# Генерисање мапа на основу сателитских снимака помоћу ГАН-а

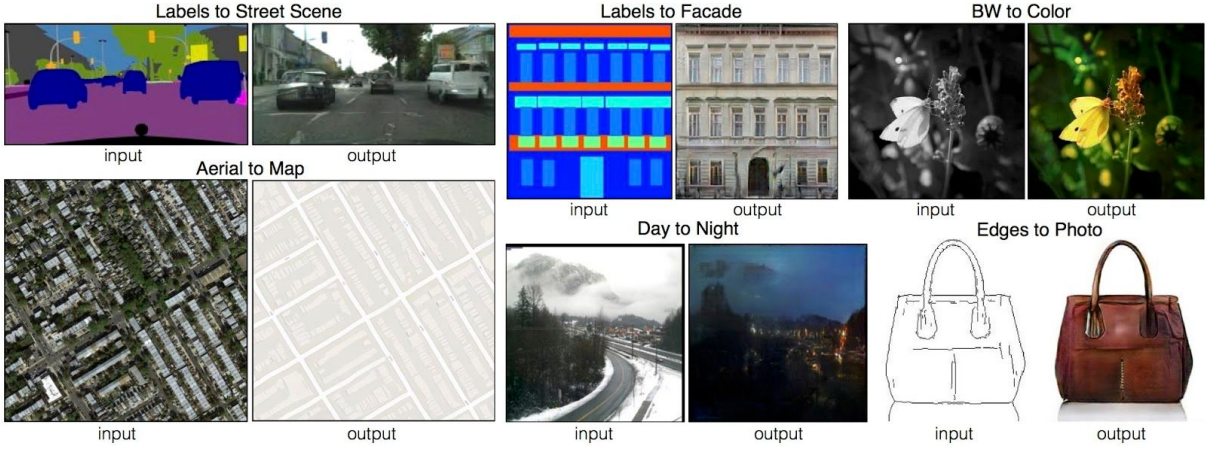
Никола Станојевић 92/2016
Дарко Васиљевић 449/2016

Математички факултет, Универзитет у Београду
Јануар 2020.

# 1 Увод

У многим областима рачунарства попут компјутерске графике, обраде слика и рачунарског вида јавља се проблем превођења улазних слика у одговарајуће излазне слике. Спектар проблема је широк, а најпознатији од њих су:

- пресликавање лабела у уличне приказе
- грађење фасада на основу скица
- трансоформисање дневних у ноћне сцене
- креирање мапа од сателитских снимака
- дизајјнирање објеката на основу скица




*Слика 1. Примери превођења слика*

Традиционално, сваки од ових задатака је био обрађиван засебним техникама, међутим, јавила се идеја да сви ови проблеми буду обрађивани једним алгоритмом, јер је и сама суштина свих њих иста. Зато се и тежило откривању мета-приступа који би решавао читаву класу проблема.

Овај нови приступ по први пут је приказао тим истраживача са Универзитета у Монтреалу, у раду ‘Generative Adversarial Nets’ [1] 2014. године, који су како сами кажу, приступили задатку примарно са теоријске стране и дали значајну полазну тачку за будуће радове. Код је доступан на `goodfeli/adversarial: Code and hyperparameters for the paper "Generative Adversarial Networks".`

Још једна група истраживача са Берклија је дала велики допринос овом пољу. Њихов рад ‘Image-to-Image Translation with Conditional Adversarial Networks’ [2] је дао један универзалан алгоритам применљив на целу класу проблема и то без посебног модификовања параметара, вођени само апстрактним циљем : ‘учинити да се предикција не разликује од реалности ’. Иако нису први у овој области, први су који су уопштили алгоритам за примену у свим случајевима, уз анализирање ефеката различитих архитектура. Резултати можда нису идеални али су ипак задовољавајућег квалитета. Код је доступан на `phillipi/pix2pix: Image-to-image translation with conditional adversarial nets.`

Конкретно, наше поље интереса је како на основу датог улаза у виду сателитског снимка терена, генерисати мапу. Постоји и неколицина радова који се баве специфично генерисањем сателитских снимака [3], али како су сви они вођени претходно наведеним приступом, неће бити разматрати надаље, већ ћемо се у наставку водити приступом изложеним у раду [2].

