

Pretvorba satelitske snimke u Google kartu

Marko Domagoj Benković
*Prirodoslovno-matematički
fakultet u Zagrebu*

Helena Marciuš
*Prirodoslovno-matematički
fakultet u Zagrebu*

Mihovil Stručić
*Prirodoslovno-matematički
fakultet u Zagrebu*

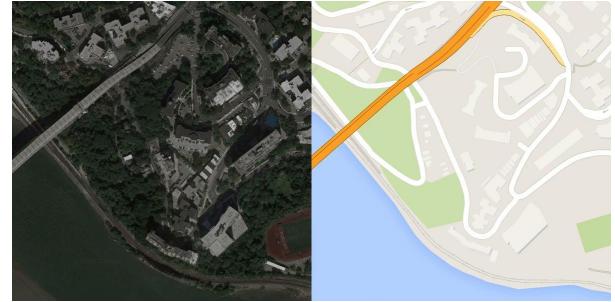
Sažetak—U ovom radu bavimo se problemom generiranja Google karata iz satelitskih snimaka pomoću *GAN* neuronskih mreža. Opisujemo podatke i njihovu pripremu za proces treniranja modela. Također, koristimo tri tipa modela: *cGAN*, *pix2pix* i *cycleGAN* te za svaki od njih opisujemo pripadnu arhitekturu mreže. Na kraju navodimo rezultate modela, međusobno ih uspoređujemo te navodimo moguća poboljšanja za buduće istraživanje ovog problema. Također, većina slika u ovom izvještaju ima prevelike dimenzije pa ih radi preglednosti navodimo zasebno na kraju, nakon literature.

I. UVOD

U današnje doba, digitalne karte koriste se svakodnevno, prvenstveno u svrhu navigacije. Koriste ih vozači taxija i autobusa, dostavljačke službe, dostavljači hrane, vojska, vozila bez vozača (npr. bespilotne letjelice) te pojedinci za navigaciju po gradu ili u automobilu. Zbog česte upotrebe karata, bitno je da su karte kvalitetne i detaljne, odnosno da čim bolje opisuju stvarni prostor. Međutim, izrada karata može biti vrlo skupa i cijena raste proporcionalno s obzirom na kvalitetu i detaljnost karte. Zbog toga se žele razviti načini koji omogućuju jeftiniju izradu karata, a da opet te karte budu što kvalitetnije. Jedan od tih načina koristi metode strojnog učenja, točnije neuronske mreže. U ovom radu rješavamo problem pretvaranja satelitske snimke u Google karte koristeći tri različita *GAN* modela - *cGAN*, *pix2pix* te *cycleGAN*.

II. OPIS PROBLEMA

Image-to-image translation je proces transformacije slike iz jedne domene u drugu, gdje je cilj naučiti mapiranje između ulazne slike jednog modaliteta u izlaznu sliku drugog modaliteta. Taj proces koristimo i za rješavanje našeg problema. Koristili smo *dataset* ([2]) koji je podijeljen na dva dijela - dio za treniranje koji sadrži 1096 parova satelitskih snimaka i pripadnih mapa te dio za validaciju koji sadrži 1098 parova slika. Sve satelitske snimke prikazuju dijelove New Yorka. Na svakoj slici, lijeva polovica slike prikazuje satelitsku snimku, a desna polovica prikazuje Google kartu.



Slika 1. Slika iz *dataset-a*

Slike su dimenzija 1200×600 piksela. Na slici 1 vidimo jednu sliku iz *dataset-a*. Napomenimo još da su slike dio većeg *dataset-a* s Kaggle-a zvanog *pixtopix* koji se može pronaći na [1].

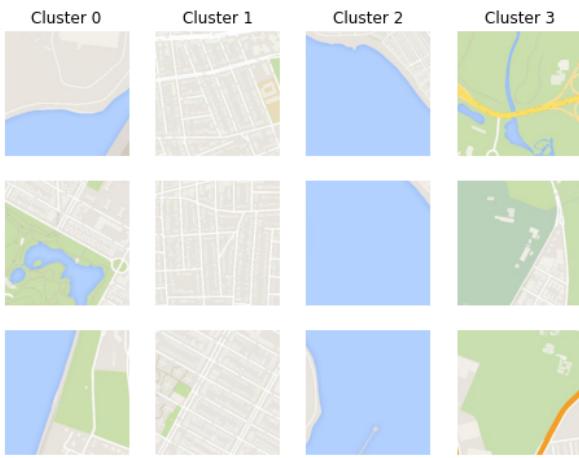
Slike smo prvo razdvojili tako da dobijemo posebno sliku satelitske snimke i posebno sliku Google karte koje su bile veličine 600×600 piksela. Svaku smo sliku umanjili da bude veličine 256×256 piksela.

A. Klasteriranje

Da bismo analizirali *dataset*, napravili smo klasteriranje *dataset-a* na 4 klastera koristeći kmeans++ algoritam. Odlučili smo se za 4 klastera jer slike koje prikazuju Google karte odokativno možemo podijeliti na 4 tipa: zgrade, vode, zelene površine te mješavina prijašnja 3 oblika. Na slici 2 vide se centroidi klastera sa veličinama pripadnog klastera, a na slici 3 vidimo po nekoliko primjera iz svakog klastera. Primijetili smo da je broj slika u klasteru sa zgradama puno veći od ostalih. Da bismo smanjili razliku, setu za treniranje dodali smo 211 slika iz seta za validaciju koje prikazuju vodene i zelene površine i time dobili 1307 parova. Da bismo proširili *dataset*, svaku smo sliku zarotirali za 180° i time dobili 2614 parova slika. Prije treniranja, sve satelitske snimke smo normalizirali, odnosno sve RGB vrijednosti



Slika 2. Centroidi klastera i pripadne veličine



Slika 3. Primjerci po klasterima

sveli na interval $[-1, 1]$.

