SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7220

Učenje modela za prevođenje crteža u slike

Iva Panić

Zagreb, srpanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 7220

Pristupnica:	**Iva Panić (0036511318)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Učenje modela za prevođenje crteža u slike**

Opis zadatka:

Prevođenje slika važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate postižu konvolucijski modeli utemeljeni na suparničkom gubitku. U ovom radu naglasak je na učenju modela za prevođenje crteža u slike. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće pristupe za prevođenje slika iz jedne domene u drugu. Odabrati slobodno dostupni skup slika te oblikovati podskupove za učenje, validaciju i testiranje. Predložiti prikladnu arhitekturu dubokog modela za prevođenje slika. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele te prikazati i ocijeniti postignutu točnost. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

Zahvaljujem se mentoru, prof.dr.sc. Siniši Šegviću, na povjerenju, savjetima i prenesenom znanju. Hvala mojoj obitelji na bezuvjetnoj podršci tijekom cijelog školovanja.

SADRŽAJ

1. Uvod

Mnogi problemi u računalnoj grafici i računalnom vidu mogu se opisati kao problemi prevođenja ulazne slike u određenu izlaznu sliku. Kao što misao možemo izreći u hrvatskom ili engleskom jeziku, tako se i scena može prikazati kao skup rubova, slika u boji ili crno-bijela slika. U ovom završnom radu obrađuje se jedna vrsta takvog prevođenja - prevođenje crteža u sliku. Kako je ovo vrlo zanimljiv problem, što u umjetnosti, dizajnu, zabavi, obrazovanju ili forenzici, u prošlosti je bilo pokušaja pronalaženja rješenja, od kojih je najuspješniji bio, zapravo, fotomontaža - program bi uz pomoć oznaka s informacijama o predmetu tražio najsličniju fotografiju pretražujući internet te ju dodao na novu fotografiju u skladu s položajem na početnoj slici. Preokret u ovom području se dogodio kada je Ian Goodfellow 2014. godine predstavio generativne suparničke modele. Budući da su se pokazali iznimno uspješnima u problemima prevođenja iz slike u sliku, tijekom sljedećih godina razvilo se mnogo specifikacija generativnih modela. Jedna od tih specifikacija je i model pix2pix koji će se proučavati u ovom završnom radu.

2. Duboke neuronske mreže

2.1. Algoritam strojnog učenja

Strojno učenje područje je umjetne inteligencije koje proučava postupke rješavanja problema na temelju skupa podataka. Programi kojima rješavamo problem ne sadrže postojeća pravila na temelju kojih se dolazi do rješenja, nego algoritmom strojnog učenja sami uče kako doći do tih pravila u zadanom skupu podataka. Neka od područja u kojima se koriste algoritmi strojnog učenja su klasifikacija slika, prepoznavanje govora i autonomna vozila. U ovom radu, strojno učenje koristi se za generiranje slika na temelju crteža. Algoritam strojnog učenja definiran je modelom, gubitkom i postupkom optimizacije.

2.1.1. Model

Model je skup hipoteza (funkcija, preslikavanja) h određenih parametrima θ. Učeći model tražimo najbolju hipotezu h u postojećem skupu. Najbolja hipoteza je ona hipoteza koja daje najtočnije rješenje našeg problema, a svaka od njih evaluirana je po kriteriju dobrote. Broj hipoteza u praksi je vrlo velik, stoga najbolju hipotezu ne pretražujemo iscrpno, već uz pomoć heuristike koja će ubrzati pretragu postojećeg skupa. U strojnom učenju postoje razne modeli koje odabiremo ovisno o prirodi problema koji se rješava, od kojih su najčešće **umjetne neuronske mreže**.

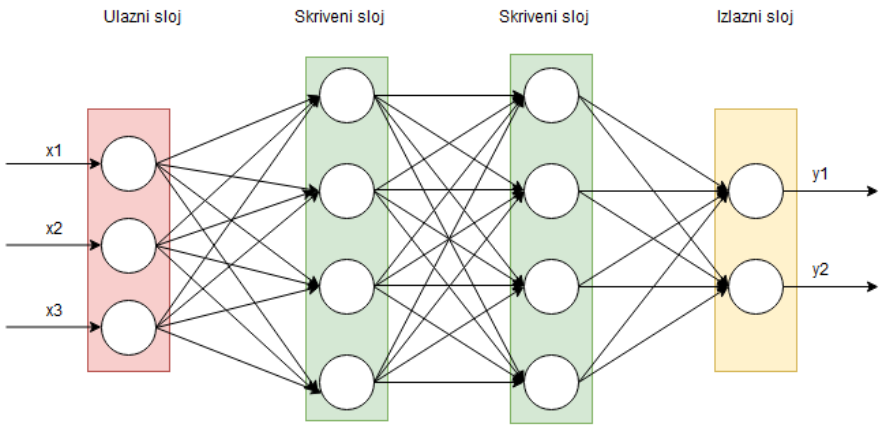
2.2. Definicija umjetne neuronske mreže

Neuronska mreža je algoritam strojnog učenja gdje je model proizvoljna nelinearna, ali jednom diferencijabilna, funkcija s velikim brojem parametara. Sastoji se od međusobno povezanih elemenata koje nazivamo jedinicama ili čvorovima čije su funkcionalnosti temeljene na funkcionalnostima biološkog neurona. Danas se smatra odličnim rješenjem za složene probleme kod kojih postoji odnos između ulaznih i izlaznih

varijabli - poput problema klasifikacije i predviđanja.

2.2.1. Arhitektura neuronske mreže

Sama neuronska mreža sadrži neurone koji su raspoređeni u n slojeva. Tri su osnovne vrste slojeva: ulazni sloj - ujedno i prvi sloj mreže, koji obrađuje ulazne podatke; izlazni sloj - posljednji sloj mreže, koji vraća predviđanja rješenja te skriveni sloj - svaki sloj između ulaznog i izlaznog sloja mreže.



Slika 2.1: Prikaz arhitekture neuronske mreže. Mreža na slici sadrži četiri sloja - ulazni sloj s tri neurona, dva skrivena sloja s četiri neurona i izlazni sloj s dva neurona, što možemo zapisati kao 3x4x4x2.

2.3. Vrste aktivacijskih funkcija

Neuroni se razlikuju na temelju aktivacijske funkcije koja je proizvoljna te služi da bi se postigla nelinearnost između slojeva. Kada bi aktivacijske funkcije bile linearne, tada se neuronske mreže ne bi mogle prilagoditi podacima i naučiti njihove nelinearnosti. Najjednostavnija aktivacijska funkcija svojstvena je modelu ADALINE.

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

Izlaz iz takve aktivacijske funkcije jednak je ulazu u funkciju. Umjesto takve aktivacijske funkcije, možemo se poslužiti korištenjem funkcije praga ili skoka. U tom slučaju, dobivamo procesnu jedinicu koja daje izlaz 0 ili 1. Najpoznatije i najčešće aktivacijske funkcije su sigmoidalna funkcija i zglobnica ili funckija ReLU.

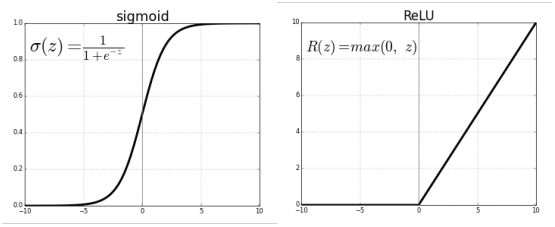
Sigmoidalna funkcija je funkcija oblika uz parametar a koji govori o nagibu funkije. Prednost ove funkcije nad ostalima je njena derivabilnost.

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

Zglobnica (engl. Rectified linear unit, ReLU) definirana je formulom

$$f(net) = \begin{cases} 0 & net = 0 \\ net & net > 0 \end{cases}. \tag{2.3}$$

Ova funkcija pokazala se vrlo korisnom u praksi zbog svoje nelinearnosti i gradijenta koji uvijek iznosi 0 ili 1.



Slika 2.2: Grafovi sigmoidalne funkcije i zglobnice. Slika je preuzeta iz [24].

2.4. Učenje neuronske mreže

Razlikujemo tri vrste učenja: nadzirano učenje (engl. supervised learning) i nenadzirano učenje (engl. unsupervised learning) i podržano učenje (engl. reinforcement learning). Kod nadziranog učenja pretpostavljamo da su podaci u obliku $(ulaz, izlaz) = (x, y)$ i da trebamo pronaći preslikavanje $\hat{y} = f(x)$. Kod nenadziranog učenja, podaci su dani bez ciljne vrijednosti i trebamo naći pravilnosti u podacima. Kod podržanog učenja, na temelju pokušaja s odgođenom nagradom učimo optimalnu strategiju. Skup primjera za učenje najčešće se dijeli na tri dijela: skup za učenje, skup za provjeru i skup za testiranje. Ti skupovi nam služe kako bismo izbjegli pretreniranje mreže - slučaj u kojem mreža gubi svojstvo generalizacije i savršeno obrađuje podatke iz skupa za učenje, dok nepoznate podatke obrađuje loše. Proces učenja mreže prilagodba je težina optimiranjem funkcije gubitka nad skupom za učenje.

2.4.1. Funkcija gubitka

Kako bismo mogli vrednovati model, trebamo definirati funkciju gubitka. Funkcija gubitka je mjera pogreške u predviđanju ciljne klase. Što je veći gubitak, tada je pogreška u predviđanju veća i naš model daje lošije rezultate. Funkcije gubitka mogu se

podijeliti u dvije skupine na temelju vrste učenja koje radimo - regresijski i klasifikacijski gubitci. U klasifikaciji, modele učimo na primjerima s oznakama razreda, dok u regresiji učimo na primjerima s brojčanim oznakama. Najčešće funkcije gubitka za svaku od ovih skupina su unakrsna entropija i srednja kvadratna pogreška.

Unakrsna entropija najčešća je klasifikacijska funkcija gubitka. Svaki predviđeni izlaz uspoređujemo sa stvarnom vrijednosti te vrijednost gubitka temelji se na tome koliko je predviđeni izlaz drugačiji od stvarne vrijednosti. Izraz kojim definiramo unakrsnu entropiju glasi

$$E(p, q) = -\sum_{x} p(x) \log q(x) \tag{2.4}$$

gdje su funkcije p i q funkcije vjerojatnosti. **Srednja kvadratna pogreška (MSE)** najčešća je regresijska funkcija gubitka. Izraz kojim definiramo srednju kvadratnu pogrešku glasi

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{y} - y_i)^2 \tag{2.5}$$

gdje je y_i stvarna vrijednost izlaza, $\hat{y}$ predviđeni izlaz, a n umnožak uzoraka i izlaza.

