UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Interdisciplinarni študijski program računalništvo in matematika – 2. stopnja

Nejc Vesel

NASLOV VAŠEGA DELA

Magistrsko delo

Mentor: prof. dr. Peter Peer

Ljubljana, 2018

Zahvala

Neobvezno. Zahvaljujem se ...

Kazalo

Program dela

Osnovna literatura

Literatura mora biti tukaj posebej samostojno navedena (po pomembnosti) in ne le citirana. V tem razdelku literature ne oštevilčimo po svoje, ampak uporabljamo okolje itemize in ukaz plancite, saj je celotna literatura oštevilčena na koncu.

Podpis mentorja:

Naslov vašega dela

Povzetek

Tukaj napišemo povzetek vsebine. Sem sodi razlaga vsebine in ne opis tega, kako je delo organizirano.

English translation of the title

Abstract

An abstract of the work is written here. This includes a short description of the content and not the structure of your work.

Math. Subj. Class. (2010): oznake kot 74B05, 65N99, na voljo so na naslovu `http://www.ams.org/msc/msc2010.html?t=65Mxx`

Ključne besede: integracija, kompleks

Keywords: integration, complex

1 Sorodna dela

Področje staranja obrazov je pred razmahom globoke učenja večinoma uporabljalo modelne pristope [6]. S pomočjo modeliranja strukture človeškega obraza in povezane anatomije se je želelo simulirati vplive staranja na človeški videz. Kot predpogoj potrebujemo način za modeliranje človeškega obraza in njegovih mimik. Veliko raziskovalnega dela v tej smeri pride iz področja animacije. Ločimo tri glavne vrste modelov in sicer **geometrične, image-based,appearance-based**.
Pri geometričnih modelih ustvarimo mrežo ključnih točk na človeškem obrazu. S transformacijami nad temi točkami pa lahko simuliramo mimiko in staranje. Potrebujemo način, ki nam ob transformacij še vedno ohrani osnovno strukturo objekta. Za to obstaja več pristopov, eden od najbolj znanih pa je predstavljen v [4], in se tudi uporablja na podrčju zaznave ključnih točk na obrazu.
Slikovno osnovani modeli se trudijo ustvariti fotorealistične slike na podlagi drugih slik. Eden od primerov je prenos teksture iz ene slike na drugo s pomočjo precesiranja slikovnih signalov in geometričnega modeliranja [11]. Podobne tehnike osnovane na prenašanju lastnostni iz prototipnega obraza se uporabljajo za prenos osvetlitve, izraza in starosti [7].
Izgledni modeli za razliko od prejšnjih dveh uporabljajo statistično učenje za izgradnjo modela. V večini primerov se iz velike baze podatkov zgradi generični prototipni model. Uporablja se tAAM model [3], ki ga s pomočjo učne množice naučimo statistični model obraza. **TUKAJ NADALJUJ — PLACEHOLDER, KER NEVEM KAKO BI TO ZASTAVU**

Za razliko od zgoraj opisanih pristopov pa modernejši pristopi uporabljajo nevronske mreže za dosego istih ciljev. Z uporabo generativnih model želimo učenje statističnega modela stranja prepustiti nevronski mreži, ki se na osnovi večje baze fotografij oseb različnih starosti sama nauči kako poteka staranje človeškega obraza.

1.1 Staranje z uporabo pogojnih generativnih nasprotniških mrež

Eden od pristopov, opisan v [1], simulira staranje s pomočjo **pogojne generativne nasprotniške mreže**. Glavna ideja razdeli postopek na tri dele

1. Naučimo pogojno generativno nasprotniško mrežo
2. Glede na podano sliko x starosti y_0, poišči latentni vektor z^* pri katerem generator zgenerira sliko, ki je najbolj podobna podani. Torej želimo minimizirati razliko med x in $\hat{x} = G(z^*, y_0)$
3. Staranje dosežemo tako, da generatorju ob optimalnem vektorju z^* namesto originalne starosti podamo ciljno starost y_{cilj}, torej $x_{cilj} = G(z^*, y_{cilj})$.

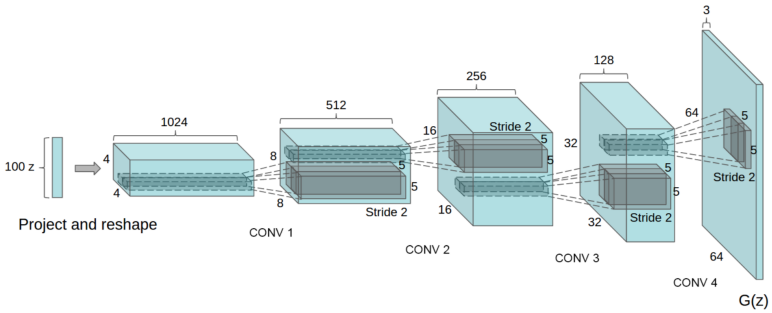
Zelo pomembna je informacija o arhitekturi nevronske mreže, ki nam generira slike. Tukaj se znotraj mreže uporabljajo konvolucijski sloji v generatorju in dekonvolucije v diskimnatorju. Za doseganje stabilnosti učenja je priporočeno upoštevanje nekaterih osnovih smernic [13]

- Vse pooling layerji (?) zamenjamo s koračnimi konvolucijami v diskriminatorju in obratno koračnimi (fractional strided) konvolucijami v generatorju
- Paketna normalizacijo uporabljamo tako v generatorju kot v diskriminatorju
- Pri bolj globokih arhitekturah ne smemo uporabljati polno povezanih slojev
- V generatorju uporabljamo ReLU aktivacijski funkcijo v vseh slojih, razen v zadnjem
- V diskriminatroju za vse sloje uporabljamo LeakyReLU aktivacijsko funkcijo