Svi modeli implementirani su pomoću Pythonovih biblioteka *Tensorflow* i *Keras* te *Kerasovog Functional API*-ja koji olakšava izradu modela koji nisu tipa *Sequential*. Za treniranje modela korišten je *Google Colab* kako bismo ubrzali proces.

III. METODE I PRISTUP RJEŠAVANJU PROBLEMA

Za implementaciju *GAN* neuronske mreže koja pretvara satelitsku snimku u Google mapu, odlučili smo koristiti 3 modela.

A. *cGAN* model

Prvi model je tzv. *cGAN* model, odnosno uvjetna generativna suparnička mreža. *cGAN* se satoji od dvije neuronske mreže - generatora i diskriminatora. Za razliku od običnog *GAN*-a gdje generator samo generira slike koje nalikuju slikama iz skupa za treniranje, generator *cGAN*-a na ulazu dobiva i uvjet po kojem generira sliku. U našem slučaju, taj je uvjet bila satelitska snimka po kojoj je generator morao generirati Google kartu. Ulaz generatora je tenzor (satelitska

snimka) dimenzija $256 \times 256 \times 3$. Kao aktivacijska funkcija korištena je funkcija tangens hiperbolni. Izlaz generatora je tenzor (Google karta) dimenzija $256 \times 256 \times 3$. Generator je treniran u sklopu cijele *cGAN* mreže. Pripadni optimizator je Adam s parametrima *learning rate* = 0.0002 i *beta₁* = 0.5. Funkcija gubitka je kombinacija dvije funkcije gubitka - binarna *cross-entropy* funkcija te *MAE (mean absolute error)* funkcija gubitka s težinama 1: 100 u korist funkcije *MAE*. Arhitektura generatora prikazana je na slici 4.

Diskriminatoreva uloga je prepoznavanje je li slika dobivena na ulazu dio *dataset-a* ili je generirana od strane generatora. Za razliku od običnog *GAN*-a gdje diskriminatorev ulaz dobiva samo sliku za prepoznavanje, kod *cGAN*-a diskriminatorev ulaz dobiva uvjet - u našem je slučaju uvjet satelitska snimka. Dakle, diskriminatorev ulaz dobiva satelitsku snimku i Google kartu te mora odlučiti je li Google karta generirana ili je dio *dataset-a*. Ulaz diskriminatoreva su 2 tenzora (satelitska snimka i Google karta) dimenzija $256 \times 256 \times 3$ koji prolaze kroz blokove konvolucije s *kernelom* 4×4 i *strideom* 2×2 . Kao aktivacijska funkcija korištena je sigmoidna funkcija. Izlaz diskriminatoreva je matrica dimenzija 8×8 vrijednosti 0 ili 1. Funkcija gubitka diskriminatoreva je binarna *cross-entropy* funkcija. Pripadni optimizator je Adam s parametrima *learning rate* = 0.0002 i *beta₁* = 0.5. Tijekom treniranja, u svakoj iteraciji, diskriminatorev ulaz dobiva satelitsku snimku i Google kartu iz *dataset-a*, odnosno kartu sa labelom *real*. Nakon toga, diskriminatorev ulaz dobiva istu satelitsku snimku, ali ovaj put s kartom s labelom *fake*, odnosno kartom generiranom generatorom. Arhitektura diskriminatoreva prikazana je na slici 5.

B. *Pix2pix* model

Drugi model je *pix2pix* model, koji je zapravo specijalan slučaj uvjetnog *cGAN*-a. Kompozitna arhitektura se i dalje sastoji od generatora i diskriminatoreva, ali ovdje generator na ulazu prima satelitsku snimku koju pretvara u pripadnu (umjetnu/lažnu) Google mapu koja odgovara satelitskoj snimci na ulazu. Zatim se dobivena mapa konatenira sa stvarnom satelitskom snimkom na ulazu. Time dobivamo prvu polovicu ulaza za diskriminatore, s lažnom (*fake*) labelom. Druga polovica ulaza je "uređeni par" ili, konkretnije, konatenacija stvarne satelitske snimke na ulazu i pripadne stvarne Google mape, s istinitosnom (*real*) labelom. Labeli *real* i *fake* su potrebni za treniranje diskriminatoreva, kako bi mogao naučiti razliku između stvarnog i lažnog ulaza. Dakle, ukoliko imamo primjerice *batch* veličine 16 na ulazu diskriminatoreva, pola *batch-a* su spomenute konatenacije s

labelom *real*, a pola s labelom *fake*. Iz uloge diskriminatora vidi se da je on zapravo binarni klasifikator. U tu svrhu funkcija gubitka diskriminatora je tzv. *binary cross-entropy*. S druge strane, mreža generatora ne ažurira svoje težine direktno, nego joj je potrebna povratna informacija diskriminatora. Dakle, generator se trenira u sklopu kompozitne *GAN* mreže koja sadrži generator i diskriminator (prilikom treniranja generatora, diskriminator je fiksiran i ne trenira se u tom periodu, ali vrijedi i obrat: prilikom treniranja diskriminatora, generator je fiksiran). Za vrijeme treniranja generatora, kompozitna mreža diskriminatoru šalje *fake* ulaz, odnosno već spomenutu konkatenaciju stvarne snimke i lažne mape, ali sa *real* labelom jer je cilj "prevariti" diskriminatora. Funkcija gubitka za ažuriranje težina generatora je *mean absolute error*, a koristi se za usporedbu stvarne Google mape na ulazu s lažnom mapom koju je generator izgenerirao. Arhitektura generatora sastoji se od tzv. *U-Net-a* s preskočnim vezama (često korišten za semantičku segmentaciju) koji se sastoji od *encoder* i *decoder* bloka, ali slika koja prikazuje strukturu je prevelika za ovaj format pa ju iz tog razloga ne prikazujemo. Arhitekturu diskriminatora čini tzv. konvolucijski *Patch-GAN* veličine 70×70 , koji je dao rezultate najbolje rezolucije. Dakle, umjesto da diskriminator u obzir uzima punu dimenziju ulaza, zapravo se fokusira na sve 70×70 lokalne podmatrice i svaku klasificira zasebno kao *real* ili *fake*. Time dobivamo 16×16 maticu vrijednosti 0 ili 1, pri čemu svaka vrijednost odgovara jednom 70×70 *patch-u*. Konačno, te vrijednosti se uprosječuju kako bismo dobili globalnu procjenu klasifikacije. Pripadni optimizator mreže diskriminatora je *Adam*, a korišteni hiperparametri su *learning rate* = 0.0002, *beta₁* = 0.5. Navedena funkcija gubitka koristi težinski faktor 0.5. Kompozitni model *GAN-a* koristi isti optimizator s istim postavkama te koristi kombinaciju dvije funkcije gubitka, a odnos težinskih faktora je 1 : 100 u korist *MAE* funkcije gubitka u odnosu na *binary cross-entropy*. Sve korištene postavke hiperparametara navedene su u radu [6], a detaljnije arhitekture diskriminatora i kompozitne *GAN* mreže prikazane su na slikama 6 i 7.