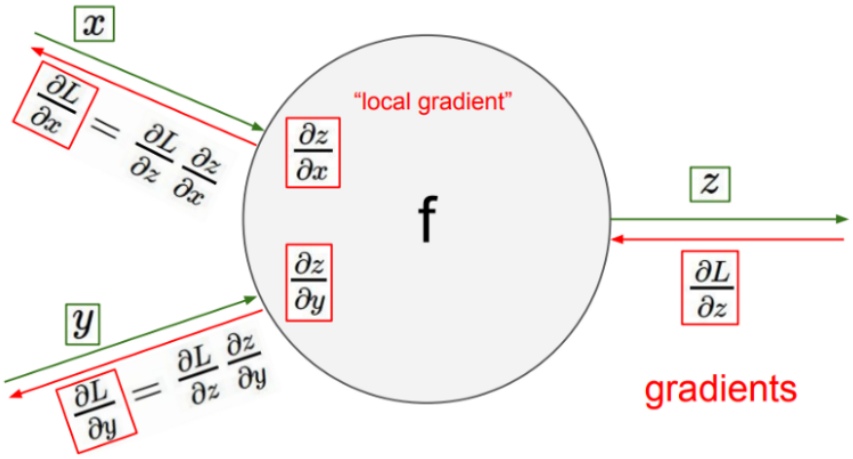
2.4.2. Algoritam *backpropagation*

Algoritam *backpropagation* koristi se za računanje gradijenata gubitka po parametrima modela. Propagirajući ulaze x kroz model da bismo dobili izlaz $\hat{y}$, informacije putuju unaprijed kroz skrivene slojeve prema izlaznom sloju modela. Izlaz $\hat{y}$ koristi se kako bi se izračunao gubitak za dane primjere za učenje x. *Backpropagation* omogućuje propagaciju dobivenog gubitka unazad kroz izlazni sloj i skrivene slojeve kako bi se odredio gradijent gubitka po parametrima modela. Algoritam se sastoji od tri koraka:

- **Prolaz unaprijed** - izračun izlaza modela $\hat{y}$ i funkcije gubitka
- **Prolaz unatrag** - izračun gradijenta funkcije gubitka na temelju parametara modela
- **Osvježavanje težina** - ažuriranje parametara modela metodom gradijentnog spusta na sljedeći način:

  $$z \leftarrow z - \eta \frac{\partial L}{\partial z} \tag{2.6}$$

  gdje su parametri promatranog sloja modela označeni sa z, L je gubitak modela za ulaz x, a η predstavlja stopu učenja, pri čemu je $\eta \in (0, 1)$



Slika 2.3: Prikaz algoritma *backpropagation*. Neka su z parametri promatranog i-tog sloja modela, L gubitak modela za ulaz x. Model se sastoji od n slojeva. Gradijent gubitka po z može se napisati kao umnožak gradijenata po izlazima i ulazima slojeva koji se nalaze nakon promatranog i-tog sloja, pravilom ulančavanja. Slika je preuzeta iz [25].

2.4.3. Stohastički gradijentni spust

Stohastički gradijentni spust (SGD) osnovni je i najčešće korišteni algoritam za optimizaciju neuronskih mreža. Ovaj algoritam osvježava vrijednost funkcije gubitka uz pomoć stope učenja i gradijenta funkcije gubitka na sljedeći način:

$$z \leftarrow z - \eta \nabla Q_i(z) \tag{2.7}$$

pri čemu je z trenutna vrijednost funkcije gubitka, η stopa učenja, a $\nabla Q(z)$ gradijent funkcije gubitka. Gradijent funkcije gubitka procjena je stvarnog gradijenta jer se izračunava iz slučajno odabranog podskupa podataka pa se na taj način smanjuje računsko opterećenje. Ovo osvježavanje događa se u svakom koraku algoritma, stoga uvijek postoji šum i u optimumu vrijednost procijenjenog gradijenta ne postaje nula. Kako bi optimizacija konvergirala, trebamo uvesti u algoritam promjenjivu stopu učenja, to jest, stopu učenja smanjivati za vrijeme učenja kako bi se smanjio utjecaj šuma.

2.4.4. Optimizacijski algoritam *Adam*

Vrijednost stope učenja ima velik utjecaj na uspješnost modela. Ako nam je stopa učenja premala, to će uzrokovati iznimno sporu konvergenciju i povećati mogućnost zapinjanja u lokalnom minimumu, dok prevelika stopa učenja uzrokuje nestabilnosti i, u nekim slučajevima, divergenciju. Kako bismo maksimizirali uspješnost modela, stopa učenja ne bi trebala biti konstantna, već se prilagoditi procesu učenja. U početku,

veća stopa može ubrzati konvergenciju, no kasnije ju treba smanjiti kako bismo izbjegli divergenciju i nestabilnosti. Takvu stopu učenja nazivamo *prilagodljivom* i koristi se u optimizacijskom algoritmu *Adam* (engl. Adaptive Moments). Algoritam Adam za ubrzavanje učenja koristi *momente* - prati prosječno kretanje gradijenta i kvadrata gradijenta uz eksponencijalno zaboravljanje. Korekciju parametara provodi na temelju prosječnih gradijenata, umjesto na izravno izračunatom gradijentu. Implementacija mu je dosta robusna, no radi sa širokim skupom parametara te pokazuje dobre rezultate u praksi. Algoritam se implementira na sljedeći način:

Algorithm 8.7 The Adam algorithm

Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
  Initialize time step $t = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
    Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

Slika 2.4: Pseudokod algoritma Adam. Slika je preuzeta iz [21].

2.5. Duboki konvolucijski modeli

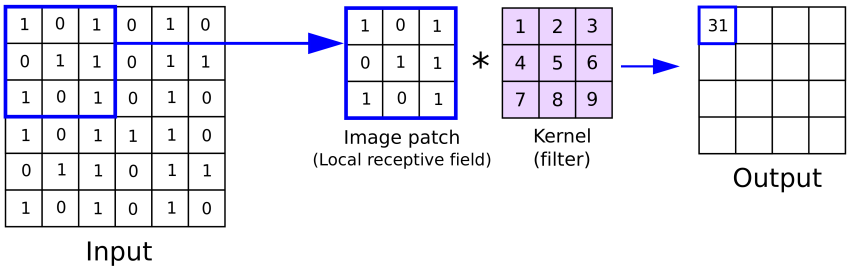
Konvolucijske neuronske mreže podvrsta su višeslojnih neuronskih mreža. Poput prethodno opisanih neuronskih mreža, sastoje se od ulaznog sloja, izlaznog sloja i skrivenih slojeva, no razlikuje ih način na koji su ti slojevi povezani. Prethodno opisane mreže imale su potpuno povezane skrivene slojeve i zbog toga ih nazivamo potpuno povezanim modelima. Potpuna povezanost skrivenih slojeva može uzrokovati prenaučenost ili usporeno učenje zbog prevelikog broja parametara i poveznica, što je kod zadataka poput obrade slike i signala posebno izraženo zbog sklonosti modela da uči šum. Rješenje ovog problema je uvođenje konvolucijskog sloja koji smanjuje broj parametara te poboljšava generalizaciju.

2.5.1. Konvolucijski sloj

Konvolucijski sloj koristimo za podatke koji imaju pravilnu topološku strukturu, poput rečenica i slika. Kod takvih podataka konvolucijski sloj je koristan jer u model unosi pristranost - isti objekti u različitim dijelovima dovode do istih podražaja. Ovaj oblik pristranosti naziva se **translacijska ekvivarijantnost**. Točnije, za funkciju $f(x)$ kaže se da je ekvivarijantna funkciji $g(x)$ ako vrijedi $f(g(x)) = g(f(x))$. Na primjer, imamo funkciju g koja pomiče svaki piksel slike S za jedan piksel udesno, $S'(x, y) = S(x - 1, y)$. Ako prvo primijenimo transformaciju g na sliku S pa onda konvoluciju, rezultat će biti isti kao da smo prvo primijenili konvoluciju, a zatim g na izlaz. Kako bismo definirali konvolucijski sloj, prvo treba definirati konvoluciju, po kojoj je i dobio ime. **Konvolucija** u strojnom učenju definirana je izrazom

$$I * K(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n) \tag{2.8}$$

gdje I označava ulaznu mapu značajki, a K jezgru konvolucijskog sloja. Princip rada konvolucije prikazan je na slici 2.5.



Slika 2.5: Prikaz rada konvolucije nad ulazom dimenzija 6x6 (na ilustraciji: Input) uz jezgru dimenzija 3x3 (na ilustraciji: Kernel). Jezgra K kreće se po ulazu I računajući izlaz $I * K$ (na ilustraciji: Output). Dio ulaza veličine jezgre koji se množi jezgrom naziva se receptivno polje. Slika je preuzeta iz [26].

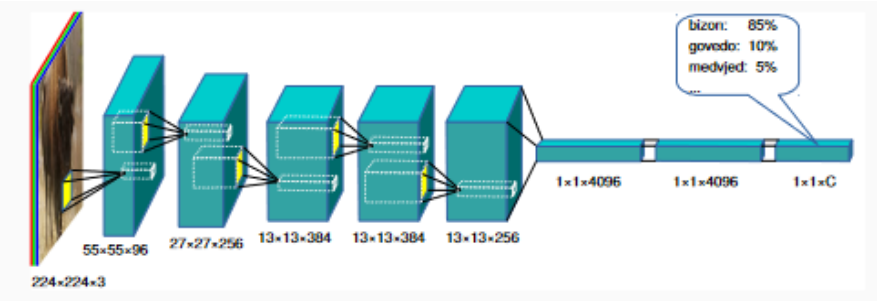
Jezgra konvolucijskog sloja treba biti malih dimenzija za koje se najčešće koriste neparni brojevi kako bi postojao piksel koji predstavlja sredinu jezgre. Konvolucijski sloj određuju četiri parametra:

- veličina jezgre
- dubina - broj mapi značajki na izlazu iz sloja
- korak - broj koji određuje za koliko se elemenata treba pomaknuti jezgra. Što je korak veći, izlazna prostorna dimenzija je manja.