V tem primeru generator na vhod sprejme vektor šuma z dimenzije 100x1. Nato ga s pomočjo polnopovezanega sloja in preoblikovanja razširi na dimenzijo 1024x4x4, katero nato s pomočjo zaporedja obratno koračnih kovolucij spremenimo v dimenzijo slike, ki je 64x64x3. To je najbolje vidno na sliki 1, ki nazorno prikaže arhitekturo. Oblika diskriminatorja je simetrična tej, z razliko da namesto obratno koračnih, uporabljamo koračne kovolucije z velikostjo koraka 2. Ločite se tudi v tem, da je zadnji sloj v diskriminatorju polnopovezan z velikostjo 1, saj je cilj samo vračanje enega bita informacije.
V drugem koraku želimo gelde na podano sliki x starosti y_o poiskati latentni vektor, ki ustvari sliko $\hat{(x)}$, ki je najboljši približek podani sliki x. V nasprotju z autoenkoderji, nam generativne nasprotniške mreže ne podajo možnosti za preslikavo slike x z atributi y v latentni vektor z. Imamo torej definirano preslikavo $f : (z, y) \mapsto x$ želimo pa dobiti preslikavo $f^{-1} : (x, y) \mapsto z$
To lahko rešimo tako da naučimo enkoder E, nevronska mrežo zadolženo za aproksimiranje f^{-1}. Najprej ustvarimo sintetično množico sto tisoč parov $(x_i, G(z_i, y_i))$, kjer so $z_i \sim N(0, I)$ naključni latentni vektorji ter $y_i \sim U$ enakomerno naključno izbrane oznake starostnih skupin. $G(z_i, y_i)$ je na učni množici obrazov in starostni naučena pogojna nasprotniška generativna mreža (CGAN). Enkoder je naučen tako, da minimiziramo evklidsko razdaljo med aproksimiranimi latentnimi vektorji $E(x_i)$ in z_i, kjer je $x_i = G(z_i, y_i)$
Dobljeni rezultat je aproksimacija, ki jo želimo še dodatno izboljšati z uporabo optimizacijskih algoritmov. Cilj je minimizirati razdaljo med x in $\hat{(x)}$. Rezultati se precej razlikujejo odvisno od izbrane mere razdalje. Najenostavnejši pristop je uporaba evklidkse razdalje na razliki med slikovnimi točkami. Težava nastane, ker metoda gleda razlike med posameznimi piksli, ki velikokrat nimajo vpliva na ohranjanje identitete. Optimizacijska metoda npr. optimizira razlike v ozadju, frizuri ipd, čeprav bi želeli da se identiteti oseb čim bolj približati. Pomanjkljivost je tudi to, da nam slike zabriše.
Alternativni način je uporaba optimizacije ki ohranja identitete. Če imamo nevronsko mrežo FR naučeno v namene razpoznave obrazov, lahko definiramo razdaljo kot razliko med reprezentacijah v tej mreži. V največ primerih gledamo evklidsko razdaljo med tenzorji definiranimi znotraj enega od slojev. Velja torej:

$$z^* = \underset{z}{\text{argmin}} = ||FR(x) - FR(\hat{x})||_{L_2}$$

Ker sta tako generator $G(z, y)$ kot nevronska mreža FR odvedljiva glede na vhodne podatke, potem se optimizacijo lahko rešuje z uporabo **L-BFGS-B** algoritma, ki mu kot začetni približek podamo vrednost z_0, ki smo jo dobili iz enkoderja E.



Slika 1: Struktura generatorja uporabljenega v [1].Vir: [13]

1.2 Staranje z uporabo ponavljajočih nevronskih mrež

Klasične ponavljajoče nevronske mreže podajajo dobre rezultate, kadar se ukvarjamo z zaporednimi podatki, kjer so členi medsebojno odvisni, to pa lahko izkoristimo za simuliranje postopka staranja [**?**] Staranje si lahko predstavljamo kot zaporedje stanj, kjer je vsako novo stanje odvisno od prejšnjega. Starosti razdelimo na starostne skupine, cilj pa je najti prehode iz enega stanja v naslednjega. Postopek je sestavljen iz dveh glavnih korakov, kjer je prvi normalizacija obraznik slik, drugi pa staranje s pomočjo mreže. Pomembno je, da nam normalizacija ohrani podatke o starosti ter lepe prehode med starostnimi skupinami. Za dosego tega cilja se poslužimo tehnike, kjer normaliziramo slike sosednih starostnih skupin skupaj. Uporablja se optični tok, saj ohrani teksturne podrobnosti na slikah. Podrobnosti in matematična formulacija so podrobneje predstavljene v [**?**]. Ko imamo slike normalizirane, se poslužimo ponavljajoče nevronske mreže, za izvajanje postopka staranja. Osnovna komponenta mreže so dvoslojna GRU (gated recurrent unit) vrata, kjer spodnji GRU zakodira vhodni obraz v skrito visokodimenzionalno reprezentacijo, ki jo zgornji del odkodira v postaran obraz. Kot našo kriterijsko funkcijo določimo razliko med vhodno sliko in referenčno sliko. Uteži postopoma povečujemo tako, da ima največjo težo razlika med najstarejšo starostno skupino in referenco. S tem omogočimo bolj realne prehode med stanji.

1.3 Staranje s pogojnim nasprotniškim autoenkoderjem

Predpostavimo da slike oseb pri različnih starostih ležijo na mnogoterosti $\mathcal{M}$. Premikanje po tem prostoru v določeni smeri nam ohrani identiteto, kljub temu da se starost spreminja. Želimo določiti preslikavo med nižjedimenzionalnim latentnim prostorom in mnogoterostjo, saj je direktna preslikava med sliko in mnogoterstjo pretežka za modeliranje.
Imamo sliko obraza x, ki jo enkoder E preslika v latentno reprezentacijo z, ki jo nato združimo z informacijo o starosti osebe l. S tem dobimo točko na $\mathcal{M}$. Glavna ideja je, da sta v latentnem prostoru informaciji o identiteti z in starosti l ločeni. To pomeni, da lahko samo spremenimo starost in s tem ohranimo identiteto osebe. S pomočjo generatorja G to preslikamo nazaj v prostor slik, ki so človeku berljive. Za enkoder uporabimo konvolucijsko nevronsko mrežo, kjer se za namene downsamplinga (?) uporablja koračenje s korakom 2, kot je priporočeno v [13]. Podobno

velja za generator. Imamo tudi dva diskriminatorja:

- D_z je povezan z enkoderjem in skrbi, da latentni vektorji z sledijo normalni porazdelitvi. S tem želimo prisiliti E da je latentni prostor enakomerno zapolnjen.
- D_{img} prisili generator, da so rezultati fotorealistični, še posebej se to vidi pri teksturi starejših obrazov.