C. *cycleGAN* model

Treći i ujedno završni model koji promatramo je *cycleGAN* model. Kod *pix2pix* modela smo koristili parove odgovarajućih satelitskih snimki i Google mapa. Ovdje koristimo isti originalni *train dataset*, ali ovog puta razdvajamo slike. Dakle, nemamo više parove snimka-mapa, nego imamo zasebno 1096 satelitskih snimki i zasebno 1096 mapa. To je čest slučaj u praksi - jedan skup sadži slike jednog

modaliteta, a drugi skup slike drugog modaliteta. U tim situacijama od koristi može biti *cycleGAN* - mreža koja uči kako preslikati karakteristike jednog modaliteta u karakteristike drugog modaliteta uz odsustvo parova slika. Važno svojstvo preslikavanja je tzv. *cycle consistency*. Formalnije, ako imamo preslikavanja $f : X \rightarrow Y$ i $g : Y \rightarrow X$ tada mora vrijediti: f i g su bijekcije te g je "inverz" od f . Također, pratimo *cycle consistency loss* koji potiče da je $f(g(y)) \approx y$ i $g(f(x)) \approx x$. Drugo važno svojstvo je *identity mapping*, odnosno očuvanje modaliteta. Primjerice, ako generator pretvara snimku u mapu, onda on za svaku mapu koju dobije na ulazu generira mapu koja je njoj "identična", tj. pokušavamo smanjiti pogrešku promjene ulaza koji je istog modaliteta kao i izlaz generatora. To svojstvo se pokazalo dosta uspješnim za očuvanje boje. Arhitektura kompozitnog modela je nešto složenija od arhitekture *pix2pix* modela - sastoji se od 2 generatora i 2 diskriminatora. Oba diskriminatora su već spomenuti konvolucijski 70×70 *Patch-GAN-ovi* i imaju istu arhitekturu, samo što ovdje koriste *instance normalization* umjesto *batch normalization-a* kao što je slučaj kod *pix2pix* modela. Još jedna razlika s *pix2pix* diskriminatorom je u tome što ga ovdje treniramo na *buffer-u* od 50 prethodno izgeneriranih slika umjesto samo na jednoj. Diskriminator koristi isti optimizator i iste postavke hiperparametara kao i u prethodnom modelu. Također, postoje 2 generatora - jedan pretvara satelitske snimke u Google mape, a drugi radi obrnuto. Zbog toga imamo 2 diskriminatora. Uloga prvog je klasifikacija izlaza prvog generatora (procjenjuje je li generirana mapa *real* ili *fake*), a uloga drugog je klasifikacija izlaza drugog generatora (procjenjuje je li generirana snimka *real* ili *fake*). Arhitektura generatora koristi tzv. *ResNet* blokove sa preskočnim vezama. Takvi blokovi svoj izlaz konkateniraju sa ulazom. Slično kao i kod prethodnog modela, treniranje generatora se provodi kroz kompozitni model, pri čemu su diskriminatori u tom trenutku fiksirani te ih ne treniramo za vrijeme treniranja generatora, i obrnuto. Ukupno minimiziramo 4 funkcije gubitka: *mean squared error* ili *MSE* za *adversarial loss*, tj. ažuriramo težine generatora na temelju povratne informacije diskriminatora i 3 funkcije *MAE* - jedna za *identity loss* (želimo postići što idealniji *identity mapping*) i dvije za *cycle consistency loss* (preslikavanja $X \rightarrow Y \rightarrow X$ i $Y \rightarrow X \rightarrow Y$). Težinski faktori su u omjeru 1 : 5 : 10 : 10. Za kompozitni model koristimo isti optimizator s istim postavkama kao i u prethodnom modelu. Detaljnije arhitekture diskriminatora i kompozitnog modela, vidljive su na slikama 8 i 9, a slika arhitekture generatora je prevelika kao i kod prethodnog

modela pa ju iz istog razloga ne navodimo u ovom izvještaju.

IV. REZULTATI

Jedna od specifičnosti *GAN* neuronskih mreža je to što je teže odrediti konkretnu metriku kojom bismo procjenili točnost modela. Odnosno, prilikom pretvorbe satelitske snimke u Google mapu nemamo testni skup, kao na primjer kod klasičnog problema klasifikacije znamenki, na kojem bismo vidjeli kvantitativni rezultat metrike pa zapravo provodimo "eye-test". Pratimo treniranje modela kroz epohe i spremamo model svakih n -epoha. Za svaki spremljeni model spremamo i jedan primjerak predikcije, odnosno generirane mape u ovisnosti o trenutnim težinama mreže. Na kraju vidimo koji model nam daje najrealističnije rezultate i njega biramo za konačni model koji ćemo koristiti u aplikaciji. Sve postavke treniranja koje smo koristili su navedene u pripadnim radovima pa isto vrijedi i za veličinu *batch-a*.