– nadopunjavanje nulama - širina ruba koji će se popuniti nulama kako bi se sačuvala izvorna prostorna dimenzija

2.5.2. Sloj sažimanja

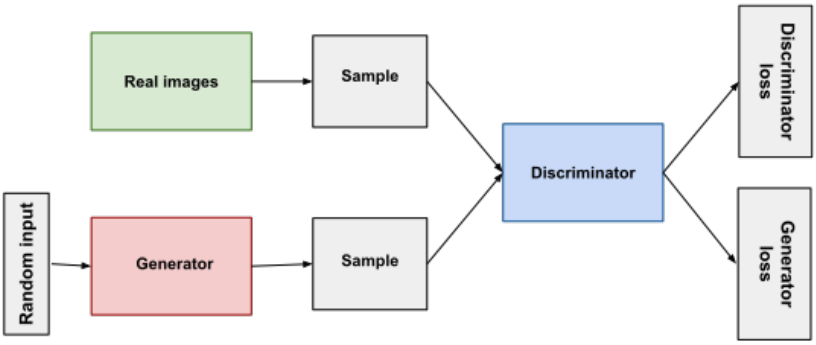
Sloj sažimanja uvodi se u mrežu kada nakon nekoliko konvolucijskih slojeva želimo dodatno smanjiti dimenzije mape značajki. Smanjivanjem njenih dimenzija, povećava se efektivno receptivno polje kasnijih slojeva i mogućnost generalizacije. Funkcija sažimanja mapira skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu te definirana je s veličinom jezgre i korakom, a princip rada sažimanja, prikazan na slici 2.6, sličan je radu konvolucije. Sažimanje koristimo kada tenzor (sliku, rečenicu) želimo prevesti u simboličku kategoriju.



Slika 2.6: Prikaz rada sažimanja. Postoji nekoliko vrsta sažimanja, a najčešće su sažimanje srednjom vrijednosti i sažimanje maksimalnom vrijednosti. U procesu sažimanja, uzimaju se značajke iz grupa ulaznih podataka te se mapiraju u jednu značajku na izlazu. Kod sažimanja srednjom vrijednosti, ta značajka dobiva se aritmetičkom sredinom vrijednosti na ulazu, a kod sažimanja maksimalnom vrijednosti, ta značajka prima vrijednost najveće značajke na ulazu. Slika je preuzeta iz [22].

3. Generativni suparnički modeli

Generativni suparnički modeli prvi su puta predstavljeni 2014. godine u znanstvenom radu Iana Goodfellowa i ostalih istraživača na Sveučilištu u Montrealu [4]. U usporedbi s ostalim generativnim modelima, poput autoenkodera i Boltzmannovog stroja, pokazalo se da generiraju vizualno najbolje uzorke. Ponašanje generativnog suparničkog modela može se jednostavno opisati kao natjecanje između dvije neuronske mreže: generatora, kojemu je cilj generirati nove uzorke te diskriminatora, čija je svrha da pri svakom ulazu odluči pripada li uzorak skupu za učenje ili je umjetno generiran od strane generatora. Generator pokušava nadmudriti diskriminator generirajući umjetne uzorke što sličnije pravim uzorcima, dok diskriminator pokušava što točnije klasificirati uzorke koji mu dolaze na ulaz. Ovakvo ponašanje uzrokuje da generator i diskriminator napreduju u izvršavanju svojih poslova sve do trenutka kada diskriminator više ne može odlučiti je li uzorak koji je primio na ulaz umjetno generiran ili pripada skupu za učenje.



Slika 3.1: Struktura generativnog suparničkog modela. Generator na ulaz prima slučajni šum te generira lažne uzorke koji se uz uzorke iz skupa za učenje šalju na ulaz diskriminatora povezanog s dvije funkcije gubitka - funkcijom gubitka generatora i funkcijom gubitka diskriminatora. Diskriminator klasificira uzorak te se unatražnom propagacijom i stohastičkim gradijentnim spustom osvježavaju njegove težine ovisno o točnosti klasifikacije, dok se težine generatora na isti način osvježavaju ovisno o ocjeni diskriminatora. Slika je preuzeta iz [27].

3.1. Diskriminator

Diskriminator se matematički označava s $D(x, \theta_d)$ te je prikazan kao funkcija D sa skupom parametara θ_d te uzorkom x na ulazu koja određuje klasifikaciju primljenog uzorka - pripada li on skupu za učenje ili skupu umjetno generiranih uzoraka. Cilj diskriminatora je istovremena maksimizacija funkcije $D(x)$ i minimizacija funkcije $D(G(z))$, gdje je G funkcija generatora.

3.2. Generator

Generator se matematički prikazuje kao diferencijabilna funkcija G sa slučajnim šumom na ulazu z i vlastitim parametrima θ_g te se označava s $G(z, \theta_g)$. Ulaz u generator je vektor slučajnog šuma z s uniformnom distribucijom $p(z)$ koji prolazi kroz generator čiji je zadatak približiti distribuciju generiranih podataka $G(z|\theta_g)$ distribuciji stvarnih podataka što se postiže maksimizacijom gubitka ovisnog o $D(G(z))$, gdje je D funkcija diskriminatora.

3.3. Učenje

Postupkom učenja generativnog suparničkog modela optimizira se sljedeća funkcija cilja s obzirom na parametre obje mreže:

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

Iz izraza (3.1) u kojem su zadani maksimacijski izraz diskriminatora i minimimacijski izraz generatora $log(1 - D(G(z))$ vidi se da je optimizacija funkcije cilja minimax igra u kojoj se usporedno događa diskriminatorova maksimizacija $D(x)$ i generatorova minimizacija $log(1 - D(G(z)))$. U praksi, umjesto minimizacije $log(1 - D(G(z)))$ koja daje slabije gradijente, maksimiziramo $log(D(G(z))$, što odgovara generatorovoj namjeri. U funkciji cilja treba vrijediti da je suma funkcija cilja diskriminatora i generatora jednaka nuli.

$$J^{(D)} + J^{(G)} = 0 \quad (3.2)$$

Kad se izraz (3.1) razdvoji na funkcije gubitka diskriminatora i generatora koristeći pravila minimax igre, dobiju se sljedeći izrazi:

$$J^{(D)} = -\mathbb{E}_{x \sim p_{data}(x)}[log(D(x))] - \mathbb{E}_{z \sim p_{data}(z)}[log(1 - D(G(z))] \quad (3.3)$$

$$J^{(G)} = -J^{(D)} \quad (3.4)$$

Funkcija gubitka generatora u minimax igri (3.4) najčešće se koristi u teorijskoj analizi, ali ne pokazuje rezultate u praksi. Promatrajući gornje izraze može se primijetiti da oni odgovaraju unakrsnoj entropiji koju tijekom igre diskriminator minimizira, a generator maksimizira. Takvo ponašanje dovodi do slučaja gdje, kada diskriminator gotovo savršeno odbaci uzorke generatora, gradijent generatora iščezava. To se događa zato što $log(1-D(G(z))$ poprima jako male vrijednosti. Ovo ponašanje je problematično jer je gradijent potreban za daljnje učenje modela pa umjesto minimizacije $log(1-D(G(z))$ za učenje generatora koristimo maksimizaciju $log(D(G(z))$. Takav pristup se pokazao korisnim jer u tom slučaju gradijenti iščezavaju tek kada učenje modela postane savršeno, a tada ono i završava. Umjesto da funkciju gubitka generatora računamo kao negativnu vrijednost funkcije gubitka diskriminatora, može se zapisati kao maksimizacija vjerojatnosti da je diskriminator u krivu, što je prikazano u izrazu (3.5):

$$J^{(G)} = -\mathbb{E}_{z\sim p_{data}(z)}[log(D(z))] \tag{3.5}$$

Uz korištenje prethodno navedenih funkcija gubitka, algoritam učenja modela je sljedeći:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

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

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

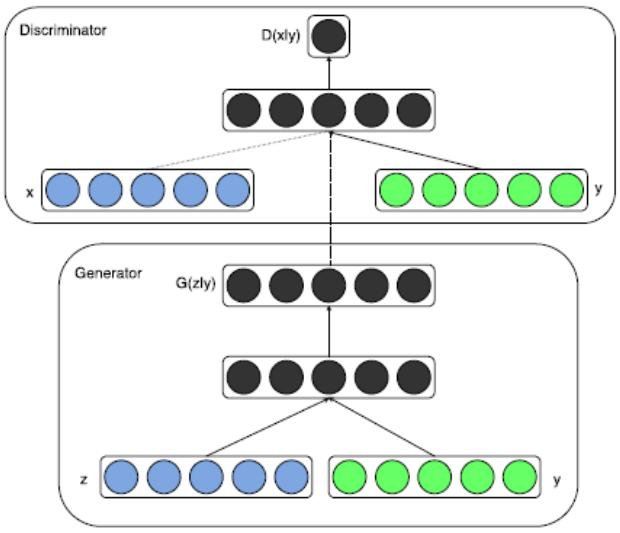
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Slika 3.2: Algoritam učenja modela. Kroz svaku iteraciju algoritma, diskriminator se uči broj koraka k kako bi se uštedilo na vremenu, a samo učenje diskriminatora se temelji na osvježavanju parametara povećanjem vrijednosti stohastičkog gradijenta. Nakon što je završeno učenje diskriminatora, izvršava se osvježavanje vrijednosti parametara generatora smanjenjem vrijednosti stohastičkog gradijenta. Slika je preuzeta iz [4].

3.4. Generativni uvjetni suparnički modeli

Generativni se suparnički model može proširiti i učiti tako da se generator i diskriminator uvjetuju nekim dodatnim informacijama y. Takav se model naziva uvjetni suparnički model i prvi puta su ga opisali Mehdi Mirza i Simon Osindero u radu "Conditional Generative Adversarial Nets" [4] 2014. godine. Dodatne informacije mogu biti bilo kakve informacije koje bi pomogle pri učenju, poput oznaka razreda. To znači da, kada se model koristi kao samostalni model za generiranje slika u nekoj domeni, mogu se generirati slike određenog tipa ili oznake klase. Uvjetovanje modela može se izvršiti dodavanjem informacija y u diskriminator i diskriminator kao dodatni ulaz. U generatoru se prethodni ulazni šum z i dodatne informacije y kombiniraju te u ovom slučaju funkcija minimax igre izgleda ovako:

$$min_G\, max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[log(D(x|y))] + \mathbb{E}_{z\sim p_{data}(z)}[log(1 - D(G(z|y))] \tag{3.6}$$



Slika 3.3: Arhitektura uvjetne suparničke mreže. Dodatna informacija y uvodi se u diskriminator i generator kao dodatni sloj što pridonosi učenju na više načina - bržim učenjem ili boljom kvalitetom generiranih slika. Slika je preuzeta iz [3].