Pri podanem latentnem vektorju z ter oznaki l, nam generator G ustvari novo sliko $\hat{x} = G(z, l) = G(E(x), l)$. Želimo, da $\hat{x}$ leži na na $\mathbb{M}$ in ima isto identiteto kot x. Matematično je naš cilj dobiti rešitev enačbe

$$\min_{E,G} \mathcal{L}(x, G(E(x), l))$$

. ter hkrati posrbeti, da je z normalno porazdeljen ter da je kriterij diskriminatorja D_{img} čim bolj izpolnjen.
Porazdelitev učnih podatkov definiramo kot $p_{data}(x)$ in porazdelitev z definiramo kot $q(z|x)$. Naj bo $p(z)$ priorna porazdelitev, potem z $z^* \sim p(z)$ definiramo vzorčenje iz $p(z)$. Skupno učenje E in D_z predstavimo s kriterijsko funkcijo

$$\min_{E} \max_{D_z} \mathbb{E}_{z^* \sim p(z)}[\log D_z(z^*)] + \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_z(E(x)))]$$

. Analogno lahko definiramo kriterijsko funkcijo za učenje diskrimanatorja D_{img} in generatorja G z oznako l kot

$$\min_{G} \max_{D_{img}} \mathbb{E}_{x,l \sim p_{data}(x,l)}[\log D_{img}(x, l)] + \mathbb{E}_{x,l \sim p_{data}(x,l)}[\log(1 - D_{img}(G(E(x), l)))]$$

Vse skupaj sedaj seštejmo in združimo v celotno kriterijsko funkcijo sistema

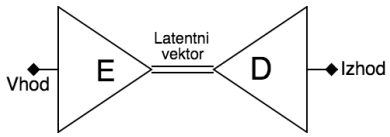
$$\begin{aligned} \min_{E,G} \max D_z, D_{img} \lambda \mathcal{L}(x, G(E(x), l)) \\ + \mathbb{E}_{z^* \sim p(z)}[\log D_z(z^*)] \\ + \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_z(E(x)))] \\ + \mathbb{E}_{x,l \sim p_{data}(x,l)}[\log D_{img}(x, l)] \\ + \mathbb{E}_{x,l \sim p_{data}(x,l)}[\log(1 - D_{img}(G(E(x), l)))] \end{aligned} \tag{1.1}$$

2 Osnovni generativni modeli

2.1 Autoenkoder

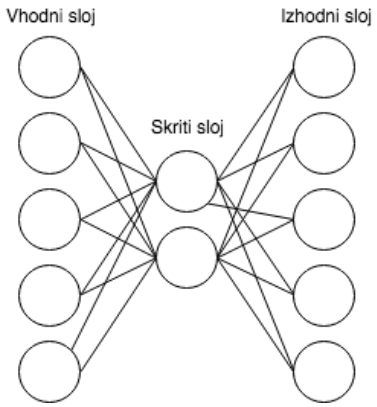
Ena od glavnih arhitektur na področju generativnih modelov je autoenkoder. Avtoenkoderji so sestavljeni iz dveh glavnih delov, **enkoderja** E in **dekoderja** D. Cilj enkoderja je stisniti vhodne podatke v latentno reprezentacijo manjše dimenzije, cilj dekoderja pa je iz latente reprezentacije rekonstruirati vhodne podatke. Želimo torej

$$E(x) = y \text{ in } D(y) \approx x$$



Slika 2: Idejna shema oblike autoenkoderja.

V splošnem se bo v procesu kodiranja in odkodiranja vedno zgodila izguba informacij, saj je latentni prostor manjše dimenzije kot vhodni podatek. Naprimer, sliko velikost 28x28 točk stisnemo v vektor velikost 10x1.
Nevronsko mrežo učimo tako, da želimo da je rekonstuiran podatek čim bolj podoben vhodnemu. Na prvi pogled se zdi, da je uporabnost avtoenkoderjev omejena na kompresijo podatkov, vendar obstaja velika uporabnost na področju generativnih modelov. Ker je latentni prostor manjše dimenzije, prisilimo mrežo, da v latentni prostor zakodira čim več informacije in tako ohrani najpomembnejše značilke na slikah.
V praksi je to koristno v namene razšumljanja slik, inpaintinga. V teh primerih mrežo naučimo da iz nepopolnih (manjkajočih, šumnatih) slik zgenerira čiste slike.
V najbolj enostavni obliki je autoenkoder sestavljen iz treh slojev. Vhodni sloj je polno povezav z edinim skritim slojem, ki je nato polno povezan z izhodnim slojem. V praksi je najpogostejša izboljšava osnovne arhitekture oblika z več polnopovezanimi skritimi sloji, ki se zmanjšujejo v velikosti na strani enkoderja in povečujejo na strani dekoderja. Če delamo s podatki kot so slike, lahko polnopovezane sloje zamenjamo s konvolucijskimi sloji različnih velikosti.
Dimenzije skritih slojev in latentnega vektorja so odvisne od dimenzij vhodnih podatkov ter od tipa podatkov s katerimi delamo. Parametri modela morajo biti prilagojeni našim podatkom in cilju, ki ga želimo doseči.

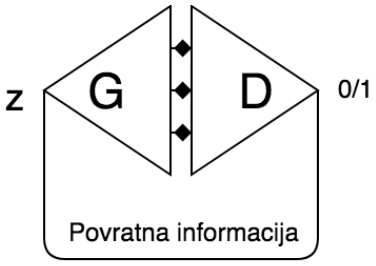


Slika 3: Diagram najenostavnejšega autoenkoderja.

2.2 Generativne nasprotniške mreže

Generativne nasprotniške mreže so bile prvič predstavljene v članku [8], kjer je predstavljen algoritem tudi podkrepljen s teoretičnimi dokazi. Glavna ideja algoritma je, da pomerimo generativni model proti nasprotniku, ki določa ali je generiran rezultat

podoben tistemu, ki ga želimo modelirati. Predstavljamo si lahko bitko med ponarejevalcem denarja in strokovnjakom, ki določa ali je kos denarja pristen. Želimo, da oba akterja skozi iterativni nasprotniški proces izboljšujeta drug drugega.
I dejno shemo mreže lahko vidimo na sliki 4. To lahko primerjamo s shemo autoenkoderja 2 in razlike v arhitekturi so očitne.



Slika 4: Generator na vhod dobi šum z iz katerega zgenerira rezultat, diskriminator pa pove ali misli, da je slika iz učne množice ali ne

Rečemo, da sta oba diskriminator kot generator večslojni nevronski mreži. Matematično gledano, želimo naučiti generator G, da bo proizvajal vzorce, katerih porazdelitev je podobna porazdelitvi podatkov, katere želimo modelirati. Da bi se naučili porazdelitev generatorja p_g na podatkih x, definiramo priorni porazdelitev $p(z)$, kjer z predstavlja vektor šuma.
Preslikavo v prostor podatkov predstavimo z $G(z;\theta_g)$, kjer je G odvedljiva funkcija, ki jo predstavlja večslojna nevronska mreža in θ_g njeni parametri. Prav tako definiramo $D(x,\theta_d)$, katerega izhod je skalar $D(x)$,ki predstavlja verjetnost, da je x prišel iz podatkov in ne iz p_g (torej ni bil generiran s pomočjo generatorja). Hočemo torej, da diskriminator za vhod vedno pravilno določi ali je prišel iz množice realnih podatkov oz. ali je bil generiran s pomočjo generatorja G. To zapišemo kot:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