# 2 Технички приступ

У овом поглављу направљен је осврт на теоријску и практичну архитектуру мреже.

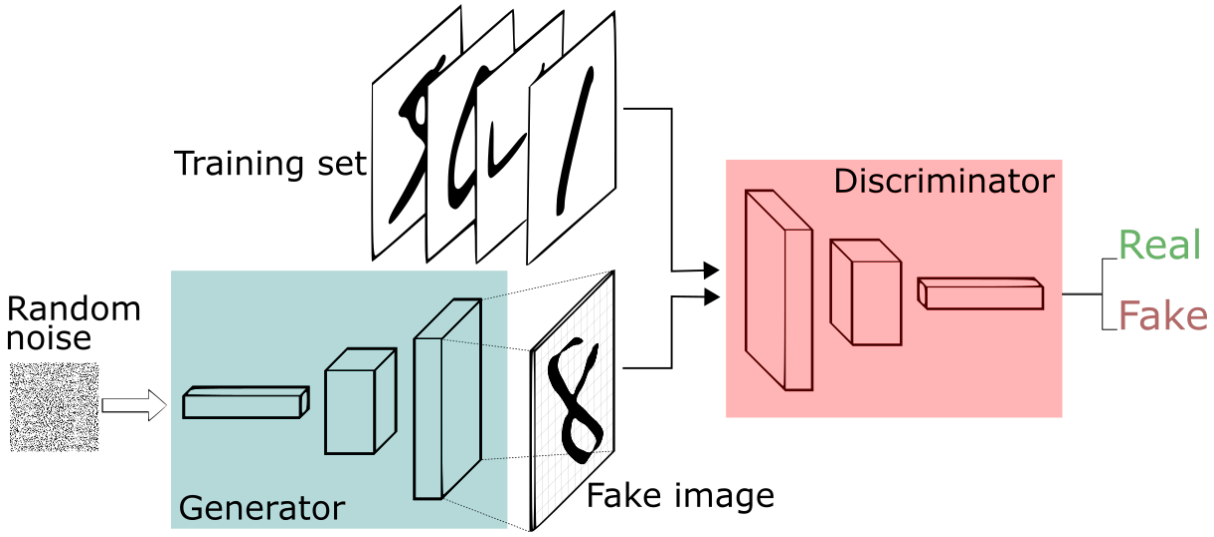
## 2.1 Скуп података

У раду су коришћене слике из скупа података '*maps*' које су такође употребљене у раду [2]. Овај скуп података се састоји од сателитских снимака Њујорка и околине и њима одговарајућих мапа, добијених на основу Google Maps API-a, подељених на тренинг и валидациони скуп који садрже редом 1.096 и 1.098 слика. Називи су цифарски у JPEG формату, а димензије су 1.200×600 пиксела. Подаци су јавни и доступни за преузимање на адреси [4].