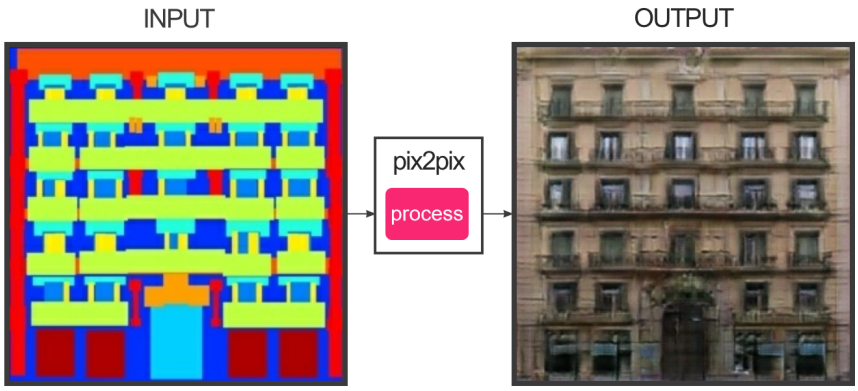
Kako uvjetni suparnički modeli uz preslikavanje uče i karakteristike vezane uz dodatnu informaciju y, pokazalo se da daju iznimno dobre rezultate na problemima poput obrade slika, grafike i računalnog vida koji uključuju prevođenje ulazne slike u odgovarajuću izlaznu sliku.

4. Prevođenje crteža u slike

Kako možemo brzo vizualizirati predmet ili prizor? Prvo što bismo odgovorili je crtežom ili fotografijom. Za razliku od fotografije, crtež ne zahtijeva posjedovanje uređaja za snimanje te može biti rezultat mašte, no vizualizacija crtežom ipak zahtijeva vještost kako bismo vjerno prikazali objekt. Prevođenje crteža u slike jedan je od načina kojim se omogućuje stvaranje realističnih slika bez potrebe za posjedovanjem umjetničkih vještina ili stručnosti u području sinteze slika. Složenost tog problema iskazana je u činjenici da su prosječni ljudski crteži izrazito pojednostavljeni i ne prate strogo rubove predmeta, što znači da je vjerodostojno prikazivanje problema izrazito komplicirano, osobito ako uzmemo u obzir da bi slika prevedena iz crteža trebala poštivati namjeru umjetnika što je više moguće. Jedna od prvih tehnika prevođenja crteža u slike je "Sketch2Photo: Internet Image Montage" [9] nastala 2009. godine. Sketch2Photo stvara sliku na temelju jednostavnog crteža s oznakama tako da pronalazi fotografije pretražujući Internet te ih sastavlja u skladu s crtežom. Primjer montaže Sketch2Photo prikazan je na slici 4.1.

Slika 4.1: Primjer montaže SketchPhoto. Slika je preuzeta iz [9].

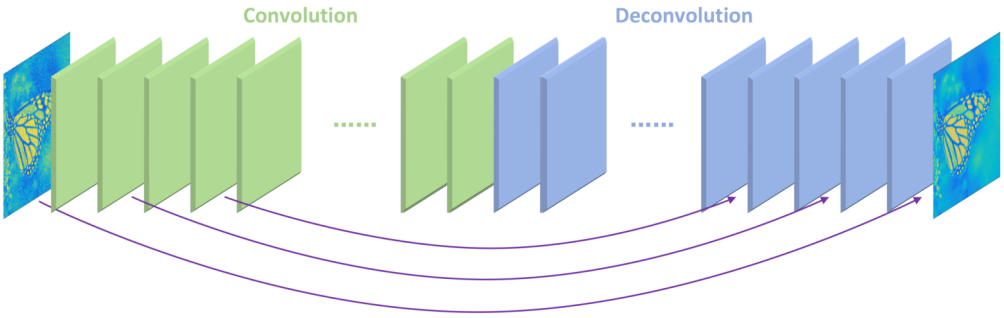
Prevođenje crteža u slike podskup je problema prevođenja iz slike u sliku. Općenito, problemi prevođenja iz slike u sliku oblikuju se kao semantička segmentacija ili gusta regresija. Ovakav se oblik ponaša prema izlaznom prostoru na način da se svaki izlazni piksel smatra uvjetno neovisnim od svih ostalih s obzirom na ulaznu sliku. U posljednje vrijeme se kao rješenje razmatraju uvjetni modeli koji postižu obećavajuće rezultate. U sklopu ovog završnog rada, proučava se uvjetni model pix2pix.



Slika 4.2: Primjer prevođenja iz slike u sliku. S lijeve strane prikazana je mapa oznaka zgrade. Mapa oznaka kodirana je bojama na temelju objekta koji predstavlja, poput zidova, vrata i prozora. S desne strane, nalazi se fotografija zgrade koja je generirana iz dane mape oznaka. Slika je preuzeta iz [10].

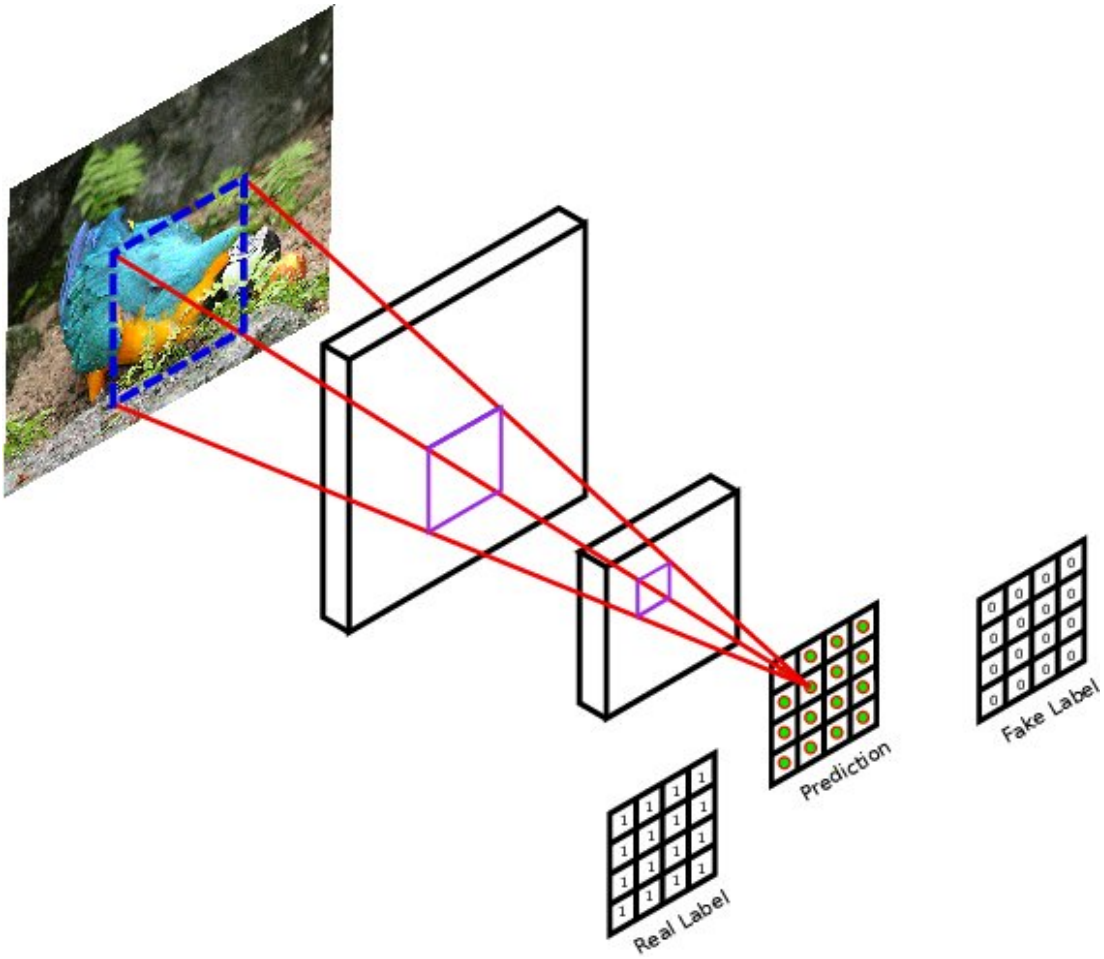
4.1. Arhitektura modela pix2pix

Glavna značajka problema prevođenja iz slike u sliku jest da preslikavaju ulazne podatke visoke razlučivosti na izlazne podatke visoke razlučivosti. Uz to, ulaz i izlaz razlikuju se po izgledu površine, no prikazuju jednake osnovne strukture. Na temelju ovih značajki oblikuje se arhitektura generatora. Mnoga prethodna rješenja ovog problema za generator koriste arhitekturu pisač-čitač (engl. encoder-decoder) u kojoj se ulazni slijed prosljeđuje kroz niz slojeva čiji se broj postupno smanjuje sve dok se ne dođe do posljednjeg sloja gdje se nad izgrađenom reprezentacijom ulaznog slijeda vrši naduzorkovanje sve dok se ne generira izlazni slijed. Za mnoge problema s prevođenjem slika poželjno je da se informacije dijeljene između ulaza i izlaza proslijede izravno preko mreže. Na primjer, ako želimo obojiti sliku, ulaz i izlaz trebaju dijeliti mjesta rubova na slici. Kako bi te informacije sačuvale, pix2pix koristi konvolucijsku mrežu U-net u kojoj dodajemo poveznice između svakog sloja pisača i i sloja čitača $n - i$, gdje je n ukupan broj slojeva. Svaka poveznica spaja sve kanale na sloju i s onima na sloju $n - i$.



Slika 4.3: Arhitektura mreže U-net s poveznicama koja se koristi za generator. Slika je preuzeta iz [11].

Model pix2pix sadrži dvije vrste generatora ovisne o dimenzijama ulaznih slika. Ulazne slike mogu biti dimenzija 128x128 i 256x256 piksela. Prvi dio mreže U-net, pisač, koristi korak dimenzija 2x2 kako bi vršio poduzorkovanje ulaza. Također, koristi se razrjeđivanje (engl. dropout) od 50% kao metoda regularizacije kako bismo izbjegli prenaučenost modela. Pix2pix za diskriminator koristi arhitekturu PatchGAN. PatchGAN diskriminator promatra samo strukturu na razini odlomaka veličine NxN piksela te donosi odluku o tome je li ona stvarna ili umjetno generirana. Diskriminator provodimo kroz sliku uzimajući odlomke za koje se donose klasifikacijske odluke sve dok se ne obradi slika te za konačnu odluku D uzimamo prosjek svih dobivenih odgovora.