Psevdokoda algoritma, ki uporablja gradientni spust za implementacijo našega generativnega nasprotniškega modela, je sledeča:

Parameter k za število iteracij notranje zanke, določimo glede na eksperimentalne korake. Pove nam kolikokrat naredimo korak optimizacije diskriminatorja za vsak korak generatorja. Želimo, da generator in diskriminator ostajata v ravnovesju. Če eden od njiju postane preveč učinkovit, težko pride do nadaljnjega izboljšanja v kvaliteti rezultatov, kar je tudi ena glavnih slabosti tega pristopa. Na začetku učenja, ko je G slab, se v praksi lahko zgodi, da D z lahkoto zavrne vse generirane vzorce, saj so očitno različni od podatkov iz učne množice. V tem primeru se vrednost $\log(1 - D(G(z)))$ močno poveča in težko najdemo globalni minimum. Da se izognemo temu problemo, reformuliramo problem tako, da namesto minimizacije $\log(1 - D(G(z)))$ učimo G, da maksimizira $\log D(G(z))$. To nam omogoča. da na začetku učenja dobimo močnešje gradiente, ki omogoča boljše premike v smer optimalne rešitve.
Ena od možnih razširitev omenjenih v članku je razširitev na pogojni generativni model $p(x|c)$, kar dobimo tako, da dodamo pogoj c na vhod tako generatorju kot diskriminatorju.

Algoritem 1 Učenje generativnega nasprotniškega modela s pomočjo gradientnega spusta

for št. iteracij treninga **do**
  for k korakov **do**
    Vzorči m vzorcev $\{z^{(1)}, \ldots, z^{(m)}\}$ iz priorne porazdelitve $p_g(z)$
    Vzorči m vzorcev $\{x^{(1)}, \ldots, x^{(m)}\}$ iz porazdelitvne podatkov $p_{data}(x)$
    Posodobi diskriminator z vzpenjanjem po gradientu

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

  end for
  Vzorči m vzorev $\{z^{(1)}, \ldots, z^{(m)}\}$ iz priorne porazdelitve $p_g(z)$
  Posodobi generator s pomočjo gradientnega spusta

$$\nabla\theta_d \frac{1}{m} \sum_{i=1} m \log\left(1 - D(G(z^{(i)}))\right)$$

end for

2.3 Variacijski autoenkoder

Eden od glavnih pristopov pri generativnem strojnem učenju je uporaba variacijskih autoenkoderjev [10]. TI so posplošitev autoenkoderja, kjer enkoder pogojimo da ustvarjeni latentni vektorji okvirno sledijo normalni porazdelitvi. Generiranje novih slik je torej samo vzorčenje iz normalne porazdelitve, z določeno srednjo vrednostjo in standardno deviacijo, ki jo dobimo iz mreže. Vedno je potreben kompromis med rekonstrukcijsko napako in prileganjem normalni porazdelitvi. Statistično gledano imamo spremenljivko z, katera generira x. Izračunali bi radi $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$. Izkaše se, da je izračun te porazdelitve v praksi problematičen, zato aproksimiramo to porazdelitev z porazdelitvijo $q(z|x)$, ki ji je podobna in jo znamo izračunati. Za mero podobnosti med dvema porazdelitvama pa uporabimo KL divergenco. V naslednjem delu predstavimo bolj natančno in formalno izpeljavo variacijskega autoenkoderja.
Predstavljajmo si množico $X = \{x^{(i)}\}_{i=n}^{N}$, ki vsebuje N neodvisnih in enakomerno porazdaljenih slučajnih spremenljivk x. Predpostavimo, da so podatki generirani s pomočjo nekega naključnega procesa, ki vključuje slučajno spremenljivko z, ki pa jo ne vidimo.
Ta proces je sestavljen iz dveh korakov

1. Vrednost $z^{(i)}$ je generirana iz predhodne porazdelitve $p_{\theta^*}(z)$
2. Vrednost x^i je generirana iz pogojne porazdelitve $p_{\theta^*}(x|z)$

Predpostavimo, da $p_{\theta^*}(z)$ in $p_{\theta^*}(x|z)$ prihajata iz družine porazdelitev $p_\theta(z)$ in $p_\theta(x|z)$ in da je njihova gostota verjetnosti povsod odvedljiva glede na parametra θ in z. V praksi so vrednosti $z^{(i)}$ ter vrednost θ^* neznane.
Zanima nas rešitev, ki deluje tudi v primeru večje podatkovne množice ter kadar je intergral marginalne verjetnosti $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$ neobladljiv (ne moremo ga

odvajati oz. izračunati) in kjer je posteriorna gostota $p_\theta(z|x) = p_\theta(z|x)p_\theta(z)/p_\theta(x)$ neobvladljiva. Ta dva primera, se velikokrat pojavljata prav v nevronskih mrežah z nelinearnimi skritimi sloji.
Pristop z variacijskimi autoenkoderji nam omogoča učinkovito ocenjevanje parametrov θ, kar nam omogoča generiranje umetnih podatkov, ki so podobni naravnim. Kadar imamo neko vrednost x ter izbrane parametre θ, nam VAE omogoča dobiti aproksimacijo latentne spremenljivke z. To se uporablja v namene kodiranja, kjer informacije iz x zakodiramo v z. Še ena od uporabnih lastnosti pa je aproksimiranje inference spremenljivke x, kar nam omogoča uporabo v smeri razšumenja, "inpaintinga"itd.
Uvedemo prepoznavni model $q_\phi(z|x)$, ki je aproksimacija $p_\theta(z|x)$. Izpeljali bomo metodo, ki se ϕ nauči skupaj s parametri θ Neopazovano spremenljivko z, si lahko predstavljamo kot zakodirano informacijo. Zato model $q_\phi(z|x)$ imenujemo **enkoder**, saj nam glede na podatek x izračuna porazdelitev možnosti z, ki bi lahko generirale ta podatek. Analogno bomo $p_\theta(x|z)$ imenovali **dekoder**, saj nam iz z producira porazdelitev čez vrednosti x.
Izpeljemo lahko spodnjo mejo:

$$\mathcal{L}(\theta, \phi, x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \mathbb{E}_{q\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]$$

ki jo želimo optimizirati glede na parametre ϕ in θ. Zaradi velike variance gradienta, je naivna Monte Carlo metoda, za ta primer neučinkovita in potrebujemo boljšo metodo.
Zanima nas cenilka za spodnjo mejo $\mathcal{L}$ in njene odvode. Upoštevajoč nekaj pogojev, ki so bolj natančno razloženi v [10], lahko reparametriziramo $\tilde{z} \sim q_\phi(z|x)$ z odvedljivo preslikavo šuma ϵ, torej $\tilde{z} = g_\phi(\epsilon, x)$, kjer velja $\epsilon \sim p(\epsilon)$. Sedaj lahko vpeljemo Monte Carlo aproksimacijo za pričakovano frednost neke funkcije $f(z)$, glede na $q_\phi(z|x)$. kot

$$\mathbb{E}_{q_\phi(z|x^{(i)})}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g_\phi(\epsilon, x^{(i)}))] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\epsilon^{(l)}, x^{(i)}))$$