*Слика 2. Узорак из скупа података*

## 2. 2 Методологија

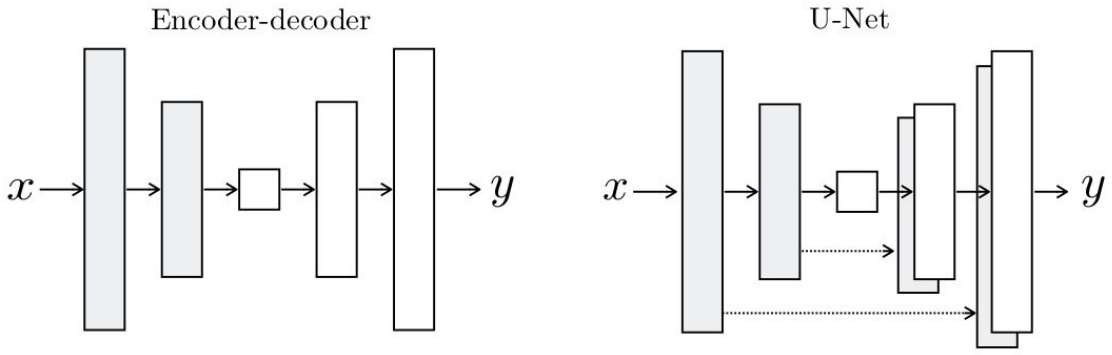
Методологија рада је заснована на генеративним супарничким мрежама (*eng. generative adversarial networks*) тзв. ГАН-овима.
Како се наводи у [5] “основна идеја супарничког приступа обучавању генеративних модела је да се користе два модела - генератор, који генерише податке и дискриминатор који класификује слике у праве и генерисане. Дискриминатор се обучава тако да што боље разликује генерисане слике од правих, а генератор се обучава да што боље генерише слике које дискриминатор неће разликовати од правих. Очигледно, како се генератор мења, мора и дискриминатор да се мења и обрнуто, тако да се ова два модела тренирају наизменично.”




*Слика 3. Приказ ГАН архитектуре*

Дискриминатор је дубока конволутивна мрежа која врши класификацију слика. Специфично, условну класификацију. То ради тако што узима и полазни сателитски снимак и циљну слику (приказ мапе) као улаз и предвиђа функцију веродостојности односно да ли је циљна слика права или лажна трансформација полазне.

Генератор је енкодер-декодер модел који користи *U-Net* архитектуру. За ралику од традиционалног ГАН приступа, где се генерисање врши на основу вектора случјног шума *z* (*eng. random noise*) излазну слику *y*, $G : z \rightarrow y$ као у [1], ми користимо тзв. кондиционо мапирање које је предложено у [2], а које при генерисању узима у обзир и посматрану слику *x,* тако да важи $G : \{x, z\} \rightarrow y$.
Ово се постиже тако што се прво улаз енкодира у простор мање димензије, тзв. уског грла (*eng. bottleneck layer*) а потом се декодира до величине излазне слике. U-Net архитектура значи да су додате конекције између одговарајућих симетриних слојева енкодера и декодера.




*Слика 4. Две могуће архитектуре генератора*

## 2. 3 Модел

Архитектуре генератора и дискриминатора имплементирају у потпуности приступ и детаље приказане у [2]. И генератор и дискриминатор користе модуле у форми '*convolution-BatchNorm-ReLu*' односно врши се конструкција слоја, нормализација серије података и потом активација слоја ReLu функкцијом. Све конволуције су 4x4 просторни филтери са кораком 2. Конволуције у енкодеру и дискриминатору, уситњавају улаз са фактором 2, а оне у декодеру увећавају за фактор 2.

Енкодер-декодер архитектура се састоји од:

- енкодер `C64-C128-C256-C512-C512-C512-C512-C512`
- декодер `CD512-CD512-CD512-C512-C256-C128-C64`

Након последњег слоја у декодеру, примењена је конволуција да се излаз из декодера измапира у број излазних канала, праћен активацијом '*tanh*' функцијом. Све ReLu функције у енкодеру су пропустљиве, са коефицијентом 0.2, док ReLu у декодеру нису пропустљиве.

Архитектура дискриминатора је: `C64-C128-C256-C512`. Након последњег слоја, конволуција је примењена за мапирање у 1-димензиони излаз, активиран сигмоидном функцијом. На први слој није примењена нормализација. Све ReLU функције су пропустљиве, са коефицијентом 0.2.