Slika 4.4: Diskriminator arhitekture PatchGAN. Svaka vrijednost izlazne matrice predstavlja vjerojatnost da je odlomak slike stvaran ili umjetno generiran. Slika je preuzeta iz [12].

U pix2pix modelu, PatchGAN koristi odlomke dimenzija 70x70 piksela. Veličina receptivnog polja ovisna je o konfiguraciji modela i njegove dimenzije računaju se ovisno o dimenzijama izlaza, koraku i dimenzijama jezgre. Dimenzije jezgre su 4x4 piksela, a dimenzije koraka su 2x2. Za pix2pix model implemetirano je više konfiguracija arhitekture PatchGAN, uključujući konfiguraciju s receptivnim poljem dimenzija 1x1 piksela nazvanu PixelGAN i konfiguraciju ImageGAN s dimenzijama receptivnog polja 2x2 piksela. U praksi se pokazalo da su optimalne dimenzije 70x70, uzimajući u obzir performanse i kvalitetu slike.

4.2. Optimizacija i učenje modela pix2pix

Kako bismo optimizirali mrežu, slijedimo uobičajeni pristup: izmjenjujemo se između gradijentnog spusta na diskriminatoru D i gradijentnog spusta na generatoru G. Kao

što je opisano u poglavlju o generativnim suparničkim modelima, umjesto da učimo G da minimizira $log(1 - D(x, G(x, z))$, učimo ga da maksimizira $log(D(x, G(x, z)))$. Kako bismo usporili učenje diskriminatora u usporedbi s generatorom, funkciju cilja dijelimo s 2. Za optimizaciju koristimo stohastički gradijentni spust nad malim grupama i primijenjujemo algoritam Adam s propadajućom stopom učenja početne vrijednosti 0.0002 te parametrima $\beta 1 = 0.5$ i $\beta 2 = 0.999$.

5. Programska izvedba

5.1. Korištene tehnologije

Implementacija ovog modela napisana je u programskom jeziku Python [28] te prilagodba je parametara i algoritama definiranih u [2] na zadatak ovog završnog rada. Učenje modela izvršeno je u sklopu besplatnog servisa Google Colab [29] koji omogućuje interaktivnu uporabu bilježnica Jupyter Notebook [30], udaljenih repozitorija te učitavanje podataka s besplatnog oblaka Google Drive [31] uz korištenje udaljenih grafičkih i središnjih procesorskih jedinica. Zbog složenosti ovog zadatka, korištena je grafička jedinica NVIDIA Tesla K80 [32].

5.2. Korištene biblioteke

Glavna biblioteka korištena za implementaciju modela je biblioteka PyTorch [33], a za obradu podataka korištene su biblioteke OpenCV [34] i Pillow [35].

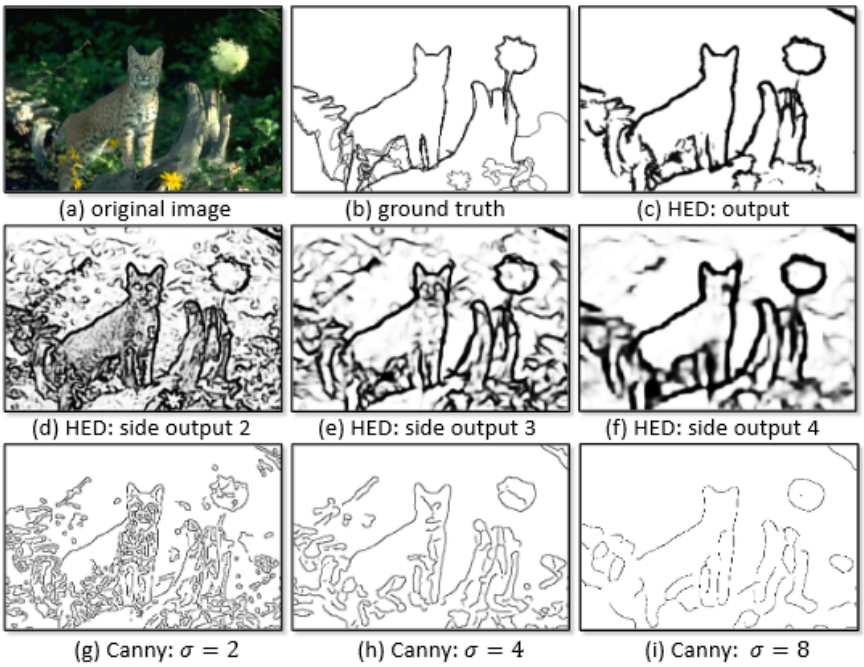
5.2.1. Biblioteka PyTorch

Biblioteka PyTorch je biblioteka otvorenog koda za strojno učenje. Razvio ju je Facebookov istraživački tim za umjetnu inteligenciju (FAIR) 2006. godine. Najčešće se primjenjuje u računalnom vidu i obradi prirodnog jezika. Njezine glavne prednosti su transparentno izvođenje na grafičkim jedinicama te automatsko računanje gradijenata po parametrima. Osnovni razred u biblioteci PyTorch je razred Tensor (torch.Tensor) koji služi za rukovanje matricama brojeva. Tenzori su slični razredu Array u biblioteci NumPy [36], no posjeduju dodatnu mogućnost izvršavanja na grafičkim procesorskim jedinicama koji koriste tehnologiju za paralelno programiranje CUDA [37]. Važni moduli koje PyTorch nudi su Autograd (torch.autograd), Optim (torch.optim) i nn (torch.nn). Modul Autograd koristi se za automatsko izračunavanje gradijenta, što je posebno korisno u algoritmu *backpropagation*. Optim je modul koji sadrži im-

plementacije raznih optimizacijskih algoritama, a modul nn koristi se za oblikovanje neuronskih mreža na višoj razini.

5.3. Skupovi podataka za učenje i testiranje modela

Skupovi za učenje i testiranje modela pix2pix sastoje se od slika dimenzija WxH piksela, gdje je H visina, a $W = 2H$ širina, podijeljenih u dva dijela: prvi dio, A, predstavlja domenu iz koje prevodimo, dok drugi dio, B, predstavlja kodomenu u koju učimo prevoditi. U sklopu problema prevođenja crteža u sliku, logično je kao domenu predložiti crteže, no učenje na crtežima izrazito je zahtjevno i dugotrajno zbog činjenice da oblici predmeta na crtežima često nisu dovoljno bliski njihovim oblicima u stvarnosti. Uz to, teško je nabaviti velik broj parova slika i odgovarajućih crteža za učenje modela. Kao alternativu crtežima, u sklopu ovog rada za učenje koristimo rubove objekata jer su dovoljno slični jednostavnim crtežima. Rubovi objekata dobiveni su korištenjem algoritma HED (engl. *Holistically-Nested Edge Detection*) koji se pokazao izrazito uspješnim u detekciji rubova.



Slika 5.1: Rubovi dobiveni korištenjem algoritma HED. Slika je preuzeta iz [13].

5.3.1. Skup podataka edges2shoes

Skup edges2shoes sastoji se od 50025 primjera slika cipela i njihovih rubova podijeljenih u skup za učenje, skup za testiranje i skup za provjeru. Edges2shoes nastao je na Sveučilištu Berkeley u Kaliforniji i jedan je od skupova za učenje modela pix2pix korištenih u radu [7]. Skup za učenje preuzet je sa službene stranice Sveučilišta Berkeley [38].

Slika 5.2: Primjer iz skupa edges2shoes.

5.3.2. Skup podataka dog

Skup dog sastoji se 320 primjera slika pasa i njihovih rubova podijeljenih u skup za učenje, skup za testiranje i skup za provjeru. Preuzet je sa udaljenog repozitorija ShibaGAN [39]. Ovaj skup se razlikuje od prošlog po tome što su rubovi slika puno detaljniji i sličniji naprednijim crtežima.

Slika 5.3: Primjer iz skupa dog.

6. Eksperimentalni rezultati

Za razliku od ostalih modela koji se uče funkcijom gubitka sve dok ne konvergira, u generativnim modelima uče se naizmjenično dvije mreže. Kako se generator učenjem poboljšava, diskriminatoru se performanse pogoršavaju zato što više ne može razlikovati prave i generirane slike što dovodi do tog da njegove povratne informacije postaju sve manje značajne. Zbog tog svojstva, za generativne modele ne postoji način da se objektivno procjeni napredak učenja ili kvaliteta modela temeljen samo na gubitku. Umjesto toga, razvile su se drugačije mjere za procjenu kvalitete generativnog modela. Neke od njih su ljudska procjena, mjera Inception Score te mjera Fréchet Inception Distance. Rezultati ovog rada evaluirani su na temelju mjere Fréchet Inception Distance.

6.1. Fréchet Inception Distance

Fréchet Inception Distance (FID) mjera je korištena za ocjenu kvalitete slika koje je stvorio generator. Za razliku od mjere Inception Score, koja računa statistiku izlaznih slika tako da pokušava klasificirati svaku generiranu sliku u njenu pripadajuću klasu, FID računa razliku između stvarne i generirane slike kao udaljenost između Gaussovih distribucija značajki stvarnih slika i generiranih slika. Rezultat toga je broj jednak ili veći od 0, gdje 0 znači da su generirane slike jednake ulaznim slikama, a svaki sljedeći veći broj predstavlja odmak od originalne slike te su generirane slike sve manje uvjerljive. Izraz za izračun metrike FID je sljedeći:

$$FID = |\mu - \mu_w|^2 + Tr(\sum + \sum_w -2(\sum \sum_w)^{1/2})) \tag{6.1}$$

gdje μ i μ_w predstavljaju vektore koji sadrže prosječne vrijednosti značajki generiranih i stvarnih slika, a $\sum$ i $\sum_w$ njihove matrice kovarijance. Operacija Tr izračunava zbroj elemenata dijagonale matrice.