A. CGAN

Model smo trenirali kroz 50 epoha. Veličina *batch-a* bila je 1 te je bilo 2614 *batcheva* po epohi. Kroz treniranje pratili smo funkciju gubitka generatora i diskriminatora. Graf funkcija gubitka prikazane su na slici 13. Na slici 10 vidi se generiranje iste slike svakih 5 epoha. Na slici 11 vide se neke satelitske snimke te pripadne generirane Google karte nakon 50 epoha kao i Google karte iz *dataset-a*. Na slici 12 vide se generirane Google karte za satelitske snimke iz *dataset-a* [3]. Možemo primijetiti da u slučaju generiranja karti iz satelitskih snimaka iz [3] model često zamjeni zelene površine sa vodenim površinama. To se događa jer su na satelitskim snimkama iz *dataset-a* za treniranje vodene površine tamno zelene boje.

B. Pix2pix

Model je treniran na 50 epoha, uz veličinu *batch-a* u iznosu 1, s ukupno 1096 slika po epohi. Dakle, treniran je na originalnom *train dataset-u*. Proces treniranja je trajao oko 2.5 sata na platformi Google Colab uz pomoć njihovog GPU-a, a grafovi koji prikazuju greške treniranja vidljivi su na slici 16. Nakon navedenog vremenskog perioda, Colab se redovito odspoji pa nije bilo moguće trenirati model na većem broju epoha. Rezultate smo spremali nakon svake desete epohе. Na slikama 14 i 15 vide se rezultati treniranja nakon 10 i nakon 50 epoha, a upravo model istreniran na 50 epoha smo uzeli za završni model.

C. cycleGAN

Za treniranje koristimo 1096 satelitskih snimki i 1096 Google mapa. Model je treniran na samo 4.5 epohe, iz razloga što je puno sporiji od *pix2pix* modela te se nakon 2.5 sata prekida veza s Colabom pa nije moguće trenirati na većem broju epoha. Na slici 17 vide se rezultati nakon prve epohe i praktički je nemoguće uočiti da bi to trebala biti Google mapa. Nakon 4 epohe situacija je bolja (slika 18), ali slike su i dalje neprimjenjive. Što se tiče pretvorbe Google mape u satelitsku snimku, situacija je "na oko" bolja. Slike 19 i 20 prikazuju stanje nakon 1. odnosno 4. epohe. Konačno, na slikama 21 i 22 nalaze se grafovi greški treniranja, pri čemu greška treniranja diskriminatora, koji provjerava pretvorbu snimke u mapu, nema nikakav silazni trend, što nas je iznenadilo.

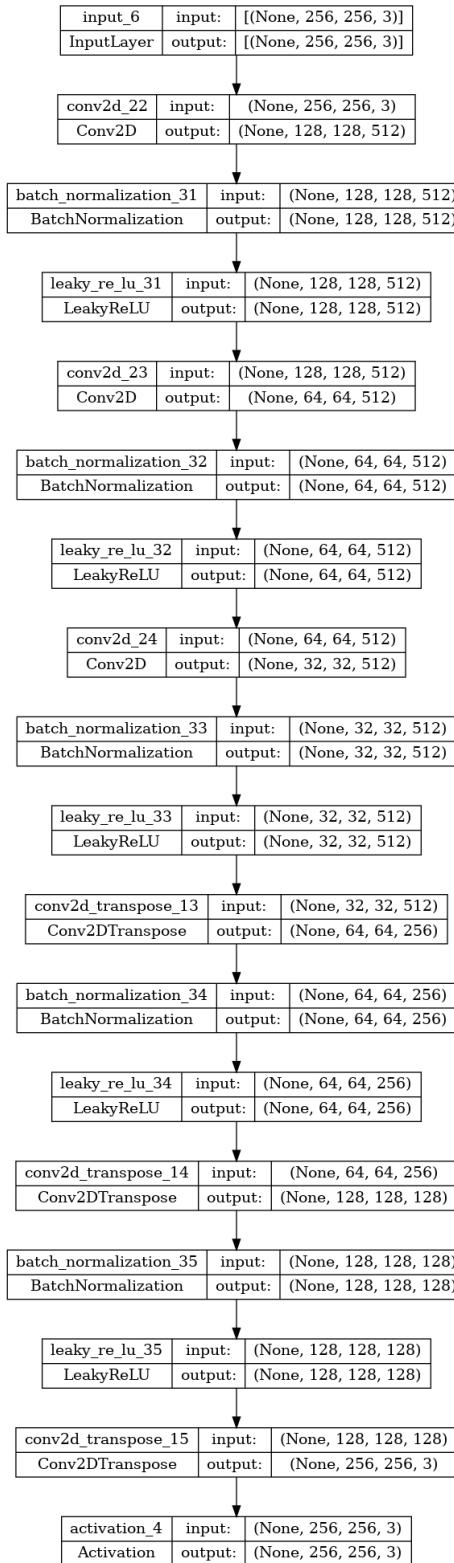
V. DRUGI PRISTUPI I MOGUĆI NASTAVAK ISTRAŽIVANJA

Iz dobivenih rezultata vidimo da je *cGAN* ostvario najbolji rezultat. Nešto lošiji bio je *pix2pix* model koji je treniran na dvostruko manjem broju epoha. Najlošiji rezultat ostvario je *cycleGAN*, što je bilo očekivano jer je treniran na samo 4 epohe. Dakle, prvo što bismo mogli poboljšati u našim modelima je povećanje broja epoha za zadnja dva modela, ali trenutno nismo mogli bolje od ovoga obzirom na naše dostupne resurse. Također, zanimljivo bi bilo s određenim alatima, primjerice gradCAM-om, provjeriti kako diskriminatori donose svoje odluke, odnosno koja područja na slici imaju najveću aktivaciju. Tako bismo možda saznali zašto je diskriminator za određenu sliku odlučio da je *real*, iako je ona možda *fake*. Bez toga je diskriminator praktički *blackbox*, a to nije prihvatljivo u određenim granama industrije ili znanosti - npr. u medicini. Na kraju, poželjno bi bilo dodatno istražiti neke konkretnije metrike za procjenu naših modela. Međutim, smatramo da nam je spremanje modela kroz n -epoha i "eye-test" praćenje rezultata bila prihvatljiva početna točka za evaluaciju naših modela.

