Generativne suparničke mreže i primjene

Peroš, Mate

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Mate Peroš

GENERATIVNE SUPARNIČKE MREŽE I PRIMJENE

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Zvonimir Bujanović

Zagreb, veljača, 2022.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem mentoru izv. prof. dr. sc. Zvonimiru Bujanoviću na pomoći pri izradi ovog rada, prijateljima na nezaboravnim uspomenama tijekom školovanja, a posebno svojoj obitelji bez koje sve ovo ne bi bilo moguće.

Sadržaj

Uvod

Povećanje razlučivosti (eng. *super-resolution*) česta je tema istraživanja u području računalnog vida. Problem povećanja razlučivosti je proces dobivanja slike visoke razlučivosti iz odgovarajuće slike niske razlučivosti i pronalazi veliku primjenu u satelitskom snimanju, obradi medicinskih prikaza, analizi fotografija teksta te biometrijskom prepoznavanju.

Konvolucijske neuronske mreže smatraju se najsuvremenijom tehnologijom u području povećanja razlučivosti slike iz razloga što pružaju najbolje rezultate na raznolikim skupovima podataka. Mana takvih modela su vizualno nezadovoljavajuće slike zbog zamućivanja sitnih detalja.

U ovom radu predlaže se model koji se zasniva na suparničkom treniranju, preciznije generativnim suparničkim mrežama. Generativni suparnički modeli pokazali su se kao kvalitetna zamjena za duboke konvolucijske modele jer postižu finije detalje na problemima povećanja razlučivosti slike bez potrebe za jako velikim brojem konvolucijskih slojeva.

Model generativnih suparničkih mreža sastoji se od dvije neuronske mreže, generatorske i diskriminatorske. Generatorska mreža ima zadatak naučiti distribuciju ulaznih podataka, gdje provođenjem podatka kroz slojeve mreže kao izlaz dobivamo uvjerljivu sliku. Diskriminatorska mreža ima zadatak odrediti je li primjer na njenom ulazu stvaran ili generiran.

Poglavlje 1 daje uvod u neuronske mreže i ključne pojmove u tom polju, poglavlje 2 obrađuje generativne suparničke mreže dok primjenu generativnih suparničkih mreža na problemima povećanja razlučivosti slike prikazujemo u poglavlju 3.

Poglavlje 1

Neuronske mreže

1.1 Duboko učenje

Duboko učenje (eng. *deep learning*) je grana strojnog učenja temeljena na algoritmima zvanim umjetne neuronske mreže, koji su inspirirani strukturom i funkcijom mozga. Modeli dubokog učenja pronašli su primjenu u područjima računalnog vida, prepoznavanja govora, obrade prirodnog jezika, dizajniranja lijekova te analize medicinskih prikaza gdje su postigli rezultate usporedive s ljudskim performansama, a često ih i prestigli.

Model neuronske mreže sastoji se od velikog broja umjetnih neurona posloženih u slojeve. Pridjev „duboko" iz naziva duboko učenje proizlazi upravo iz arhitekture modela neuronske mreže, gdje dubina modela predstavlja broj slojeva u modelu.

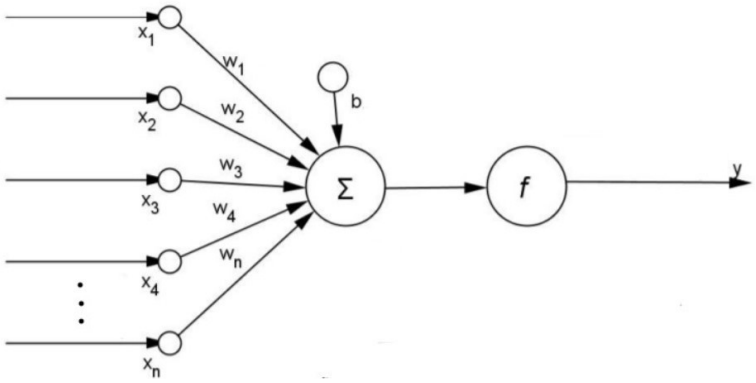
1.2 Umjetni neuron

Prije nego što definiramo neuronsku mrežu, potrebno je definirati njenu osnovnu građevnu jedinicu - umjetni neuron. Osnovni model neurona sastoji se od ulaza u neuron (vektor x) koji se skalarno množi s vektorom težina w te zbraja s tzv. *biasom* b, a na zbroj konačno djeluje aktivacijska funkcija f kao što je prikazano na slici 1.1, odnosno izlaz neurona glasi

$$f\left(\sum_{i=1}^{n} x_i \cdot w_i + b\right)$$

gdje x_i predstavlja ulaz u neuron, w_i predstavlja težinski faktor, b predstavlja bias, a f predstavlja aktivacijsku funkciju. Parametri w_i i b mogu se postaviti na proizvoljnu vrijednost, a izbor vrijednosti bit će cilj treniranja neuronske mreže.

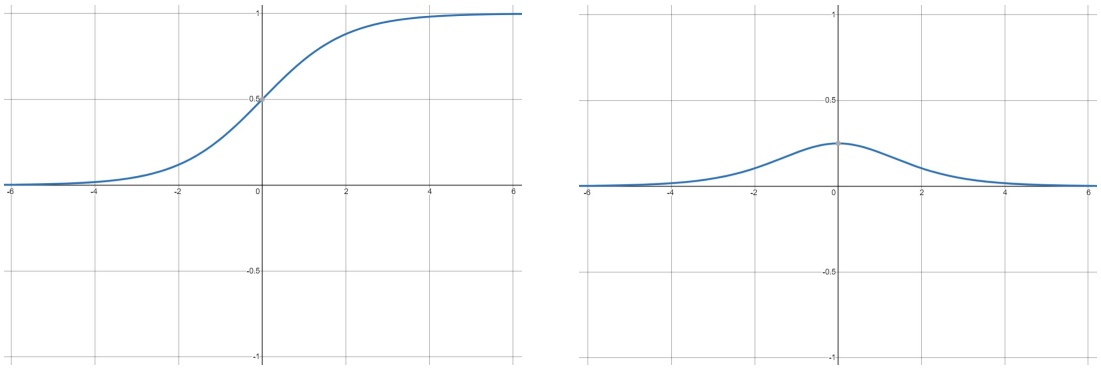
Kako je osnovni model neurona kroz vrijeme ostao isti, izbor aktivacijske funkcije ključna je odluka pri konstruiranju neurona. U modernom dubokom učenju najčešće korištene aktivacijske funkcije su sigmoidalna, tangens hiperbolni i funkcija ReLU.



Slika 1.1: Model umjetnog neurona

Sigmoidalna funkcija definirana je jednadžbom (1.1) te njen graf možemo vidjeti na slici 1.2a, a graf derivacije na slici 1.2b.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.1}$$



(a) Sigmoidalna funkcija

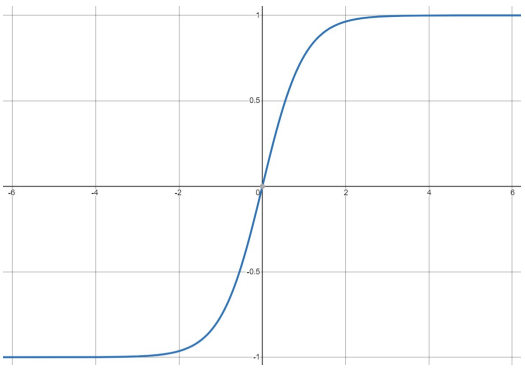
(b) Derivacija sigmoidalne funkcije

Iz grafa primjetimo da se izlaz sigmoidalne funkcije može interpretirati kao vjerojatnost, primjerice kod problema klasifikacije. Međutim, promatrajući prvu derivaciju zamjećujemo manu sigmoidalne funkcije. Kako se pri treniranju svaka težina neuronske mreže ažurira proporcionalno parcijalnoj derivaciji funkcije pogreške u ovisnosti o toj težini, mali nagib na repovima derivacije može dovesti do problema nestajućeg gradijenta, odnosno, kako propagiramo gradijent kroz mrežu, gradijent će postati toliko malen da težina koju ažuriramo gotovo pa neće promijeniti vrijednost.

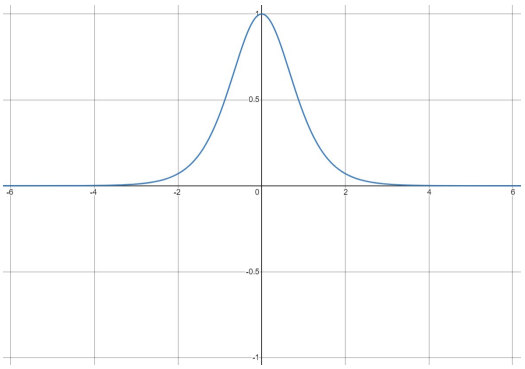
Alternativa za sigmoidalnu funkciju je funkcija tangens hiperbolni. Tangens hiperbolni definiran je jednadžbom (1.2) te njen graf možemo vidjeti na slici 1.3a, a graf derivacije na slici 1.3b.

$$f(x) = \tanh x \tag{1.2}$$

Promatrajući graf zamjećujemo da za razliku od sigmoidalne funkcije, tangens hiperbolni centriran je oko nule te ima po modulu jače gradijente što su poželjna svojstva pri treniranju mreže.



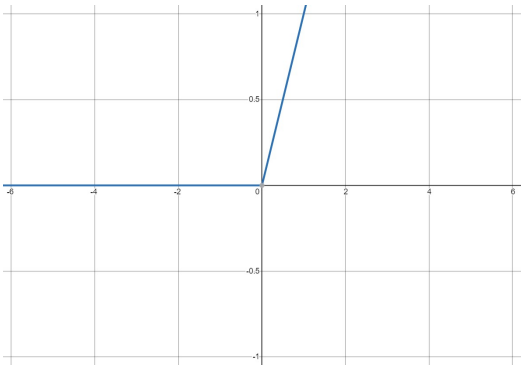
(a) Tangens hiperbolna funkcija



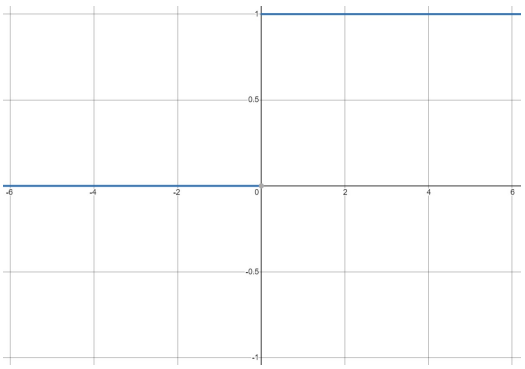
(b) Derivacija tangens hiperbolne funkcije

Trenutno najpopularnija aktivacijska funkcija pri treniranju dubokih neuronskih mreža je funkcija *ReLU*, dok se sigmoidalna i tangens hiperbolni najčešće koriste u izlaznom sloju. Funkcija ReLU definirana je jednadžbom (1.3) te njen graf možemo vidjeti na slici 1.4a, a graf derivacije na slici 1.4b.

$$f(x) = \max(0, x) \tag{1.3}$$



(a) Funkcija ReLU



(b) Derivacija funkcije ReLU

ReLU se vrlo efikasno računa, kao i njen gradijent. Nedostatci funkcije ReLU su necentriranost oko nule, nediferencijabilnost u nuli i problem umirućih neurona. Diferencijabilnost u nuli rješavamo dodefiniranjem gradijenta u nuli (najčešće u nulu). Do problema umirućih neurona najčešće dolazi kada tijekom treniranja težine mreže poprime velike negativne vrijednosti pa ulaz u ReLU aktivaciju postane negativan broj, odnosno, izlaz nula. Kako problem leži u izlazu nula, problem rješavamo uvođenjem funkcije *Leaky ReLU* definirane jednadžbom (1.4). Konstanta a je fiksna i najčešće se uzima iz intervala $\langle 0, 1\rangle$.

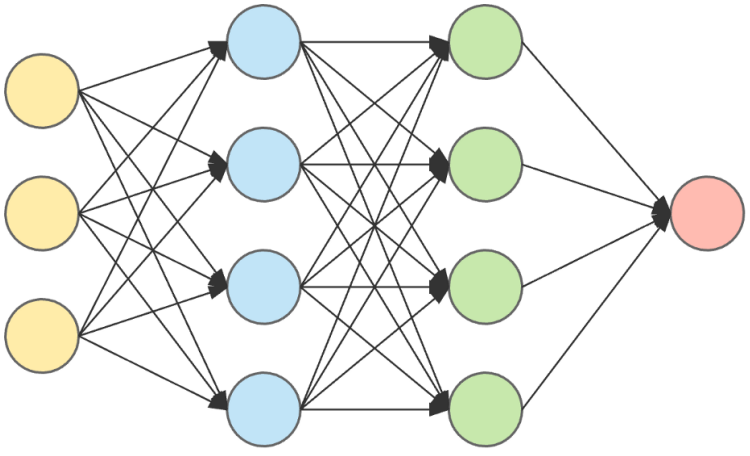
$$f(x) = \begin{cases} ax, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{1.4}$$

Varijanta funkcije Leaky ReLU u kojoj je konstanta a parametar koji učimo zovemo parametrizirana ReLU (tzv. PReLU).

1.3 Unaprijedne neuronske mreže

Motivacija za nastanak neuronskih mreža bila je stvaranje sustava koji može učiti. Neuronske mreže najbolje je interpetirati kao funkcije aproksimacije, gdje za dani ulazni podatak x aproksimiramo neki izlazni podatak y.

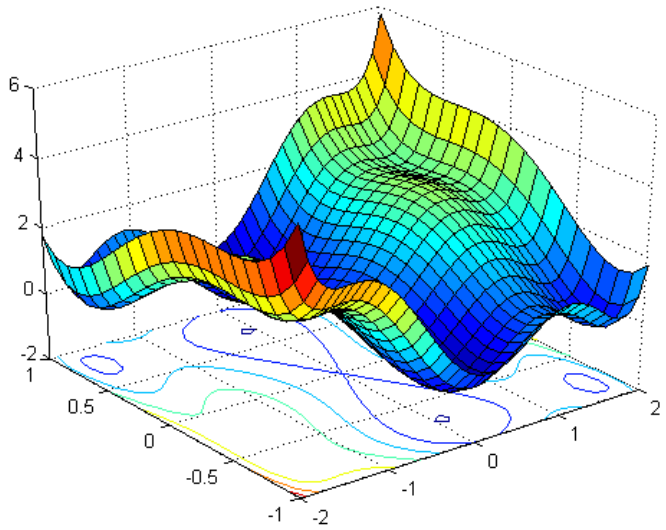
Unaprijedna neuronska mreža (eng. *feedforward network*) temelji se na skupu povezanih umjetnih neurona, organiziranih u slojeve kao na slici 1.5. Svaka neuronska mreža sastoji se od točno jednog ulaznog i izlaznog sloja te potencijalno velikog broja skrivenih slojeva. Ulazni sloj prima značajke ulaznog podatka kojeg obrađujemo te ih prosljeđuje skrivenim slojevima sve do izlaznog sloja.



Slika 1.5: Model neuronske mreže s dva skrivena sloja.

