Eötvös Loránd Tudományegyetem

Czifra Domonkos

Msc szakdolgozat

Generatív modellek kiértékelési metrikái

Témavezető:
Zombori Zsolt
Rényi Alfréd Matematikai Kutatóintézet

2019. május 31.

Köszönetnyilvánítás

Nagy köszönettel tartozom témavezetőmnek, Zombori Zsoltnak a szakdolgozat elkészítésében nyújtott támogatásáért. Mind elméleti, mind gyakorlati problémákkal bátran fordulhattam hozzá. Külön köszönöm gondos lektorálását, a vázlat elkészülésétől a (remélem utolsó) sajtóhibáig.

Köszönöm a tavalyi Deep-Learning szeminárium négy oktatójának, Zombori Zsoltnak, Lukács Andrásnak, Varga Dánielnek, és Csiszárik Adriánnak, hogy felkeltették érdeklődésemet a Deep-Learning, és a szakdolgozat témája: a generatív modellek iránt.

Köszönöm továbbá szüleimnek az egyetemi (és nem egyetemi) évek alatt nyújtott szeretetét, és támogatását.

Tartalomjegyzék

1. Bevezetés **3**

2. Háttérismeretek **5**

2.1. Modern Mesterséges Neurális hálók 5

2.2. Stochasztikus gradiens, Backpropagation 6

2.3. Wasserstein távolság 7

2.4. Minimális költségű párosítás: Magyar módszer, Munkres algoritmus 8

2.5. Minimális költségű Maximális folyam 10

3. Generatív modellezés **13**

3.1. Generatív Autoenkóderek 14

3.1.1. Deep autoenkóder 14

3.1.2. Variációs AutoEnkóder (VAE) 15

3.2. Generativ Adversarial Networks (GAN) 19

3.2.1. GAN elmélet 19

3.2.2. GAN hibái 21

3.3. Wasserstein GAN (WGAN) 22

3.3.1. Kantorovitch-Rubinstein dualitás tétel 23

3.3.2. WGAN algoritmus 24

3.4. Improved Wasserstein GAN 25

4. Generatív modellek kiértékelése **26**

4.1. Alapprobléma 26

4.2. Az irodalomban felmerülő kiértékelő metrikák 27

5. Leimplementált kiértékelési metrikák **34**

5.1. Wasserstein távolság alkalmazása a kiértékelésben, és az alkalmazás nehézségei 34

5.2. Wasserstein távolság közelítése teljes párosítással 35

5.3. Eredmények MNIST adathalmazon 36

5.4. Eredmények Syn-circles adathalmaz 39

5.5. Eredmények Celeba adathalmaz 42

5.6. Wasserstein távolság közelítése maximális párosítással, ritkított gráfon 43

6. Implementáció **46**

7. Összefoglaló **48**

1. Bevezetés

A kép, és hangfelismerés két alapvető Machine Learning (ML) problémekör, ahol a klasszikus ML algoritmusokat (Decision Tree [1], Support Vector Machine (SVM) [2] stb.) az utóbbi 10 évben lehagyták a modern Deep Learning algoritmusok [3].

Mi változott az elmúlt 10-15 évben, hiszen a mesterséges neuronhálók elmélete már jóval azelőtt ki lett dolgozva, mint 1-2 klasszikusnak mondott ML algortimus, mint például az SVM?

Az egyik ok az, hogy az utóbbi időben a GPU-k számítási kapacitása nagyon megnőtt, amiken a párhuzamosíthatóság miatt a neuronhálók tanítási ideje lecsökkent. De ezen kívül legalább olyan fontosak a jól megírt, és széles körben elerjedt, és GPU-ra optimalizált Deep Learning könyvtárak (Tensorflow [4], Theano [5], Pytorch). Továbbá azok az apró eredmények is, amik bár nem számítanak olyan jelentős újításnak, de sok gyakorlati alkalmazásnál hozták meg az átütő sikert.

A deep-learning-nek az iparban számos alkalmazása született az elmúlt években, amelyek életképesnek bizonyultak:

- Az orvosi műszerekben: MR, CT-feldolgozás [6]
- Fordításban (lásd Google)
- Hangfeldolgozásban: hangfelismerés [7]
- Játékprogramokban: Go (AlphaGo [8]), Sakk

És számos olyan alkalmazás is született, amik bár nagyon meghökkentőek, de az iparba való szerves beépülésük még várat magára:

- Kép leírása szövegesen [9]
- Valós idejű objektumfelismerés: Yolo2 klasszifikáló háló [10]
- Deep Dream projekt: adott kép adott stílussal való átfestése (lásd az 1.1 ábrát)
- Képkitöltés, képjavítás (Mint például pislogó szem eltüntetése [11])

Ebbe a kategóriába tartozik az a téma is, amiről a szakdolgozatom szól: a képgenerálás, és ezek "jóságának" kiértékelése. Bár már volt akinek sikerült ebből értéket teremteni: Edmond de Belamy portréja (lásd a 3.4 ábrán, és hogy hogyan készült, ahhoz lásd a 3.2 fejezetet) egy aukción több mint 400 ezer dollárért kelt el [12]. Ennek ellenére a képgenerálás még főleg csak egy meghökkentő alkalmazás arra, hogy mi mindenre képes a deep-learning.

1.1. ábra. A Deep Dream projekt egyik alkalmazása: Egy adott fénykép egy adott festő (egyetlen festménye alpján) stílusában lefestve

A generatív modellek fő feladata egy eloszlás megtanulása tanító adatokból úgy, hogy képes legyen új, a tanító adathalmazban nem szereplő, de abba jól illeszkedő elemek generálására.

Példa erre, ha a generatív modellünktől azt várjuk el, generáljon minél jobb képeket, amelyek egy hálószobára hasonlítanak (mint mondjuk a híres LSUN bedrooms adathalmaz [13]). Ilyen generatív modellek már születtek, és néhányan nagyon jól teljesítenek: nagy felbontású, nagyon éles képeket képesek generálni, amikről egyáltalán nem is látszik, hogy egy gép készítette őket.

Azonban géppel eldönteni, hogy az újonnan generált képek mennyire élethűek, egyáltalán nem egyszerű. Milyen algoritmus mondja meg, hogy a generált kép nem illeszkedik bele a tanító adathalmazba? Vagy esetleg túlságosan hasonlítanak a képek a tanító adathalmazbeliekre, azaz nem csinál mást a generátor, mint memorizálja a tanító képeket, és nem képes ezeket általánosítani, és újakat generálni? Ezek még megoldatlan problémák, bár születtek próbálkozások ezek javítására.

A szakdolgozatomnak a célja a legfontosabb generatív modellek ismertetése (3. fejezet), valamint kiértékelési nehézségeinek bemutatása (4. fejezet). Majd bemutatom az általam javasolt kiértékelési metrikákat, és az implmentációjukat, az 5-6. fejezetben. A 2. fejezetben pedig az ezek megértéséhez szükséges elméleti alapokat foglaltam össze.

