels_A = dataset[i]
```

```

14     real_A, real_B, labels_A = real_A.cuda(), real_B.cuda(),
15     labels_A.cuda()
16     ##### GENERATOR #####
17     # Forward pass
18     fake_B = netG_A(real_A)
19     fake_A = netG_B(real_B)
20     reconstructed_A = netG_B(fake_B)
21     reconstructed_B = netG_A(fake_A)
22     # Detach discriminator from the graph
23     set_requires_grad([netD_A, netD_B], requires_grad=False)
24     # Gan Loss
25     logits_da = netD_A(fake_B)
26     logots_db = netD_B(fake_A)
27     loss_G_A = bce_logit_loss(logits_da, fake_label)
28     loss_G_B = bce_logit_loss(logits_db, fake_label)
29     # Cycle Loss
30     loss_cycle_A = torch.abs(reconstructed_A - real_A)
31     loss_cycle_B = torch.abs(reconstructed_B - real_B)
32     cylce_loss = loss_cycle_A + loss_cycle_B
33     # Identity mapping Loss
34     loss_idt = torch.abs(netG_A(real_B) - real_B) + torch.abs(netG_B(
35     (real_A) - real_A)
36     # Total GAN Loss & optimization
37     loss_G = loss_G_A + loss_G_B + cycle_lambda*cylce_loss +
38     idt_lambda*loss_idt
39     loss_G.backward()
40     optimizer_G.step()
41     ##### DISCRIMINATOR #####
42     # Attach discriminator
43     set_requires_grad([netD_A, netD_B], requires_grad=True)
44     optimizer_D.zero_grad()
45     # Discriminator A
46     loss_da_real = bce_logit_loss(netD_A(real_A), real_label)
47     loss_da_fake = bce_logit_loss(netD_B(fake_A.detach()), fake_label)
48     loss_da = (loss_da_real + loss_da_fake)*0.5
49     loss_da.backward()
50     # Discriminator B
51     loss_db_real = bce_logit_loss(netD_B(real_B), real_label)
52     loss_db_fake = bce_logit_loss(netD_B(fake_B.detach()), fake_label)
53     loss_db = (loss_db_real + loss_db_fake)*0.5
54     loss_db.backward()

```

```
52     optimizer_D.step()
53     model.update_learning_rate()
```

5.3. Model za adaptaciju domene

Zbog toga što CycleGAN model [Zhu et al. (2017)] nije dizajniran za prepoznavanje i očuvanje sadržaja na snimkama potrebno je doraditi funkciju gubitka koja se optimira.

```
1 def semantic_consistency(self, model, fake_A, fake_B, real_A,
2     real_B, label):
3     noisy_labels = model(fake_B).argmax(1)
4     semantic_loss = self.ce_loss(fake_A, label) + self.ce_loss(
5         fake_B, noisy_labels)
6     return semantic_loss
7
8 loss_G = loss_G_A + loss_G_B + cylce_loss + loss_idt +
9     semantic_consistency(model, fake_A, fake_B, real_A, real_B,
10    labels)
```

U ukupni gubitak generatora dodajemo novi član - semantičku konzistentnost. Taj član koristi prethodno naučeni model f_s i oznake u izvornoj domeni kako bi kaznio odstupanje sadržaja sintetičkih podataka od oznaka i od predikcija modela f_s .

6. Sustav za semantičku segmentaciju zgrada - DeepSat

U sklopu ovog rada razvijen je programski sustav za olakšanu kontrolu nad procesom učenja dubokih modela. Sustav je nazvan DeepSat sustav. U članku [Sculley et al. (2015)] opisan je koncept *tehničkog duga* u području ML-a i argumentira se zašto je on izraženiji u ML sustavima nego u klasičnim programskim sustavima. Tehnički dug je koncept koji opisuje situaciju koja se dešava ne obraćanjem pažnje na kvalitetu koda, a on se isplaćuje redovitim prepravljanjem koda i poboljšavanjem njegove kvalitete. Bitno je naglasiti kako je kod za implementaciju modela tek mali dio u cijelokupnom ML sustavu, a kako većina koda u ML sustavu služi kao potpora i kao sustav zaslužan za kontrolu nad procesom.

Za razliku od većine projekata u programskom inženjerstvu gdje je potrebno pratiti samo kod projekta u ML projektima postoji 3 osi varijacije koje je potrebno istodobno pratiti, a to su podaci, model i potporni kod.