6.2. Rezultati na skupu primjera dog

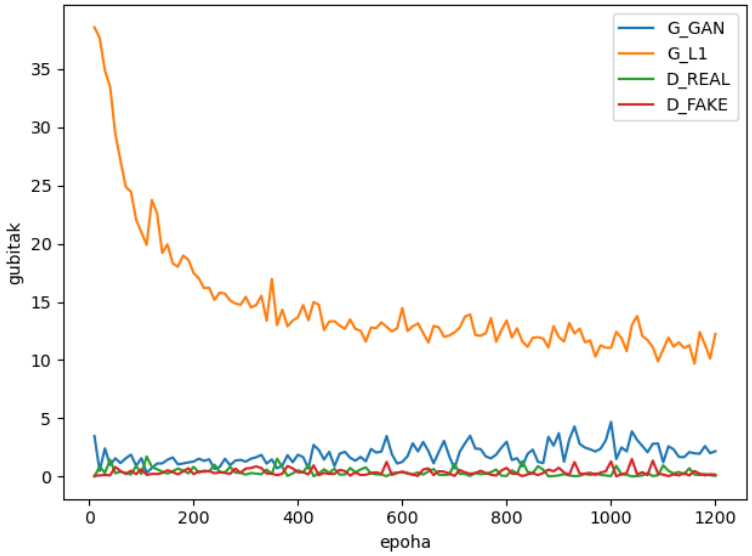
Učenje modela na skupu dog izveli smo u nekoliko navrata s različitim veličinama grupa za učenje kako bismo usporedili performanse. Originalne dimenzije ulaznih slika su 768x384 piksela, no u svrhe učenja dimenzije se na ulazu smanjuju na 512x256 piksela. Skup za učenje sadrži 260 slika, a skupovi za testiranje i provjeru 30 slika. Učenje modela trajalo je 1200 epoha. Kao arhitektura diskriminatora korištena je mreža U-net, a za generator smo izabrali mrežu PatchGAN. Diskriminator za funkciju gubitka koristi binarnu unakrsnu entropiju (VanillaGAN). Za optimizaciju postupka učenja korišten je algoritam Adam s propadajućom stopom učenja početne vrijednosti 0.0002 te parametrima $\beta 1 = 0.5$ i $\beta 2 = 0.999$.

Slika 6.1: Ulaz modela - stvarna fotografija

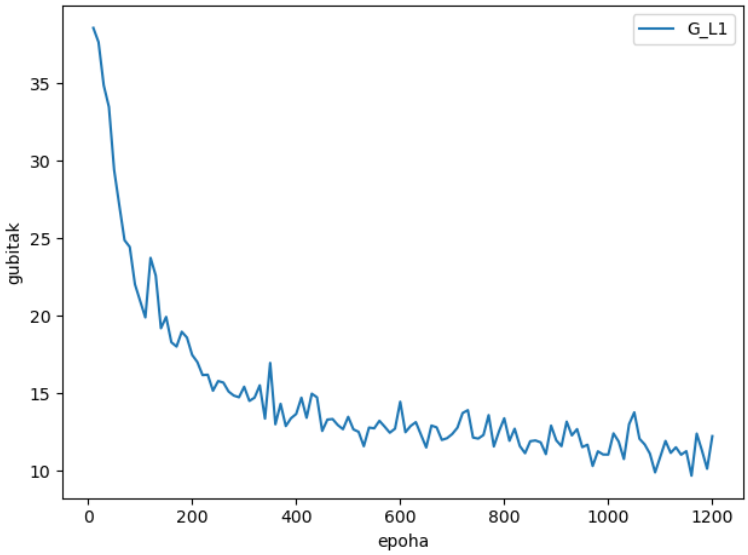
Slika 6.2: Ulaz modela - rubovi

Slika 6.3: Izlaz modela - slika generirana iz rubova

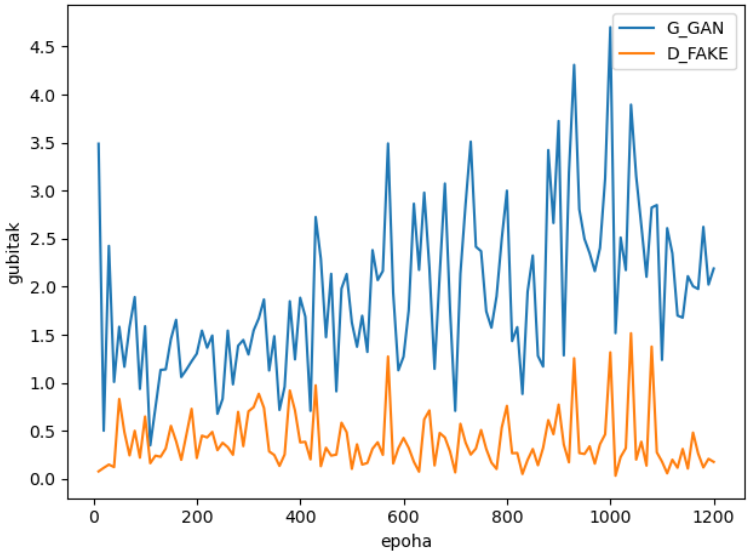
Pokazalo se da najbolje rezultate dobijemo koristeći male grupe za učenje. Model učen na većim grupama za učenje slabije je otporan na šum i dobiva lošije rezultate u metrici FID. Odnos veličine grupe za učenje na rezultate prikazan je u tablici 6.1. Drugi način evaluacije modela je praćenje ponašanja gubitaka generatora i diskriminatora. Gubitak L1 predstavlja razliku između početne slike i generirane slike te se u našem slučaju kroz epohe njegova vrijednost sa 44.15 smanjila na 14.28. Na slici 6.5 prikazano je kretanje vrijednosti ovog gubitka tijekom učenja modela.



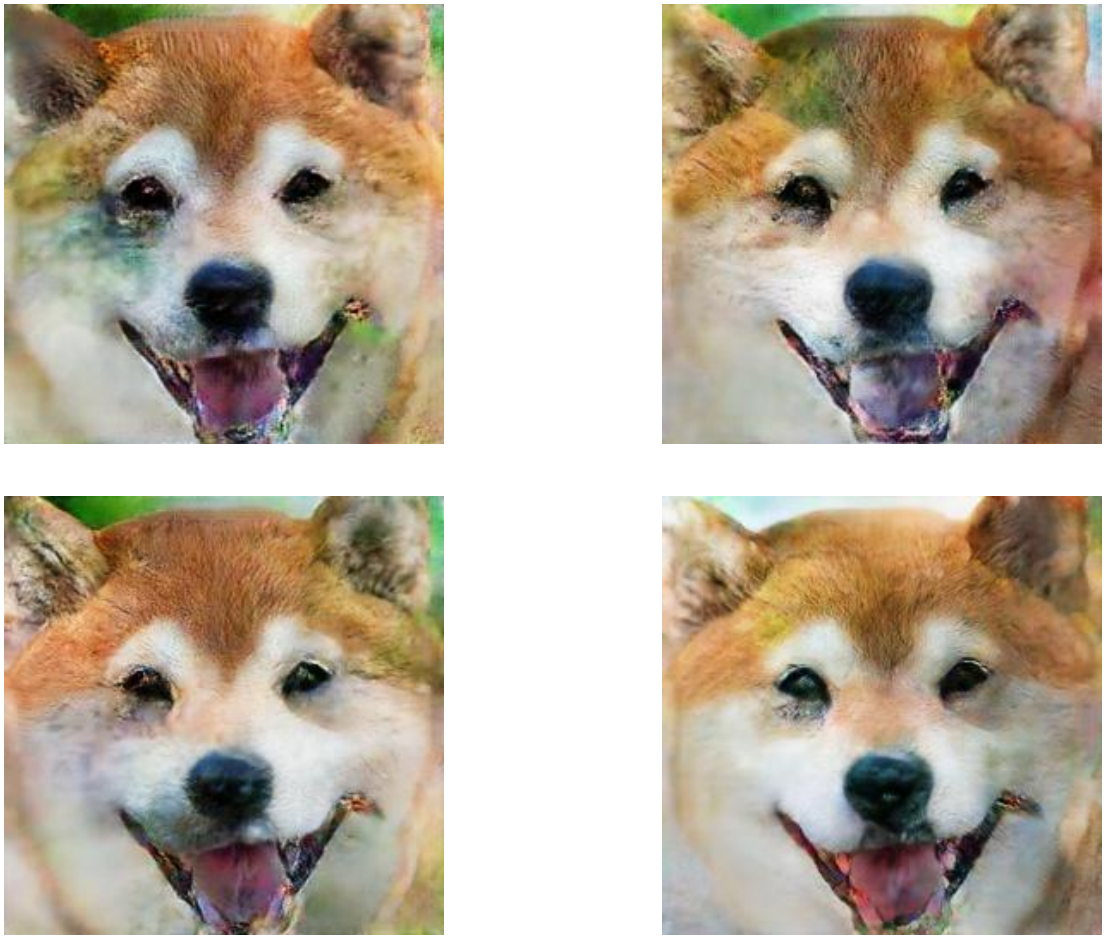
Slika 6.4: Prikaz vrijednosti svih gubitaka kroz epohe. G_GAN predstavlja gubitak generatora, G_L1 predstavlja gubitak L1, D_REAL je gubitak diskriminatora u klasifikaciji primjera kao stvarnih, a D_FAKE je gubitak diskriminatora u klasifikaciji primjera kao lažnih.



Slika 6.5: Prikaz vrijednosti gubitka generatora kroz epohe. Gubitak L1 koristi se u učenju generatora kako bi generirane slike bile što sličnije slikama na ulazu. Njegova bi vrijednost tijekom učenja trebala težiti nuli.