, kjer je $\epsilon^{(l)} \sim p(\epsilon)$.
To tehniko apliciramo na naš problem in dobimo cenilko $\tilde{\mathcal{L}}^A(\theta, \phi; x^{(i)}) \simeq \mathcal{L}(\theta, \phi; x^{(i)})$ za katero velja

$$\tilde{\mathcal{L}}^A(\theta, \phi; x^{(i)}) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^(i), z^{(i,l)}) - \log q_\phi(z^{(i,l)}|x^{(i)})$$

kjer

$$z^{(i,l)} = g_\theta(\epsilon^{(i,l)}, x^{(i)}) \text{ in } \epsilon^{(l)} \sim p(\epsilon)$$

Velikokrat lahko $KL - divergenco$ integriramo analitično, tako da je ocena z vzorčenjem potrebna le za rekonstrukcijsko napako. $\mathbb{E}_{q\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)$. KL-divergenco si lahko predstavljamo kot regularizacijski člen ϕ, kar nam da drugo različico cenilke $\widetilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) \simeq \mathcal{L}(\theta, \phi; x^{(i)})$ ki je definirana kot

$$\widetilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(x^{(i)}|z^{(i,l)}))$$

in velja

$$z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)}) \text{ in } \epsilon^{(l)} \sim p(\epsilon)$$

Če Iz podatkovnem množice X z N elementi vzorčimo po M vzorcev, lahko skonstruiramo cenilko osnovano na "minibatchih":

$$\mathcal{L}(\theta, \phi; X) \simeq \widetilde{\mathcal{L}}(\theta, \phi; X^M) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\theta, \omega; x^{(i)})$$

Minibatch $X^M = \{x^{(i)}\}_{i=1}^M$ je naključno izbran vzorec velikosti M iz množice X. V psevdokodi je algoritem predstavljen kot

Algoritem 2 Minibatch verzija AEVB algoritma. Lahko se uporablja cenilka $\widetilde{\mathcal{L}}^A$ ali $\widetilde{\mathcal{L}}^B$

$\theta, \phi \leftarrow$ inicializacija parametrov
repeat
  $X^M \leftarrow$ Naključen minibatch M vzorcev iz X
  $\epsilon \leftarrow$ Naključni vzorec šuma iz porazdelitve $p(\epsilon)$
  $g \leftarrow \nabla_{\theta,\phi} \widetilde{\mathcal{L}}^M(\theta, \phi; X^M, \epsilon)$ (Gradient minibatch cenilke)
  $\theta, \phi \leftarrow$ Posodobimo parametre s premikom v smeri gradienta (SGD ali ADAGRAD)
until parametra (θ, ϕ) skonvergirata
return θ, ϕ

2.4 Nasprotniški autoenkoder

Nasprotniški autoenkoderji [12] so razširitev autoenkoderjev in so po svoje zasnovi precej podobni variacijskim avtoenkoderjem. Glavna razlika se pojavi v načinu zagotavljanja porazdelitve latentnega vektorja. Variacijski autoenkoder uporablja KL-divergenco kot vodilo za vodenje pravilne porazdelitve, medtem ko se pri nasprotniških avtoenkoderjih uporablja nasprotniško učenje, kjer želimo s pomočjo diskriminatorja doseči isti cilj.
Naj bo x vhodni podatek, z latentni vektor autoenkoderja in $p(z)$ porazdelitev kateri želimo da koda z sledi. Definirajmo $q(z|x)$ kot porazdelitev enkoderja, $p(x|z)$ kot porazdelitev dekoderja, $p_d(x)$ kot porazdelitev podatkov ter $p(x)$ kot porazdelitev našega modela. Enkoder definira porazdelitev $q(z)$ na latentnem vektorju kot

$$q(z) = \int_x q(z|x) p_d(x) dx$$

Nasprotniški autoenkoder je autoenkoder, ki je regulariziran z ujemanjem med $q(z)$ in $p(z)$.
Nasprotniška mreža je zadolžena za vodenje porazdelitve $q(z)$ proti $p(z)$, medtem ko je autoenkoder zaslužen za minimiziranje rekonstrukcijske napake. Velja, da je generator nasprotniške mreže tudi enkoder autoenkoderja $q(z|x)$, ki je zadolžen da

prelisiči diskriminator, da ne loči med $q(z)$ in $p(z)$. Učenje vedno izvajamo v dveh delih, najprej učimo nasprotniško mrežo (diskriminator) in s tem regulariziramo porazdelitev latentnega vektorja. S tem se posodobi tudi generator mreže, ki je hkrati enkoder autoenkoderja tako, da bolje zmede diskriminator. V drugem koraku pa učimo autoenkoder in s tem izboljšujemo rekonstrukcijsko napako.
Možnih je nekaj različnih izbir za enkoder $q(z|x)$

- **Deterministični** Predpostavimo, da je $q(z|x)$ deterministična funkcija x-a. V tem primeru je enkoder podoben enkoderju standardnega autoenkoderja in edini vir stohastičnosti (nepredvidljivosti) v $q(z)$ ostane učna množica $p_d(x)$.

- **Normalno porazdeljeni** Privzamemo, da je $q(z|x)$ normalna porazdelitev, katere srednja vrednost in varianca je dobljena iz enkoderja. Velja torej

  $$z_i \sim \mathcal{N}(\mu_i(x), \sigma_i(x))$$

  V tem primeru stohastičnost dobimo tako iz učne množice kot iz naključnosti naravne porazdelitve.