2. Háttérismeretek

Ebben a fejezetben összefoglalom azokat az ismereteket, amelyek szükségesek az ismertetett generatív modellek, és kiértékelési metrikáik megértéséhez.

2.1. Modern Mesterséges Neurális hálók

A neuronhálók célja általában egy adott feladatot optimálisan megoldó függvény (f^*) approximációja, amely adott x bemenetre, az $y = f^*(x)$ optimális megoldást minél jobban közelíti.

Gondoljunk csak a klasszifikáció problémájára, ahol x egy bemeneti kép, és keressük azt az y címkét, ami megmondja, x mely osztályhoz tartozik. Vagy egy sakkprogram esetében legyen x a játék állása, f^*-tól pedig a legjobb lépést várjuk el. De ennek az optimális függvénynek a megtalálása sokszor reménytelen, de még az approximációja is általánosságban nagyon nehéz. Ezért f^*-t csak egy függvénycsaládon belül keressük. Ez a függvénycsalád egy adott neurális háló, θ paraméterezéssel. Így tehát keressük azt az adott θ paramétert, melyre f^* a legjobb approximáció a függvénycsaládon belül.

Azért hálónak (hálózatnak) nevezzük a modellt, mert f^*-ot valójában függvények konvolúciójaként akarjuk előállítani: $f^*(x) = f^3(f^2(f^1(x)))$.

1. Definíció (réteg, mélység). *Ezeket az f^i függvényeket **réteg**eknek hívjuk, a rétegek számát pedig a háló **mély**ségének nevezzük.*

Az utolsó réteg a **kimeneti réteg**, amely $\mathbf{R}^{Dim(y)}$-ba képez. A többi, **belső** (hidden) rétegre pedig teljesülnie kell, hogy a megelőző réteg képtartományából a következő réteg értelmezési tartományába képez. Az előrecsatolt háló kifejezés arra utal, hogy a hálóban mint gráfban nincsen kör. Ebben különbözik a rekurrens hálóktól, amelyben előfordulhatnak körök, leggyakrabban hurkok formájában. Előre csatolt hálókat alkalmaznak legtöbbször képklasszifikációra, objektumfelismerésre, de sok generatív modellben is (mint például a következő fejezetben bemutatott Variational Autoenkóder, és Generativ Adversarial Networks).

A neuron jelző az idegtudományra utal, ahonnan a mesterséges neurális hálók ötlete származik. Ugyanis a fent említett réteg függvények: $f^i : \mathbf{R}^n \to \mathbf{R}^m$ felbomlanak m darab $\mathbf{R}^n \to \mathbf{R}$ skalárfüggvényre. És ezeket úgy is tekinthetjük, mint önálló számítási egységeket (neuront), amelyek függetlenül működnek és tanulnak. De természetesen, a mesterséges neuronhálók sok mindenben nem hasonlítanak a mintául vett emberi agyhoz, mert hatékony működésükhöz elengedhetetlen az a rengeteg beleépített mérnöki és matematikai trükk/módszer,

ami a természetben nem található meg. És így az sem céja, hogy a mintául szolgált emberi agyat pontosan leutánozza, ugyanis a modern, precíz agy-szimulációk rettentően erőforrásigényesek: sokszor nem is futnak le valós időben [14].

Bár a neuronhálók sok feladaton sokkal jobban teljesítenek, mint az ember(lásd Bevezető), de azért sok olyan terület maradt még, ahol az emberi agy jobban teljesít, és ahonnan még meríthetnek inspirációt a kutatók a természettől. Ilyen például a hatékonyság: az emberi agy 10^4-ed annyi energiasűrűséggel dolgozik, mint a mai processzorok ($10mW/cm^2$ vs $100W/cm^2$), és így a fogyasztása is sokkal kisebb ($20W$). Ezen az újfajta hardverek megjelenése segíthet; melyek a neurális hálót mint processzor egységet tartalmazzák. Így ezeken a mesterséges neurális hálók futtatása gyorsabb, és energiatakarékosabb. Az első ilyen processzor, aminek ipari alkalmazása is van, az az IBM TrueNorth chipje [15].

2.2. Stochasztikus gradiens, Backpropagation

Adott tehát egy neurális háló (f) amivel f^*-ot szeretnénk közelíteni. Az alkalmazástól függően többféle veszteségfüggvénnyt definiálunk, hogy mennyire van közel adott x bemenetre $\hat{y} = f(x)$ az optimális $y = f^*(x)$ megoldáshoz, és így mérhetjük a tanulás során, hogy a modellünk mennyit javul/romlik.

A két alapvető ML feladathoz általában különböző veszteségfüggvényeket is alkalmazunk:

- **Klasszifikáció:** Az inputot szeretnénk C kategóriába besorolni (például egy kézzel írt számot 0-9 kategóriába).

  Talán a leggyakrabban használt veszeteségfüggvény a Kereszt-Entrópia (Cross Entropy Loss)

  $$CE(y, \hat{y}) = -\sum_{i=1}^{m} t_i log(s_i) \tag{1}$$

  ahol $y = (t_1, \ldots, t_k)$, $\hat{y} = (s_1, \ldots, s_k)$.

  Ennek számos változata ismert: ha 'softmax' aktiváló függvény után alkalmazzuk ($softmax(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}$), akkor **Softmax loss**-nak, vagy Categoriacal Cross Entropy-nak hívják.

  Ha sigmoid aktiváló függvény után alkalmazzák, akkor **Binary Cross Entropy**-nak hívják (a bináris elnevezés a sigmoid függvény értékkészletéből jön: $sigmoid(x) = \frac{1}{1+e^x} \in (0,1)$). Ezt azon feladatoknál szokták alkalmazni, ahol több címke is illik egy bemenetre: ugyanis ezt a lossfüggvényt minden címkére függetlenül kell alkalmazni, azok nem befolyásolják egymást, míg a $softmax$ esetben ez nem igaz.

– **Regresszió**: Az inputot jellemezni szeretnénk egy (vagy több) folytonos változóval (Mondjuk egy házat az árával/méretével stb.)

  A legismertebb regresszió veszteségfüggvény a Legkisebb Négyzetes Hiba (MSE=Mean Squared Error):

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

Szeretnénk tehát a háló paramétereit úgy beállítani, hogy a veszteségfüggvény minél alacsonyabb legyen. Ezt gradienscsökkentéssel érhetjük el, minden iterációban a paraméterek hangolásával. Mivel a háló minden paramétere a bemenet valamilyen differenciálható függvénye, célunk ezen gradiens kiszámítása, és a paraméter módosítása úgy, hogy a hiba csökkenjen. A gradiens kiszámítása

– az egész adathalmaz

– egy részhalmaz (minibatch)

– akár csak egyetlen bemenet

szerint történhet. Azonban a batchméret alkalmas megválasztása nagyon fontos. Túl pici batchméret zajos tanuláshoz(túltanuláshoz) vezethet, túl nagy batchméret esetén pedig lassú tanulást eredményezhet (elvész a tanító jel).