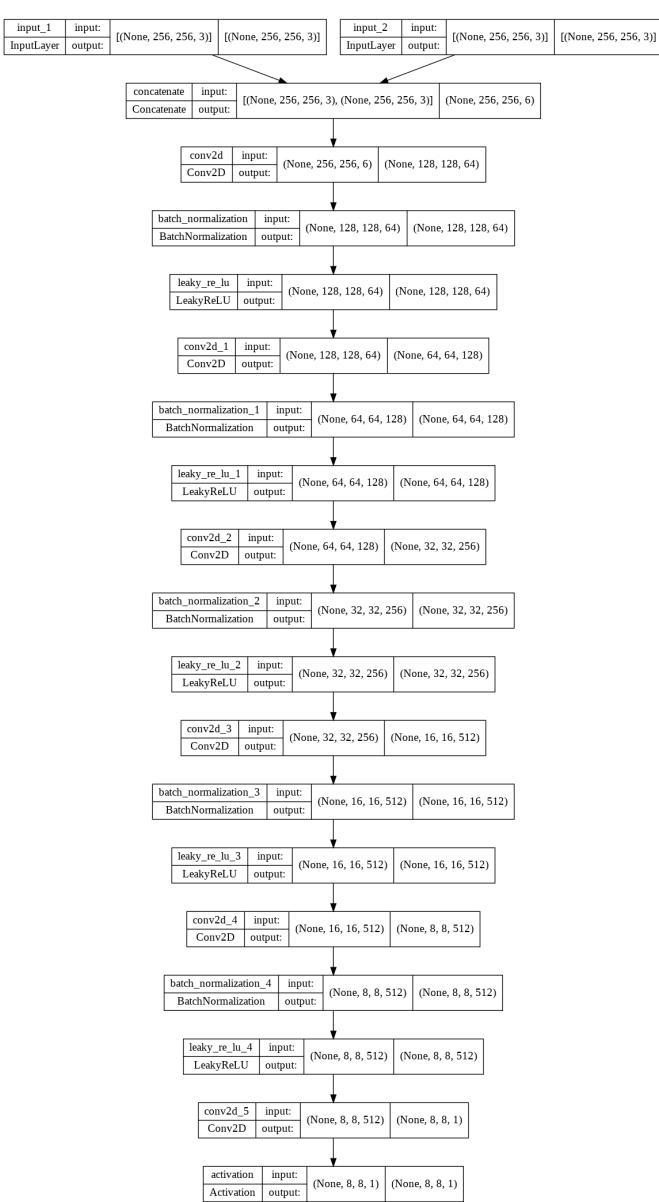
LITERATURA

- [1] URL: <https://www.kaggle.com/datasets/vikramtiwari/pix2pix-dataset>.
- [2] URL: <https://drive.google.com/file/d/1s5a2UeJR4HKJ-nV4NmRMkBHr3zn20Tf/view>.
- [3] URL: <https://www.kaggle.com/datasets/balraj98/deepglobe-road-extraction-dataset>.
- [4] URL: <https://www.kaggle.com/datasets/balraj98/deepglobe-land-cover-classification-dataset>.

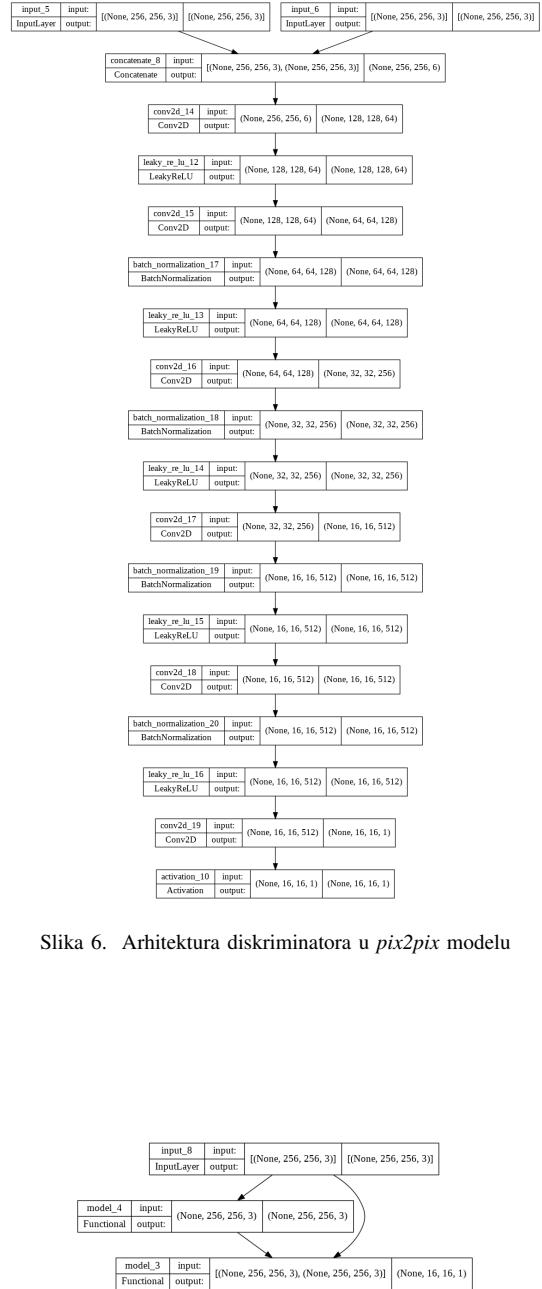
- [5] Vaishali Ingale, Rishabh Singh i Pragati Patwal. „Image to Image Translation: Generating Maps From Satellite Images”. (2019.). URL: <https://arxiv.org/pdf/2105.09253.pdf>.
- [6] Phillip Isola i dr. „Image-to-Image Translation with Conditional Adversarial Networks”. (2018.). URL: <https://arxiv.org/pdf/1611.07004.pdf>.
- [7] Zili Yi i dr. „DualGAN: Unsupervised Dual Learning for Image-to-Image Translation”. (2018.). URL: <https://arxiv.org/pdf/1704.02510.pdf>.
- [8] Jun-Yan Zhu i dr. „Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. (2020.). URL: <https://arxiv.org/pdf/1703.10593.pdf>.