- **Univerzalni aproksimator** To si predstavljamo kot posplošitev prejšne možnosti. Cilj je, da $q(z|x)$ naučimo kot univerzalni aproksimator. Naj bo enkoder funkcija $f(x, \eta)$, ki na vhod dobi x in naključni šum η s fiksno porazdelitvijo, potem lahko vzorčimo iz katerekoli posteriorne porazdelitve $q(z|x)$, tako da evaluiramo $f(x, \eta)$ pri različnih vzorcih η. Matematično gledano, predpostavimo da velja $q(z|x, \eta) = \delta(z - f(x, \eta))$ in

  $$q(z|x) = \int_\eta q(z|x, \eta)p_\eta(\eta)d\eta \implies q(z) = \int_x \int_\eta q(z|x, \eta)p_d(x)p_\eta(\eta)d\eta dx$$

  Tu je stohastičnost v $q(z)$ dobljena tako iz učne množice kot iz šuma η na vhodu enkoderja. V tem primeru nismo več omejeni na naravno porazdelitev, ampak si lahko izberemo kakršnokoli porazdelitev želimo, saj to določuje porazdelitev šuma η.

Nasprotniški autoenkoder je mogoče razširiti v pogojno obliko kjer vsem učnim vzorcem dodelimo oznako kateremu razredu pripadajo. Na vhod diskriminatorja dodamo one-hot (?) oznako, ki predstavlja razred, kateremu vzorec pripada. Ko enkoder izračuna latentni vektor združimo oznako skupaj s latentnim vektorjem in to podamo dekodirnemu delu mreže.

3 Preizkušeni pristopi

Med raziskovanjem načina za doseganje najboljših rezultatov staranja in pomlajevanja smo se poslužili različnih pristopov, ki jih bomo predstavili v tem poglavju. Predstavili bomo teoretično podlago vsakega od njih ter dosežene rezultate. Analizirali bomo pomanjkljivosti in težave pri implementaciji.
Za implementacijo različnih modelov, je bila uporabljena programska knjižnica **Keras 2.0.0**, ki v ozadju uporablja **Tensorflow 1.0.0**. Za doseganje enakovrednih

rezultatov je verzija pomembna, saj lahko starejše ali novejše različice dajejo bistveno drugačne rezultate.
V vseh poizkusih, razen tam kjer je specifično navedeno drugače, smo za učno množico uporabljali bazo slik UTKFace [**?**], ki vsebuje 20000 slik oseb različnih etničnih izvorov, spola ter starosti. Slike v bazi so obrezane ter poravnane ter vsebujejo slike oseb pri različnih osvetlitvah, obraznih mimikah ter stopnjah pokritosti.

3.1 Pogojni Wasserstein GAN

Eden od prvih preizkušenih modelov je bila pogojna različica wasserstein GAN [2] modela. Eden od glavnih izzivov pri učenju GAN modelov je nastabilnost učenja, kar pa naj bi wassersteinova različica odpravila.

3.1.1 Teoretično ozadje

Pri učenju generativnih modelov predpostavimo, da podatki prihajajo iz neke porazdelitve, katere aproksimacijo P_θ se želimo naučiti. Da to dosežemo, lahko definiramo slučajno spremenljivko Z, ki ima fiksno porazdelitev $p(z)$. To lahko s pomočjo nevronske mreže (parametrične funkcije) preslikamo v novo slučajno spremenljivko, ki generira vzorce s porazdelitvijo P_θ.

$$g_\theta : \mathcal{Z} \leftarrow \mathcal{X}$$

S spreminjanjem parametrov θ, lahko dosežemo da se P_θ približna porazdelitvni podatkov P_r. Želimo izbrati pravilno metriko za definiranje razdalje med dvema porazdelitvama. Obstaja nekaj standardnih možnosti, ki jih lahko uporabimo:

- **TV razdalja**
$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

- **KL divergenca**
$$KL(\mathbb{P}_r || \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x)$$

  Kjer je največja težava asimetričnost, torej $KL(\mathbb{P}_r||\mathbb{P}_g) \neq KL(\mathbb{P}_g||\mathbb{P}_r)$ ter dejstvo, da je mera neskončna, kadar velja $P_g(x) = 0$ in $P_r(x) > 0$. To nam lahko povzroča težave v začetnih fazah učenja nevronske mreže.

- **Jensen-Shannon (JS) divergenca**
$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r||\mathbb{P}_m) + KL(\mathbb{P}_g||\mathbb{P}_m)$$

  kjer je $\mathbb{P}_m$ enak $\mathbb{P}_r + \mathbb{P}_g)/2$. To nam odpravi največje pomanjkljivosti KL divergence, saj je simetrična ter vedno manjša od neskončnosti, saj lahko izberemo $\mu = \mathbb{P}_m$

- **EM oz. Wasserstein-1 razdalja**

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$

  kjer nam $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ predstavlja množico vseh skupnih porazdelitev $\gamma(x, y)$, katerih "marginalašta $\mathbb{P}_r$ in $\mathbb{P}_g$. Lahko si predstavljamo kot koliko mase mora biti prestavjene da iz ene porazdelitve dobimo drugo.

Izkaže se, da je za nas najbolj zanimiva EM razdalja, saj določena enostavna zaporedja porazdelitev konvergirajo glede na to razdaljo in ne glede na druge. V tej obliki te razdalje ni mogoče izračunati, vendar po Kantarevich-Rubensteinovi dualnosti je zgornja definicija EM razdalje ekvivalentna

$$W(P_r, P_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_\theta}[f(x)]$$

kjer supremum gledamo glede na vse 1-Lipschitz funkcije. Naj bosta d_x in d_y funkciji razdalje definirani na prostorih X in Y. Funkcija f je **K-Lipschitzova** če $\forall x_1, x_2 \in X$ velja $d_y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$.

Psevdokoda WGAN učenja se zapiše kot

Algoritem 3 WGAN algoritem.

Require: α hitrost učenja, c parameter rezanja, m velikost batcha, n_{kritik} število iteracij kritika na število iteracij generatorja

Require: w_0 začetne uteži kritika, θ_0 začetne uteži generatorja

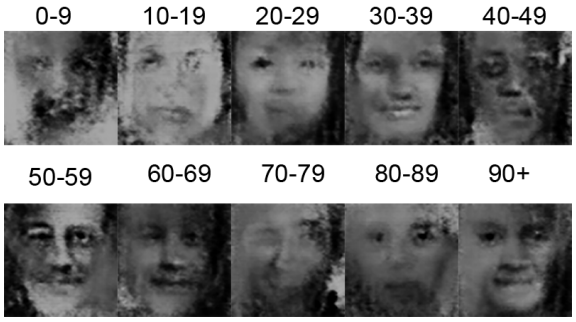
while θ ne skonvergira **do**
  for $t = 0, \ldots, n_{kritik}$ **do**
    Vzorči $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ iz učnih podatkov
    Vzorči $\{z^{(i)}\}_{i=1}^m \sim p(z)$ iz prejšnje (prior) porazdelitve
    $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$
    $w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$
    $w \leftarrow \mathrm{obreži}(w, -c, c)$
  end for
  Vzorči $\{z^{(i)}\}_{i=1}^m \sim p(z)$ iz prejšnje (prior) porazdelitve
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \mathrm{RMSProp}(\theta, g_\theta)$
end while

Izkaže se [2], da je za doseganje Lipschnitzovega pogoja dovolj, da funkcije obrežemo (clip) med dve možni vrednosti, ki jih nastavimo s parametrom c, kar moramo storiti po vsaki posodobitvi uteži w.