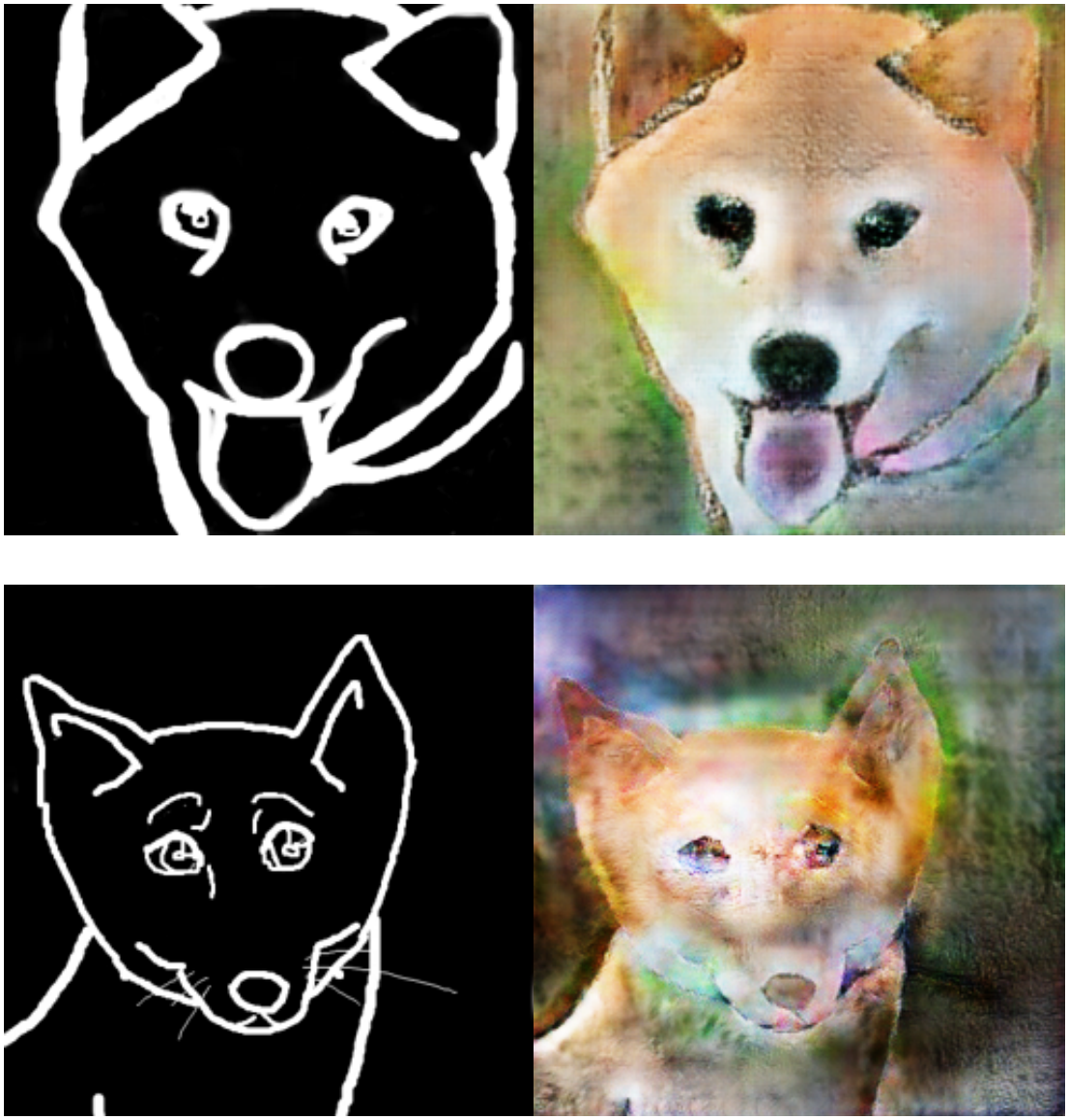
Slika 6.6: Prikaz odnosa gubitka generatora i gubitka diskriminatora u klasifikaciji primjera kao lažnih. U našem slučaju, vrijednosti gubitaka kreću se između 0 i 4.47.



Slika 6.7: Prikaz izlaza za različite veličine grupa za učenje. Redom: 32, 16, 8, 4. Brojčani rezultati prikazani su u tablici 6.1.

Veličina grupe za učenje	FID
32	149.493
16	146.490
8	133.338
4	124.092

Tablica 6.1: Odnos veličine grupe za učenje i rezultata metrike FID na skupu za provjeru dog. Vidimo da smanjenje grupe za učenje utječe pozitivno na rezultat metrike FID.



Slika 6.8: Primjeri prevođenja neviđenih crteža u sliku. Crteže sam nacrtala koristeći program Paint.

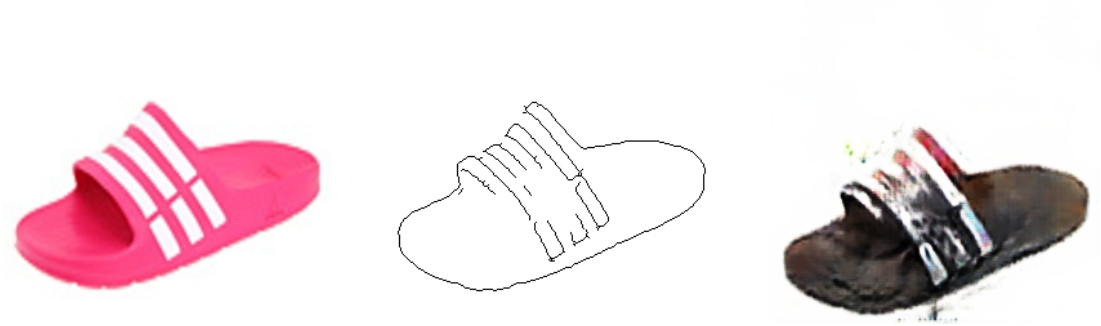
6.3. Rezultati na skupu primjera edges2shoes

Učenje modela na skupu edges2shoes izvelo se u dva navrata s različitim arhitekturama modela kako bismo usporedili performanse. Dimenzije ulaza u ovom skupu su 512x256 piksela. Cijeli skup sadrži oko 50 tisuća slika cipela i njihovih rubova. Zbog opsežnosti skupa, učenje modela trajalo je 30 epoha. Načini izračuna funkcije gubitka diskriminatora korišteni za usporedbu su VanillaGAN (unakrsna entropija) i lsGAN (engl. *Least Squares Generative Adversarial Networks*), opisan u [41]. Za optimizaciju postupka učenja korišten je algoritam Adam s propadajućom stopom učenja

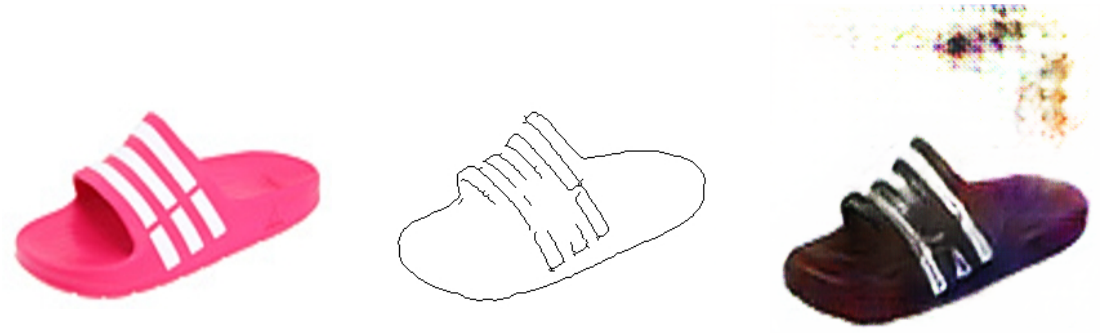
početne vrijednosti 0.0002 te parametrima $\beta 1 = 0.5$ i $\beta 2 = 0.999$. Pri vrednovanju ovog modela, vanillaGAN pokazao se boljim izborom. Iako rezultati nisu pretjerano različiti, korištenjem VanillaGAN-a dobivamo puno bolji rezultat pri izračunu metrike FID. Rezultati su prikazani u tablici 6.2.

Funkcija gubitka	FID
VanillaGAN	172.910
lsGAN	224.844

Tablica 6.2: Odnos načina izračuna funkcije gubitka i rezultata FID na skupu za provjeru edges2shoes.

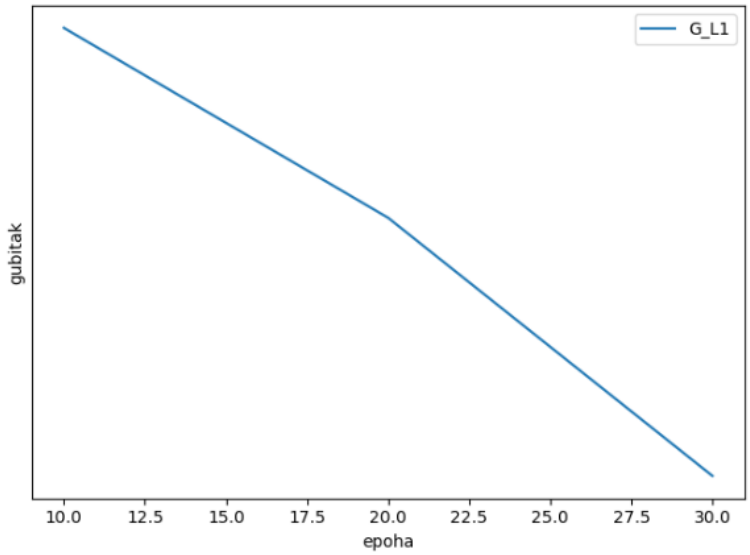


Slika 6.9: Primjer rezultata na skupu za provjeru koji se sastoji od rubova predmeta dobivenih korištenjem lsGAN.

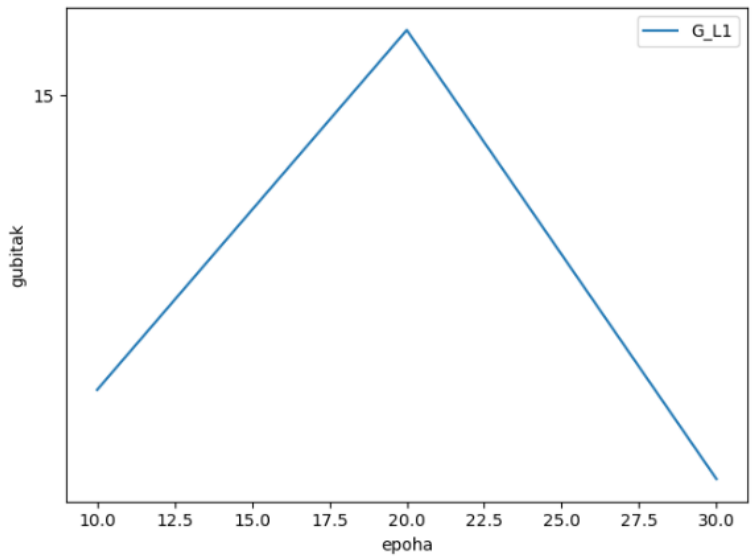


Slika 6.10: Primjer rezultata na skupu za provjeru koji se sastoji od rubova predmeta dobivenih korištenjem VanillaGAN.