Додатно, архитектура читавог система, са свим мрежама и слојевима, дата је у фајлу *modelsummary.txt.*

## 2. 4 Имплементација

Приликом имплементације, коришћен је програмски језик Python[6], верзија 3.7.5 као и следеће библиотеке:

- NumPy [7] - Библиотека отвореног кода за нумеричко израчунавање. Коришћена је верзија 1.17.3
- Keras [8] - Библиотека отвореног кода која пружа високо апстрактни API, што је чини једноставном за коришћење. Ова библиотека захтева позадинску библиотеку за израчунавање; у ову сврху је коришћена библиотека TensorFlow[9]. Верзија библиотеке Keras која је употребљавана је 2.3.0, док је верзија библиотеке TensorFlow 2.0

Имплицитно су коришћене и библиотеке од којих наведене библиотеке зависе. У склопу имплементације коришћени су и неки модули који су део стандардне библиотеке језика Python.
Структура кода је описана у наставку, на апстрактном нивоу. Ово значи да су описи неких функција, које нису од важности за схватање кода, изостављени. Изворни код се може наћи на следећој адреси [10].

На самом почетку, покретањем скрипта *util.py* извшено је претпроцесирање података . Учитане слике се трансформишу до димензија 256×512 пиксела, деле на сателитски снимак и мапу и затим компресују у NumPy формат који се користи у даљој обради.

Саме неуронске мреже имплементиране су у фајлу *maps.py***.** Најзначајније функције су:

- **`define_generator(image_shape)`** - имплементира *U-Net* енкодер-декодер генератор модел. Користи помоћну функцију `define_encoder_block()` да креира блокове слојева за енкодер и `decoder_block()` функцију да креира блокове слојева за декодер.

- **`define_discriminator(image_shape)`** - имплементира модел дискриминатора као што је приказано у дизајну. Модел узима две улазне слике које су спојене заједно и врши предвиђање. Врши се оптимизовање бинарном крос-ентропијом(eng. *binary_crossentropy*) и скалирање тежина на пола, како је предложено у [2].

- **`define_gan(g_model, d_model, image_shape)`** - узима већ дефинисане моделе генератора и дискриминатора и спаја их у композитни модел.

- **`train(d_model, g_model, gan_model, dataset, n_epochs=100, n_batch=1)`** - тренира модел, у 100 епоха, да би се скратило време извршавања, иако је у раду [2] предложено 200 епоха

- **`summarize_performance(iter, g_model, dataset)`** - прави преглед модела и чува га у H5 формату фајла. Уз то се чувају и 3 упоредна приказа оригинала, предикције и стварне слике из датог модела

Фајлови *loader.py* и *single_img_loader.py* су скриптови који врше демострацију рада тренираног модела на случајно одабраним узорцима.