Najjednostavniji model neuronske mreže je potpuno povezana neuronska mreža (eng. *fully connected neural network*). U takvom modelu ulazni sloj predstavlja značajke podatka kojeg obrađujemo i u njemu se ne vrši nikakva obrada, već se podaci prosljeđuju prvom skrivenom sloju. Slojevi modela su potpuno povezani, odnosno, svaki neuron iz prethodnog sloja je povezan sa svakim neuronom u idućem sloju. Konačno, izlazni sloj daje obrađene podatke, aproksimaciju željenog izlaza našim modelom. Slika 1.5 osim što predstavlja unaprijednu, predstavlja i potpuno povezanu neuronsku mrežu.

Rad neuronske mreže evaluiramo funkcijom pogreške, odnosno računajući mjeru nepodudaranja između izlaza iz mreže i stvarnog izlaza podatka trening skupa. Za vrijeme treniranja neuronske mreže, cilj je minimizirati funkciju pogreške (eng. *loss function*) koja ovisi o parametrima zadane mreže tj. težinama i biasu na svakom umjetnom neuronu.



Slika 1.6: Primjer grafa funkcije pogreške.

Često korištena funkcija pogreške kod problema regresije je srednja kvadratna pogreška (eng. *mean squared error*) definirana jednadžbom

$$MSE(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{1.5}$$

gdje y predstavlja vektor stvarnih vrijednosti y_i, a $\hat{y}$ vektor vrijednosti $\hat{y}_i$ dobivenih provlačenjem svakog od n ulaznih podataka x_i vektora x kroz mrežu. Preciznije, $\hat{y} = M(x;\theta)$, gdje M predstavlja neuronsku mrežu, a $\theta = (W, b)$ uređeni par svih težina W i biasa b mreže.

Kod problema klasifikacije (razvrstavanja ulaznog podatka u jednu od n klasa), preferira se korištenje funkcije unakrsne entropije (eng. *cross entropy*) definirane jednadžbom

$$CE(\hat{y}, y) = -\sum_{i=1}^{n} y_i \log \hat{y}_i, \tag{1.6}$$

gdje n predstavlja broj klasa.

Snaga neuronskih mreža leži u nelinearnosti aktivacijskih funkcija čime se postiže sposobnost modeliranja funkcija vrlo kompleksnih preslikavanja ulaza u izlaz. Kada bi aktivacijske funkcije bile linearne, tada bi i izlaz modela bio linearna kombinacija značajki ulaznog podatka. Broj skrivenih slojeva kao i broj neurona u skrivenim slojevima ovisi o zadanom problemu, a kroz skrivene slojeve neuronska mreža uči različite reprezentacije ulaznih podataka s ciljem što uspješnijeg rješavanja problema. Unaprijedna neuronska mreža s ReLU aktivacijama i dovoljno slojeva, od kojih se svaki sastoji od $n + 4$ neurona, sposobna je aproksimirati svaku Lebesgue integrabilnu funkciju $f : \mathbb{R}^n \to \mathbb{R}$ do proizvoljne preciznosti s obzirom na L^1 udaljenost [12], što ih čini univerzalnim funkcijskim aproksimatorom.

Gradijentni spust

Pri učenju parametara skrivenih slojeva neuronska mreža koristi gradijentni spust (eng. *gradient descent*) koji računamo algoritmom unazadne propagacije (eng. *backpropagation*). Gradijentni spust je iterativni optimizacijski algoritam za traženje lokalnog minimuma diferencijabilne funkcije. Kako je gradijent zapravo vektor smjera najbržeg rasta funkcije, ideja gradijentnog spusta je napraviti korak u smjeru suprotnom od gradijenta u svakom koraku, odnosno formalno

$$V_{n+1} = V_n - \eta \nabla F(V_n), \tag{1.7}$$

gdje V_n predstavlja vektor vrijednosti parametara funkcije u n-toj iteraciji, F diferencijabilnu funkciju kojoj tražimo lokalni minimum, a η skalar koji određuje veličinu koraka.

U kontekstu dubokog učenja koristimo varijantu gradijentnog spusta kojim tražimo minimum funkcije pogreške, odnosno pokušavamo minimizirati pogrešku neuronske mreže. Kako je izvršavanje gradijentnog spusta samo na jednom primjeru nestabilno, gradijentni spust u pravilu izvršavamo na malim podskupovima podataka. Takvu varijantu gradijentnog spusta zovemo gradijentni spust s mini-grupama (tzv. stohastički gradijentni spust). Formalno, neka je C funkcija pogreške, θ vektor vrijednosti svih parametara neuronske mreže, $B = \{(x_i, y_i) \mid i = 1, \ldots, n\}$ slučajno odabran podskup (mini-grupa) trening podataka gdje je x_i ulaz u model, a y_i stvarni („ciljani") izlaz. Tada jednadžba gradijentnog spusta s

mini-grupama glasi

$$\theta \leftarrow \theta - \eta \nabla_\theta \left(\frac{1}{n} \sum_{i=1}^{n} C(y_i, x_i, \theta) \right). \tag{1.8}$$

U kontekstu dubokog učenja, η nazivamo stopom učenja (eng. *learning rate*). U svakoj iteraciji slučajno se odabire nova mini-grupa dok se ne iskoristi cijeli trening skup. Prolaz po cijelom trening skupu nazovamo epoha. Postupak optimizacije neuronske mreže izvršavamo sve do zadovoljavanja zadanih kriterija konvergencije ili do isteka zadanog maksimalnog broja epoha.

Algoritam unazadne propagacije

Problem računanja gradijenta u neuronskim mrežama rješava se uporabom algoritma unazadne propagacije. Algoritam unazadne propagacije zahtijeva da su aktivacijske funkcije diferencijabilne. Algoritam se odvija u dva koraka:

- Unaprijedni prolaz kroz neuronsku mrežu kojim za zadani ulaz računamo izlaz mreže.
- Unazadni prolaz kroz neuronsku mrežu gdje računamo odstupanje izlaza mreže od traženog izlaza zadanog primjera te se izračunata pogreška propagira unazad kroz mrežu. Pogreške se računaju za svaki neuron pomoću kojih se ažuriraju parametri mreže.

Preciznije, cilj unazadne propagacije je izračunati parcijalne derivacije

$$\frac{\partial C}{\partial w_{ij}^l}, \frac{\partial C}{\partial b_i^l},$$

gdje C predstavlja funkciju pogreške, b_i^l bias i-tog neurona u l-tom sloju te w_{ij}^l težinski faktor kojim aktivacijska vrijednost j-tog neurona $(l-1)$-vog sloja ulazi u i-ti neuron l-tog sloja. Aktivacijsku vrijednost i-tog neurona l-tog sloja označit ćemo sa a_i^l. Aktivacijska vrijednost a_i^l povezana je s aktivacijskim vrijednostima $(l-1)$-vog sloja preko jednadžbe

$$a_i^l = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l \right). \tag{1.9}$$

Jednadžbu (1.9) možemo zapisati i u matričnom obliku (1.10),

$$a^l = f\left(W^l a^{l-1} + b^l \right), \tag{1.10}$$

gdje $W^l = (w^l_{ij})$ predstavlja matricu težinskih faktora l-tog sloja, b^l vektor biasa l-tog sloja te a^l vektor aktivacijskih vrijednosti l-tog sloja.

Zbog jednostavnosti zapisa jednadžbi kojim računamo parcijalne derivacije, uvodimo i pomoćnu varijablu z^l_i koja predstavlja vrijednost i-tog neurona l-tog sloja prije primjene aktivacijske funkcije, odnosno $a^l_i = f\left(z^l_i\right)$, odnosno vektorski $a^l = f\left(z^l\right)$.

Promotrimo li vrijednost z^l_i, lako se primjeti da mala perturbacija Δz^l_i na i-tom neuronu l-tog sloja mijenja aktivacijsku vrijednost sa $f(z^l_i)$ na $f(z^l_i+\Delta z^l_i)$. Kako nastavimo propagirati tu vrijednost kroz kasnije slojeve mreže, ukupna vrijednost funkcije pogreške se približno mijenja za vrijednost $\frac{\partial C}{\partial z^l_i}\Delta z^l_i$. Ako $\frac{\partial C}{\partial z^l_i}$ poprima veliku vrijednost, tada malom perturbacijom Δz^l_i suprotnog predznaka možemo smanjiti vrijednost funkcije pogreške. Također, ako $\frac{\partial C}{\partial z^l_i}$ poprima malu vrijednost, tada perturbacija Δz^l_i i ne pravi neku razliku. Motivirani ovim činjenicama, $\frac{\partial C}{\partial z^l_i}$ možemo interpretirati kao „pogrešku" na i-tom neuronu l-tog sloja.

Kako bismo izračunali tražene parcijalne derivacije uvodimo posrednu vrijednost

$$\delta^l_i = \frac{\partial C}{\partial z^l_i} \tag{1.11}$$

koju zovemo pogreška na i-tom neuronu l-tog sloja. Unazadna propagacija daje nam postupak kojim ćemo izračunati δ^l_i te ga povezati sa $\frac{\partial C}{\partial w^l_{ij}}$ i $\frac{\partial C}{\partial b^l_i}$.

Algoritam unazadne propagacije temelji se na četiri jednadžbe do kojih dolazimo uz pomoć „lančanog pravila". Prva jednadžba (1.12) služi za račun „pogreške" na neuronima u posljednjem, L-tom sloju:

$$\begin{aligned}\delta^L_i &= \frac{\partial C}{\partial z^L_i} = \frac{\partial C}{\partial a^L_i}\frac{\partial a^L_i}{\partial z^L_i} \\ &= \frac{\partial C}{\partial a^L_i} f'(z^L_i).\end{aligned} \tag{1.12}$$

Ako su δ^L_j komponente vektora δ^L slijedi

$$\delta^L = \nabla_a C \odot f'(z^L), \tag{1.13}$$

gdje $\odot$ predstavlja Hadamardov produkt ($(A\odot B)_{ij} = (A)_{ij}\cdot(B)_{ij}$), a $\nabla_a C$ vektor čije su komponente parcijalne derivacije $\frac{\partial C}{\partial a^L_i}$. Druga jednadžba (1.14) služi za izražavanje „pogreške" na neuronima $(l-1)$-vog sloja preko pogreške l-tog sloja. Kako bi izveli jednadžbu moramo zapisati $\delta^l_i = \frac{\partial C}{\partial z^l_i}$ preko $\delta^{l+1}_j = \frac{\partial C}{\partial z^{l+1}_j}$, odnosno

$$\begin{aligned}\delta_i^l &= \frac{\partial C}{\partial z_i^l} = \sum_j \frac{\partial C}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_j \frac{\partial z_j^{l+1}}{\partial z_i^l} \delta_j^{l+1} \\ &= \sum_j \frac{\partial \left(\sum_k w_{jk}^{l+1} f(z_k^l) + b_j^{l+1} \right)}{\partial z_i^l} \delta_j^{l+1} \\ &= \sum_j w_{ji}^{l+1} \delta_j^{l+1} f'(z_i^l).\end{aligned} \tag{1.14}$$

Ako su δ_j^l komponente vektora δ^l slijedi

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot f'(z^l) \tag{1.15}$$

gdje $(W^{l+1})^T$ predstavlja transponiranu matricu težinskih faktora $(l+1)$-vog sloja. Preko jednadžbi (1.13) i (1.15) možemo izračunati „pogrešku" na svakom neuronu mreže. Izračunate pogreške preostaje povezati s parcijalnim derivacijama $\frac{\partial C}{\partial w_{ij}^l}$ i $\frac{\partial C}{\partial b_i^l}$. Primjenom „lančanog pravila" dolazimo do

$$\begin{aligned}\frac{\partial C}{\partial b_i^l} &= \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial b_i^l} = \delta_i^l, \\ \frac{\partial C}{\partial w_{ij}^l} &= \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} = a_j^{l-1} \delta_i^l,\end{aligned} \tag{1.16}$$

odnosno, ako zapišemo u matričnom obliku

$$\begin{aligned}\frac{\partial C}{\partial b^l} &= \delta^l, \\ \frac{\partial C}{\partial W^l} &= a^{l-1} (\delta^l)^T\end{aligned} \tag{1.17}$$

gdje sa a^{l-1} označavamo vektor aktivacijskih vrijednosti neurona $(l-1)$-vog sloja, a sa $(\delta_l)^T$ transponirani vektor „pogrešaka" na neuronima l-tog sloja.

Sada kada raspolažemo jednadžbama algoritma unazadne propagacije, preostaje zapisati rad algoritma:

1. Unos podatka x: računamo $a^1 = f(W^1 x + b^1)$
2. Unaprijedna propagacija: $z^l = W^l a^{l-1} + b^l$, $a^l = f(z^l)$, za $l = 2, 3, \ldots, L$
3. Izračun vektora δ^L: $\delta^L = \nabla_a C \odot f'(z^L)$

4. Unazadna propagacija: $\delta^l = W^{l+1}\delta^{l+1} \odot f'(z^l)$, za $l = L-1, L-2, \ldots, 1$
5. Izračun gradijenta: $\frac{\partial C}{\partial b^l} = \delta^l$, $\frac{\partial C}{\partial W^l} = a^{l-1}(\delta^l)^T$, za $l = 1, 2, \ldots, L$

Ključni detalj algoritma unazadne propagacije je da uspijeva izračunati sve parcijalne derivacije u jednom prolasku unaprijed iza kojeg slijedi jedan prolazak unatrag kroz mrežu. Grubo govoreći, računski trošak prolaska unatrag otprilike je isti kao i prolaz unaprijed, stoga je ukupni trošak algoritma unazadne propagacije otprilike jednak kao i dva prolaska unaprijed kroz mrežu.

1.4 Konvolucijske neuronske mreže

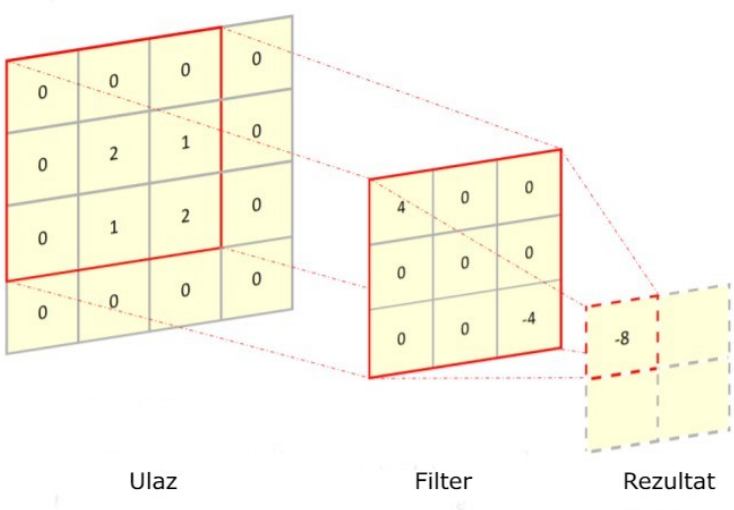
Potpuno povezane neuronske mreže imaju nekoliko nedostataka kada ih primjenjujemo na probleme obrade slika. Prvi problem je jako velik broj parametara za treniranje zbog veličine samog ulaza pa je takve modele teško trenirati. Također, kada koristimo unaprijedne neuronske mreže ulaznu sliku (matricu) prvo moramo spljoštiti u vektor (npr. tako da sve stupce matrice posložimo jedan na drugi) čime gubimo sve prostorne informacije koje nam je matrični zapis dao.