6.1. Implementacija infrastrukture za učenje

Prije provođenja eksperimenta napravili smo cijelokupnu implementaciju cjevovoda za učenje dubokih modela. Cjevovod obrade podataka sastoji se od nekoliko faza izvođenja koje se razlikuju prema njihovim funkcijama. Faze međusobno komuniciraju preko dijeljenog repozitorija. Svaka faza cjevovoda komunicira s dijeljenim repozitorijem iz kojeg uzima ulaze i u njega zapisuje izlaz i dodatne objekte. Izlaz faze predstavlja rezultat izračuna te faze i njega koriste sljedeće faze cjevovoda. Dodatni objekti faze su podaci čija je funkcija objašnjenje unutrašnjeg rada te faze i služe za stjecanje dodatnih informacija o samom procesu. Ponajprije služe za vizualnu inspekciju procesa, te su dobar način kako prepoznati ili objasniti nepredviđene okolnosti. Najvažnije karakteristike cjevovoda i zbog čega svaki ozbiljniji ML projekt mora imati kvalitetno napravljenu ML infrastrukturu za učenje modela su:

- **Reproducibilnost izvođenja cjevovoda.** Svaka konfiguracija ML cjevovoda preslikava se u jedinstven skup artefakta i izlaza. Ovo je bitno svojstvo koje nam služi kako bismo mogli međusobno usporediti eksperimente.
- **Ponovno iskorištanje faza.** Neke faze su dugotrajne. Zbog toga, faze pamte prijašnje konfiguracije nad kojima su pokrenute i izbjegavaju uzaludno računanje istih rezultata.
- **Praćenje karakteristika procesa.** ML projekti se razlikuju od običnih programskih projekata u tome što nas uz rezultate također zanima kako je sustav došao do tih rezultata i kakve su karakteristike unutrašnjosti procesa.
- **Jednostavnije uklanjanje grešaka.** Olakšano je pisanje *unit* testova ako su faze sustava odvojene jer se svaka faza može izolirati i za nju pripremiti posebna okolina.

Ponašanje cjevovoda konfigurira se na temelju jedne konfiguracijske datoteke. Zbog toga je moguće jednostavnije verzioniranje eksperimenta cjevovoda gdje se jednostavno povezuje pokretanje cjevovoda, njegovi rezultati i konfiguracijska datoteka. Ispod ovog paragrafa možemo vidjeti primjer konfiguracije faze za učenje. U toj konfiguraciji istaknute su najbitnije komponente koje se mogu mijenjati. Zbog jednostavnosti i čitkosti razvijen je poseban skriptni jezik koji omogućuje lakše uređivanje takvih datoteka.

Izrazi koji se nalaze između znakova \$ indeksiraju ostale vrijednosti unutar konfiguracijske datoteke, npr. izraz *batch_size*: *\$sharding.shard_size\$*, zapisuje u *batch_size* istu vrijednost koja je upisana u varijablu *shard_size* faze *sharding*.

Izrazi koji se nalaze između znakova % predstavljaju Python kod koji se dinamički izvršava u trenutku poziva funkcije za obradu takvog niza znakova. Recimo, nakon obrade dijela rječnika "metrics": ["\%accuracy%", "\%precision%", "\%recall%"] lista nizova znakova pretvorit će se u listu funkcija s istim imenom. Funkcija koja obrađuje takav niz unutra svog konteksta mora imati dostupne te objekte koji su unaprijed implementirani.

```

1 {
2   "trainer" : {
3     "active": false,
4     "amp": false,
5     "mixup_factor": -1,
6     "epochs": 100,
7     "device": "cpu",
8     "model": {

```

```

9      "src.models.pyramid_swiftnet.model.PiramidSwiftnet": {
10        "num_classes": 2
11      }
12    },
13    "dataloader": {
14      "train": {
15        "torch.utils.data.DataLoader": {
16          "dataset": "%train_db%",
17          "batch_size": "$sharding.shard_size$"
18        }
19      },
20      "valid": {
21        "torch.utils.data.DataLoader": {
22          "dataset": "%valid_db%",
23          "batch_size": "$sharding.shard_size$"
24        }
25      }
26    },
27    "loss_function": {
28      "src.losses.losses.Focal_Loss": {
29        "gamma": 0.5,
30        "type": "exp"
31      }
32    },
33    "hypertuner": {
34      "active": false,
35      "search_space": {
36        "lrl1": "%ray.tune.uniform(2e-4, 6e-4)%",
37        "lr2": "%ray.tune.uniform(1e-4, 4e-4)%",
38        "wd1": "%ray.tune.uniform(1e-4, 4e-4)%",
39        "wd2": "%ray.tune.uniform(1e-4, 4e-4)%",
40        "beta11": "%ray.tune.uniform(0.5, 1)%",
41        "beta12": "%ray.tune.uniform(0.5, 1)%",
42        "beta21": "%ray.tune.uniform(0.5, 1)%",
43        "beta22": "%ray.tune.uniform(0.5, 1)%"
44      },
45      "search_alg": "basic",
46      "resources_per_trial": {"cpu": 1},
47      "iterations": 1,
48      "num_samples": 1
49    }
50  }
51 }

```

Listing 6.1: Prikaz konfiguracije modula za učenje (*engl. trainer*). Osim standardnih PyTorch komponenti koje se koriste prilikom učenja modela, ova faza također sadrži i mogućnost optimizacije hiperparametra i mogućnost dodavanja komponenti promatrača (*engl. observers*).