2.3. Wasserstein távolság

2. Definíció (Wasserstein-p metrika). *Legyen $\nu, \eta : \mathbb{B}(\mathbb{R}^n) \to [0,1]$ valószínűségi mértékek, véges p-edik momentummal ($\int_{\mathbb{R}^n} x^p d\nu < \infty$). Ekkor a*

$$\mathbb{W}(\nu, \eta) = \Big(\inf_{\gamma \in \Pi(\nu,\eta)} \int d(x,y)^p d\gamma(x,y) \Big)^{\frac{1}{p}}$$

kifejezést Wasserstein-p metrikának hívjuk.

Ahol $\Pi(\nu, \eta)$ az a halmaz, amely az összes ν, η összetett eloszlást tartalmazza, amelynek marginális eloszlásai rendre ν és η.

Egy ilyen $\gamma(x, y)$ diszkrét eloszlások között pont egy szállítási feladatot jelent. Legyen ugyanis $P = x_1, \ldots, x_n$, $Q = y_1, \ldots, y_n$. γ ebben az esetben egy $n * n$-es mátrix, amely egy konkrét szállítási kiosztás (a marginális feltételek miatt):

$$\sum_i \gamma(i,j) = y_j$$
$$\sum_j \gamma(i,j) = x_i$$

.

Így tehát $\mathbb{W}(\nu,\eta) = \inf_{\gamma\in\Pi(\nu,\eta)} \sum_{i,j} d(i,j)\gamma(i,j)$. Azaz Lineáris Programozási (LP) feladatként felírva:

$$\begin{aligned} \min_{\gamma} \quad & d\gamma \\ \text{s.t.} \quad & \\ \sum_i \gamma(i,j) = \quad & y_j \qquad \forall j \in N \\ \sum_j \gamma(i,j) = \quad & x_i \qquad \forall i \in N \end{aligned} \tag{3}$$

Ezt például egy folyamfeladattal meg lehet oldani $O(n^2m)$ időben.

Folytonos esetben is egy szállítási feladatot kell megoldani, de annak megoldása már jóval bonyolultabb, hiszen hagyományos LP feladatként már nem lehet felírni.

Emiatt is szokták ezt a metrikát "homok-kupac" távolságnak is nevezni, hiszen adott két eloszlás (ha az olvasó 2 Gauss eloszlásra gondol, akkor közelebb kerül az elnevezéshez), ezeket mint homok-kupacokat tekintve a Wasserstein távolságuk azt mondja meg, legalább mennyi munkát (föld mennyiség $*$ megtett út) kell befektetnünk, hogy az egyik homok-kupacot a másik formájára alakítsuk.

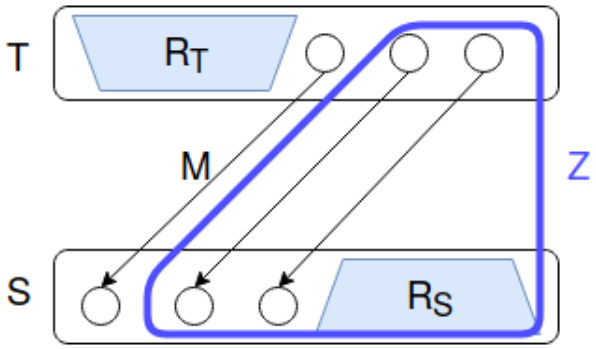
2.4. Minimális költségű párosítás: Magyar módszer, Munkres algoritmus

Mivel a magyar módszer viszonylag ismert algoritmus, ezért teljes részletességgel itt nem fogom tárgyalni, a témában nem jártas olvasónak az alábbi jegyzetet javaslom [16]. De az alapvető fogalmakat, algoritmusokat, amit később is használok, azt itt is ismertetem.

3. Definíció (Minimális költségű párosítás). *Adott egy $G = (V = [S,T], E)$ páros gráf, $c : V \to \mathbf{R}^+$ költségfüggvénnyel. Keressük $|S|$-et fedő független élek halmazát (feltesszük hogy létezik ilyen) úgy, hogy ezen élek költsége minimális legyen. Egy ilyen független élhalmazt minimális költségű párosításnak nevezünk.*

A feladat erősen polinomiális időben történő megoldása Kuhn nevéhez köthető (1955) [17]. Azonban a feladat megoldása során nagyban segítette a gráfelmélet magyar üttörői, König Dénes és Egervári Jenő, korábbi munkái [18], [19], ezért Kuhn az algoritmusát az ő tiszteletükre **magyar módszer**nek nevezte el.

1. Tétel (König). *Egy tetszőleges $G = (S,T)$ páros gráfban maximális független élek száma ($\nu(G)$) egyenlő az éleket lefogó pontok minimális számával ($\tau(G)$)*



2.2. ábra. König: alternáló út keresése [16]

A nemtriviális $\nu(G) \geq \tau(G)$ irány belátásához König azt vette észre, hogy egy nem maximális méretű M párosítás, M-alternáló utakkal mindig bővíthető. Pontosabban, ha R_S, R_T azon pontok halmaza, amiket nem fed M. Z pedig az R_T pontokból M-alternáló utakon elérhető pontok halmaza (lásd 2.2), akkor

- Ha $R_S \cap Z$ nem üres, akkor van R_T, R_S között alternáló út, amin ha megcseréljük a párosításbeli, és a nem párosításbeli éleket, akkor egy eggyel nagyobb méretű párosítást kapunk
- Különben pedig $(Z \cap S) \cup (T - Z)$ egy $|M|$ méretű, minden élet lefogó ponthalmaz.

Így tetszőleges élből kiindulva az alternáló utak mentén javítva egy teljes párosítást kaphatunk.

4. Definíció (halmaz hiánya). *Egy $X \subset S$ halmaz hiányán a $max(0, |X| - |\Gamma(X)|)$ értéket értjük. Ahol $\Gamma(X)$ alatt X szomszédait értjük.*

A König algoritmusból könnyen látszik, hogy mekkora S-ben a maximális hiányos halmaz:

2. Tétel (König-Hall). *Egy párosítás által fedetlenül hagyott csúcsok minimális száma egyenlő a $X \subset S$ halmazok maximális hiányával.*

hiszen $S \cap Z$ halmaz hiánya pont R_S.

Egerváry Jenő teszőleges egész súlyozásra adott egy primál-duál feltételt a maximális súlyú párosításra (ő eredetileg csak egészértékűre mondta ki).

3. Tétel (Egerváry). *Adott $G = (S, T, E)$ páros gráf, $c : V \to \mathbf{R}^+$ súly-függvénnyel. A súlyozott lefogások minimuma egyenlő a súlyozott párosítások maximumával ($\nu_c = \tau_c$). Ha pedig c egészértékű, akkor, akkor τ is választható egészértékűnek.*

Egerváry tételéből könnyen jön az algoritmus, ami egy ilyen párosítást megtalál, sőt élek többszörözésével még racionális súlyokra is működik. Sajnos azonban általános súly-függvényre nem feltétlen véges [20]!