Konvolucijske neuronske mreže (eng. *convolutional network*) predstavljaju klasu neuronskih mreža koje sadrže tzv. konvolucijski sloj i zaslužne su za veliki napredak na području računalnog vida, gdje na nekim problemima postižu i nadljudske rezultate. Kod izgradnje konvolucijske neuronske mreže koristimo tri glavne vrste slojeva: konvolucijski sloj, sloj sažimanja i potpuno povezani sloj koji je jednak kao kod našeg opisa potpuno povezanih neuronskih mreža.

Konvolucijski sloj

Konvolucijski sloj je osnovna građevna jedinica konvolucijskih neuronskih mreža te obavlja većinu transformiranja značajki. Konvolucijski sloj temelji se na ideji da su bliži pikseli slike jače povezani nego oni udaljeni. Svaki konvolucijski sloj se sastoji od skupa konvolucijskih prozora (filtera) čije vrijednosti služe za „izračunavanje" odnosa između bliskih piksela. U konvolucijskom koraku svaki se filter pomiče preko cijele širine i visine ulaza i izračunava Hadamardov produkt između vrijednosti prekrivenog dijela slike i vrijednosti filtera te ih sumira (vidi sliku 1.7). Proces se nastavlja pomicanjem filtera i računanjem traženih vrijednosti koje se spremaju u novu matricu.

Empirijski se ispostavilo da konvolucijski slojevi koji su na početku mreže uče prepoznavati detalje i strukture. Primjerice, prvi sloj mreže može naučiti prepoznavati rubove objekata na slikama. Što se sloj nalazi dublje u mreži, on uči prepoznavati kompleksnije značajke. Tako drugi sloj mreže može prepoznavati jednostavne geometrijske oblike, dok posljednji sloj može prepoznavati lica, objekte iz prirode i slično. Vrijednosti težina koje



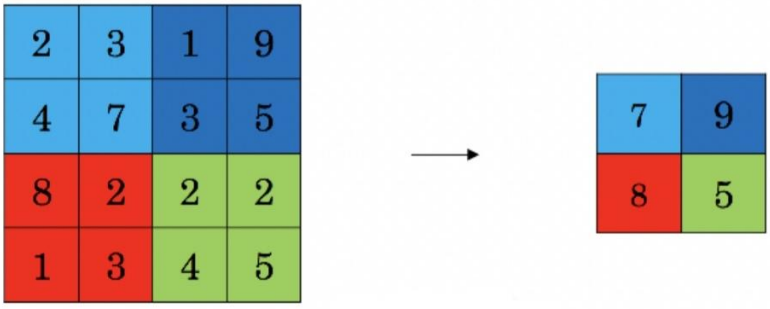
Slika 1.7: Primjena konvolucijskog prozora na matricu. U kontekstu neuronskih mreža, „Ulaz" predstavlja aktivacijske vrijednosti j-tog sloja, „Filter" težine koje treniramo, a „Rezultat" su skalarni produkti prije primjene aktivacijske funkcije. Filter klizi po matrici slijeva nadesno i odozgo prema dolje dok ne prođe po j-tom aktivacijskom sloju, odnosno dok ne izračuna sve vrijednosti u matrici „Rezultat".

su pohranjene u filterima predstavljaju parametre modela te ih mreža može naučiti. Na početku na slučajan način inicijaliziramo filtere kako bi izbjegli da dva filtera traže jedan te isti oblik.

Sloj sažimanja

Sloj sažimanja (eng. *pooling layer*) predstavlja primjenu funkcija sažimanja na aktivacijske mape iz prethodnog sloja. Na taj način smanjujemo prostornu veličinu i broj parametara za računanje u mreži. Sloj sažimanja se pokazao iznimno koristan pri čuvanju invarijantnosti prema malim translacijama piksela. Ukoliko sliku translatiramo za neki mali iznos, vrijednost rezultata nakon sažimanja neće se značajno promijeniti. Pomoću sloja sažimanja značajka se pronalazi na nekom području, a ne na „naučenom" mjestu čime izbjegavamo prenaučenost. Također, sažimanje je korisno kada nam je važno prepoznati je li neka značajka na slici, a ne gdje se točno nalazi.

Dva najčešća tipa sažimanja su sažimanje izborom maksimalnog elementa (eng. *max-pooling*) i sažimanje prosječnom vrijednošću (eng. *average-pooling*). Primjenu sloja sažimanja možemo vidjeti na slici 1.8.

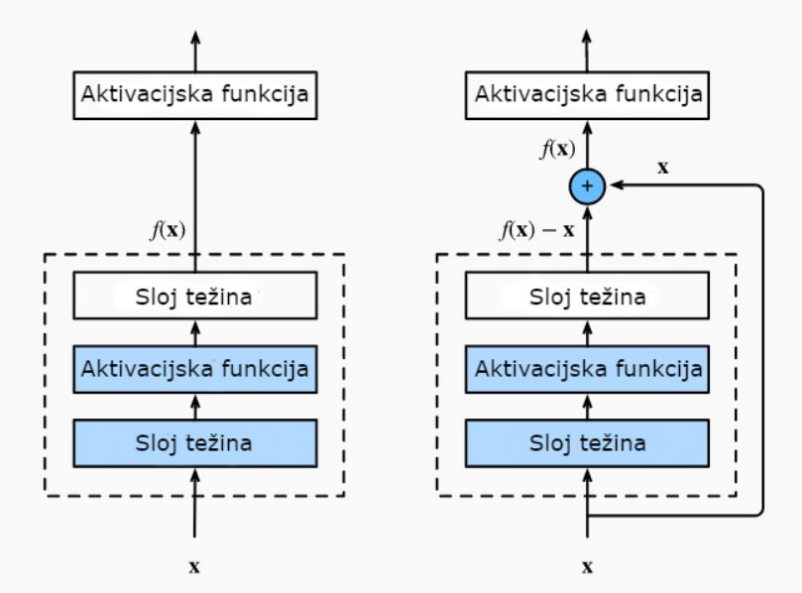


Slika 1.8: Sažimanje izborom maksimalnog elementa.

1.5 Rezidualne neuronske mreže

Kako modelu povećavamo dubinu, za očekivati je da će u najgorem slučaju plići model i dublji model postići istu preciznost, a češće da će dublji model biti precizniji. Međutim, empirijski je pokazano da postoji dubina nakon koje počinje naglo degradiranje preciznosti dubokih modela kod standardnog učenja. Do degradiranja preciznosti dolazi zbog problema nestajućeg gradijenta. Što je model dublji, postaje teže propagirati pogrešku do plitkih slojeva i model prestaje učiti. Kako bi se riješio taj problem, uvode se direktne veze između plitkih i dubokih slojeva koje omogućavaju lakše propagiranje informacije u oba smjera.

Preciznije, neka imamo dio mreže koja kao ulaz prima x, a željeni izlaz joj je $f(x)$ (vidi sliku 1.9). U lijevom primjeru taj dio mreže pokušava direktno naučiti funkciju $f(x)$, dok u desnom primjeru, kako ulaz x preskače taj dio mreže, preskočeni komad pokušava naučiti „ostatak", tj. rezidual $f(x) - x$, odakle i naziv rezidualno učenje. Ako je iz modela izvučen maksimum, željeno preslikavanje postaje funkcija identiteta $f(x) = x$ pa je takvo preslikavanje izuzetno lako naučiti guranjem težina i biasa sloja u nulu. Iscrtkano obrubljeni dio na desnom primjeru nazivamo rezidualni blok (eng. *residual block*) i ključna je građevna jedinica dubokih modela koje nazivamo rezidualne neuronske mreže (eng. *ResNets*) [7].



Slika 1.9: Primjer standardnog učenja (lijevo) i rezidualnog učenja (desno) [10].

Poglavlje 2

Generativne suparničke mreže

2.1 GAN

Generativne suparničke mreže (eng. *Generative adversarial networks*) ili GANovi [5] predstavljaju model nenadziranog učenja u kojem generator pokušava naučiti uzorke u podacima kako bi mogao generirati nove, dosad neviđene podatke.

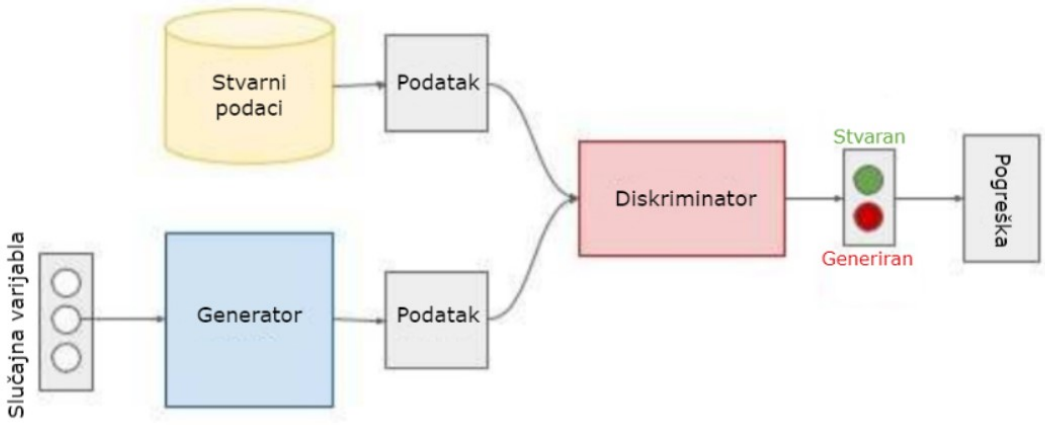
GANovi se temelje na ideji suparničkog treniranja u kojem se dvije mreže, generator i diskriminator, bore jedna protiv druge i tako međusobno poboljšavaju. Generator kao ulazni podatak prima vektor slučajnih vrijednosti i transformira ga u „lažni" (generirani) podatak. Stvarne i generirane podatke predajemo diskriminatoru koji pokušava odlučiti koji od njih su stvarni, a koji generirani. Takav postupak dovodi nas u jednu od dvije situacije:

- Generator uspješno vara diskriminator, odnosno diskriminator ne uspijeva klasificirati generirane podatke kao lažne. Tada je potrebno poboljšati diskriminator.
- Diskriminator uspješno razlikuje generirane podatke od stvarnih što znači da generirani podaci nisu dovoljno dobri za prevariti diskriminator. U tom slučaju potrebno je poboljšati generator.

Pogledamo li pobliže, diskriminator nije ništa više nego binarni klasifikator koji pokušava razlikovati stvarne podatke od generiranih. Kao što je spomenuto u poglavlju 1.3, kod problema klasifikacije često korištena funkcija pogreške je funkcija unakrsne entropije (1.6), odnosno u ovom slučaju funkcija binarne unakrsne entropije

$$BCE(\hat{y}, y) = y \log \hat{y} + (1 - y) \log(1 - \hat{y}), \tag{2.1}$$

gdje sa $\hat{y}$ označavamo izlaz modela (predviđenu vrijednost), a sa $y \in \{0, 1\}$ stvarnu klasu. Također, primjetimo da smo u funkciji binarne unakrsne entropije izostavili negativan predznak pa funkciju pogreške nećemo minimizirati već maksimizirati.



Slika 2.1: Generativna suparnička mreža.

Preciznije, neka je p_{data} distribucija stvarnih podataka, a p_z distribucija slučajne varijable z koju generator $G(z;\theta_G)$ prima na ulazu i preslikava u podatak iz distribucije p_g koja pokušava aproksimirati p_{data}, a G predstavlja diferencijabilnu funkciju reprezentiranu neuronskom mrežom s parametrima θ_G. Također, definiramo i diskriminator $D(x;\theta_D)$ zadan neuronskom mrežom s parametrima θ_D koji kao izlaz daje vjerojatnost da je podatak x došao iz distribucije stvarnih podataka p_{data} prije nego iz generatorske distribucije p_g.

Dakle, diskriminator (klasifikator) D kao ulaz prima ili $x \sim p_{data}$ ili $G(z) \sim p_g$. Željenu vrijednost za $D(x)$ označit ćemo sa 1, a željenu vrijednost za $D(G(z))$ sa 0. Promotrimo li sada jednadžbu (2.1) za ulazni podatak $x \sim p_{data}$ imamo:

$$BCE(D(x), 1) = \log D(x).$$

Očito želimo da diskriminator maksimizira $\log D(x)$, a kako je log monotona funkcija $\log D(x)$ ćemo maksimizirati ako maksimiziramo izlaz diskriminatora $D(x)$. Također, promotrimo li jednadažbu (2.1) za ulazni podatak $G(z) \sim p_g$ imamo:

$$BCE(D(G(z)), 0) = \log(1 - D(G(z))).$$

Ponovno, želimo maksimizirati vrijednost $\log(1 - D(G(z)))$, što je ekvivalentno maksimiziranju $1 - D(G(z))$, odnosno, minimiziranju $D(G(z))$. Tada funkcija pogreške za diskriminator za jedan primjer postaje

$$\log D(x) + \log(1 - D(G(z))) \tag{2.2}$$

i tražimo D koji će maksimizirati vrijednost funkcije pogreške, odnosno

$$\max_D \left(\log D(x) + \log(1 - D(G(z)))\right). \tag{2.3}$$

Cilj generatora je prevariti diskriminator generiranjem što stvarnijih slika, odnosno, generator želi stvoriti $G(z)$ takve da ih diskriminator označava sa 1. Iz (2.1) slijedi

$$BCE(D(G(z)), 0) = \log(1 - D(G(z)))$$

te kako generator radi „na štetu" diskriminatora, generator zapravo minimizira $\log(1 - D(G(z)))$, odnosno maksimizira $D(G(z))$ pa $\log(1 - D(G(z)))$ uzimamo kao funkciju pogreške za generator i tražimo G koji će funkciju pogreške minimizirati, odnosno,

$$\min_G \log(1 - D(G(z))). \tag{2.4}$$

Konačno, jednadžbe (2.3) i (2.4) možemo povezati u

$$\min_G \max_D \left(\log D(x) + \log(1 - D(G(z))\right) \tag{2.5}$$

jer prvi član ne ovisi o G.

Iz jednadžbe (2.5) lako je primjetiti da se generativne suparničke mreže temelje na minimax igri u kojoj se generatorska mreža mora natjecati s protivnikom diskriminatorom. Drugim riječima, igrači D i G igraju minimax igru s funkcijom dobitka $V(G, D)$. Tijekom učenja, diskriminator pokušava maksimizirati vlastiti dobitak dok ga generator pokušava minimizirati pa vrijednost konvergira u

$$v^* = \min_G \max_D V(G, D). \tag{2.6}$$

Motivirani izvodom na jednom primjeru, promotrimo li jednadžbu (2.5) u kontekstu slučajno odabranih mini-grupa, funkciju pogreške V možemo zapisati kao

$$V(G, D) = \mathbb{E}_{x \sim p_{data}} \left[\log D(x)\right] + \mathbb{E}_{z \sim p_z} \left[\log(1 - D(G(z))\right]. \tag{2.7}$$

Ovakav izbor funkcije pogreške tjera diskriminator da pokuša naučiti ispravno klasificirati stvarne primjere od generiranih. Istovremeno, generator pokušava natjerati klasifikator da pomisli kako su njegovi primjeri stvarni. Minimax igra definirana funkcijom (2.7) globalni optimum postiže kada je $p_g = p_{data}$, što na kraju treniranja mreže i želimo postići.