Na slici 6.1 prikazana je organizacija projekta. Slijede opisi komponenta:

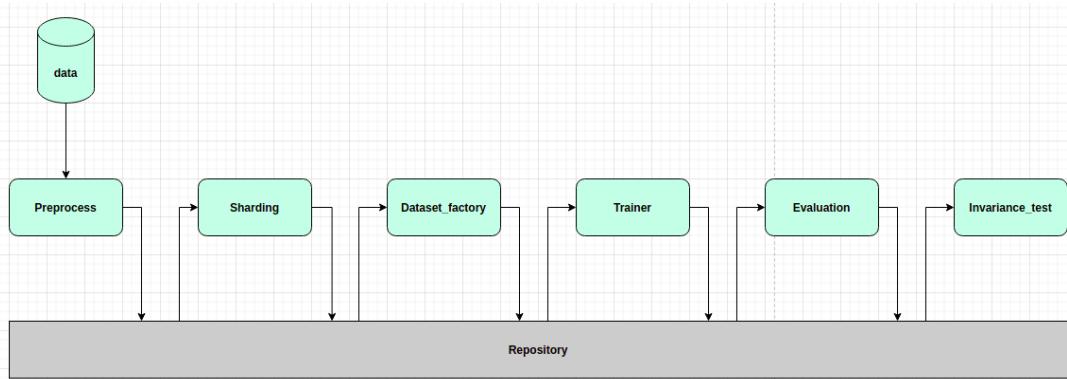
- **experiments.** Skripte za izvršavanje određenih eksperimenta.
- **pipeline.** Implementacije faza cjevodova.
- **runners.** Skripte za obradu konfiguracijske datoteke i pozivanje faza u cjevovodu.
- **src.** Biblioteke projekta.
- **tests.** Unit i integracijski testovi.
- **main.py.** Glavna skripta za pokretanje učenja.
- **makefile.** Skripta za inicijalizaciju projekta.
- **requirements.txt.** Popis stranih Python biblioteka.
- **runner.bash.** Pokretanje učenja.

dominikstipic	chore(main): printed dataset path	...	on Dec 4, 2021	192
experiments	fix(trainer): fixed problems with gpu and it's ...		2 months ago	
pipeline	feat(trainer): active flag in order to prevent c...		2 months ago	
runners	feat(trainer): active flag in order to prevent c...		2 months ago	
scripts	fix/scripts): {device}_weights.pt to weights.p...		3 months ago	
src	feat(losses): added focal loss		2 months ago	
tests	test(resources and test_infra): added test re...		3 months ago	
.gitignore	fix(augmentation experiment): save results t...		3 months ago	
LICENSE	Initial commit		4 months ago	
README.md	chore(README): updated readme file		3 months ago	
main.py	chore(main): printed dataset path		last month	
makefile	test(overfit): Completed test for checking if ...		3 months ago	
requirements.txt	chore(requirements.txt): added gutil depen...		2 months ago	
runner.bash	feat(main runner): implemened system runn...		4 months ago	

Slika 6.1: Organizacija DeepSat projekta. U početku je potrebno inicijalizirati projekt upisivanjem `make` naredbe. Ta naredba ugrađuje sve potrebne strane pakete. Podaci se postavljaju u `data` direktorij i projekt se u konačnici pokreće naredbom `python main.py` nakon čega se počinju izvršavati sve faze definirane u konfiguracijskoj datoteci `config.json`.

Na slici 6.2 može se vidjeti komunikacijski dijagram faza cjevovoda preko dijeljenog repozitorija. Zbog dijeljenog repozitorija olakšana je komunikacija između faza i puno je lakši pregled operacija koje se izvode unutar procesa. Slijedi kratak opis faza cjevovoda prikazanih na slici 6.2:

- **Preprocess.** Faza u kojoj se obrađuju originalne snimke pomoću odabranih transformacija. U okviru ovog rada u toj fazi se izvršava rezanje originalnih 5000x5000 snimka u mnogo 600x600 isječka na kojima model uči.
- **Sharding.** Grupiranje grupe isječaka u tar arhive. Svrha ove faze je serijalizacija podataka u tar arhive kako bi stvaranje mini-grupa (*engl. batches*) tijekom učenja bilo mnogo brže. Zbog toga što su primjeri serijalizirani, pristup tvrdom disku mnogo je brži nego u slučaju kada su primjeri nasumično raspoređeni po tvrdom disku.
- **Dataset factory.** Stvaranje PyTorch skupa podataka, dijeljenje podataka na podskup za treniranje, testiranje i validiranje te definiranje augmentacija za svaki podskup.
- **Trainer.** Konfiguracija procesa za učenje i njihovih komponenti.
- **Evaluation.** Evaluacija naučenog modela na podskupu za testiranje.
- **Invariance test.** Testovi invarijantnosti na odabrane transformacije.



Slika 6.2: Komunikacijski dijagram faza cjevovoda. Komponente cjevovoda komuniciraju preko zajedničkog sučelja. Zbog toga je olakšano dodavanje novih komponenti koji jedino trebaju znati koristiti protokol sučelja, a ne moraju se zamarati implementacijskim detaljima komponenti s kojima direktno komuniciraju. Komponente u zajednički repozitoriju upisuju artefakte, tj. objekte koje su namijenjene za pregledavanje i izlaze koje koriste druge komponente.