# 3 Резултати

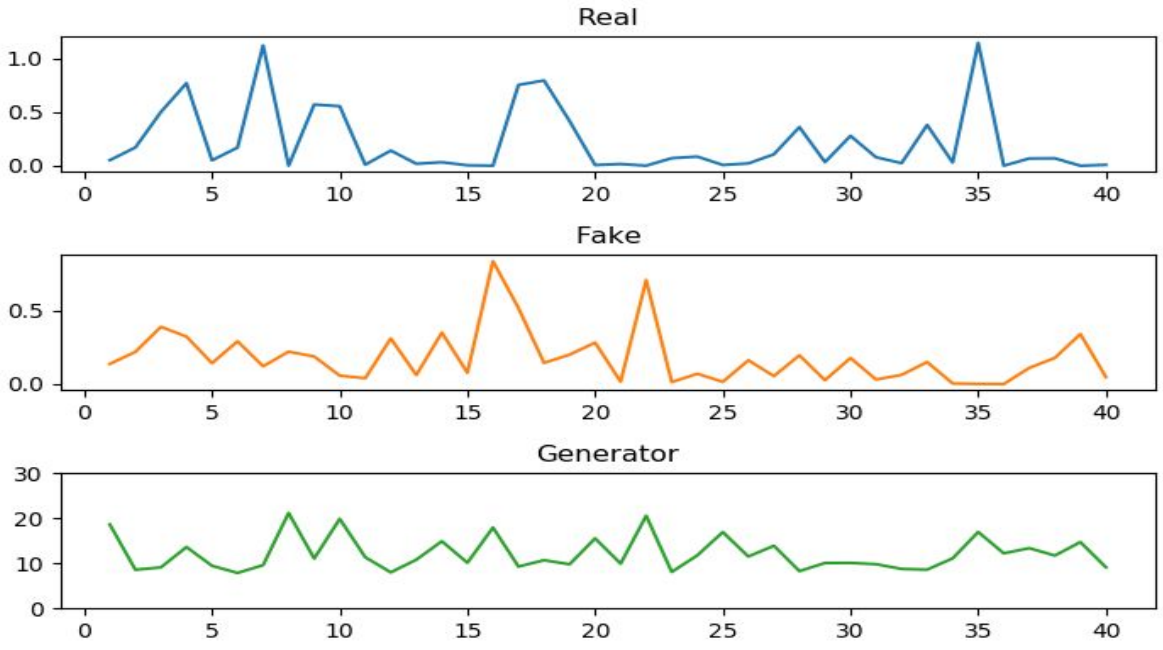
У овом поглављу направљен је осврт на тренирање модела, коришћену архитектуру, визуалне и квантитативне процене добијених модела.

## 3.1 Конфигурација

Битно је нагласити да је тренирање вршено у јако ограниченим условима, како временским, тако и хардверским. Тренирање је вршено у 40 епоха, користећи CPU: **`Intel® Core™ i5-6200U CPU @ 2.30GHz × 4`** и трајало је приближно 20 сати.

## 3.1 Анализа

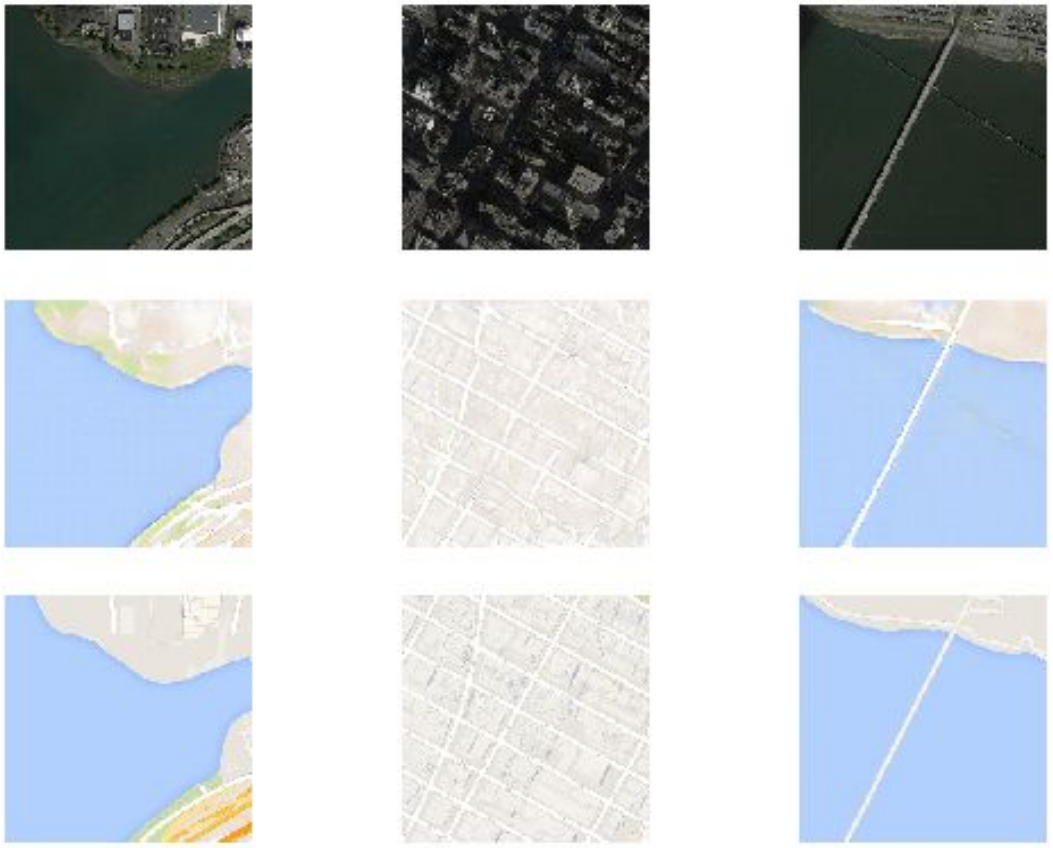
Узевши у обзир да ГАН модели углавном нису евалуирани праћењем веродостојности, већина наших евалуација ће бити квалитативне. Такође, кад су у питању ГАН-ови, више епоха не значи нужно и бољи квалитет модела, што се показало кроз наше резултате, па је модел престао са усавршавањем већ од 40. епохе