Prije nego krenemo dokazivati navedenu tvrdnju, definirat ćemo dvije funkcije za ocjenjivanje sličnosti vjerojatnostnih distribucija p i q. Prva takva funkcija je *KL (Kullback-Leibler)* divergencija definirana jednadžbom

$$KL(p \parallel q) = \int_x p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x. \tag{2.8}$$

KL divergencija postiže minimalnu vrijednost nula kada je $p = q$. Nedostatak KL divergencije je asimetričnost pa ne zadovoljava svojstva metrike. Uz KL divergenciju uvodimo i *JS (Jensen-Shannon)* divergenciju definiranu jednadžbom

$$JS(p \parallel q) = \frac{1}{2}KL\left(p \,\middle\|\, \frac{p+q}{2}\right) + \frac{1}{2}KL\left(q \,\middle\|\, \frac{p+q}{2}\right). \tag{2.9}$$

JS divergencija je simetrična, ograničena funkcija.

Prvo izvodimo jednadžbu optimalnog diskriminatora za proizvoljan generator. Za diskriminator D kažemo da je optimalan s obzirom na fiskni G ako najbolje raspoznaje između stvarnih i generatorom G generiranih podataka, odnosno, ako za D jednadžba (2.7) postiže maksimum.

Propozicija 2.1.1. *Za fiksni generator G i funkciju pogreške V, optimalan diskriminator D je*

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}.$$

Dokaz. Jednadžba (2.7) predstavlja funkciju pogreške V koju diskriminator optimizira bez obzira na generator.

$$\begin{aligned} V(G,D) &= \int_x p_{data}(x) \log D(x)\, \mathrm{d}x + \int_z p_z(z) \log\left(1 - D(G(z))\right)\, \mathrm{d}z \\ &\overset{(2.11)}{=} \int_x \Big(p_{data}(x) \log D(x) + p_g(x) \log\left(1 - D(x)\right)\Big)\, \mathrm{d}x \end{aligned} \tag{2.10}$$

Za svaki $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, funkcija $y \to a \log y + b \log(1 - y)$ postiže maksimum na intervalu $[0, 1]$ u točki $\frac{a}{a+b}$ iz čega slijedi dokaz. Pomoćna tvrdnja:

$$\begin{aligned} \int_z p_z(z) \log\left(1 - D(G(z))\right)\, \mathrm{d}z &= \mathbb{E}_{z\sim p_z}\left[\log(1 - D(G(z)))\right] \\ &= \mathbb{E}_{x\sim p_g}\left[\log(1 - D(x))\right] \\ &= \int_x \Big(p_g(x) \log\left(1 - D(x)\right)\Big)\, \mathrm{d}x \end{aligned} \tag{2.11}$$

□

Teorem 2.1.2. *Globalni minimum funkcije* $\max_D V(G, D)$ *je postignut ako i samo ako* $p_g = p_{data}$ *te u toj točki vrijednost funkcije iznosi* $-\log 4$.

Dokaz.

$$
\begin{aligned}
\max_D V(G,D) &= V(G,D_G^*) \\
&= \int_x p_{data}(x)\log D_G^*(x)\,\mathrm{d}x + \int_x p_g(x)\log\left(1-D_G^*(x)\right)\,\mathrm{d}x \\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right)\mathrm{d}x + \int_x p_g(x)\log\left(\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right)\mathrm{d}x \\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)\mathrm{d}x - \int_x p_{data}(x)\log 2\,\mathrm{d}x \\
&+ \int_x p_g(x)\log\left(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)\mathrm{d}x - \int_x p_g(x)\log 2\,\mathrm{d}x \\
&= KL\left(p_{data}\,\middle\|\,\frac{p_{data}+p_g}{2}\right) + KL\left(p_g\,\middle\|\,\frac{p_{data}+p_g}{2}\right) - 2\log 2 \\
&= 2JS(p_{data}\parallel p_g) - \log 4
\end{aligned}
\tag{2.12}
$$

Kako je JS divergencija između dvije distribucije uvijek nenegativna, a nulu postiže isključivo kada su distribucije jednake, pokazali smo da je $-\log 4$ globalni minimum funkcije $\max_D V(G,D)$ koji se postiže kada je $p_{data} = p_g$. □

Treniranje generativnih suparničkih mreža provodi se naizmjenično, gdje prvo jednu ili više iteracija treniramo diskriminator pa onda generator, i tako u krug. Tijekom treniranja diskriminatora, parametre generatora fiksiramo pa diskriminator uči prepoznati mane trenutnog generatora kako bi uspješno razlikovao stvarne podatke od generiranih. Slično, za vrijeme treniranja generatora parametre diskriminatora fiksiramo. U suprotnom bi generator pokušavao pogoditi „pomičnu metu" i možda nikada ne bi konvergirao.

Nažalost, treniranje generativnih suparničkih mreža u praksi se pokazalo zahtjevnim. Zbog istovremenog provođenja gradijentnog spusta na funkciju pogreške za generator i diskriminator, pronalaženje ravnoteže igre (sedlaste točke funkcije V(G, D)) postaje izrazito zahtjevno. U takvim situacijama parametri modela osciliraju i nikada ne konvergiraju. Također, javlja se problem nestajućeg gradijenta. Problem nestajućeg gradijenta nastaje kada diskriminator počne izrazito uspješno raspoznavati stvarne primjere od generiranih. Tada, zbog dobrog rada diskriminatora, gradijent funkcije pogreške poprima veličinu približnu nuli pa propagiranjem gradijenta kroz model ne postižemo željeni efekt, odnosno parametri modela se jako malo mijenjaju ili ostaju nepromijenjeni. Treći problem nastaje kada diskriminator zapne u nekom od lokalnih minimuma. U tom slučaju generator pronalazi jedan primjer ili više njih koje diskriminator prihvaća kao stvarne i nauči svaki ulazni vektor preslikavati, do na male varijacije, u te primjere (eng. *mode collapse*).

Algoritam 1 Treniranje generativnih suparničkih mreža stohastičkim gradijentnim spustom. Broj koraka k koji primjenjujemo na diskriminatoru je hiperparametar.

for iteracija treniranja = 1, 2, ... **do**
  for korak = 1, 2, ..., k **do**
    Uzorkuj skup od m primjera $\{z_1, \ldots, z_m\}$ iz distribucije p_z.
    Uzorkuj skup od m primjera $\{x_1, \ldots, x_m\}$ iz distribucije p_{data}.
    Ažuriraj parametre diskriminatora stohastičkim gradijentnim usponom gdje je gradijent funkcije pogreške dan sa:

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x_i\right) + \log\left(1 - D(G(z_i))\right)\right]$$

  end for
  Uzorkuj skup od m primjera $\{z_1, \ldots, z_m\}$ iz distribucije p_z.
  Ažuriraj parametre generatora stohastičkim gradijentnim spustom gdje je gradijent funkcije pogreške dan sa:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D(G(z_i))\right)$$

end for

Poboljšanje treniranja ovakvog modela koje se pokazalo uspješnim u praksi leži u drukčijoj formulaciji problema, odnosno generator više ne smanjuje log-vjerojatnost da diskriminator napravi točnu predikciju, već uvećava log-vjerojatnost da diskriminator napravi pogrešku. Preciznije, umjesto da minimizira $\log\left(1 - D(G(z))\right)$, generator maksimizira $\log D(G(z))$ [5].

2.2 Wasserstein GAN

Osim što smo pokazali da globalni minimum funkcija $\max_D V(G, D)$ postiže kada je $p_g = p_{data}$, jednadžbom (2.12) pokazali smo da funkcija pogreške u GANovima ocjenjuje sličnost između distribucija p_g i p_{data} pomoću JS divergencije kada je diskriminator optimalan. U ovom poglavlju pokazat ćemo nedostatke KL i JS divergencije, uvesti novu metriku sličnosti dvaju distribucija zvanu *Wasserstein* udaljenost te pomoću nje otkloniti neke od problema koji nastaju pri treniranju GANova. Također, uvest ćemo novi model GANa zvan *Wasserstein GAN* ili skraćeno WGAN.

Primjer 2.2.1. *Neka je* $Z \sim U(0,1)$ *slučajna varijabla s uniformnom distribucijom na intervalu* $\langle 0,1\rangle$. *Neka je* $\mathbb{P}_0$ *distribucija slučajne varijable* $(0,Z) \in \mathbb{R}^2$, *a* $\mathbb{P}_\lambda$ *distribucija slučajne varijable* (λ,Z), *gdje* λ *predstavlja fiksnu vrijednost,* $\lambda \in [0,1]$. *Promotrimo primjenu KL i JS divergencije na* $\mathbb{P}_0$ *i* $\mathbb{P}_\lambda$. *Vrijedi:*

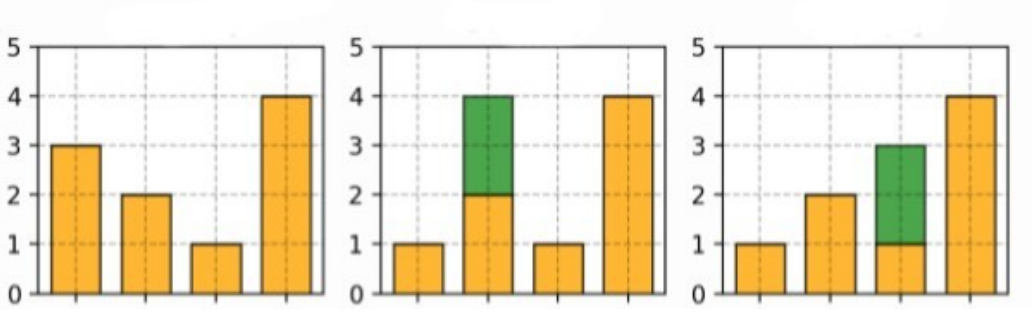
$$KL(\mathbb{P}_0 \parallel \mathbb{P}_\lambda) = \begin{cases} +\infty, & \lambda \neq 0 \\ 0, & \lambda = 0 \end{cases} \tag{2.13}$$

$$KL(\mathbb{P}_\lambda \parallel \mathbb{P}_0) = \begin{cases} +\infty, & \lambda \neq 0 \\ 0, & \lambda = 0 \end{cases} \tag{2.14}$$

$$JS(\mathbb{P}_0 \parallel \mathbb{P}_\lambda) = \begin{cases} \log 2, & \lambda \neq 0 \\ 0, & \lambda = 0 \end{cases} \tag{2.15}$$

Iz primjera 2.2.1 lako vidimo nedostatke KL i JS divergencija. KL divergencija poprima vrijednost $+\infty$, dok JS divergencija ima gradijent nula kada ne postoji preklapanje između distribucija. Također, JS divergencija nije diferencijabilna ni neprekidna za vrijednost $\lambda = 0$. Za razliku od KL i JS divergencija, željeli bismo imati mjeru sličnosti distribucija kojoj bi vrijednost bila što manja kako λ ide u nulu.

Motivirani potrebom za boljom mjerom sličnosti između dvije distribucije promotrit ćemo Wasserstein udaljenost. Za diskretne slučajne varijable, Wasserstein udaljenost još možemo zvati i „Earth-Mover" udaljenost. Ako distribucije diskretnih slučajnih varijabli p' i q' zamislimo kao dvije hrpe zemlje (vidi sliku 2.2), Wasserstein udaljenost predstavlja minimalnu količinu posla potrebnu za transformirati jednu hrpu zemlje u drugu. Količinu posla definiramo kao količinu zemlje koju je potrebno pomaknuti pomnoženu s prijeđenom udaljenosti.



Slika 2.2: Neka prvi graf predstavlja distribuciju p', a treći graf distribuciju q'. Na drugom grafu zelenom bojom smo označili količinu zemlje koju smo prenijeli s prvog stupca na drugi, a na trećem grafu količinu zemlje koju smo prenijeli s drugog stupca na treći. Ukupna količina posla predstavljena ovim planom transporta iznosi 4 [21].

Računanje Wasserstein udaljenosti je optimizacijski problem jer postoji beskonačno mnogo načina za pomicati zemlju naokolo, a potrebno je pronaći optimalni plan. Plan transporta zemlje označit ćemo sa γ, gdje $\gamma(x, y)$ kaže koliko ćemo zemlje prenijeti sa stupca x na stupac y. Naravno, kako bi plan transporta zemlje γ bio valjan, moramo osigurati da sa svakog stupca zemlje raspolažemo s točno onoliko zemlje koliko na tom stupcu ima, odnosno, $\sum_x \gamma(x, y) = p'(y)$, $\sum_y \gamma(x, y) = q'(x)$. Sa $\Pi(p', q')$ označavamo skup svih valjanih planova transporta zemlje.

Formalno, neka $\Pi(p', q')$ predstavlja skup svih mogućih zajedničkih distribucija γ' čije su marginalne distribucije p' i q'. Tada diskretna Wasserstein udaljenost glasi

$$EMD(p', q') = \inf_{\gamma' \in \Pi(p', q')} \sum_{x,y} \gamma'(x, y)\|x - y\| = \inf_{\gamma' \in \Pi(p', q')} \mathbb{E}_{(x,y)\sim\gamma'}[\|x - y\|]. \tag{2.16}$$

Sada kada smo razvili intuiciju o Wasserstein udaljenosti za slučaj diskretnih distribucija, neka su p i q dvije neprekidne vjerojatnosne distribucije. Wasserstein udaljenost na neprekidnim distribucijama definirana je jednadžbom

$$W(p, q) = \inf_{\gamma \in \Pi(p,q)} \int_x \int_y \gamma(x, y)\|x - y\| \, \mathrm{d}y\mathrm{d}x = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|], \tag{2.17}$$

gdje $\Pi(p, q)$ predstavlja skup svih mogućih zajedničkih distribucija γ čije su marginalne distribucije p i q. Intuitivno, $\gamma(x, y)$ nam govori koliko „mase" moramo prenijeti sa x na y kako bismo distribuciju p transformirali u distribuciju q. Wasserstein udaljenost predstavlja vrijednost optimalnog takvog plana. Barem intuitivno, Wasserstein udaljenost čini se kao dobra metrika za optimizirati pri treniranju GANova jer doslovno računa koliko nam je „posla" još ostalo da distribuciju generiranih podataka p_g pretvorimo u distribuciju stvarnih podataka p_{data}.

U odnosu na dosad korištene mjere sličnosti distribucija u treniranju GANova poput KL i JS divergencija, primjenimo li Wasserstein udaljenost na problem iz primjera 2.2.1, vrijedi

$$W(\mathbb{P}_0, \mathbb{P}_\lambda) = |\lambda|, \tag{2.18}$$

odnosno, Wasserstein udaljenost ne poprima konstantnu vrijednost kad nema preklapanja između distribucija i neprekidna je funkcija s „upotrebljivim" gradijentom što je vrlo važno za stabilno učenje gradijentnim spustom.

Međutim, nemoguće je isprobati sve moguće zajedničke distribucije γ iz skupa $\Pi(p, q)$ kako bismo pronašli infimum. Iz tog razloga, predlaže se reformulacija jednadžbe (2.17) [1, 8] u kojoj je cilj potpuno maknuti ograničenja s obzirom na distribuciju γ. U nastavku ćemo neformalno povezati jednadžbu (2.17) s njezinim dualnim oblikom, a rigorozni dokaz izlazi iz okvira ovoga rada i moguće je pronaći u [18].