6.2. Skup podataka

Za eksperimente odabran je skup podataka nastao u sklopu natjecanja Inria Aerial Image Labeling Dataset [Maggiori et al. (2017)]]. Zadatak tog natjecanja je stvoriti modele koji će uspješno provoditi semantičku segmentaciju kuća pomoću satelitskih snimaka. Autori zadatka prikupili su slike iz različitih područja na Zemlji kako bi pružili modelu što raznovrsnije podatke. Snimke su prikupljene s prostora ovih gradova: američki gradovi Austin, Kitsap i Chicago te austrijski gradovi Beč i Tirol. Slike su visoke kvalitete, rezolucije 5000 x 5000 piksela, posjeduju tri kanala, zapisane su u tif formatu i ukupno zauzimaju 31.5 GB memorije. Sažetak skupa podataka prikazan je u tablici 6.4.

Train	Tiles*	Total area	Test	Tiles*	Total area
Austin, TX	36	81 km ²	Bellingham, WA	36	81 km ²
Chicago, IL	36	81 km ²	San Francisco, CA	36	81 km ²
Kitsap County, WA	36	81 km ²	Bloomington, IN	36	81 km ²
Vienna, Austria	36	81 km ²	Innsbruck, Austria	36	81 km ²
West Tyrol, Austria	36	81 km ²	East Tyrol, Austria	36	81 km ²
Total	180	405 km ²	Total	180	405 km ²

Slika 6.3: Sažetak Inria skupa podataka. Slikana su područja 10 gradova s različitim stupnjem urbanizacije [Maggiori et al. (2017)]. Korisnicima su dostupne samo oznake: Austina, Chica-goa, Kitsapa, Beča i Tirola. Oznake ostalih gradova su skrivene i namijenjene za evaluaciju na istoimenom natjecanju.



Slika 6.4: Primjeri snimka svakog grada iz Inria skupa podataka. Gradovi su redom: Austin, Chicago, Kitsap, Tirol i Beč.

6.3. Eksperimenti i rezultati

U ovom dijelu diplomskog rada prikazani su rezultati različitih eksperimenta. Prvi eksperimenti odnose se na eksperimente s različitim vrstama augmentacija s ciljem uspostavljanja uvjeta invarijantnosti na određene transformacije. Nakon toga mjerimo utjecaj promjene domene podataka na performanse modela i uočavamo efekt kovarijatnog pomaka. Nakon toga eksperimentiramo s tehnikama adaptacije domene kako bi umanjili efekt kovarijatnog pomaka između dviju različitih domena. Domene s kojima eksperimentiramo su snimke gradova Tirola i Austina koji se mogu vidjeti na slici 6.5. Eksperimenti su napravljeni uz pomoć NVIDIA GTX 1070 grafičke kartice i Google Colab Pro platforme. Učenje modela semantičke segmentacije je na jednom gradu otprilike trajalo desetak sati, dok je učenje generatora za kreiranje adaptacija kroz 60 epoha trajalo otprilike 3 dana. U tablici 6.1 možemo vidjeti najbolje rezultate na gradu Austinu, a na slici 6.6 kretanje funkcije gubitka tijekom učenja i konačne predikcije modela na skupu za testiranje.



Slika 6.5: Usporedba gradova Austina i Tyrola. Tyrol je zeleniji i ruralniji grad s drugačijom distribucijom piksela. Kovarijatni pomak se dešava ako model učimo na slikama Austina, a testiramo na Tyrolu.

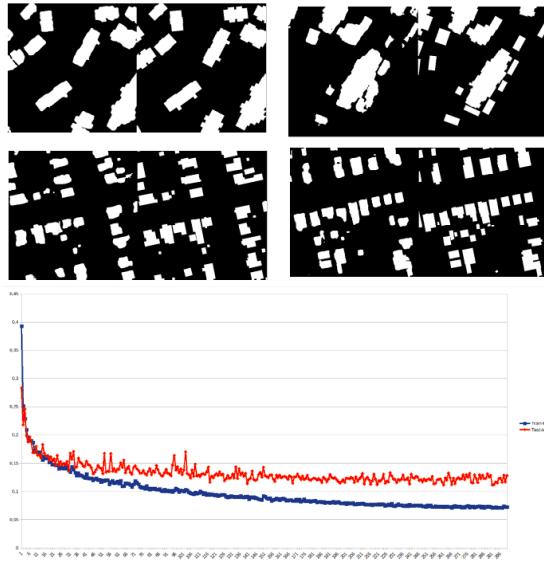
Accuracy	96,4%
Precision	90,7%
Recall	85,65%
mIoU	78,75%

Tablica 6.1: Najbolji rezultati naučenog modela na gradu Austinu. Prema službenoj tablici Inria projekta [Inria (2016)] najbolji rezultat ima $mIoU$ jednak 81.91% i točnost jednaku 97.41%

6.3.1. Eksperimenti s augmentacijama

Svrha ovih eksperimenta je pomoći augmentacije podataka razbiti lažne kauzalne veze koje je model naučio. Lažne kauzalne veze su uzročno-posljedične veze koje nisu istinite, tj. ne vrijede u svim uvjetima. Duboki modeli obično uče na skupovima visoko dimenzionalnih podataka, poput slika, koji u sebi mogu sadržavati brojne neistinite asocijacije (*engl. spurious correlations*). Jedan od načina kako se utjecaj tih podatkovnih asocijacija može smanjiti ili u potpunosti razbiti je intervencija ili korištenje augmentacija. Naravno, takva tehnika suzbijanja neistinith intervencija ograničena je metoda jer ovisi o procjeni, ali je vrlo dobra za postavljanje očitih uvjeta invarijantnosti.