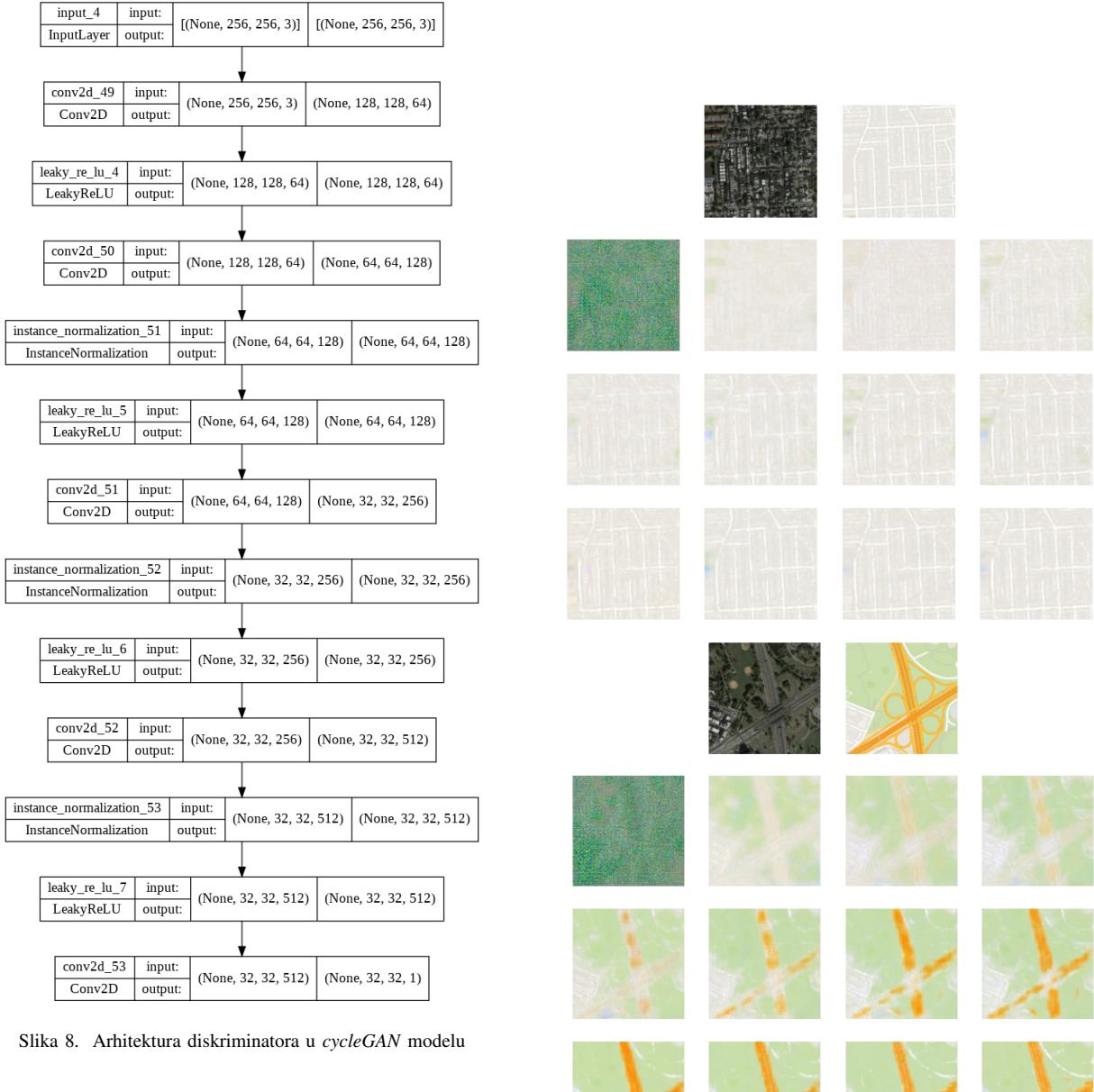
Slika 4. Arhitektura generatora u cGAN modelu



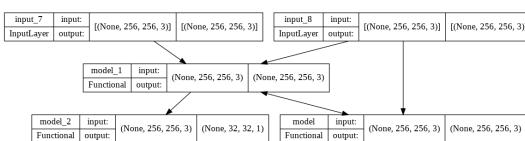
Slika 5. Arhitektura diskriminatora u *cGAN* modelu