Za funkciju $f\colon X \to Y$ kažemo da je K-Lipschitz neprekidna ako vrijedi

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2),\ \forall x_1, x_2 \in X \tag{2.19}$$

gdje je d_X metrika na prostoru X, a d_Y metrika na prostoru Y. Sa $\|f\|_L \leq 1$ označit ćemo skup svih 1-Lipschitz funkcija $f\colon X \to \mathbb{R}$. Neka je $f\colon \mathbb{R}^n \to \mathbb{R}$ neprekidna. U jednadžbu (2.17) „umjetno" uvodimo supremum po f tako da kao rješenje odbacuje sve $\gamma \notin \Pi(p, q)$:

$$W(p,q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|] \tag{2.20}$$

$$= \inf_{\gamma} \Big(\mathbb{E}_{(x,y)\sim\gamma}\Big[\|x-y\|\Big] + \underbrace{\mathbb{E}_{(x,y)\sim\gamma}\Big[\sup_{f} \Big(\mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)] - (f(x)-f(y)) \Big) \Big]}_{= \begin{cases} 0, & \gamma \in \Pi(p,q) \\ +\infty, & \text{inače} \end{cases}} \Big) \tag{2.21}$$

$$= \inf_{\gamma} \sup_{f} \mathbb{E}_{(x,y)\sim\gamma}\Big[\|x-y\| + \mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)] - (f(x)-f(y))\Big] \tag{2.22}$$

$$= \sup_{f} \inf_{\gamma} \mathbb{E}_{(x,y)\sim\gamma}\Big[\|x-y\| + \mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)] - (f(x)-f(y))\Big] \tag{2.23}$$

$$= \sup_{f} \Big(\mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)] + \underbrace{\inf_{\gamma} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\| - (f(x)-f(y))\right]}_{= \begin{cases} 0, & \|f\|_L \leq 1 \\ \text{negativno}, & \text{inače} \end{cases}} \Big) \tag{2.24}$$

$$= \sup_{\|f\|_L \leq 1} \Big(\mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)] \Big). \tag{2.25}$$

U jednadžbi (2.21), ako je $\gamma \in \Pi(p, q)$ tada $\mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)]$ i $\mathbb{E}_{(x,y)\sim\gamma}[f(x) - f(y)]$ su jednaki pa je njihov supremum nula neovisno o izboru f. Ako $\gamma \notin \Pi(p, q)$, tada su s i x, odnosno, t i y, potpuno neovisni i možemo odabrati f tako da postigne vrijednost $+\infty$. Iz tog razloga možemo stvoriti taj član i maknuti ograničenje u infimumu da γ mora biti iz skupa $\Pi(p, q)$. Zamjena infimuma i supremuma u jednadžbama (2.22) i (2.23) dozvoljena je zbog minimax principa (eng. *minimax principle*). U jednadžbi (2.24) ako je f 1-Lipschitz neprekidna vrijedi $\mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\| - (f(x)-f(y))\right] \geq 0$. Također, iz definicije Lipschitz neprekidnosti slijedi $\|x-y\| - (f(x)-f(y)) \leq 2\|x-y\|$, odnosno, $\mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\| - (f(x)-f(y))\right] \leq 2\mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\|\right]$. Sada, kada smo slobodni odabrati bilo koju distribuciju γ (ne mora imati p i q kao marginale), možemo odabrati niz multivarijatnih normalnih distribucija kojima standardna devijacija konvergira u nulu pa je i vrijednost infimuma očekivanja nula. Ako funkcija f nije 1-Lipschitz neprekidna, uz pretpostavku da je neprekidna i da postoji (x, y) takva da $f(x) - f(y) > \|x-y\|$, znamo da funkcija

$\|x - y\| - (f(x) - f(y))$ poprima negativnu vrijednost u maloj okolini od (x, y). Sada, ako za γ uzmemo multivarijatnu normalnu distribuciju sa očekivanjem (x, y) i dovoljno malom standardnom devijacijom možemo postići da je očekivanje $\mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\| - (f(x) - f(y))\right]$ negativno. Kako smo postigli negativnu vrijednost infimuma očekivanja za sve funkcije f koje nisu 1-Lipschitz neprekidne, a promatramo supremum po svim f, dovoljno je promatrati supremum po svim 1-Lipschitz funkcijama jer je za njih očekivanje nula.

Dualni oblik jednadžbe (2.17) označavat ćemo sa W', odnosno

$$W'(p, q) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{s\sim p}[f(s)] - \mathbb{E}_{t\sim q}[f(t)]\right) \tag{2.26}$$

Ako supremum po svim 1-Lipschitz funkcijama zamijenimo sa supremumom po svim K-Lipschitz funkcijama, onda se supremum mijenja iz $W'(p, q)$ u $K \cdot W'(p, q)$. Izračunati supremum po svim K-Lipschitz funkcijama $\{f : \|f\|_L \leq K\}$ i dalje je neizvedivo, ali za razliku od (2.17) lakše je za aproksimirati.

U kontekstu generativnih suparničkih mreža, neka je $\{D_{\theta_D}\}_{\theta_D\in\Theta_D}$ familija 1-Lipschitz neprekidnih funkcija parametriziranih sa θ_D, gdje D_{θ_D} predstavlja diskriminator s parametrima θ_D (U kontekstu WGANova, diskriminator još zovemo i kritičar (eng. *critic*) jer kao izlaz ne daje vrijednost iz intervala $[0, 1]$ kao klasifikator, već bilo koju vrijednost iz $\mathbb{R}$). Također, neka G_{θ_G} predstavlja generator s parametrima θ_G, odnosno $G(z, \theta_G)$ je vrijednost funkcije G u točki (z, θ_G). Neka je p_g distribucija slučajne varijable $G_{\theta_G}(z)$, gdje je z slučajna varijabla s distribucijom p_z. Pomoću Wasserstein udaljenosti (2.17) želimo izračunati sličnost između distribucija p_{data} i p_g pa slijedi

$$W(p_{data}, p_g) = \inf_{\gamma\in\Pi(p_{data},p_g)} \mathbb{E}_{(x,G(z))\sim\gamma}[\|x - G(z)\|]. \tag{2.27}$$

Jednadžbu (2.27) prebacujemo u njen dualni oblik (2.26) pa vrijedi

$$W'(p_{data}, p_g) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim p_{data}}[f(x)] - \mathbb{E}_{z\sim p_z}[f(G(z))]\right). \tag{2.28}$$

Konačno, supremum po $\|f\|_L \leq 1$ ćemo aproksimirati maksimumom po familiji 1-Lipschitz neprekidnih funkcija parametriziranih sa θ_D, $\{D_{\theta_D}\}_{\theta_D\in\Theta_D}$ pa jednadžbu (2.28) interpretiramo kao

$$\max_{\theta_D\in\Theta_D} \left(\mathbb{E}_{x\sim p_{data}}[D_{\theta_D}(x)] - \mathbb{E}_{z\sim p_z}[D_{\theta_D}(G_{\theta_G}(z))]\right) \tag{2.29}$$

i stoga kao funkciju pogreške u Wasserstein GANu uzimamo

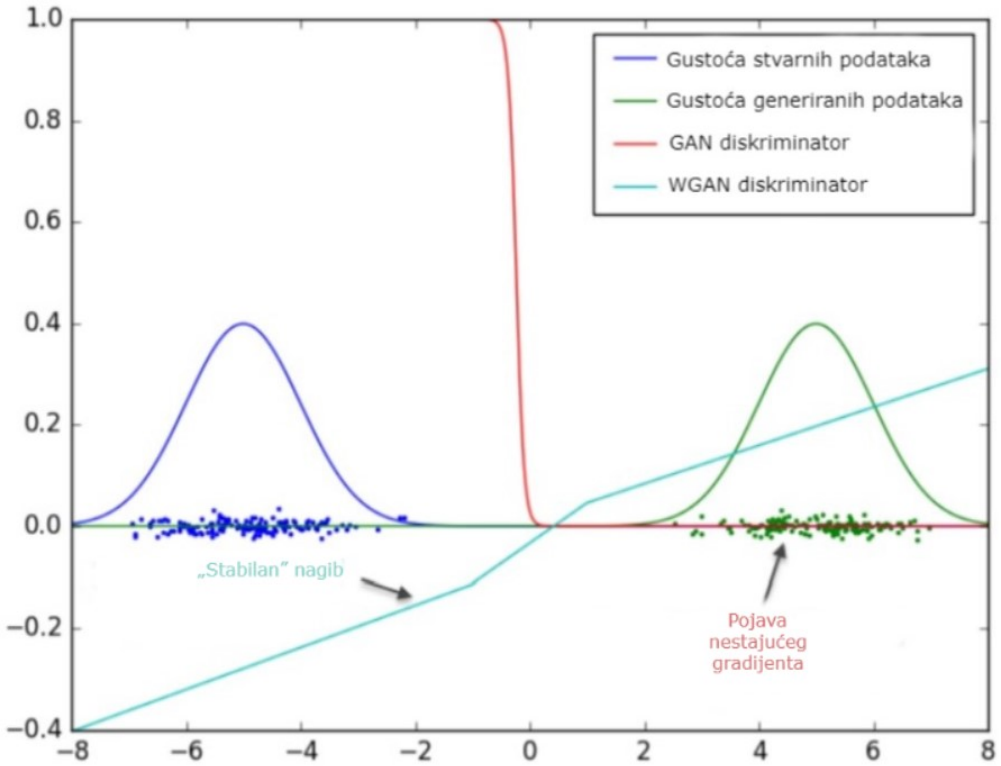
$$W_{GAN}(p_{data}, p_g) = \mathbb{E}_{x\sim p_{data}}[D_{\theta_D}(x)] - \mathbb{E}_{z\sim p_z}[D_{\theta_D}(G_{\theta_G}(z))]. \tag{2.30}$$

Cilj treniranja je postići da p_g odgovara p_{data}. Za fiksni G_{θ_G}, možemo izračunati optimalni D_{θ_D} s obzirom na funkciju pogreške (2.30). Tada, unazadnom propagacijom kroz

$W_{GAN}(p_{data}, p_g)$ računamo gradijent (2.31) za θ_G.

$$\begin{aligned}\nabla_{\theta_G} W(p_{data}, p_g) &= \nabla_{\theta_G} \left(\mathbb{E}_{x \sim p_{data}}[D_{\theta_D}(x)] - \mathbb{E}_{z \sim p_z}[D_{\theta_D}(G_{\theta_G}(z))] \right) \\ &= -\mathbb{E}_{z \sim p_z} \left[\nabla_{\theta_G} D_{\theta_D}(G_{\theta_G}(z)) \right]\end{aligned} \tag{2.31}$$

Naravno, što je diskriminator u (2.31) bolji, to će i gradijent biti „pouzdaniji". Za JS divergenciju, kako diskriminator postaje bolji, tako i gradijent postaje „pouzdaniji" ali dolazi do javljanja problema nestajućeg gradijenta. Za razliku od JS divergencije, empirijski je pokazano da se diskriminatore trenirane na Wasserstein udaljenosti preporuča trenirati do optimalnosti. Na slici 2.3 vidimo da WGAN diskriminator jako brzo konvergira u po dijelovima linearnu funkciju koja na (gotovo) cijeloj domeni daje korisne gradijente. Mogućnost treniranja WGAN diskriminatora do optimalnosti uvelike olakšava treniranje jer više ne moramo balansirati između treniranja generatora i diskriminatora.



Slika 2.3: Optimalni GAN i WGAN diskriminator kada uče razlikovati dvije Gaussove distribucije. Kao što vidimo, treniranje GAN diskriminatora do optimalnosti rezultira nestajućim gradijentom, dok WGAN diskriminator pruža „dobre" gradijente na cijeloj domeni [1].

Ovim postupkom, proces treniranja rastavili smo u tri koraka:

- Za fiksni θ_G, izračunaj aproksimaciju od $W(p_{data}, p_g)$ treniranjem D_{θ_D} do konvergencije.
- Za optimalni D_{θ_D}, izračunaj gradijent (za θ_G) $-\mathbb{E}_{z\sim p_z}\left[\nabla_{\theta_G} D_{\theta_D}(G_{\theta_G}(z))\right]$ uzorkovanjem nekoliko $z \sim p_z$.
- Ažuriraj θ_G i ponovi postupak.

Cijeli postupak vrijedi isključivo kada je svaka funkcija familije $\{D_{\theta_D}\}_{\theta_D\in\Theta_D}$ K-Lipschitzova, za neki faktor K. Svojstvo Lipschitz neprekidnosti osiguravamo rezanjem težina (eng. *weight clipping*) [1, 16]. Rezanjem težina svaki težinski faktor koji je veći od vrijednosti c postavljamo na c, a težinski faktor koji je manji od vrijednosti $-c$ postavljamo na $-c$. Tada je skup funkcija koji zadovoljava svojstvo Lipschitz neprekidnosti podskup K-Lipschitz neprekidnih funkcija za neki K koji ovisi o c i arhitekturi diskriminatora.

Algoritam 2 Wasserstein GAN algoritam. Hiperparametri: veličina mini-grupe $m = 64$, parametar RMSProp-a $\alpha = 0.00005$, faktor rezanja $c = 0.01$ te broj koraka diskriminatora $n_{diskriminator} = 5$. RMSProp je varijanta gradijentnog spusta [15]. Varijable $grad_{\theta_G}$ i $grad_{\theta_D}$ predstavljaju gradijente od θ_G i θ_D.

Input: θ_D, inicijalne vrijednosti parametara diskriminatora. θ_G, inicijalne vrijednosti parametara generatora.

while θ_G nije konvergirao **do**
  for t=1, ..., $n_{diskriminator}$ **do**
    Uzorkuj skup od m primjera $\{z_1, \ldots, z_m\}$ iz distribucije p_z.
    Uzorkuj skup od m primjera $\{x_1, \ldots, x_m\}$ iz distribucije p_{data}.
    $grad_{\theta_D} \leftarrow \nabla_{\theta_D}\left[\frac{1}{m}\sum_{i=1}^{m} D_{\theta_D}(x_i) - \frac{1}{m}\sum_{i=1}^{m} D_{\theta_D}(G_{\theta_G}(z_i))\right]$
    $\theta_D \leftarrow \theta_D + \alpha \cdot RMSProp(\theta_D, grad_{\theta_D})$
    $\theta_D \leftarrow clip(\theta_D, -c, c)$
  end for
  Uzorkuj skup od m primjera $\{z_1, \ldots, z_m\}$ iz distribucije p_z.
  $grad_{\theta_G} \leftarrow -\nabla_{\theta_G}\frac{1}{m}\sum_{i=1}^{m} D_{\theta_D}(G_{\theta_G}(z_i))$
  $\theta_G \leftarrow \theta_G - \alpha \cdot RMSProp(\theta_G, grad_{\theta_G})$
end while

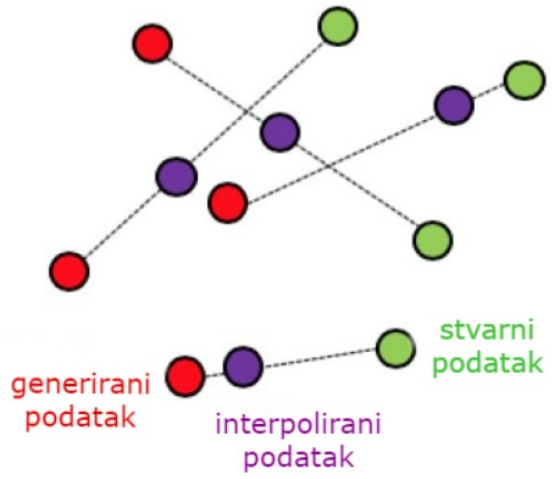
Rezanje težina je loš način osiguravanja Lipschitz neprekidnosti funkcije. Ako je parametar rezanja velik, potrebno je puno vremena dok težine konvergiraju, stoga proces treniranja diskriminatora do optimalnosti dugo traje. Ako je parametar rezanja malen, lako možemo naići na problem nestajućeg gradijenta. U sljedećem poglavlju promotrit ćemo kako efikasnije osigurati Lipschitz neprekidnost.