Mjerimo *cijenu augmentacije* *aug* na skupu za testiranje. Tu cijenu mjerimo tako da usporedimo razliku između performansi na skupu za testiranje i augmentiranog skupa



Slika 6.6: Na gornjoj slici mogu se vidjeti 4 primjera parova predikcija i oznaka na testnim podacima. Na donoj slici je prikazana funkcija gubitka tijekom 300 epoha. Crvenom krivuljom prikazano je kretanje gubitka na skupu za validiranje, a s plavom kretanje gubitka na skupu za učenje. Nakon nekog vremena dolazi do zasićenja procesa.

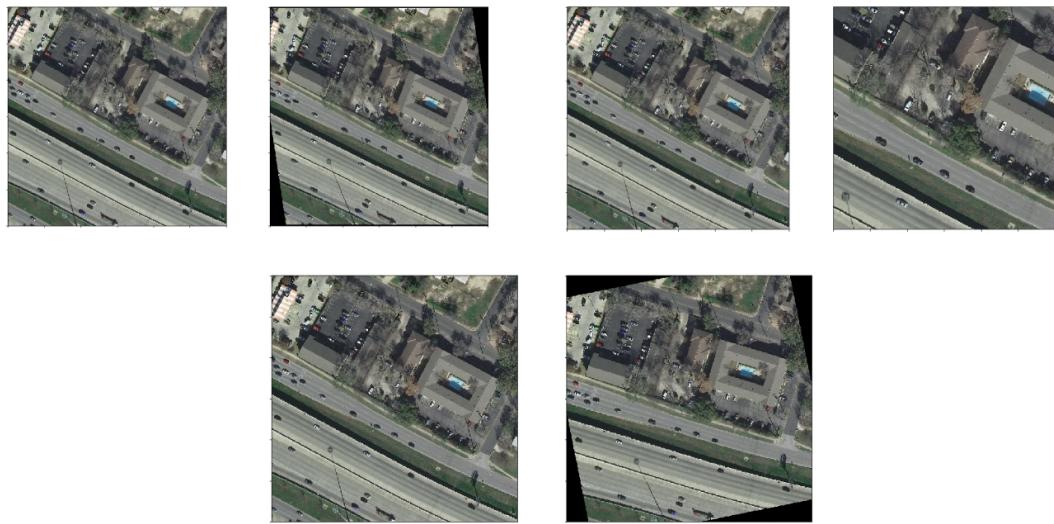
za testiranje, tj. koristimo formulu:

$$cost(model, test) = |m(model, test) - m(model, aug(test))| \quad (6.1)$$

U gornjoj formuli oznaka m označava odabranu mjeru performanse poput točnosti (*engl. accuracy*) ili usrednjjenog presjeka nad unijom (*engl. mean intersection over union ili mIoU*)... Veliko odstupanje cijene za neku vrstu augmentacije govori nam da model nije invarijantan na odabranu transformaciju i da bi podatke za učenje trebalo augmentirati s tom transformacijom kako bi se uklonile neistinite asocijacije u podacima.

U ovom eksperimentu mjerimo cijene nekoliko augmentacija na način da ih primjenjujemo na skupu za testiranje. Transformacije ili augmentacije s kojima radimo eksperimente su: skaliranje, rotacije i posmik. Povećavamo parametre augmentacija te tako pojačavamo jačinu tih augmentacija. Podaci navedenih eksperimenta mogu se naći u tablici 6.2, a na slici 6.7 se mogu vidjeti rezultati primijenjenih transformacija. Iz navedenih tablica može se vidjeti kako se povećanjem utjecaja augmentacija povećava njihova cijena i da je prije svakog treninga jako bitno kvalitetno augmentirati podatke kako bismo dobili što invarijantnije značajke. Korištenjem rotacija za augmentaciju cijena pada na 0.38% za kuteve između -180 i 180 te smo tako učinili model invarijantnim na rotacije. Nakon toga smo naučili semantički model s nasumičnim

rotacijama te dobivamo povećanje mIoU-a od 3% na skupu za testiranje na Austinu.



Slika 6.7: Primjeri transformacija koje koristimo u augmentacijama. Prikazane su transformacije posmika, skaliranja i rotacije.