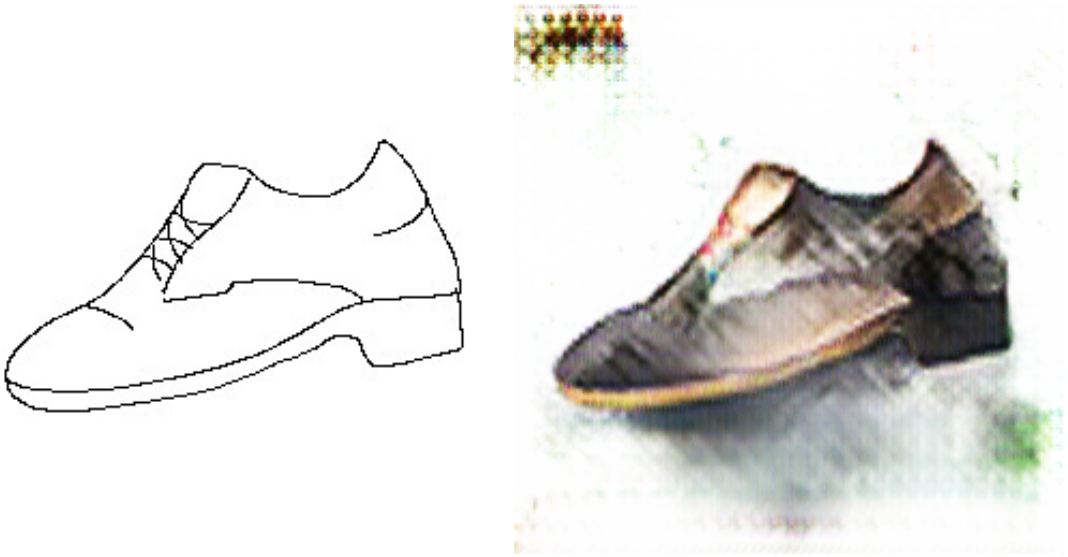
Iz slika 6.9 i 6.10 možemo primijetiti da korištenjem VanillaGAN i lsGAN dobivamo jednako dobro detektirane granice predmeta, no model koji koristi VanillaGAN proizvodi nešto uvjerljivije rezultate što se tiče teksture i boje predmeta.



Slika 6.11: Gubitak L1 u modelu s VanillaGAN načinom izračuna funkcije gubitka. U ovom slučaju, gubitak L1 kontinuirano teži prema nuli.



Slika 6.12: Gubitak L1 u modelu s lsGAN načinom izračuna funkcije gubitka. Nakon dvadesete epohe stopa učenja počinje propadati što pomaže u konvergenciji te zbog tog gubitak L1 teži nuli, dok je prije toga težio u beskonačnost.



Slika 6.13: Primjer prevođenja neviđenog crteža u sliku. Crtež sam nacrtala koristeći program Paint.

Pokazalo se da uvjerljivije rezultate dobivamo prilažući na ulaz crtež s manje detalja jer je on sličniji podacima na kojima smo učili model.

Slika 6.14: Primjer prevođenja neviđenog crteža s manje detalja u sliku. Crtež sam nacrtala koristeći program Paint.

Slika 6.15: Primjer prevođenja neviđenog crteža s više detalja u sliku. Crtež sam nacrtala koristeći program Paint.

6.4. Osvrt na rezultate

Iz eksperimenata zaključujem da model pix2pix s konfiguracijom opisanom u ovom poglavlju daje dobre rezultate u detekciji oblika predmeta te dobre slike generira iz crteža s manje detalja. Najbolje rezultate dobili smo korištenjem VanillaGAN načina za izračun funkcije gubitka, dok smo uz lsGAN dobili nešto lošije slike. Ovakvi rezultati uspješno prikazuju rad modela. Kako bismo dobili rezultate prikladne za stvarne primjene bilo bi potrebno koristiti veći skup podataka s detaljnijim rubovima predmeta ili skup podataka koji se sastoji od crteža i njihovih odgovarajućih slika. Za ovaj pristup potrebno je duže učenje koje na platformi Google Colab nije moguće provesti pa bi trebalo razmotriti drugačiju platformu ili učiti na računalu s jakom grafičkom jedinicom. Problem prevođenja crteža tada bi najviše bio koristan u dizajnu i arhitekturi te bi svoju primjenu našao i u forenzici.

7. Zaključak

U zadnjih nekoliko godina, dogodio se velik pomak u području nenadziranog učenja, posebno generativnih modela. Generativni modeli pokazali su se prilagodljivima mnogim problemima - pa tako i prevođenju crteža u slike. Prevođenje crteža u slike zanimljiv je problem primjenjiv u raznim područjima, od forenzike do umjetnosti, no relativno je i težak uzimajući u obzir da u njemu trebamo naći ravnotežu između namjere umjetnika i realističnosti generirane slike. Dobiveni rezultati pokazuju da učenje na rubovima radi dobro samo ako se bavimo prevođenjem crteža s manje detalja. Danas, osim modela pix2pix, koji je obrađen u ovom završnom radu, postoje i napredniji modeli poput SketchyGAN [40] koji pokazuju nešto bolje rezultate. U budućnosti, bilo bi potrebno istražiti druge modele i konfiguracije uz duže vrijeme učenja.

LITERATURA

[1] B. Dalbelo Bašić, M. Čupić, J. Šnajder. *Umjetne neuronske mreže*, https://www.fer.unizg.hr/_download/repository/UmjetneNeuronskeMreze.pdf

[2] J. Y. Zhu. *Implementacija modela pix2pix i CycleGAN*, URL: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/commits?author=junyanz

[3] M. Mirza, S. Osindero. *Conditional Generative Adversarial Nets*, 2014.

[4] I J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozairy, A.Courville, Y. Bengioz. *Generative Adversarial Nets*, 2014.

[5] I. Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*, 2017.

[6] Y. Pang, J. Lin, T. Qin, Z. Chen. *Image-to-Image Translation: Methods and Applications*, 2021.

[7] P. Isola, J. Y. Zhu, T. Zhou, A. A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*, 2018.

[8] T. Park, R. Zhang, A. A. Efros, J. Y. Zhu. *Contrastive Learning for Unpaired Image-to-Image Translation*, 2018.

[9] T. Chen, M. M. Chen, P. Tan, A. Shamir, S. M. Hu. *Sketch2Photo: Internet Image Montage*, 2009.

[10] C. Hesse *Image-to-Image Demo*, URL: https://affinelayer.com/pixsrv/

[11] X. J. Mao, C. Shen, Y. B. Yang. *Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections*, 2016.

[12] U. Demir, G. Unal. *Patch-Based Image Inpainting with Generative Adversarial Networks*, 2018.

[13] S. Xie. *Holistically-Nested Edge Detection*, 2015.

[14] W. Chen, J. Hays. *SketchyGAN: Towards Diverse and Realistic Sketch to Image Synthesis*, 2018.

[15] S. Kaji, S. Kida. *Overview of image-to-image translation using deep neural networks: denoising, super-resolution, modality-conversion, and reconstruction in medical imaging*, 2019.

[16] L. Fan, J. Krone, S. Woolf. *Sketch to Image Translation using GANs*

[17] K. Fugošić. *Prevođenje iz slike u sliku korištenjem uvjetnih suparničkih modela*, FER, 2017.

[18] L. Ivković. *Prijenos umjetničkog stila optimiranjem konvolucijskog modela*, seminarski rad, FER, 2019.

[19] M. Jelavić. *Polunadzirana klasifikacija rukom pisanih znakova generativnim suparničkim modelima*, završni rad, FER, 2017.

[20] F. Zelić. *Izlučivanje slikovnih reprezentacija generativnim suparničkim modelima*, diplomski rad, FER, 2018.

[21] M. Čupić. *Optimizacija parametara modela*, Duboko učenje, FER, URL: `http://www.zemris.fer.hr/~ssegvic/du/du3optimization.pdf`

[22] S. Šegvić, J. Krapac. *Konvolucijski modeli*, Duboko učenje, FER, URL: `http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf`

[23] T. Hrkač. *Generativni suparnički modeli*, FER

[24] *Activation Functions in Neural Networks* `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9`

[25] *Back-Propagation simplified*, `https://towardsdatascience.com/back-propagation-simplified-218430e21ad0`

[26] A. H. Reynolds. *Convolutional Neural Networks (CNNs)* `https://anhreynolds.com/blogs/cnn.html`

[27] *Overview of GAN Structure, Generative Adversarial Networks*, URL: https://developers.google.com/machine-learning/gan/gan_structure

[28] *Službena web-stranica programskog jezika Python*, URL: https://www.python.org/

[29] *Servis Google Colab*, URL: https://colab.research.google.com

[30] *Službena web-stranica interaktivnih bilježnica Jupyter Notebook*, URL: https://ipython.org/notebook.html

[31] *Oblak Google Drive*, URL: https://drive.google.com/drive/

[32] *Službena web-stranica grafičke jedinice NVIDIA Tesla K80*, URL: https://www.nvidia.com/en-gb/data-center/tesla-k80/

[33] *Službena web-stranica biblioteke PyTorch*, URL: https://pytorch.org/

[34] *Službena web-stranica biblioteke OpenCV*, URL: https://opencv.org/

[35] *Službena web-stranica biblioteke Pillow*, URL: https://python-pillow.org/

[36] *Službena web-stranica biblioteke Numpy*, URL: https://numpy.org/

[37] *Službena web-stranica tehnologije CUDA*, URL: https://developer.nvidia.com/cuda-zone

[38] *Službena web-stranica Sveučilišta Berkeley*, URL: https://people.eecs.berkeley.edu/~tinghuiz/projects/pix2pix/datasets/

[39] *Udaljeni repozitorij ShibaGAN*, URL: https://github.com/tony23545/Sketchy-ShibaGAN

[40] W. Chen, J. Hays *SketchyGAN*, URL: https://arxiv.org/abs/1801.02753

[41] X. Mao et al. *Least Squares Generative Adversarial Networks*, URL: https://arxiv.org/pdf/1611.04076

Učenje modela za prevođenje crteža u slike

Sažetak

Ovaj završni rad obuhvaća teoretsku razradu osnova dubokog učenja, poput neuronskih mreža te njihovih arhitektura i postojećih algoritama. Razrađuje se pojam generativnog suparničkog modela i predstavlja problem prevođenja crteža u sliku, kao i primjena generativnih modela u tom području. Opisana je arhitektura modela pix2pix te prikazani su eksperimentalni rezultati nekoliko konfiguracija modela. Spomenuta je korištena programska potpora.

Ključne riječi: strojno učenje; duboko učenje, neuronske mreže, računalni vid, generativni modeli, generativne suparničke mreže, prevođenje iz slike u sliku, crteži

Training models for translating sketches into images

Abstract

This thesis includes a theoretical elaboration of the basics of deep learning, such as neural networks, their architectures and existing algorithms. It elaborates the concept of generative adversarial networks and represents the problem of translating sketches into images, as well as the applications of generative models in this area. The architecture of the pix2pix model is described and the experimental results of several model configurations are presented. Used software support is mentioned.

Keywords: machine learning, deep learning, computer vision, neural networks, generative models, generative adversarial networks, image-to-image translation, sketches