2.3 Kažnjavanje odstupanja gradijenta

Osim ranije navedenih problema, rezanje težina „usmjerava" diskriminator prema jednostavnijim funkcijama. Glavni problem u procesu optimizacije WGANa prozilazi iz interakcije parametra rezanja c i funkcije pogreške te dovodi do problema nestajućeg ili eksplodirajućeg gradijenta (vidi sliku 2.6). Suprotno nestajućem gradijentu, u problemu eksplodirajućeg gradijenta kako propagiramo gradijent kroz mrežu, gradijent postaje toliko velik da dolazi do prevelike promjene u vrijednosti težine koju ažuriramo i model ne uspijeva konvergirati.

Kao alternativa rezanju težina, za osiguravanje Lipschitz neprekidnosti ponuđena je metoda kažnjavanja odstupanja gradijenta (eng. *gradient penalty*). Kako optimalan WGAN diskriminator ima normu gradijentnog vektora jednaku 1 [6], navedeno svojstvo pokušavamo direktno osigurati uvođenjem pribrojnika u funkciju pogreške koji kažnjava model ako norma gradijentnog vektora diskriminatora odstupa od 1. Ovakav pristup mnogo prirodnije postiže Lipschitz neprekidnost i rezultira daleko stabilnijim procesom treniranja. Model koji ovim principom osigurava Lipschitz neprekidnost zovemo WGAN-GP modelom.

Kako je nemoguće izračunati gradijent po svim točkama, gradijent ćemo računati pomoću skupa nastalog interpolacijom između stvarnih i generiranih podataka (vidi sliku 2.4).



Slika 2.4: Skup podataka nastao interpoliranjem između stvarnih i generiranih podataka.

Preciznije, neka je p_{data} distribucija stvarnih podataka, a p_g distribucija generiranih. Tada implicitno definiramo p_{in} uniformnim uzorkovanjem po dužinama koje spajaju parove točaka od kojih je jedna točka iz p_{data}, a druga iz p_g. Ovakva definicija motivirana je

činjenicom da optimalni diskriminator na dužinama između parova točaka iz p_{data} i p_g ima vrijednost norme gradijentnog vektora jednaku 1 [6].

Konačno, jednadžbu (2.30) modificiramo u

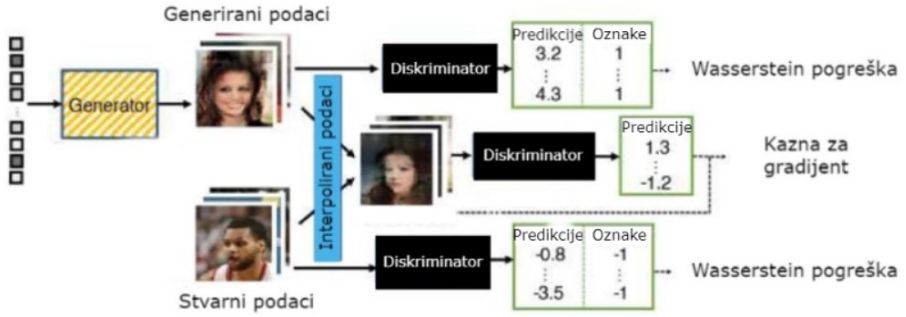
$$\max_G \min_D - \underbrace{\left(\mathbb{E}_{x\sim p_{data}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))]\right)}_{\text{originalna pogreška diskriminatora}} + \underbrace{\lambda\, \mathbb{E}_{\bar{x}\sim p_{in}}\left[(\|\nabla_{\bar{x}} D(\bar{x})\|_2 - 1)^2\right]}_{\text{kažnjavanje odstupanja gradijenta}}. \tag{2.32}$$

Glavne prednosti WGAN modela nad GAN modelima su mogućnost treniranja diskriminatora do konvergencije pa treniranje generatora, dok bi u GAN modelima takvim pristupom naišli na problem nestajućeg gradijenta. Međutim, treniranje WGAN modela s rezanjem težina je i dalje teško ako parametar c nije idealno namješten (vidi sliku 2.6). Najveća prednost WGAN-GP modela je svojstvo lagane konvergencije modela. U WGAN-GP modelu treniranje je stabilno pa potiče na korištenje mnogo kompleksnijih modela za generator i diskriminator, koji su dosada bili nepovoljni zbog problema konvergencije GAN i WGAN modela.

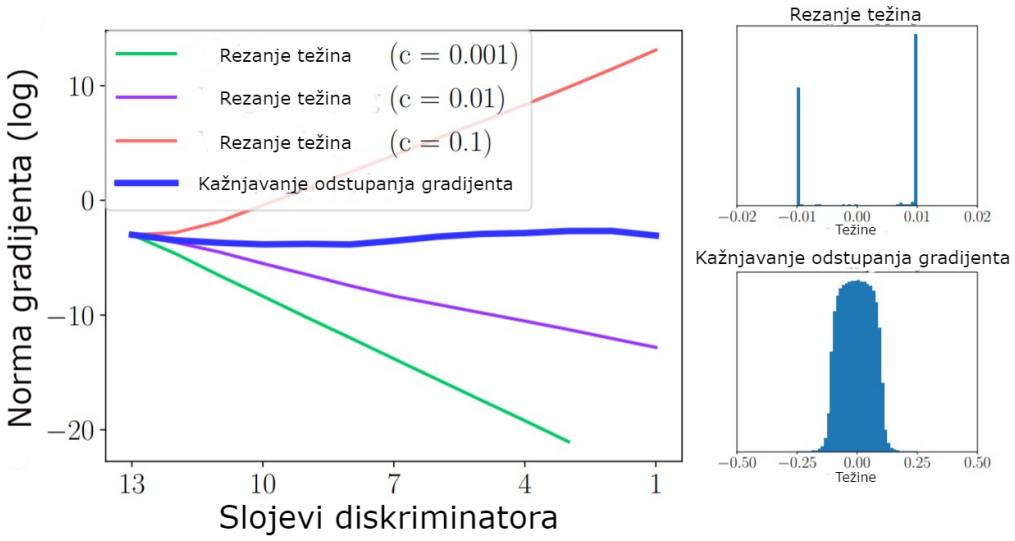
Algoritam 3 Wasserstein GAN algoritam s kažnjavanjem odstupanja gradijenta. Hiperparametri: faktor kažnjavanja odstupanja gradijenta $\lambda = 10$, veličina mini-grupe $m = 64$, parametri metode Adam $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$ te broj koraka diskriminatora $n_{diskriminator} = 5$. Adam je varijanta gradijentnog spusta [15].

Input: θ_D, inicijalne vrijednosti parametara diskriminatora. θ_G, inicijalne vrijednosti parametara generatora.

while θ_G nije konvergirao **do**
  for t=1, ..., $n_{diskriminator}$ **do**
    Uzorkuj skup od m primjera $\{z_1, \ldots, z_m\}$ iz distribucije p_z.
    Uzorkuj skup od m primjera $\{x_1, \ldots, x_m\}$ iz distribucije p_{data}.
    for i = 1, ..., m **do**
      Uzorkuj ϵ uniformno iz $[0, 1]$.
      $\hat{x}_i \leftarrow G_{\theta_G}(z_i)$
      $\bar{x}_i \leftarrow \epsilon x_i + (1-\epsilon)\hat{x}_i$
      $L_i \leftarrow D_{\theta_D}(\hat{x}_i) - D_{\theta_D}(x_i) + \lambda(\|\nabla_{\bar{x}} D_{\theta_D}(\bar{x}_i)\|_2 - 1)^2$
    end for
    $\theta_D \leftarrow Adam(\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} L_i, \theta_D, \alpha, \beta_1, \beta_2)$
  end for
  Uzorkuj skup od m primjera $\{z_1, \ldots, z_m\}$ iz distribucije p_z.
  $\theta_G \leftarrow Adam(\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} -D_{\theta_D}(G_{\theta_G}(z_i)), \theta_G, \alpha, \beta_1, \beta_2)$
end while



Slika 2.5: Model WGANa s kažnjavanjem odstupanja gradijenta [4].



Slika 2.6: Norme gradijenta WGAN diskriminatora ili eksplodiraju ili nestanu (lijevo), međutim stabilne su kada koristimo kažnjavanje odstupanja gradijenta. Rezanje težina gura težinske faktore u krajnje vrijednosti $-c$ i c, dok to nije slučaj u kažnjavanju odstupanja gradijenta [6].

Poglavlje 3

Povećanje razlučivosti slike

Povećanje razlučivosti (eng. *super-resolution*) proces je dobivanja slike visoke razlučivosti iz odgovarajuće slike niske razlučivosti i pronalazi veliku primjenu u problemima računalnog vida. Usprkos naprecima u preciznosti i brzini, duboke konvolucijske mreže ne uspijevaju obnoviti fine detalje slike kada povećavamo razlučivost za veći faktor. Takvi modeli postižu dobre rezultate na metrikama sličnosti (*PSNR*, *SSIM* [9]) između uvećane i originalne slike ali uvećane slike ljudskom promatraču često budu vizualno nezadovoljavajuće. U ovom radu promatramo SRGAN [11], generativnu suparničku mrežu namijenjenu vizualno zadovoljavajućem povećanju razlučivosti slike.

3.1 SRGAN

Funkcija pogreške

Neka je I^{SR} slika povećane razlučivosti dobivena iz slike I^{LR} niske razlučivosti, a I^{HR} odgovarajuća stvarna slika visoke razlučivosti. Za vrijeme treninga, slike niske razlučivosti dobivaju su iz slika visoke razlučivosti primjenom operacije smanjenja razlučivosti (bikubične interpolacije) s faktorom r. Tada sliku I^{LR} definiramo kao tenzor veličine $W \times H \times 3$, a I^{HR}, I^{SR} kao tenzor veličine $rW \times rH \times 3$, gdje H i W predstavljaju visinu i širinu slike, a faktor 3 predstavlja tri vrijednosti RGB modela za reprezentaciju boja.

Cilj treniranja je dobiti generatorsku funkciju G koja kao ulaz prima sliku niske razlučivosti i povećavajući joj razlučivost procjenjuje odgovarajuću sliku visoke razlučivosti. Kao generatorsku mrežu odabiremo unaprijednu konvolucijsku neuronsku mrežu G_{θ_G} s parametrima θ_G.

Po uzoru na [5] definiramo diskriminatorsku mrežu D_{θ_D}, s parametrima θ_D, koju naiz-

mjenično optimiziramo sa G_{θ_G} kako bismo riješili minimax problem

$$\min_{\theta_G} \max_{\theta_D} \left(\mathbb{E}_{I^{HR} \sim p_{data}} \left[\log D_{\theta_D}(I^{HR}) \right] + \mathbb{E}_{I^{LR} \sim p_z} \left[\log \left(1 - D_{\theta_D} \left(G_{\theta_G}(I^{LR}) \right) \right) \right] \right), \tag{3.1}$$

gdje diferencijabilni diskriminator D treniramo da raspozna slike visoke razlučivosti I^{HR} od onih s povećanom razlučivosti $I^{SR} = G(I^{LR})$. Ovakvo treniranje potiče generator da stvara finije detalje kako bi uspio zavarati diskriminator. Za početak, diskriminatorsku mrežu treniramo da riješi maksimizacijski problem iz jednadžbe (3.1), odnosno, koristimo se funkcijom pogreške iz poglavlja 2.1, bez metoda iz poglavlja 2.2 i 2.3.

Parametre θ_G mreže računamo optimizirajući funkciju pogreške C^{SR}. Preciznije, želimo pronaći θ_G^* tako da vrijedi

$$\theta_G^* = \arg\min_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} C^{SR} \left(G_{\theta_G}(I_i^{LR}), I_i^{HR} \right), \tag{3.2}$$

gdje su I_i^{HR}, $i = 1, \ldots, m$, trening slike visoke razlučivosti, a I_i^{LR}, $i = 1, \ldots, m$, odgovarajuće trening slike niske razlučivosti dobivene iz slika visoke razlučivosti. Jednadžbu za C^{SR} definiramo kao težinsku sumu gubitka sadržaja i suparničkog gubitka

$$C^{SR} = \underbrace{C_{MSE}^{SR} + 6 \cdot 10^{-3} C_{VGG}^{SR}}_{\text{gubitak sadržaja}} + \underbrace{10^{-3} C_{Gen}^{SR}}_{\text{suparnički gubitak}} . \tag{3.3}$$

Srednju kvadratnu pogrešku po pikselima C_{MSE}^{SR} definiramo kao

$$C_{MSE}^{SR} = \frac{1}{r^2 WH} \sum_{i=1}^{rW} \sum_{j=1}^{rH} \left(I_{i,j}^{HR} - G_{\theta_G} \left(I^{LR} \right)_{i,j} \right)^2, \tag{3.4}$$

i ova metrika, iako je zaslužna za visoku PSNR vrijednost, ne uspijeva osigurati vizualno zadovoljavajuće rezultate. Kao nadopunu C_{MSE}^{SR} uvodimo tzv. *VGG* pogrešku C_{VGG}^{SR}. VGG pogrešku definiramo pomoću aktivacijskih slojeva već istrenirane VGG19 [17] mreže namijenjene za prepoznavanje objekata na slici. Preciznije, neka $\phi_{k,l}$ označava funkciju koja ulazne vrijednosti provlači kroz VGG19 mrežu i preslikava u vrijednosti aktivacijskog sloja dobivenog nakon primjene l-tog konvolucijskog sloja i aktivacijske funkcije, a prije primjene k-tog sloja sažimanja VGG mreže. VGG pogrešku tada definiramo kao

$$C_{VGG_{k,l}}^{SR} = \frac{1}{W_{k,l} H_{k,l}} \sum_{i=1}^{W_{k,l}} \sum_{j=1}^{H_{k,l}} \left(\phi_{k,l} \left(I^{HR} \right)_{i,j} - \phi_{k,l} \left(G_{\theta_G} \left(I^{LR} \right) \right)_{i,j} \right)^2 \tag{3.5}$$

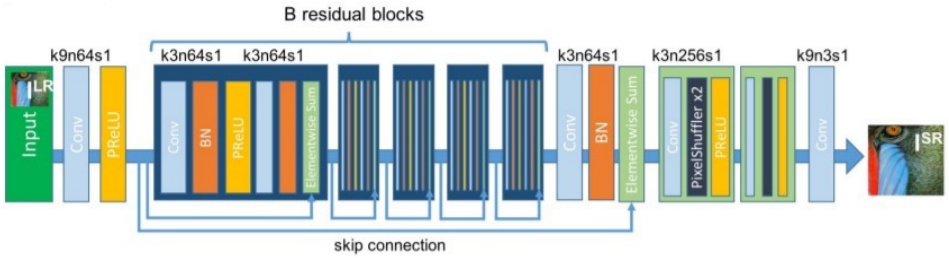
gdje $W_{k,l}$ i $H_{k,l}$ označavaju dimenzije odgovarajućeg aktivacijskog sloja. Za VGG pogrešku uzimamo $C_{VGG_{5,4}}^{SR}$. Kako bismo dobili VGG pogrešku reda veličine MSE pogreške, VGG pogrešku skaliramo za faktor $6 \cdot 10^{-3}$.