Algorithm 1 Magyar módszer [16]

Input: $G = (S, T)$ páros gráf

Output: π minimális éllefogó súlyozás, M maximális súlyú pároítás

$\pi(v) = c_{\max}\ \forall v \in S$
$M := \{\}$
while $\tilde{\pi}(V)! = c(M)$ **do**
  $G_\pi := (S, T, E_\pi)$ pontos élek gráfja
  E_π éleket irányítsuk S-ből T irányába, a maradék E beli éleket pedig T-ből S irányba.
  if $\exists\ P$ irányított út M által fedett $s \in S$ és $t \in T$ között **then**
    P egy alternáló út, így $|M|$ növelhető:
    $M := M \triangle P$
  elseR_T, R_S az M által fedetlen csúcsok halmaza S ill. T-ben:
    $Z := R_S$-ből elérhető pontok halmaza
    $H := Z \cap S$
    $\delta := \min\{\pi(u) + \pi(v) - c(u, v) : (u, v) \in E, u \in H, v \in T - \Gamma_{G_\pi}(H)\}$
    $\pi := \begin{cases} \pi(v) - \delta, & v \in H \\ \pi(v) + \delta, & v \in \Gamma_{G_\pi}(H) \\ \pi(v) & \text{különben} \end{cases}$

Futásidő

Az útkeresés az 1-es algoritmusban $O(|E|)$ időben megy, az *else* ágban H mérete legalább eggyel nő, ezért adott M méretre csak $|S|$-szer léphetünk. M-et pedig az if ágban $|S|$-szer növelhetjük (Z mérete csökkenhet, ha M-et növeljük). Így a furásidő $O(|E| * |S|^2)$.

2.5. Minimális költségű Maximális folyam

5. Definíció (Minimális költségű Maximális folyam (MCMF)). *Adott $G = (V, A)$ irányított gráf adott s, t forrás illetve nyelő csúccsal, valamint $c, g : A \to \mathbf{R}^+$ költség, illetve kapacitásfüggvénnyel. Keressük azt az $f : A \to \mathbf{R}^+$ folyamot, melyre*

$$\left\{ \sum_{e \in A} c(e) f(e) |\ \textit{azon } f\textit{-ek amelyekre } \sum_{e \in A} f(e)\ \textit{maximális} \right\} \tag{4}$$

minimális.

A folyamfeladatnál a folyamfeltételt a kitűntetett s illetve t csúcsokra nem írtuk elő. Ha nincs ilyen kitüntetett két csúcs (tehát a folyamfeltételt mindkét csúcsra feltesszük), akkor **áram**ról beszélünk. A minimális költségű áramot hasonlóan lehet definiálni.

A minimális költségű folyam-, és áramfeladat ekvivalens, ez egy-egy egyszerű átalakítással egyszerűen belátható:

- ha az áramfeladatra van egy szubrutinunk: a folyamfeladat t, s csúcsa közé húzzunk be egy elegendően nagy kapacitású ($> g_{max} * m$), és elegendően pici költségű ($< c_{max} * m$)) e élet. Könnyen látható, hogy az így definiált áramfeladat optimális megoldásából az e élet kihagyva a folyamfeladat optimális megoldását kapjuk.

- ha a folyamfeladatra van egy szubrutinunk: az áramfeladathoz egy-egy izolált s, t csúcsot véve kapunk egy folyamfeladatot. Triviális, hogy ennek az optimális megoldása is optimális az áramfeladatra.

Minimáli költségű folyam megtalálása

A minimális költségű folyamfeladatot meg lehet oldani nagyon hasonlóan [21], mint ahogyan Edmonds-Karp [22] megoldotta a maximális folyamfeladat problémát.

Először irányítatlan értelemben egyszerű, irányított gráfot vizsgáljunk. $u(i, j)$ jelölje a kapacitást i, j csúcs között (0 ha nincs él), $c(i, j)$ jelölje a költséget egységnyi folyamra (i, j) élen, valamint magát a folyamot pedig $f(i, j)$, amit kezdetben azonosan nullának választunk.

Új éleket veszünk a gráfhoz: minden (i, j) élre egy párhuzamos, de ellentétes irányú (j, i) élet húzunk be, $c(j, i) = -c(i, j)$ költséggel, $u(j, i) = 0$ kapacitással. Az algoritmus során arra mindig figyelünk, hogy $f(j, i) = -f(i, j)$ maradjon. Mivel egyszerű, irányított gráfból indultunk ki, a létrejövő gráfban nem lesznek irányított értelemben párhuzamos élek.

Hasonlóan a Fold-Fulkersson algoritmushoz [23], egy reziduális (maradék) folyamot veszünk, amibe csak telítetlen éleket veszünk be (amikre tehát $f(i, j) < < u(i, j)$), $r(i, j) = u(i, j) - f(i, j)$ kapacitással.

Az Edmonds-Karp algoritmus a legkevesebb élet használó utat keresi meg a reziduális gráfban, de ha a költségek nem egyformák az éleken, akkor a c szerinti legrövidebb utat keressük meg a reziduális gráfban. Az úton megkeressük a legkisebb reziduális kapacitást (δ), és δ-val növelhető a folyam-gráfban a folyam a legrövidebb út mentén, valamint minden egyes élnek a visszafele élén ugyanany-

nyival csökkentenünk kell a folyamértéket, hogy a $f(j,i) = -f(i,j)$ érvényben maradjon.

Irányítatlan gráf esetén megtehetjük, hogy egy (i,j) irányítatlan élet helyettesítünk $(i,j),(j,i)$ irányított élekkel. Azonban így meg kell birkóznunk a párhuzamos élekkel, ami elméletben nem jelent gondot, csak az implementálásnál.

1. A legrövidebb út keresésnél nem elég az újonnan hozzávett csúcsokat, hanem az odavezető élet is tárolni kell.

2. Amikor az algoritmusban visszafelé élen is csökkentjük a folyamot, párhuzamos élek során ki kell választani a megfelelőt.

Futásidő

Jelölje $T(n,m)$ azt az időt, amennyibe n csúcson, m élen leggyorsabban megtalálunk egy legrövidebb folyamot. Mivel minden útkeresés után egy él telítődik, és csak akkor telítődhet újra ugyanaz az él, ha a forrástól távolabb került, így tehát egy útkeresést legfeljebb $O(|E||V|)$-szer ismételjük meg. Az legrövidebb út keresésére a legismertebb algoritmus a Dijkstra algoritmus (szomszédsági mátrixal, és Fibonacci kupacot alkalmazva $O(|E|+|V|\log(|V|))$ a futásideje), azonban az csak nemnegatív költségekre fut helyesen, ezért az itt csak módosítással alkalmazható (lásd [21]). Ha pedig a negatív élekre is alkalmazható Bellmann-Ford algoritmussal számoljuk a legrövidebb utat, akkor a végső futásidő $O(|E|^2|V|^2)$.