*Слика 5. Графици грешака кроз епохе*

Са графика се види да је већ од самог почетка дискриминатор јако успешно класификовао како праве, тако и лажне генерисане слике, већ после 20 итерација грешка је ограничена на сегмент [0, 1]. Грешка који је производио генератор приликом тренирања на групама података такође је већ након првих 20 итерација драстично ограничена, а након 30 епоха углавном је ограничена на сегмент [0,10].

На сваких 10 епоха, упоредо са генерисањем модела, приказан је и визуелно стадијум тренирања у коме се модел назали. Као што се помгло и очекивати, најбоље се показао 4. модел одн. након 40 епоха. На њему се види да генерисане слике изгледају јако реалистичне, иако линије улица нису у потпуности праве линије и поједине слике садрже мрље. Међутим, све велике структуре су на правим местима уз одговарајуће боје.



*Слика 6. Извештај током тренирања: сателитски снима - предикција - стварна мапа*


*Слика 7. Пример тестирања на случајном узорку*

Још оваквих приказа, као и тестова  је доступно у Github репозиторијуму пројекта [10].

# 4 Закључак

Узевши у обзир све аспекте и резултате, може се закључити да је успешно репродукован жељени приступ односно конструисана мрежа за генерисање мапа на основу сателитских снимака. Иако квалитет развијен алгоритма није дао резултате као онај у раду који је био водиља [2] при свим ограничењима које смо поставили, показао се довољно добро.

Међутим, оно што је сигурно је да ГАН-ови представљају обећавајући приступ у многим областима, поготов оним које укључују задатке са захтевним графичким излазом.

Даљи правац развоја се може усмерити у више праваца. Могу се подешавати параметри постојећег модела или може се постојећи модел искористити за неки нови приступ. Такође огроман простор за надоградњу отварају бројне технике оптимизације ГАН модела, попут оних предложених у раду [11].

# 5 Референце


[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. *Generative adversarial nets*

[2] Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2016. *Image-to-Image Translation with Conditional Adversarial Networks*

[3] Swetava Ganguli, Pedro Garzon, Noa Glaser, 2019. *GeoGAN: A Conditional GAN with Reconstruction and Style Loss to Generate Standard Layer of Maps from Satellite Images*

[4] Department of Electrical Engineering and Computer Sciences at UC Berkeley, *Maps Dataset*

[5] Mladen Nikolić, Anđelka Zečević:*Mašinsko učenje*. Matematički fakultet, Univerzitet u Beogradu, 2018

[6] *Python programski jezik*

[7] *NumPy programska biblioteka*

[8] *Keras programska biblioteka*

[9] *TensorFlow programska biblioteka*

[10] *GitHub repozitorijum*

[11] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Alec Radford, Vicki Cheung. *Improved Techniques for Training GANs*