Slika 7. Arhitektura kompozitnog *pix2pix* modela

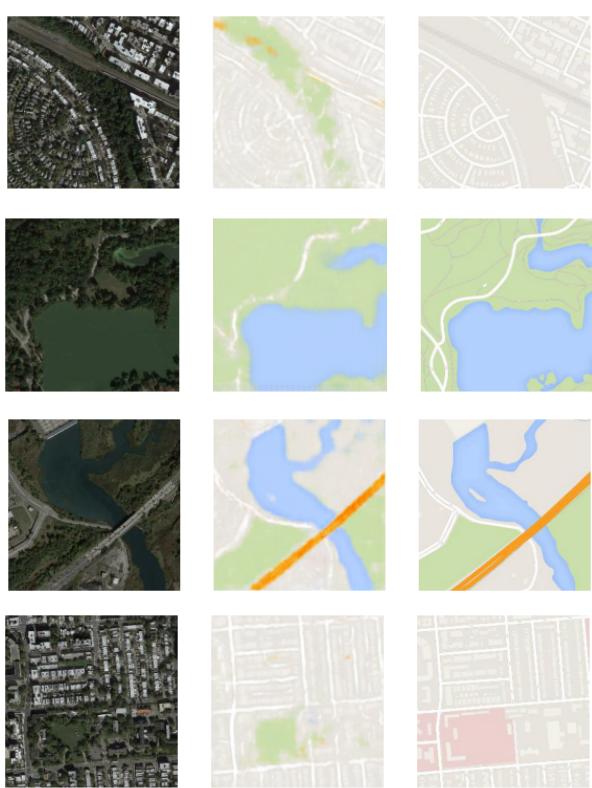


Slika 8. Arhitektura diskriminatora u *cycleGAN* modelu



Slika 9. Arhitektura kompozitnog *cycleGAN* modela

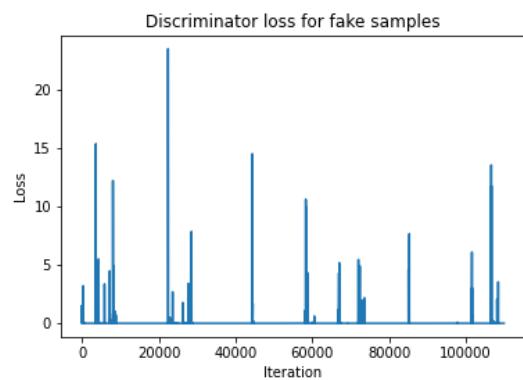
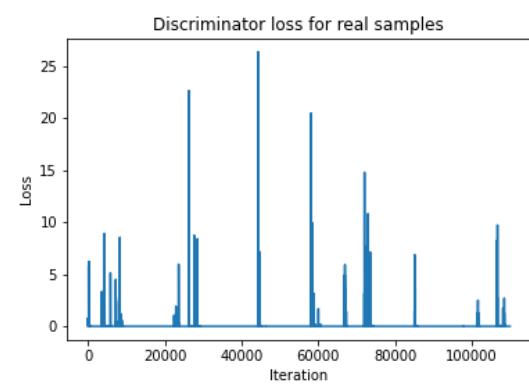
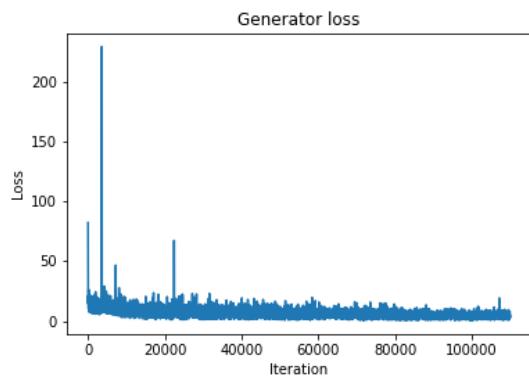
Slika 10. Generirane karte nakon svake pete epohe *cGAN* modelom



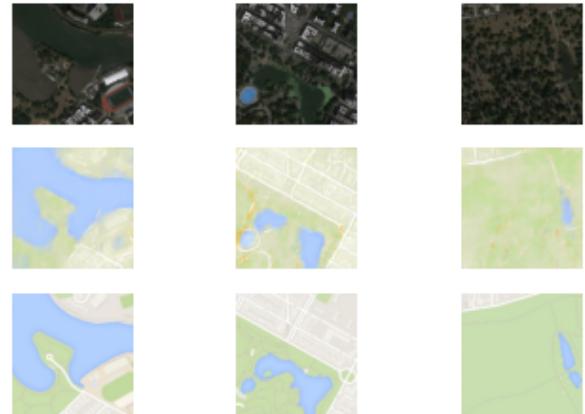
Slika 11. Satelitske snimke, generirane karte *cGAN* modelom te karte iz dataseta