3. Generatív modellezés

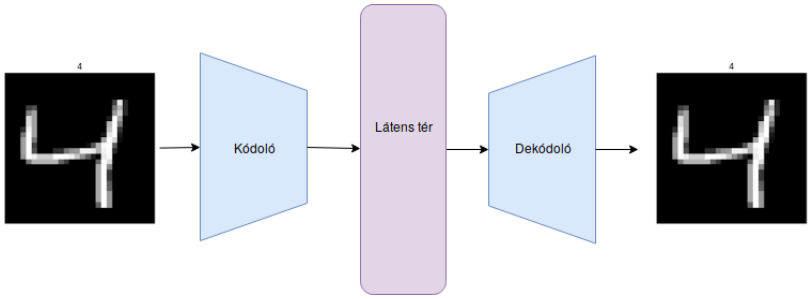
A Deep Learning egyik legnagyobb ígérete, hogy mint ahogyan az ember is, megismerje és leképezze a bennünket körülvevő világot. Ehhez nagyfokú általánosítási képesség, és az adatokból fontos tulajdonságok kinyerése szükséges. Mint ahogyan egy klasszifikáló háló képes vonalakat, objektumokat (pl.: testrészeket) megkülönböztetni, és ezek meglétét/hiányát összevetve képes a legmegfelelőbb osztályt kiválasztani.

A generatív modellek célja a beadott adathalmaz (képek, hanganyagok, videók stb.) megismerése. De nemcsak olyan egyszerű tulajdonságok megtanulása, mint dimenzió, kategorizálás stb, hanem egy olyan szintű absztrakció, amivel képes olyan új adat generálására, ami hasonlít a tanító adathalmazbeliekhez.

Hiszen ahogy a híres fizikus, Richard Feynmann is mondja [24]: "Amit nem tudok reprodukálni, azt nem is értem".

Nagyon fontos, hogy a modell minél általánosabb tulajdonságokat sajátítson el a tanító adathalmazból, és minél kevésbé csak megjegyezze/memorizálja bemenő adathalmazt. Például autóképeket generáló modell esetén, azt várjuk el a modelltől, hogy tanulja meg, hogy néz ki egy autó formája, hogy 4 kereke van stb., de az egyes autó részeket ne csak "összeollózza" más képekről, hanem valami újat hozzon létre.

A generatív modellek alkalmazása még nem jellemző az iparban, de számtalan egyszerű, és meghökkentő alkalmazása van: a generált művészképtől, a photoshop arckép-deformálós applikációjáig. De használható képkitöltésre, zajszűrésre, és képélesítésre is.



3.3. ábra. Autóenkóder folyamatábra

3.1. Generatív Autoenkóderek

Az első komoly generatív tulajdonsággal rendelkező autóenkóder, a Variációs Autóenkóder (továbbiakban VAE) megjelenése nem túl régi: [25] [26] pulikálták először 2014-ben. Bár mély mesterséges háló alapú autóenkóderek előtte is léteztek, mint generatív modell azonban rosszul teljesítettek. Később azonban a képgenerálás, és a megerősítéses tanulás [27] területén is előremutató eredmények születtek.

Mára azonban a lentebb ismertetett, és ugyanekkor felbukkanó Generatív Ellenséges Hálók (Generativ Adversarial Networks, továbbiakban GAN) meghaladták a képgenerálásban a VAE modelleket. Ugyanis a VAE egyik nagy hibáját, amit nem sikerült kiküszöbölni, hogy nagy felbontású képeknél (kisebb felbontásban is, de ott ez nem ennyire feltünő) elveszik az élesség.

3.1.1. Deep autoenkóder

Egy mély autóenkódernek három fontos része van: egy kódoló, egy dekódoló, és a veszteségfüggvény. A kódoló egy neurális háló valamilyen θ paraéterrel (Enc_p), és a célja, hogy egy x inputot elkódoljon valamilyen látens, alacsonybb dimenziós térbe ($z = Enc_p(x)$). Ezt a z kódot kapja meg a dekóder, ami egy q-val paraméterezett neurális háló (Dec_q), és a feladata az, hogy minél kisebb veszteséggel visszaállítsa az eredeti képet (lásd a 3.3 folyamatábrát).
Egy jó autóenkódertől azt várjuk el, hogy:

- a visszaállítás sikeres legyen, azaz $x \approx Dec_q(Enc_p(x))$ közel legyen egymáshoz (mondjuk l_2 normában).
- hatékonyan tömörítsen: azaz a látens tér dimenziója minél kisebb legyen
- de ne csak megjegyezze az adatokat, hanem a tömörítést a bemenet tulajdonságai szerint végezze: azaz ne csak a tanító adathalmazon teljesítsen jól, hanem a tanító adatpontokhoz hasonló inputon is.

A fentieket jobban formalizálva úgy is megfogalmazhatjuk a feladatot: Van egy $\mathbb{D} = \{x_i\}_{i=1}^n$ minták halmaza valamilyen eloszlásból. A mintákat valamilyen random generátor, $P(X|z)$ generálta z segítségével, $P(z)$ eloszlásból. A feltételezésünk szerint ez a z változó tartalmazza mindazt az információt, ami szükséges ahhoz, hogy a megfelelő X képet visszatömörítsük. Például ha X egy virágról készült festmény, akkor z tartalmazza a lefestett virág tulajdonságait: a virág fajtáját, színét, leveleinek számát, stb. Az ideális kódolót így egy feltételes eloszlással írhatjuk le: $P(z|X)$-el (adott képet különbözőféleképpen is leírhatunk), a dekódolót pedig $P(X|z)$-vel (adott leírásból a festő másféle képeket is lefesthet). $P(X)$-et így a feltételes eloszlásból is ki lehet számítani: $P(X) = \int P(X|z)p(z)dz$.

Sajnos azonban az autoenkóderek nem feltétlen ezeket az általános tulajdonságokat kódolják, hanem hajlamosak "memorizálni" a tanító adathalmazt, általánosítás nélkül. Többféleképpen is elérhetjük, hogy az autóenkóder ne csak memorizálja az inputokat, hanem z-ben valóban értékes tulajdonságok tárolódjanak.