Skaliranje	Cijena	Rotacija	Cijena	Posmik	Cijena
[1,1]	0,16%	[0,0]	0,00%	[0,0]	0,16%
[0,7, 1,2]	0,53%	[-10,10]	1,65%	[-2,2]	0,26%
[0,7, 1,4]	0,90%	[-30,30]	4,19%	[-4,4]	0,34%
[0,7, 1,8]	1,60%	[-90,90]	7,40%	[-8,8]	0,44%
[0,7, 2]	2,32%	[-120,120]	10,07%	[-16,16]	0,62%
[0,7, 3]	8,86%	[-180,180]	12,70%		

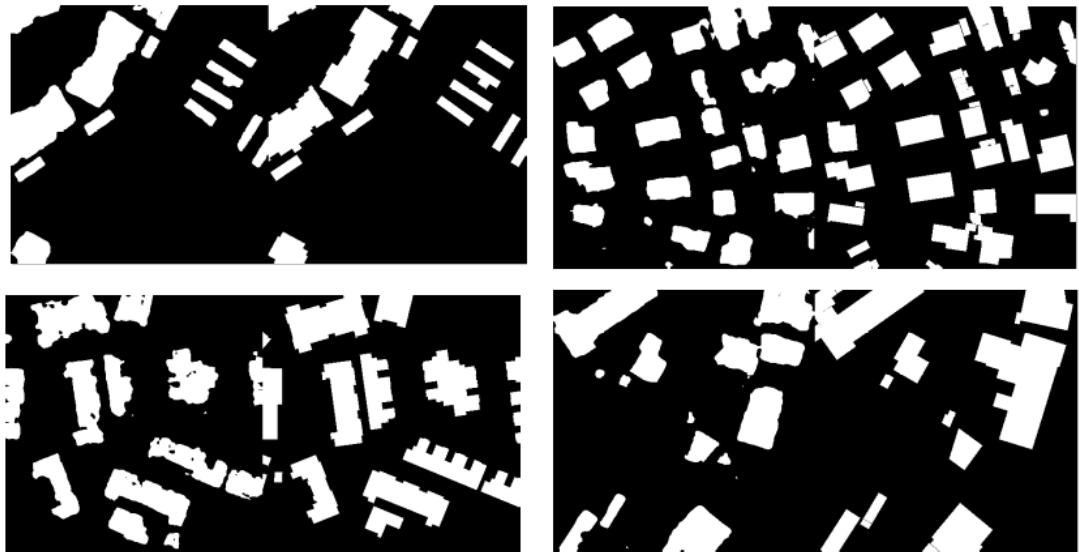
Tablica 6.2: Mjerenja invarijantnosti modela na transformacije rotacije, skaliranja i posmika. Koliko je model otporan na određenu transformaciju mjerimo uz pomoć cijene augmentacije definirane formulom 6.1. U prvoj tablici nalaze se podaci cijena augmentacija za transformaciju skaliranja, u drugoj tablici za rotaciju, a u trećoj tablici za transformaciju posmicanja.

6.3.2. Kovarijatni pomak

Evaluacija modela na drugim domenama gotovo uvjek rezultira lošijim rezultatima nego na onim domenama na kojima je model učio. U ovom eksperimentu mjerimo performansu modela na različitim domenama. Iz takvih mjerjenja možemo uočiti smanjenje performansi ako je model učio na stranim domenama. Rezultati se mogu vidjeti u tablici 6.3, a primjeri predikcija modela za sve kombinacije modela i podataka mogu se vidjeti na slici 6.9. Pad performansi je veoma izražen i predstavlja bi problem u praksi. Za model koji je učio na snimkama grada Austina performanse padaju za 17.33%, a za model koji je naučen na snimkama grada Tirola performanse padaju za velikih 22.82% pri promjeni domene kod evaluacije modela.

Evaluacija		
Učenje	Austin	Tyrol
Austin	77.56%	60.23%
Tyrol	52.23%	75.05%

Tablica 6.3: Rezultati (mIoU) na kombinacijama podataka na kojima je model učio i na kojima je model testiran. Testiranje modela na drugačijim domenama nego li na onima na kojima je model naučen rezultira lošijom performansom zbog efekta kovarijatnog pomaka.



Slika 6.8: Primjeri nasumično odabranih predikcija modela na različitim domenama. Na lijevoj strani slike nalaze se predikcije, a na desnoj strani oznake kuća. Krenuvši od prvog stupca i retka ka drugom stupcu i retku u smjeru kazaljke na sat slike predstavljaju: model učen na **Austinu** i testiran na **Austinu**, model učen na **Austinu** i testiran na **Tirolu**, model učen na **Tirolu** i testiran na **Austinu** i konačno model učen na **Tirolu** i testiran na **Tirolu**.

6.3.3. Ograničenja metode i primjeri rezultata

Adaptacija i translacija domene. U ovoj inačici eksperimentiramo s tehnikama prevođenja i adaptacijama domene. Za prevođenje podataka iz jedne domene u drugu korišten je CycleGAN model, a za metodu adaptacije razvijen je model inspiriran CyCada modelom [Hoffman et al. (2017)]. Originalni CyCada model je vremenski složen model pa je zbog toga razvijena pojednostavljena verzija. CycleGAN model za prevođenje podataka ne razlikuje piksele objekta od piksela okoline, što je problematično jer se tijekom prevođenja gubi sadržaj zgrada. Zbog toga testiramo rezultate na translacijama čiji su pikseli sadržaja dopunjeni s pikselima iz originalne snimke. Također testiramo dvije verzije CyCada modela, jedan koji je naučen kroz 30 epoha i onaj koji je naučen kroz 60 epoha. Na slici 6.9 se redom mogu vidjeti originalne slike, translatirane slike, translatirane slike s dopunom i adaptirane slike. Kvantitativni rezultati mogu se vidjeti unutar tablice 6.4.