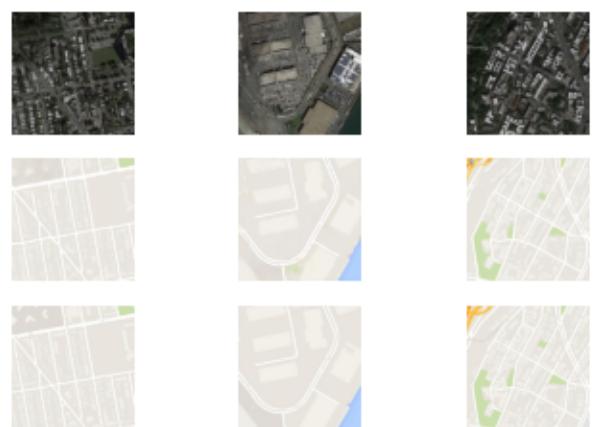
Slika 12. Rezultati za *cGAN* model



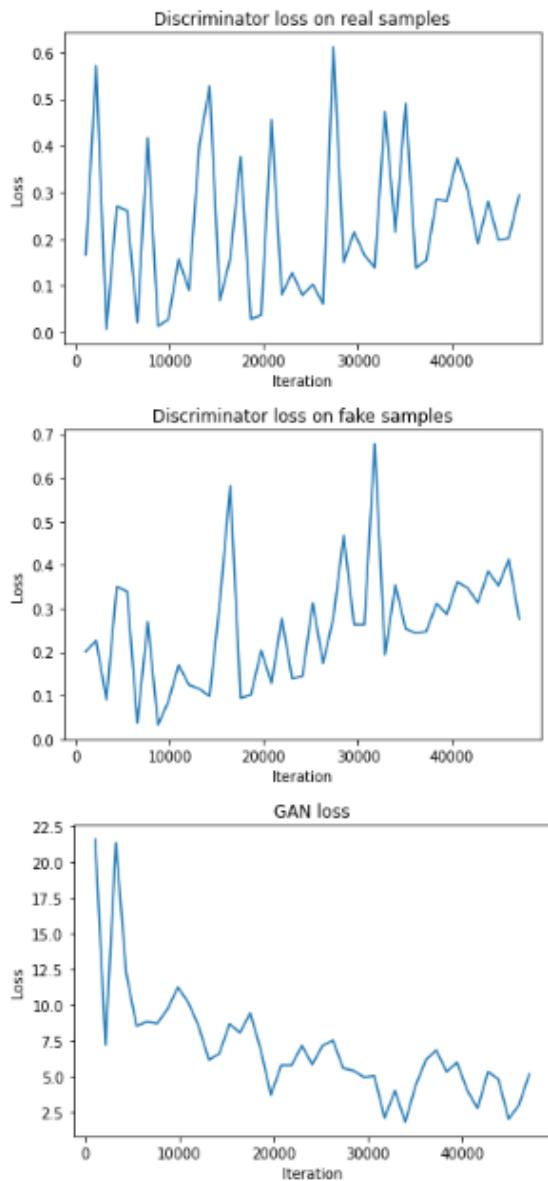
Slika 13. Greške treniranja *cGAN* modela



Slika 14. Rezultati treniranja *pi2pix* modela nakon 10 epoha - snimka, generirana mapa i stvarna mapa



Slika 15. Rezultati treniranja *pi2pix* modela nakon 50 epoha - snimka, generirana mapa i stvarna mapa



Slika 16. Greške treniranja za *pix2pix* model

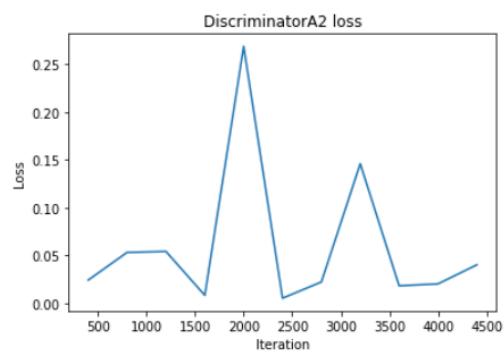
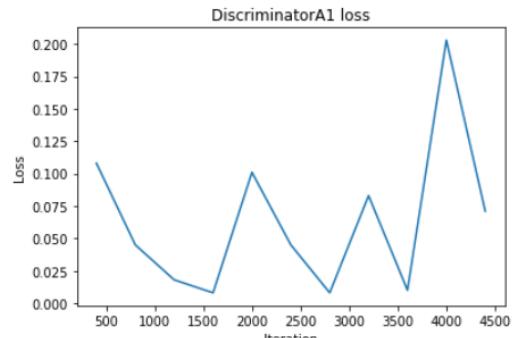


Slika 17. *cycleGAN* snimka u mapu, nakon 1. epohe

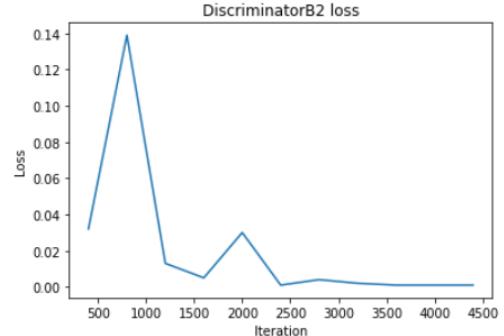
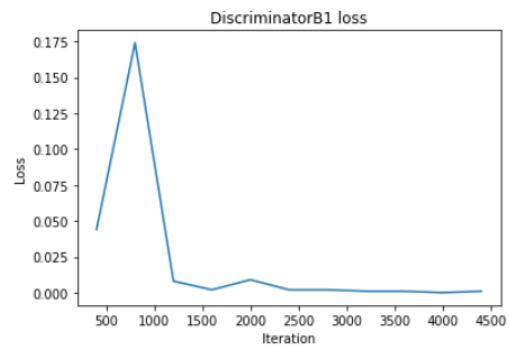
Slika 18. *cycleGAN* snimka u mapu, nakon 4. epohe



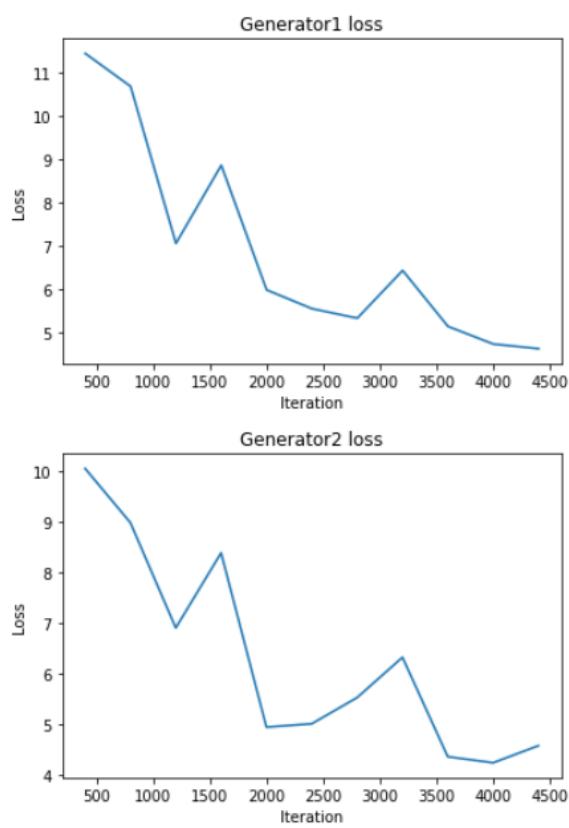
Slika 19. *cycleGAN* mapa u snimku, nakon 1. epohe



Slika 20. *cycleGAN* mapa u snimku, nakon 4. epohe



Slika 21. Greške treniranja diskriminatora *cycleGAN* modela



Slika 22. Greške treniranja generatora *cycleGAN* modela