1. a legegyszerűbb, ha lecsökkentjük a látens tér dimenzióját
2. ha a látens tér neuron rétegét megritkítjuk: kevesebb neuronnak engedjük meg az aktivációt (sparse autoencoder [28])
3. z-re valamilyen eloszlást írunk elő

Az utolsó adja a variációs autoenkóder ötletét:

3.1.2. Variációs AutoEnkóder (VAE)

A variációs autoenkóder ötlete tehát az, hogy úgy tanítja be a hálót, hogy $z = Enc(x)$ olyan eloszlású legyen ($P(z)$), amiből könnyű mintavételezni (ez a továbbiakban valamilyen paraméterű, normális eloszlás lesz: $N(\mu, \Sigma)$), valamint hogy $P(z|X)$ magas legyen. A dekódoló által definiált $Q(z|X)$ eloszlást pedig megpróbálja úgy optimalizálni, hogy minél közelebb legyen $P(z|X)$ eloszláshoz, a Kullback-Leibner divergencia szerint [29].

6. Definíció (Kullback-Leibner divergencia). *Legyen P, Q folytonos, valós valószínűségi változók.*

$$KL(P, Q) = \int_{-\infty}^{+\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \tag{5}$$

Így pedig $P(x)$-ből is könnyű lesz mintavételezni, hiszen z-ket úgy választottuk, hogy azok nagy valószínűséggel előállították x_i-ket. Tehát a kódoló neurális

hálónk a posteriori $P(z|X)$ eloszlás approximációját, míg a dekódoló a $P(X|z)$ eloszlást fogja megtanulni.

Autóenkóder célfüggvényének levezetése

Tehát minimalizálni akarjuk $Q(z|X)$ és $P(z|X)$ KL távolságát:

$$D_{KL}[Q(z|X)||P(z|X)] = \sum_z Q(z|X) \log\left(\frac{Q(z|X)}{P(z|X)}\right) \tag{6}$$

$$= E_{z\sim Q}\left[\log \frac{Q(z|X)}{P(z|X)}\right] \tag{7}$$

$$= E_{z\sim Q}[\log Q(z|X) - \log P(z|X)] \tag{8}$$

$P(z|X)$ helyére a Bayes szabály szerint behelyettesítünk:

$$D_{KL}[Q(z|X)||P(z|X)] = E_{z\sim Q}\left[\log Q(z|X) - \log \frac{P(X|z)P(z)}{P(X)}\right] \tag{9}$$

$$= E_{z\sim Q}[\log Q(z|X) - (\log P(X|z) + \log P(z) - \log P(X))] \tag{10}$$

$$= E_{z\sim Q}[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X) \tag{11}$$

$P(X)$ nem függ z-től, ezért vihettük ki a várhatóértékből. Átrendezve kapjuk:

$$\log P(X) - D_{KL}[Q(z|X)||P(z|X)] = E_{z\sim Q}[\log P(X|z) - (\log Q(z|X) - \log P(z))] \tag{12}$$

$$= E_{z\sim Q}[\log P(X|z)] - E[\log Q(z|X) - \log P(z)] \tag{13}$$

$$= E_{z\sim Q}[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)] \tag{14}$$

Megkaptuk így tehát a VAE célfüggvényét:

$$\log P(X) - D_{KL}[Q(z|X)||P(z|X)] = E_{z\sim Q}[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)]$$

A bal oldalon $\log P(X)$-et akarjuk maximalizálni, kiegészítve egy hibával, ami z reprodukálhatóságának a veszteségfüggvénye. A jobb oldal pedig pont egy autoenkóder formáját ölti: Q kódolja X-et z-be, míg P dekódolja z-t, hogy rekonstruálja X-et. A veszteség függvény is így két részből áll:

– rekonstrukciós hibából

– regularizációs hibából

Ahhoz, hogy a regularizációs hibát ki tudjuk számítani, szükségünk lesz egy újraparaméterezésre: $Enc(x)$ nem egy valós vektort fog kiadni, hanem kettőt: egyet a normális eloszlás közepeivel, egyet pedig a normális eloszlás eltéréseivel. Két normálisnak a KL távolságát egyszerű beleépíteni a veszteségfüggvénybe, ugyanis:

$$KL\left(N(\mu_1, \Sigma_1^2), N(\mu_2, \Sigma_2^2)\right) = \frac{1}{2}\left(tr(\Sigma_1\Sigma_2) + (\mu_1 - \mu_2)^T\Sigma_2^{-1}(\mu_1 - \mu_2) - k - \log\left(\frac{\det\Sigma_1}{\det\Sigma_2}\right)\right)$$

Autóenkóderek alkalmazása

Az autóenkóderek 4 fontos tulajdonságát emelném ki:

- **adatspecifikus:** az autoenkóder csak a tanító halmaz, és az ahhoz hasonló inputokra teljesít jó, de az attól nagyon eltérőekre nem. Például ha az autóenkódert egy járműveket tartalmazó adathalmazon tanítottuk, valószínűleg nem fog jól teljesíteni emberi arcokat tartalmazó adatbázison. Ez különbözik a hagyományos hang és képtömörítő eljárásoktól: a JPEG vagy MP3 tömörítő csak olyan általános feltételezésekre építenek, milyen egy általános hang, vagy kép, de hogy az minek a hangja/képe már nem.

- **veszteséges:** Mint ahogyan a JPEG vagy MP3 tömörítés is, egy bemenet be- és annak kitömörítése nem (feltétlen) fogja ugyan azt a képet eredményezni.

- **félig-megerősítéses tanítás:** az autóenkóderek az úgynevezett semi-supervised tanuláshoz tartoznak, hiszen bár nem címkézett adathalmazon dolgozik, a tanítás során amikor elkódol és dekódol egy inputot, az eredményt pedig összeveti a kiinduló inputtal. Így bár eredetileg egy nem címkézett adathalmazból indul ki, csak önmagának gyártja a címéket. Így tehát a supervised (megerősítéses), és az unsupervised tanulás között helyezkedik el.

- **automatikus tanulás** ha más adathalmazt szeretnénk tanítani, nem kell kézzel paramétereket az adathalmazra hangolni

Ezért nem terjedtek még el kép és hangtömörítő autóenkóderek, mert csak specifikus adathalmazokon teljesítenek jobban, mint a hagyományos tömörítők.
Van azonban két sikeres alkalmazási területe az autóenkódereknek:

- zajcsökkentés

– dimenzócsökkentés, adatvizualizáció: a leggyakoribb magas dimenziós adatvizualizációs eszköz a t-SNE (t-eloszlású stochasztikus szomszédos beágyazás). Azonban ez csak 100 alatti dimenzióra működik jól. Ezért ennél magasabb dimenziós adat esetén általában egy autóenkóderrel leképezik egy 100 dimenziós térbe, és arra alkalmaznak egy t-SNE módszert.