Podaci za učenje	mIoU (Testiranje)	mIoU (Tyrol)	mIoU (Austin)
Austin	-	60,23%	77,56%
CycleGAN	62,86%	34,06%	60,98%
CycleGAN + dopuna	94,70%	14,27%	21,83%
CyCada_30	62,00%	56,53%	58,35%
CyCada_60	59,7%	56,51%	56,93%

Tablica 6.4: mIoU rezultati za modele koji su učeni na različitim inačicama podataka. Snimke grada Tirola predstavljaju ciljnu domenu, a snimke grada Austina izvornu domenu za metode adaptacije i prevođenja domene. Podaci na kojima učimo su dobiveni pomoću CycleGAN modela, CycleGAN modela gdje je sadržaj podataka dopunjeno, CyCada modela učenog 30 epoha i CyCada modela učenog 60 epoha. Iz rezultata na izvandistribucijskim podacima (Tyrol) vidi se da predložene metode nisu uspjеле smanjiti efekt kovarijatnog pomaka, ali može se vidjeti kako je CyCada ipak uspjela zadržati dio sadržaja zgrada jer su rezultati na izvandistribucijskim podacima mnogo bolji u usporedbi s CycleGAN podacima.



Slika 6.9: Primjeri dobivenih adaptacija i njihova usporedba. Slike grada Austina predstavljaju ishodišnu domenu, dok slike grada Tirola ciljnu domenu. S lijeva na desno prikazana je originalna slika, CycleGAN slika, CycleGAN slika s dopunama i CyCada slika. CycleGAN slika s dopunama je nastala na način da smo piksele sadržaja iz CycleGAN slike dopunili s pikselima iz originalne slike. Poneke slike su zrcaljene jer je kao augmentacija korištena nasumična horizontalna rotacija.

7. Zaključak

U ovom radu ispitujemo i razvijamo prototip sustava za detekciju kuća iz satelitskih snimka pomoću modela semantičke segmentacije. Opisujemo razvijenu arhitekturu sustava za učenje dubokih modela i naglašavamo dobre prakse pri izradi takvog sustava. Nakon toga, opisujemo problem generalizacije dubokih modela te iskorištavamo tehniku adaptacije podataka na druge domene kako bismo povećali performanse modela semantičke segmentacije na izvandistribucijskim podacima. Također, eksperimentiramo s tehnikama augmentacije podataka kako bi umanjili efekt kovarijatnog pomaka. Rad bi mogao biti interesantan svakome tko želi napraviti prediktivne sistave temeljene na dubokom učenju i tko nastoji izbjegći efekt kovarijatnog pomaka na izvandistribucijskim podacima.

Adaptacija podataka iz jedne domene u drugu daje realistične primjere koji imaju potencijala za uklanjanje kovarijatnog pomaka na izvandistribucijskim podacima. Primjeri su naizgled prave fotografije i ne odaju dojam da su umjetno generirane. Naučili smo da šira generalizacija modela najviše ovisi o podacima na kojima model uči i da je uspostava invarijantnih značajki u različitim domenama i uz različite transformacije ključno za izradu kvalitetnih i pouzdanih ML sustava. Eksperimentima smo pokazali kako kvalitetna augmentacija podataka smanjuje efekt kovarijatnog pomaka, tj. pomaže nam da naučimo invarijantne značajke koje će biti stabilne u različitim domenama. Također smo primijenili metodu adaptacije podataka na izvandistribucijske podatke. Ta metoda daje realistične sintetičke podatke koji ne odaju dojam da su ti podaci sintetički. Tijekom procesa adaptacije podatka dešava se situacija da se sadržaj zgrada u nekim slikama gubi ili slabi. Zbog tog problema nismo u potpunosti ublažili efekt kovarijatnog pomaka na zadovoljavajuću mjeru.

Za budući rad voljeli bi podesiti model za izradu adaptacija i smanjiti nestajanje ključnih piksela kako bismo spriječili gubitak sadržaja sa slikama. Također je potrebno proučiti i eksperimentirati s drugaćijim gubitcima na zadatku semantičke segmentacije kako bi se spriječile zrnate predikcije zgrada.

LITERATURA

Petra Bevandic, Marin Orsic, Ivan Grubisic, Josip Saric, i Sinisa Segvic. Multi-domain semantic segmentation with overlapping labels. *CoRR*, abs/2108.11224, 2021. URL <https://arxiv.org/abs/2108.11224>.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, i Yoshua Bengio. Generative adversarial networks, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. *CVPR 2015*, abs/1512.03385, 2015. URL <http://arxiv.org/abs/1512.03385>.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, i Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation, 2017.