Konačno, funkciji pogreške dodajemo i suparnički gubitak koji potiče mrežu da pokuša prevariti diskriminator priklanjajući se rješenjima koji leže u prostoru stvarnih slika. Suparnički gubitak definiramo jednadžbom (3.6),

$$C_{Gen}^{SR} = \sum_{i=1}^{m} \left(-\log D_{\theta_D} \left(G_{\theta_G} \left(I_i^{LR} \right) \right) \right), \tag{3.6}$$

gdje su I_i^{LR}, $i = 1, \ldots, m$ trening slike niske razlučivosti. Vrijednost $D_{\theta_D}\left(G_{\theta_G}\left(I^{LR}\right)\right)$ predstavlja vjerojatnost da diskriminator sliku povećane razlučivosti $G_{\theta_G}\left(I^{LR}\right)$ prihvati kao stvarnu sliku. Kako bismo osigurali bolje ponašanje gradijenta, minimiziramo $-\log D_{\theta_D}\left(G_{\theta_G}\left(I_i^{LR}\right)\right)$ umjesto $\log\left(1 - D_{\theta_D}\left(G_{\theta_G}\left(I_i^{LR}\right)\right)\right)$ [5].

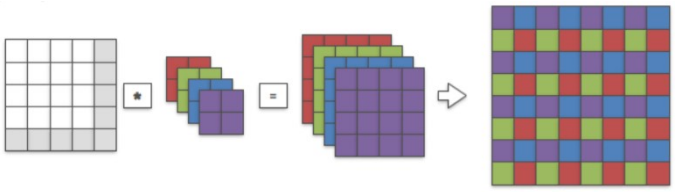
Generator



Slika 3.1: Model generatorske mreže po uzoru na [11], gdje k predstavlja veličinu konvolucijskog prozora, n broj konvolucijskih prozora, a s veličinu pomaka konvolucijskog prozora.

Generatorska mreža (vidi sliku 3.1) kao ulaz prima sliku niske razlučivosti I^{LR} te na nju primjenjuje konvolucijski sloj s parametriziranom ReLU aktivacijom i tako priprema ulaz za $B = 16$ identičnih rezidualnih blokova (vidi poglavlje 1.5). Rezidualni blok sastoji se od konvolucijskog sloja popraćenog normalizacijom nad grupom i parametrizirane ReLU aktivacije iza čega slijedi konvolucijski sloj i normalizacija nad grupom. Konačno, vrijednost na ulazu prvog konvolucijskog sloja direktnom vezom prenosimo i zbrajamo s vrijednosti nakon druge normalizacije nad grupom i dobiveni rezultat predstavlja izlaz rezidualnog bloka. Iza rezidualnih blokova slijedi konvolucijski sloj s normalizacijom nad grupom. Vrijednost aktivacijskog sloja iza parametrizirane ReLU aktivacije direktnom vezom prenosimo i zbrajamo s vrijednostima aktivacijskog sloja iza normalizacije s grupom. Nakon što smo semantički obradili sliku, primjenjujemo dva bloka za povećanje razlučivosti slike

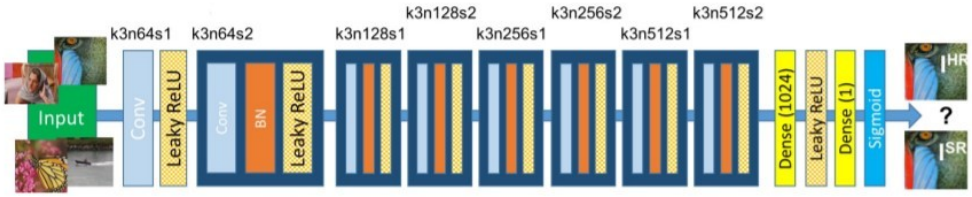
(vidi sliku 3.2). Konačno, primjenjujemo posljednji konvolucijski sloj koji kao izlaz daje sliku povećane razlučivosti I^{SR}.



Slika 3.2: Blok za povećanje razlučivosti prvo primjenjuje konvolucijski sloj pa koristi periodično miješanje vrijednosti za postizanje uvećane matrice.

Kako se generator sastoji isključivo od konvolucijskih slojeva, možemo ga primjenjivati na slike proizvoljnih dimenzija.

Diskriminator



Slika 3.3: Model diskriminatorske mreže [11], gdje k predstavlja veličinu konvolucijskog prozora, n broj konvolucijskih prozora, a s veličinu pomaka konvolucijskog prozora.

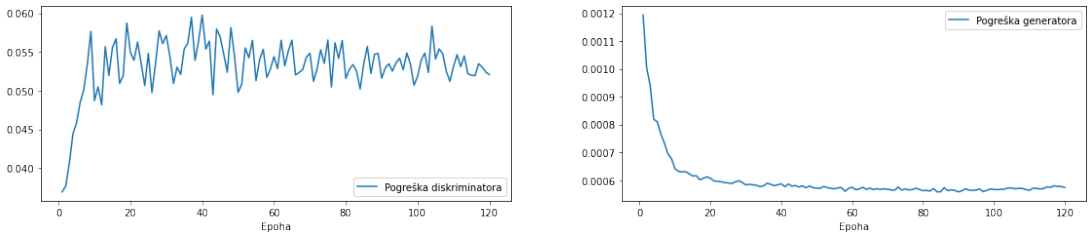
Kako bi razlikovali stvarne slike visoke razlučivosti I^{HR} od onih povećane razlučivosti I^{SR} treniramo diskriminatorsku mrežu (vidi sliku 3.3). Mreža se sastoji od osam konvolucijskih slojeva gdje svaki drugi sloj povećava broj konvolucijskih prozora za faktor dva, odnosno, ukupno od početnih 64 do konačnih 512 prozora. Konvolucije s pomakom dva koristimo kako bismo smanjili dimenzije slike te se nakon takvog sloja visina i širina slike prepolove. Izlaz iz posljednjeg konvolucijskog sloja dočekuju dva potpuno povezana sloja. Na kraj mreže dolazi sigmoidalna aktivacijska funkcija koja računa vjerojatnost pripada li ulazni podatak u slike visoke razlučivosti ili slike povećane razlučivosti.

Treniranje

SRGAN smo trenirali na NVIDIA Tesla P100-PCIE-16GB GPU i na 16000 fotografija iz *VOC2012* skupa podataka [3]. Validacija je provedena na 400 slika iz skupa podataka

VOC2012 različitih od onih iz trening skupa. Proces treniranja trajao je 120 epoha i ukupno vrijeme treniranja je 9 sati. Za svaku mini-grupu nasumično bez ponavljanja odabiremo 16 slika iz trening skupa i nasumično izrežemo 16 podslika visoke razlučivosti dimenzija 96×96. Slike niske razlučivosti postižemo primjenom bikubične interpolacije s faktorom $r = 4$ na slike visoke razlučivosti i dobivamo 16 slika dimenzije 24×24. Intenzitet (vrijednost) svakog piksela slike visoke i niske razlučivosti skaliran je s intervala $[0, 255]$ na interval $[0, 1]$. Za optimizaciju parametara generatora i diskriminatora koristi se Adam optimizator s parametrima $\beta_1 = 0.9$, $\beta_2 = 0.999$ te stopom učenja 0.0001. Treniranje generatora i diskriminatora provedeno je naizmjenično, odnosno, nakon svakog ažuriranja diskriminatora, ažurirali bi i generator, i tako u krug. Kod je implementiran u PyTorchu [14].

Slika 3.4: Lijeva slika predstavlja originalni trening podatak, srednja slika trening podatak visoke razlučivosti (dobiven izrezivanjem), a desna slika trening podatak niske razlučivosti dobiven zamućivanjem (bikubičnom interpolacijom) podatka visoke razlučivosti.



Slika 3.5: Lijeva slika predstavlja pogrešku diskriminatora za vrijeme treniranja SRGANa, dok desna slika predstavlja pogrešku generatora. Kako diskriminator i generator rade jedan protiv drugog, za očekivati je da kako se pogreška jednog poboljša, pogreška drugog će se pogoršati. U početnim epohama generirane slike su loše pa je za očekivati da će diskriminator imati malu, a generator veliku pogrešku. Kako generator postaje sve bolji, pogreška mu opada, a pogreška diskriminatora raste. Konačno, nakon otprilike 50 epoha pogreška generatora i diskriminatora se stabilizira, odnosno modeli polako konvergiraju.

Rezultati

Testiranje provodimo na tri standardna skupa podataka za mjerenje preciznosti povećanja razlučivosti: *Set5* [2], *Set14* [22] i *BSD100* [13]. Sva testiranja provode se na slikama gdje je visina i širina slike niske razlučivosti 4× umanjena u odnosu na sliku visoke razlučivosti, odnosno, slika visoke razlučivosti ima 16× više piksela nego slika niske razlučivosti. Kao metrike sličnosti slika koristili smo SSIM (eng. *structural similarity index measure*) [20] i PSNR (eng. *peak signal-to-noise ratio*) [9] gdje obje metrike postižu višu vrijednost što su slike sličnije. Za identične slike SSIM postiže vrijednost 1, dok PSNR (zbog MSE pogreške u nazivniku) postiže vrijednost $+\infty$.

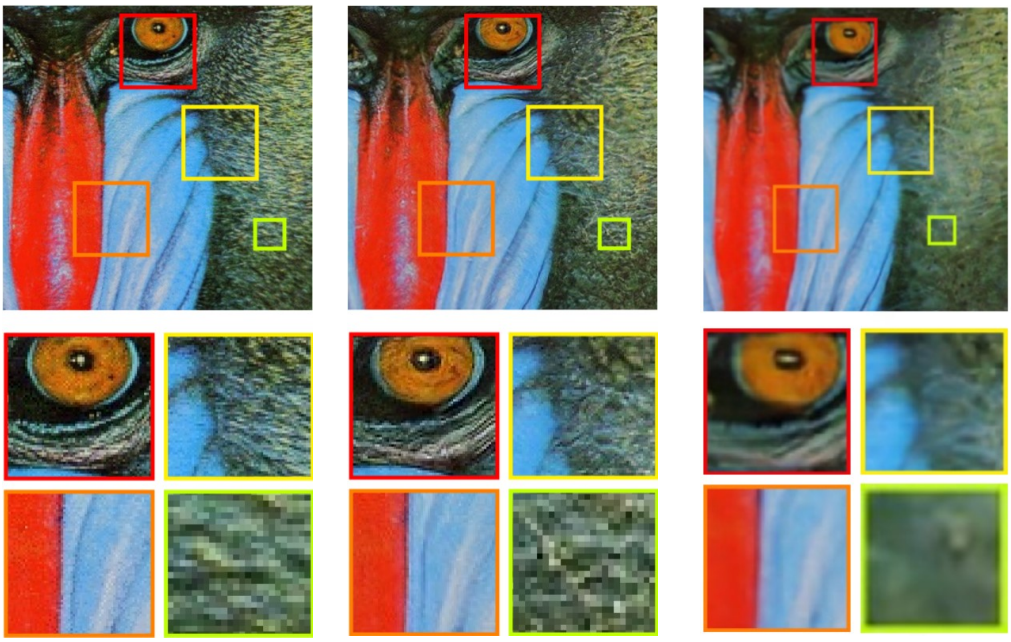
Slika 3.6: Primjer rezultata na jednoj slici iz test skupa Set14. Lijeva slika predstavlja sliku niske razlučivosti, srednja slika originalnu sliku, a desna slika sliku povećane razlučivosti.

	Set5		Set14	
	SRGAN[11]	SRGAN[naš]	SRGAN[11]	SRGAN[naš]
PSNR	29.40	27.15	26.02	24.87
SSIM	0.8472	0.7904	0.7397	0.6779

Tablica 3.1: Prosječna PSNR i SSIM vrijednost SRGAN modela na test skupovima Set5 i Set14.

BSD100	SRGAN [11]	SRGAN [naš]
PSNR	25.16	23.77
SSIM	0.6688	0.6108

Tablica 3.2: Prosječna PSNR i SSIM vrijednost SRGAN modela na test skupu BSD100.



(a) Originalna slika (b) SRGAN [11] (c) SRGAN [naš]

Slika 3.8: Primjer rezultata na jednoj slici iz test skupa Set14. Lijeva slika predstavlja sliku niske razlučivosti, srednja slika originalnu sliku, a desna slika sliku povećane razlučivosti.

Promotrimo li tablice 3.1 i 3.2, te slike 3.6, 3.8 i 3.9 vidimo da su postignuti rezultati našeg SRGAN modela nešto lošiji nego u originalnom radu, ali unatoć tome su vizualno zadovoljavajući i predstavljaju veliko poboljšanje u odnosu na sliku niske razlučivosti.

Slika 3.9: Primjer rezultata na tri slike iz test skupa BSD100. Lijeva slika predstavlja sliku niske razlučivosti, srednja slika originalnu sliku, a desna slika sliku povećane razlučivosti.

3.2 SRWGAN-GP

Kako je SRGAN mreža [11] objavljena prije Wasserstein GANa s kažnjavanjem odstupanja gradijenta koji omogućuje stabilnije treniranje i bolje rezultate, funkciju pogreške iz [6] primjenjujemo na SRGAN mrežu i uspoređujemo rezultate dvije mreže. Mreži s Wasserstein GAN funkcijom pogreške i kažnjavanjem odstupanja gradijenta dat ćemo naziv SRWGAN-GP.

Funkcija pogreške

Kao kod SRGAN modela, cilj je istrenirati unaprijednu konvolucijsku mrežu G_{θ_G}, s parametrima θ_G, da uspješno generira primjerke koje diskriminatorska mreža D_{θ_D} s parametrima θ_D prihvaća. Po uzoru na [6], treniranjem želimo pronaći parametre θ_D tako da jednadžba (3.7) poprimi svoj minimum.

$$-\left(\mathbb{E}_{x\sim p_{data}}[D_{\theta_D}(x)] - \mathbb{E}_{z\sim p_z}[D_{\theta_D}(G_{\theta_G}(z))]\right) + \lambda\, \mathbb{E}_{\bar{x}\sim p_{in}}\left[(\|\nabla_{\bar{x}} D_{\theta_D}(\bar{x})\|_2 - 1)^2\right] \tag{3.7}$$

Za razliku od [5] diskriminator je poželjno trenirati do konvergencije pa često radimo više ažuriranja parametara diskriminatora nego generatora. Cilj treniranja generatora je pronaći parametre θ_G^* kao u (3.2), odnosno optimizirati funkciju pogreške C^{SR} (3.3). U SRWGAN-GPu jednadžbe za C^{SR}_{MSE} (3.4) i C^{SR}_{VGG} (3.5) ostaju iste, dok se jednadžba za suparnički gubitak C^{SR}_{Gen} mijenja iz (3.6) u

$$C^{SR}_{Gen} = -\frac{1}{m}\sum_{i=1}^{m} D_{\theta_D}\left(G_{\theta_G}\left(I_i^{LR}\right)\right), \tag{3.8}$$

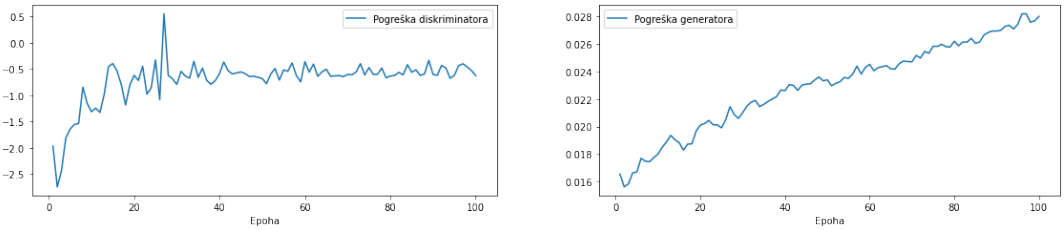
gdje su I_i^{LR}, $i = 1, \ldots, m$ trening slike niske razlučivosti.