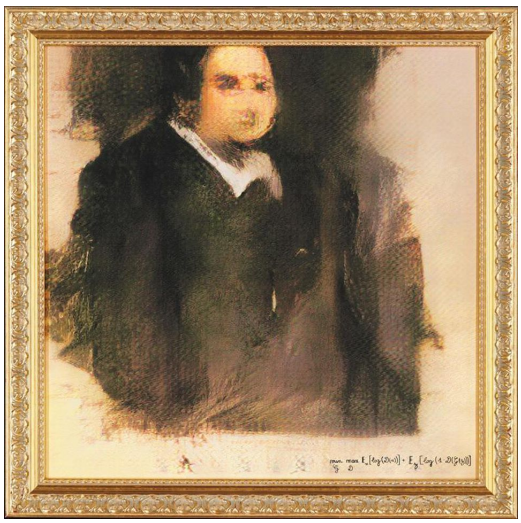
3.2. Generativ Adversarial Networks (GAN)

A GAN-ok egy nem túl rég megjelent kutatási terület, amit Ian Goodfellow [30] 2014-ben publikált először. A GAN-ok célja egy eloszlás megtanulása, és abból példányok generálása.

Az addigi konvolúciós hálók már jól meg tudtak tanulni (kép)eloszlásokat, pontosabban nagy valószínűséggel le tudták vetíteni az egyes képeket a megfelelő címke-osztályokba. Azonban egyes osztályokból a tanuló adathalmazban nem szereplő képek generálása egy nehéz feladat volt.

Ugyan a klasszifikáló hálóknak azt is meg lehet tanítani, ismerje fel egy kilógó kép felbukkanását az adathalmazban (pl új címke hozzávételével), de ezek a módszerek tipikusan csődöt mondanak egy hamisítóval szemben. Ezek kiküszöbölésére nyújtanak megoldást a GAN-ok.

A GAN-oknak annak ellenére, hogy nemrég jelentek csak meg, gyakorlati haszna is van: egy 2018 novemberi aukción egy GAN által "festett" kép több mint $400 ezer dollárért kelt el [31]! Érdemes megnézni a 3.4 ábrán.



3.4. ábra. "Edmond de Belamy, from La Famille de Belamy" című festmény az első New York-i AI festmény aukción $432 ezer dollárért kelt el. A képet generáló hálót 15 ezer 14-20. századi festményen tanították.

3.2.1. GAN elmélet

A GAN-ok célja tehát egy eloszlás megtanulása, és abból példányok generálása. De honnan tudjuk, hogy egy generált kép jó-e, vagy sem, és honnan kapunk információt (pl.: gradienst) a modellünk számára, hogy egy generált képet milyen irányban lehetne még élethűbbé tenni?

A GAN alapötlete, hogy 2 neurális háló egymásal verseng/egymástól tanul, tehát nincs szükség külső segítségre (unsuperwised learning), ami megmondja, hogy egy kép mennyire jó. A két neurális háló:

- A Diszkriminátor: az a célja, hogy az bemenetről minél nagyobb valószínűséggel megmondja, az adott eloszlásból jön-e, vagy sem.
- A Generátor: az a célja, hogy átverje a diszkriminátort, azaz hogy olyan kimeneteket gyártson, amire a diszkriminátor minél nagyobb valószínűséggel mondja, hogy az adott eloszlásból jön.

Egy generátum "jóságát" tehát nem külsőleg mondjuk meg (például egy címke segítségével), hanem az dönti el, sikerült-e átverni a diszkriminátort. Ennek viszont az a hátulütője, hogy nehéz összehasonlítani két GAN teljesítményét, mert mindig csak a két háló egymáshoz mért teljesítményét látjuk, de alapvetően nincs egy abszolút mérőszám, amihez viszonyíthatnánk. Erre alább még bővebben viszszatérek.

Tehát úgy is felfoghatjuk, hogy a diszkriminátor, és a generátor egy zéró összegű játék két játékosa: egy pénzhamisító, és egy rendőr. A hamisító próbál minél megtévesztőbb bankjegyeket/eloszlábeli elemeket gyártani, a rendőr pedig pedig próbálja a hamis bankjegyeket/eloszlásbeli elemeket lebuktatni.

1. Megjegyzés. *A generátor nyílván nem verheti át $\frac{1}{2}$-nél nagyobb valószínűséggel a diszkriminátort, hiszen ha az $\frac{1}{2}$-$\frac{1}{2}$ valószínűséggel módja egy inputra hogy hamis, illetve hogy igaz, akkor $\frac{1}{2}$ valószínűséggel lebuktatja a hamisítót!*

A generátornak szüksége van random bemenetre, mert ha determinisztikusan működne, akkor a diszkriminátor könnyen lebuktathatná. Tehát egy adott látens térből képez egy, a képeket (vagy az bemeneteket) tartalmazó térbe. A diszkriminátor pedig ebből a térből képez a [0,1] halmazba, annak megfelelően, hogy szerinte az input a hamis eloszlásból, avagy az igazi eloszlásból jött. Fromálisan tehát:

- p_z legyen a látens téren vett eloszlás
- p_r legyen az adathalmazon vett eloszlás
- p_g pedig legyen a generátor eloszlása x inputon.

A generátort ezentúl egy G, a diszkriminátort pedig egy D neurális hálóval helyettesítjük (θ_g, illetve θ_d paraméterrel).
A diszkriminátor célja tehát a $x \sim p_r$ inputokon a

$$\mathbb{E}_{x\sim p_r} log(D(x)) \tag{15}$$

az $x \sim G(z), z \sim p_z$ inputokon pedig

$$\mathbb{E}_{z \sim p_z} log(1 - D(G(z))) \tag{16}$$

maximalizálása. A zéró összegű játék tehát így írható fel:

$$\min_G \max_D \mathbb{E}_{z \sim p_z} log(1 - D(G(z))) + log(D(x)) \tag{17}$$

A tanítást backpropagation szerint végezzük. Minden epochban veszünk $\{z_1, ..., z_k\}$ k minibatch-nyi véletlen bemenetet, valamint $\{x_1, ..., x_k\}$ k minibatch-nyi elemet az adathalmazból. A diszkriminátor paramérereit növeljük a stochasztikus gradiens szerint:

$$\theta_d := \theta_d + \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{k} [log(1 - D(G(z_i))) + log(D(x_i))] \tag{18}$$

A generátort pedig csökkentjük a stochasztikus gradins szerint:

$$\theta_g := \theta_g - \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{k} log(1 - D(G(z_i))) \tag{19}$$

Kezdetben azonban $log(1 - D(G(z)))$ nulla körüli lesz, hiszen a generátor még kezdetben gyengén teljesít, ezért ha eszerint minimalizálunk, a gradiens itt kicsi. Így érdemes a generátornak $log(D(G(z)))$ szerint maximalizálni, hiszen így amikor backpropagation-t végzünk, akkor a gradiens már kezdetben is jó irányba vezeti a generátort.

A gyakorlatban érdemes a diszkriminátort többször frissíteni egy epochon belül, mint a generátort, mert ha a generátor túl nagy valószínűséggel veri át a diszkriminátort, akkor a generátor sem fejlődik, hiszen a gradiens közel 0 lesz.

3.2.2. GAN hibái

A GAN-oknak sajnos számos ismert hátulütője van.