Maximilian Ilse, Jakub M. Tomczak, i Patrick Forré. Selecting data augmentation for simulating interventions, 2020.

Inria. Leaderboard, 2016. URL <https://project.inria.fr/aerialimagelabeling/leaderboard/>.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, i Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL <http://arxiv.org/abs/1611.07004>.

Justin Johnson, Alexandre Alahi, i Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. URL <http://arxiv.org/abs/1603.08155>.

Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. 2013.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, i Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. U *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017.

Mehdi Mirza i Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. URL <http://arxiv.org/abs/1411.1784>.

Marin Orsic, Ivan Kreso, Petra Bevandic, i Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. *CoRR*, abs/1903.08469, 2019. URL <http://arxiv.org/abs/1903.08469>.

Judea Pearl. Statistics and causal inference: A review, 2003.

Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL <http://arxiv.org/abs/1505.04597>.

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, i Dan Dennison. Hidden technical debt in machine learning systems. U C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, i R. Garnett, urednici, *Advances in Neural Information Processing Systems*, svezak 28. Curran Associates, Inc., 2015. URL <https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcacf2674f757a2463eba-Paper.pdf>.

V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999. doi: 10.1109/72.788640.

Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, i Michael Plöd. Ml-ops.org, Dec 2021. URL <https://ml-ops.org/content/mlops-principles>.

Mei Wang i Weihong Deng. Deep visual domain adaptation: A survey. *CoRR*, abs/1802.03601, 2018. URL <http://arxiv.org/abs/1802.03601>.

Garrett Wilson i Diane J. Cook. Adversarial transfer learning. *CoRR*, abs/1812.02849, 2018. URL <http://arxiv.org/abs/1812.02849>.

Jun-Yan Zhu, Taesung Park, Phillip Isola, i Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. U *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Semantička segmentacija zgrada u satelitskim snimkama

Sažetak

Problem s kojom se ovaj rad bavi je izrada sustava za semantičku segmentaciju zgrada iz satelitskih snimki koji ima sposobnost izvandistribucijske generalizacije. Skup podataka na kojem radimo su satelitske slike visoke rezolucije proizašle iz natjecanja za segmentaciju slika- "Inria Aerial Image Labeling Dataset". Model semantičke segmentacije koji koristimo temelji se na metodi rezolucijskih piramida. Dok za poboljšanje generalizacije na izvandistribucijske podatke koristimo metodu adaptacije i prevođenja domene. Model za adaptaciju podataka temelji se na dorađenoj CycleGAN arhitekturi koja je bliska CyCADA arhitekturi. U početku razvijamo programsku potporu za učenje dubokih modela temeljenu na MLOps načelima kako bi imali što kvalitetniju kontrolu nad procesom učenja i pomoću nje radimo eksperimente. Istraživački dio ovog rada odnosi se na razvoj i usporedbu metoda za izvandistribucijsku generalizaciju. Izvandistribucijski podaci su zapravo slike gradova čiji se krajobraz i arhitektura razlikuju od gradova na kojima smo učili. U eksperimentima na početku mjerimo kovarijatni pomak između različitih domena. Potom pokušavamo smanjiti kovarijatni pomak upotreborazličitih augmentacija kojima je cilj razbiti lažne asocijacije u podacima. Nakon toga, koristimo metodu za prevođenje podataka između dviju domena i adaptaciju domene kako bismo povećali performanse modela na izvandistribucijskim podacima. Glavni doprinos ovog rada je izrada ML sustava za kontrolu nad procesom učenja i prikaz eksperimenta s razvijenim modelom za adaptaciju domene.

Ključne riječi: Semantička segmentacija, satelitske snimke, adaptacija domene, prevođenje domene, izvandistribucijska generalizacija

Semantic segmentation of buildings in satellite images

Abstract

The problem that this thesis deals with is the development of a system for semantic segmentation of buildings from satellite images that has the ability of out-of-distribution generalization. The dataset we are working on is high-resolution satellite images that came from the Inria Aerial Image Labeling Dataset competition. The semantic segmentation model which we use is based on the resolution pyramid method. For the improvement of out-of-distribution generalization, we use the domain translation and domain adaptation method. The domain adaptation model is based on a refined CycleGAN architecture that is similar to the CyCADA architecture. Initially, we develop software support for learning deep models based on MLOps principles to have the best possible control over the learning process, and then we use it to conduct further experiments. The research part of this paper deals with the development and comparison of methods for out-of-distribution generalization. Out-of-distribution data are images of cities whose landscape and architecture differ from the cities where the model was initially trained. In the experiments, we initially measure the covariate shift between different domains. We then try to reduce the covariate shift by using various augmentations aimed at breaking spurious correlations in the data. After that, we use the domain translation method and domain adaptation method to increase the performance of the model on out-of-distribution datasets. The main contribution of this work is the creation of the end-to-end ML pipeline for automatically learning deep models and for process control. Also, we show the results of the experiments with the proposed model for domain adaptation on out-of-distribution domains.

Keywords: Semantic segmentation, satellite imagery, domain adaptation, domain translation, out-of-distribution generalization