3.1.2 Eksperimentalni rezultati

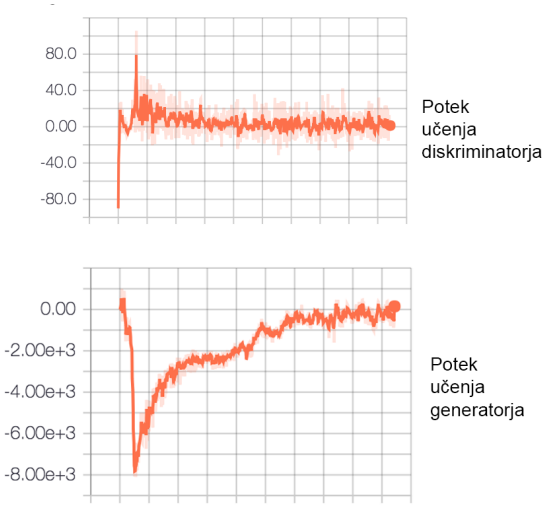
Za doseganje staranja s pomočjo WGAN modela, smo želeli najprej preizkusiti WGAN kot klasični pogojni generativni model. Želeli smo, da bi mreža glede na vhodni šumni vektor ter oznako starosti ustvarila osebo, ki deluje dovolj stara. Preizkusili smo dve standardni arhitekturi za obliko generatorja in diskriminatorja. Prva

arhitektura je bila konvolucijska in je sledila predlogom iz [13]. Želeli smo mrežo najprej preizkusiti na enostaven način, zato smo se odločili za velikost slike **64x64** slikovnih točk ter za enokanalne slike. Vsakega od različnih modelov, smo učili za dolžino 20000 batchov.
Pri konvolucijskem modelu, smo dobili dobili rezultate, ki so prikazani na sliki 5. Vidimo, da je bila naša mreža učena na učni množici obraznih slik, vendar rezultati



Slika 5: Rezultati učenja CWGAN mreže z konvolucijsko arhitekturo

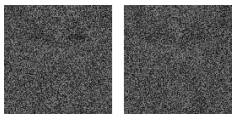
niso sprejemljivi, saj so slike zamazane in polne artefaktov. Prav tako so starostni razredi slabo naučeni, saj večina obrazov na slikah na pogled ne pripada pravilni starostni skupini. V sklopu učenja smo uporabljali **RMSProp** optimizator z nastavljeno hitrostjo učenja kot 0.0005. Parameter rezanja je bil standardno nastavljen na (-1,1) ter $n_{kritik} = 5$. Velikost batch-a pa smo nastavili na 32. Začetne uteži so bile inicializirane z vzorčenjem iz normalne porazdelitve. Razlog za slab uspeh učenja lahko najdemo na grafu vrednosti kriterijske funkcije, ki je viden na sliki 6



Slika 6: Kriterijska funkcija pri učenju konvolucijskega CWGAN modela

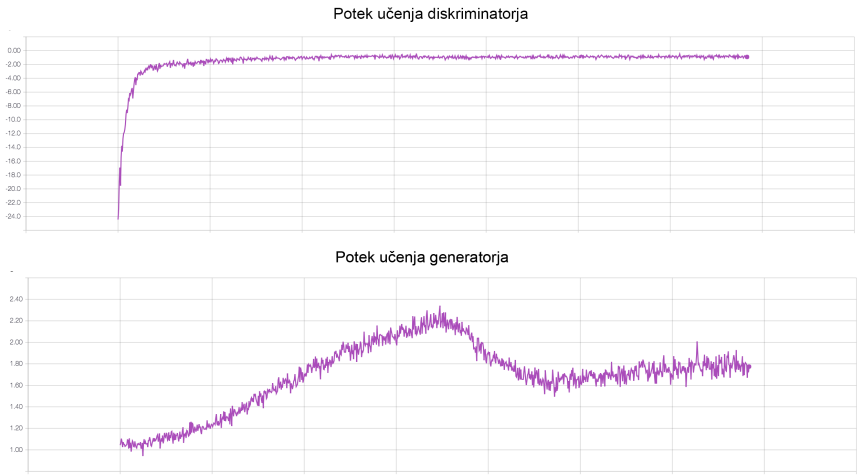
Opazimo, da nam diskriminator zelo hitro skonvergira v okolico ničle, vendar se nato tam ne ustali. V času učenja skače po okolici, kar pomeni, da ne pride do nekih večjih vizualnih izboljšanj. Generator pa na začetku učenja dobi izrazito negativno vrednost svoje kriterijske funkcije in se nato počasi približuje ničli, a se tudi on tam

ne ustali.
Da bi izboljšali rezultate, smo poskušali z empiričnim spreminjanjem hiperparametrov (hitrost učenja, n_{kritik}, velikost batch-a, število filtrov v konvolucijskih slojih ipd. Prav tako smo poskušali dodati Dropout sloj [14] z različnimi parametri, vendar se rezultati niso vidno izboljšali.
Preizkusili smo tudi uporabo CWGAN arhitekture s polnopovezanimi sloji, kjer generator s pomočjo zaporednih kaskadnih slojev iz vektorja šuma zgenerira sliko, diskriminator pa samo z uporabo le-teh presodi ali je slika prišla iz množice realnih podatkov ali ne.
Za začetek smo uporabili iste parametre za učenje kot pri konvolucijskem modelu, a smo žal dobili rezultate, ki so bili še manj kvalitetni. Po končanem učenju je večino slik vsebovalo le naključni šum, kar je vidno na sliki 7.



Slika 7: Vzorec generiranih slik pri CWGAN modelu s polnopovezanimi sloji

Če pogledamo potek učenja na sliki 8,



Slika 8: Kriterijska funkcija pri učenju CWGAN modela s polnopovezanimi sloji

lahko vidimo da je v tem primeru diskriminator zelo hitro skonvergiral proti optimalni vrednosti 0, medtem ko kriterijska funkcija generatorja ni skonvergirala. To nam lepo prikaže najbolj klasično težavo pri učenju GAN-ov in sicer neravnovesje v moči med diskriminatorjem D in generatorjem G. V tem primeru je D postal tako dober, da ga G, ni znal prelisičiti in medsebojno učenje ni bilo uspešno. Poskusili smo z empiričnim spreminjanjem parametrov, da bi izboljšali rezultate učenja, vendar nam to ni uspela.