Arhitektura SRWGAN-GP

Generatorska i diskriminatorska mreža SRWGAN-GP modela ostaju iste kao u SRGAN modelu, promatramo utjecaj promjene funkcije pogreške na model.

Treniranje

Postupak treniranja SRWGAN-GPa isti je kao i postupak treniranja SRGANa do na sljedeće razlike. Proces treniranja trajao je 100 epoha (8 sati), veličina mini-grupe je 32 te za optimizaciju parametara generatora i diskriminatora koristi se Adam optimizator s parametrima $\beta_1 = 0.0, \beta_2 = 0.9$ [6] te stopom učenja 0.0001.



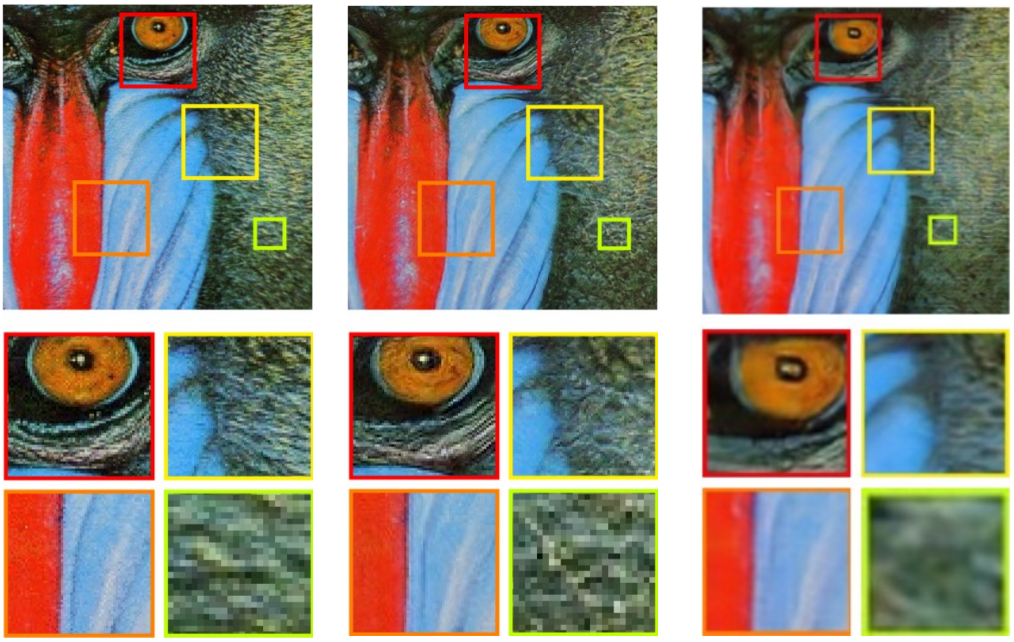
Slika 3.10: Lijeva slika predstavlja pogrešku diskriminatora za vrijeme treniranja SRWGAN-GPa, dok desna slika predstavlja pogrešku generatora. Pogreška diskriminatora naglo propada u negativnu vrijednost te se polako penje prema nuli kako diskriminator konvergira. Vrijednost pogreške diskriminatora stabilizira se oko pedesete epohe. Kako diskriminator postaje bolji, pogreška generatora polako raste, a kako se diskriminator stabilizira, pogreška generatora usporava rast.

Rezultati

Testiranje provodimo na tri standardna skupa podataka za mjerenje preciznosti povećanja razlučivosti: *Set5* [2], *Set14* [22] i *BSD100* [13]. Sva testiranja provode se na slikama gdje je slika niske razlučivosti 4× umanjena u odnosu na sliku visoke razlučivosti, odnosno, slika visoke razlučivosti ima 16× više piksela nego slika niske razlučivosti.

Slika 3.11: Primjer rezultata na jednoj slici iz test skupa Set14. Lijeva slika predstavlja sliku niske razlučivosti, srednja slika originalnu sliku, a desna slika sliku povećane razlučivosti.

Promotrimo li tablice 3.3 i 3.4, vidimo da naš SRWGAN-GP model postiže lošiju PSNR i SSIM vrijednost od SRGANa u originalnom radu. Također, promotrimo li slike 3.11, 3.13 i 3.14 primjećujemo da su slike generirane SRWGAN-GP modelom više zamućene nego slike generirane originalnim SRGANom ali i da postižu fine detalje u usporedbi sa slikom niske razlučivosti.



(a) Originalna slika (b) SRGAN [11] (c) SRWGAN-GP [naš]

Slika 3.13: Primjer rezultata na jednoj slici iz test skupa Set14. Lijeva slika predstavlja sliku niske razlučivosti, srednja slika originalnu sliku, a desna slika sliku povećane razlučivosti.

	Set5		Set14	
	SRGAN [11]	SRWGAN-GP [naš]	SRGAN [11]	SRWGAN-GP [naš]
PSNR	29.40	26.94	26.02	24.77
SSIM	0.8472	0.7669	0.7397	0.6674

Tablica 3.3: Prosječna PSNR i SSIM vrijednost SRWGAN-GP modela na test skupovima Set5 i Set14.

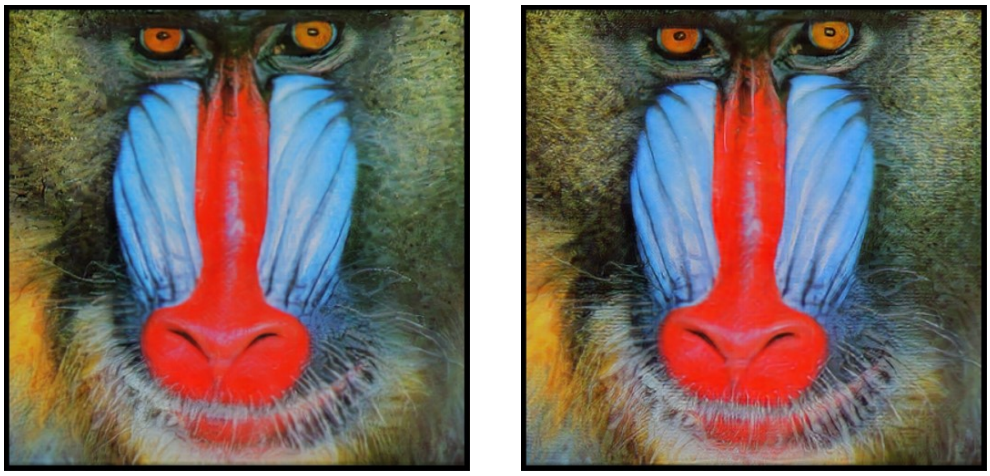
Slika 3.14: Primjer rezultata na tri slike iz test skupa BSD100. Lijeva slika predstavlja sliku niske razlučivosti, srednja slika originalnu sliku, a desna slika sliku povećane razlučivosti.

BSD100	SRGAN [11]	SRWGAN-GP [naš]
PSNR	25.16	23.66
SSIM	0.6688	0.6041

Tablica 3.4: Prosječna PSNR i SSIM vrijednost SRWGAN-GP modela na test skupu BSD100.

3.3 Zaključak

Konačno, usporedimo li naš SRGAN sa SRWGAN-GPom, primjećujemo da na slici 3.15 SRWGAN-GP postiže nešto višu razinu detalja i oštrinu nego SRGAN, naročito ako usporedimo dlaku babuna. Međutim, promotrimo li tablice 3.5 i 3.6 primjećujemo da SRGAN postiže nešto višu PSNR i SSIM vrijednost od SRWGAN-GPa. Puni potencijal WGAN-GP modela u problemu povećanja razlučivosti zvan ESRGAN moguće je vidjeti u [19], gdje ESRGAN premašuje sve prethodne pristupe kako u oštrini slike, tako i u razini detalja.



(a) SRGAN [naš] (b) SRWGAN-GP [naš]

Slika 3.15

	Set5		Set14	
	SRGAN [naš]	SRWGAN-GP [naš]	SRGAN [naš]	SRWGAN-GP [naš]
PSNR	27.15	26.94	24.87	24.77
SSIM	0.7904	0.7669	0.6779	0.6674

Tablica 3.5: Prosječne PSNR i SSIM vrijednosti na test skupovima Set5 i Set14.

BSD100	SRGAN [naš]	SRWGAN-GP [naš]
PSNR	23.77	23.66
SSIM	0.6108	0.6041

Tablica 3.6: Prosječne PSNR i SSIM vrijednosti na test skupu BSD100.

Bibliografija

[1] M. Arjovsky, S. Chintala i L. Bottou, *Wasserstein GAN*, Proceedings of the 34th International Conference on Machine Learning **70** (2017), 214–223.

[2] M. Bevilacqua, A. Roumy, C. Guillemot i M. Alberi Morel, *Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding*, Proceedings of the British Machine Vision Conference, BMVA Press, 2012, str. 135.1–135.10, ISBN 1-901725-46-4.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn i A. Zisserman, *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[4] D. Foster, *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*, O'Reilly Media, 2019, ISBN 9781492041894.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville i Y. Bengio, *Generative Adversarial Networks*, Advances in neural information processing systems **27** (2014), 2672–2680.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin i A. Courville, *Improved training of Wasserstein GANs*, Advances in Neural Information Processing Systems, sv. 30, 2017.

[7] K. He, X. Zhang, S. Ren i J. Sun, *Deep residual learning for image recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, str. 770–778.

[8] V. Herrmann, *Wasserstein GAN and the Kantorovich-Rubinstein Duality*, 2017, `https://vincentherrmann.github.io/blog/wasserstein/`, posjećena u siječnju 2022.

[9] A. Horé i D. Ziou, *Image Quality Metrics: PSNR vs. SSIM*, 2010 20th International Conference on Pattern Recognition, 2010, str. 2366–2369.

[10] Dive into Deep Learning, *Residual Networks (ResNet)*, 2020, `https://d2l.ai/chapter_convolutional-modern/resnet.html`, posjećena u veljači 2022.

[11] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., *Photo-realistic single image super-resolution using a generative adversarial network*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, str. 4681–4690.

[12] Z. Lu, H. Pu, F. Wang, Z. Hu i L. Wang, *The expressive power of neural networks: a view from the width*, Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, 2017, str. 6232–6240.

[13] D. Martin, C. Fowlkes, D. Tal i J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, sv. 2, 2001, str. 416–423.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai i S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, str. 8024–8035.

[15] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv preprint arXiv:1609.04747 (2016), `https://arxiv.org/abs/1609.04747`.

[16] K. G. Sargent, *Lipschitz Constraints in Generative Adversarial Networks*, Magistarska radnja, Harvard, 2019.

[17] K. Simonyan i A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 3rd International Conference on Learning Representations, ICLR, 2015.

[18] C. Villani, *Optimal Transport: Old and New*, Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg, 2008, ISBN 9783540710509.

[19] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, Dong C., C. Loy, Y. Qiao i X. Tang, *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*, Computer Vision – ECCV 2018 Workshops, Springer International Publishing, 2019, str. 63–79, ISBN 978-3-030-11021-5.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh i E. P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE Transactions on Image Processing **13** (2004), br. 4, 600–612.

[21] L. Weng, *From GAN to WGAN*, 2017, http://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html, posjećena u veljači 2022.

[22] R. Zeyde, M. Elad i M. Protter, *On Single Image Scale-Up Using Sparse-Representations*, Curves and Surfaces, 2012, str. 711–730, ISBN 978-3-642-27413-8.

Sažetak

Povećanje razlučivosti česta je tema istraživanja u području računalnog vida. Problem povećanja razlučivosti je proces dobivanja slike visoke razlučivosti iz odgovarajuće slike niske razlučivosti i pronalazi veliku primjenu u satelitskom snimanju, obradi medicinskih prikaza, analizi fotografija teksta te biometrijskom prepoznavanju.

U ovom radu dali smo pregled teorije generativnih suparničkih mreža, modela koji se sastoje od dvije suprotstavljene mreže, generatorske i diskriminatorske, gdje generatorska ima zadatak naučiti distribuciju ulaznih podataka s ciljem generiranja novih, uvjerljivih podataka, a diskriminatorska nastoji postići što veći uspjeh u raspoznavanju stvarnih podataka od generiranih. Također, promotrili smo dvije funkcije pogreške, prvu temeljenu na binarnoj unakrsnoj entropiji, a drugu na Wasserstein udaljenosti i kažnjavanju odstupanja gradijenta.

Konačno, predložili smo dva modela za rješavanje problema povećanja razlučivosti temeljena na generativnih suparničkim mrežama, prvog s funkcijom pogreške temeljenom na binarnoj unakrsnoj entropiji, a drugog s funkcijom pogreške temeljenoj na Wasserstein udaljenosti i kažnjavanju odstupanja gradijenta. Takvi modeli pokazali su se kao kvalitetna zamjena za duboke konvolucijske modele i postigli su fine detalje na slikama povećane razlučivosti bez potrebe za velikim brojem konvolucijskih slojeva.

Summary

Single image super-resolution is a common research topic in the field of computer vision. The problem of single image super-resolution is the process of obtaining a high-resolution image from a corresponding low-resolution image and finds great application in satellite imaging, medical imaging, text-to-photo analysis and biometric recognition.

In this thesis, we provide an overview of the theory of generative adversarial networks which are models of two adversarial networks, generator and discriminator, where the generator has the task of learning the distribution of input data to generate new, compelling data, and the discriminator seeks to achieve success in recognizing real data from generated. We also considered two error functions, the first based on binary cross entropy and the second on Wasserstein distance and gradient penalty.

Finally, we proposed two models to solve the problem of single image super-resolution based on generative adversarial networks, the first with a loss function based on binary cross entropy, and the second with a loss function based on Wasserstein distance and gradient penalty. Such models have proven to be a quality replacement for deep convolutional models and have achieved fine detail in high-resolution images without the need for a large number of convolutional layers.

Životopis

Rođen sam u Karlovcu, 7. siječnja 1998. godine. Odrastao sam u Mravincima. Pohađao sam Osnovnu školu kraljice Jelene u Solinu, a 2012. upisao sam III. gimnaziju u Splitu. Tijekom osnovne i srednje škole razvio sam interes za matematiku i informatiku.

Preddiplomski sveučilišni studij matematike na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu upisao sam 2016., a završio 2019. godine. Iste sam godine upisao diplomski studij *Računarstvo i matematika*, također na Prirodoslovno-matematičkom fakultetu. Tijekom preddiplomskog studija aktivno sam sudjelovao u radu udruge *Mladi nadareni matematičari "Marin Getaldić"*.