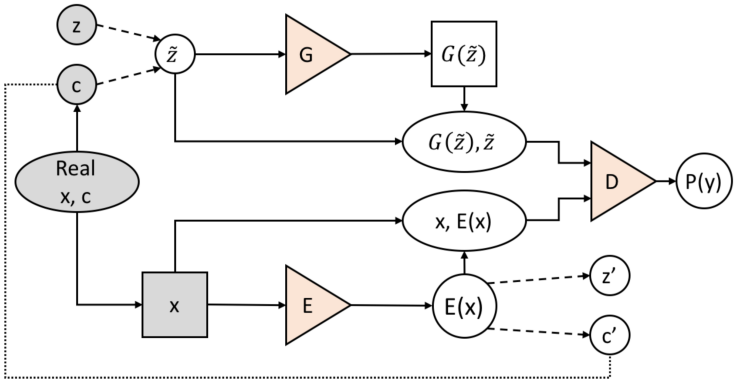
3.2 BiCOGAN

3.2.1 Teoretično ozadje

Klasični GAN modeli nam omogočijo dobiti preslikavo iz prostora šuma z v prostor generiranih podatkov x, vendar v procesu očenja žal ne dobimo obratne preslikave $x \mapsto z$, kar je v določenih primerih zaželjeno, saj nam omogoča dobiti kompaktno reprezentacijo naše informacije x.
Rešitev tega problema je model, ki pogojno generativno nasprotniško mrežo poveže z enkoderjem. ki je zadolžen za določanje inverzne preslikave [9]. Velja [5], da v idealnem primeru morata biti G in E inverzna, da lahko prelisičita diskriminator D. V tem primeru primeru se GAN enačba, zapisana v 2.1 spremeni v

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

Nas so take mreže zanimale zato, saj bi lahko postopek iz [1] poenstovali tako, da bi bila potrebna uporaba samo ene nevronske mreže. Pri omenjenem pristopu, so za določanje preslikave iz prostora slik v prostor latentnih vektorjev, uporabili dodatno nevronsko mrežo, kar pa zahteva izgradnjo dodatnega modela ter učenja, ki je ločen od generiranja slik.
Želja je bila, da bi z uporabo dvosmernega učenja, hkrati učili tako generator in diskriminator kot tudi enkoder. Idejno arhitekturo naše mreže v shematski obliki vidimo na 9.



Slika 9: Idejna zasnova BICOGAN arhitekture.Vir:[9]

Če se nanašamo na oznake, definirane v poglavju o navadnih GAN mrežah, potem formalno rečemo, da se generator nauči preslikavo $G(\tilde{z}, \theta_G)$ iz porazdelitvne $p_{\tilde{z}}$, kjer je $\tilde{z} = [z\ c]$ v porazdelitev p_G, kjer je glavni cilj da se p_G in p_{data} čim manj razlikujeta. Naloga enkoderja $E(x; \theta_E)$ je da slika iz p_{data} v p_E, kjer želimo da je p_E čim bližje $p_{\tilde{z}}$. Diskriminator se odloča o avtentičnosti vzorca glede na $D(\tilde{z}, G(\tilde{z}); \theta_D)$ in $D(E(x), x; \theta_D)$.
V splošnem primeru BICOGAN modela se mora enkoder naučiti inverzni preslikavo iz x v z **in** c. Ker pa je to v našem primeru nepotrebno, smo naš model spremenili tako, da je informacija o oznaki c enkoderju vedno podana.

3.2.2 Eksperimentalni rezultati

Poizkusili smo z implementacijo BICOGAN modela v orodju Keras. Kot najenostavnejšo obliko smo za vse tri notranje mreže (diskriminator, generator, enkoder) uporabili polnopovezano arhitekturo. Rezultati, ki smo jih dobili so prikazali solidne vizualne rezultate s strani generatorja, vendar je do težav prišlo pri enkoderju. Ni nam uspelo, da bi se enkoder pravilno naučil preslikave iz x v z. Veljati bi namreč moralo, $G(E(x)) \sim x$, že na prvi pogled pa je očitno, da to ni uspelo.
Še ena od težav je, da arhitektura s polnopovezanimi sloji generira slike, ki so polne šume in zatorej precej različne od realnih podatkov.

Literatura

[1] G. Antipov, M. Baccouche in J.-L. Dugelay, *Face aging with conditional generative adversarial networks*, v: Image Processing (ICIP), 2017 IEEE International Conference on, IEEE, 2017, str. 2089–2093.

[2] M. Arjovsky, S. Chintala in L. Bottou, *Wasserstein generative adversarial networks*, v: International Conference on Machine Learning, 2017, str. 214–223.

[3] T. F. Cootes, G. J. Edwards in C. J. Taylor, *Active appearance models*, IEEE Transactions on Pattern Analysis & Machine Intelligence (6) (2001) 681–685.

[4] T. F. Cootes in dr., *Active shape models-their training and application*, Computer vision and image understanding **61**(1) (1995) 38–59.

[5] J. Donahue, P. Krähenbühl in T. Darrell, *Adversarial feature learning*, arXiv preprint arXiv:1605.09782 (2016).

[6] Y. Fu, G. Guo in T. S. Huang, *Age synthesis and estimation via faces: A survey*, IEEE transactions on pattern analysis and machine intelligence **32**(11) (2010) 1955–1976.

[7] Y. Fu in N. Zheng, *M-face: An appearance-based photorealistic model for multiple facial attributes rendering*, IEEE Transactions on Circuits and Systems for Video technology **16**(7) (2006) 830–842.

[8] I. Goodfellow in dr., *Generative adversarial nets*, v: Advances in Neural Information Processing Systems 27 (ur. Z. Ghahramani in dr.), Curran Associates, Inc., 2014, str. 2672–2680.

[9] A. Jaiswal in dr., *Bidirectional conditional generative adversarial networks*, arXiv preprint arXiv:1711.07461 (2017).

[10] D. P. Kingma in M. Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114 (2013).

[11] Z. Liu, Z. Zhang in Y. Shan, *Image-based surface detail transfer*, IEEE Computer Graphics and Applications **24**(3) (2004) 30–35.

[12] A. Makhzani in dr., *Adversarial autoencoders*, v: International Conference on Learning Representations, 2016.

[13] A. Radford, L. Metz in S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, arXiv preprint arXiv:1511.06434 (2015).

[14] N. Srivastava in dr., *Dropout: a simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research **15**(1) (2014) 1929–1958.