- Mivel a neurális hálók gradienscsökkentését függetlenül végezzük, egyáltalán nem biztos, hogy a zéró összegű játékban megtalálnak egy Nash egyensúlyt, tehát könnyen megeshet, hogy a fent leírt algoritmus nem konvergál.
- Ha a diszkriminátor túl rosszul teljesít, a generátor nem kap releváns információt (gradienst), hogy milyen irányba lehetne fejlődni. Viszont ha a diszkriminátor túl jó, akkor pedig a a generátor gradiense szinte eltűnik, és így a generátor csak nagyon lassan tanul.
- Gyakran megesik, hogy a GAN pici változatossággal rajzol képeket, és nem tanulja meg az input változatosságát (Pl.: MNIST esetén nem rajzolja ki az összes számjegyet).

– Mint már említettem nehéz összehasonlíani két GAN teljesítményét, valamint a túltanulásra sincsenek jó mérőszámok.

3.3. Wasserstein GAN (WGAN)

A tanítás során a G generátorunk P_{θ_g} eloszlást generál. Azt szeretnénk, hogy ez konvergáljon, azaz hogy létezzen P_∞ eloszlás, hogy $\lim_\theta \rho(P_{\theta_g}, P_\infty)$ létezzen, valamilyen ρ metrikára. Ennek a metrikának a megválasztásán sok múlik, hiszen ez épül be a loss függvénybe, valamint a konvergencia is függ a metrika megválasztásától.

A továbbiakban jelölje Σ az $\mathbb{R}^d$ feletti borel halmazokat. Továbbá legyenek $\mathbb{P}_r, \mathbb{P}_\theta$, $\mathbb{R}^d$–en értelmezett valószínűségi mértékek.

Kullback-Leibner, Jensen-Shannon divergencia és Wasserstein távolság

– Totális variancia:

$$\delta(\mathbb{P}_g, \mathbb{P}_r) = \sup_{A \in \Sigma} |\mathbb{P}_g(A) - \mathbb{P}_r(A)| \tag{20}$$

– Kullback-Leibner divergencia (KL):

$$KL(\mathbb{P}_g, \mathbb{P}_r) = \int P_r \log(\frac{P_r}{P_g}) d\mu \tag{21}$$

ahol P_g, P_r abszolút folytonos μ-re nézve. A definícióból látszik, hogy nem szimmetrikus a KL divergencia. Az is látszik, hogy az értéke végtelen, ha létezik $P_g(x) = 0$, $P_r(x) > 0$ pont.

– Jensen-Shannon divergencia (JS):

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r, \mathbb{P}_m) + KL(\mathbb{P}_g, \mathbb{P}_m) \tag{22}$$

ahol $P_m = \frac{P_r + P_g}{2}$. A Jensen-Shannon divergencia a KL divergenciával ellentétben szimmetrikus, és mindig értelmezve van, hiszen μ-t választhatjuk P_m-nek!

– Earth-Mover avagy Wasserstein távolság:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} ||x - y|| \tag{23}$$

Ahol $\Pi(P_r, P_g)$ olyan $\gamma(x, y)$ összetett eloszlások halmaza, amik marginális eloszlása P_r illetve P_g. Tehát ha a két eloszlásra mint földkupacokra gondolunk, a Wasserstein távolság megmondja, mennyi energiát kell befektetnünk, hogy az egyik földkupacot a másik földkupaccá alakítsuk ($\gamma(x, y)$ mondja meg, x pontból mennyi földet mozgassunk át y pontba).

Az eredeti WGAN cikk [32] egy egyszerű példával mutatja meg a WGAN előnyeit: Legyen Z [0,1] uniform random. $P_0 = (0, Z)$, $P_\theta = (\theta, Z)$ eloszlások $\mathbb{R}^2$ felett. Ekkor ha $\theta! = 0$:

- $W(P_0, P_\theta) = |\theta|$
- $JS(P_0, P_\theta) = log(2)$
- $KL((P_0, P_\theta) = KL((P_\theta, P_0) = +\infty$
- $\delta(P_0, P_\theta) = 1$

$\theta = 0$ esetén mindegyik távolság természetesen 0-t ad. Tehát látszik, a fenti négy távolság közül egyedül a Wasserstein távolság nyújt némi simaságot, és P_θ csak a Wassertein távolság szerint konvergát P_0-hoz. A Wasserstein távolság alkalmazását a feladatra az alábbi tétel is alátámasztja:

4. Tétel. *Legyen $\mathbb{X}$ kompakt tér, $\mathbb{P}_n$ eloszlások X felett. Ekkor az alábbiak ekvivalensek:*

- $W(\mathbb{P}_n, \mathbb{P}) \to 0$
- $\mathbb{P}_n \to \mathbb{P}$ *gyengén.*

Továbbá ha $JS(\mathbb{P}_n, \mathbb{P}) \to 0 \implies W(\mathbb{P}_n, \mathbb{P}) \to 0$.
Valamint $KL(\mathbb{P}_n || \mathbb{P}) \to 0$ vagy $KL(\mathbb{P}_n || \mathbb{P}) \to 0 \implies JS(\mathbb{P}_n, \mathbb{P}) \to 0$.

3.3.1. Kantorovitch-Rubinstein dualitás tétel

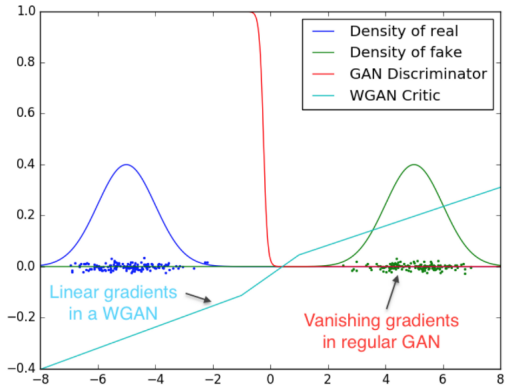
(23) kiszámolása még diszkrét eloszlások között is meglehetősen költséges lehet. Érdemes ezért alkalmazni rá a Kantorovits-Rubinstein dualitástételt:

$$W(\mathbb{P}_n || \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\ P_r}[f(x)] - \mathbb{E}_{x\ P_\theta}[f(x)] \tag{24}$$

Itt f-ről azt követeljük meg, hogy 1-*Lipschitz* függvény legyen. Ám vegyük észre, hogy ha csak K Lipschitz tulajdonságot követelünk f-től akkor a Wasserstein távolság K szorosát kapjuk. Így ha van egy $\{f_w : w \in W\}$ K-Lipschitz családunk, akkor elegendő

$$\max_{w \in W} \mathbb{E}_{x \sim P_r(x)}[f(x)] - \mathbb{E}_{x \sim p(z)}[f_w(g_\theta)] \tag{25}$$

feladatot megoldani, ha (24)-ben a szuprénum valamilyen $w \in W$ elemén felvétetik, vagy elegendően közel van hozzá. Ami a mi feladatunk esetében annyit jelent, hogy a θ-val paraméterezett diszkriminátor hálót úgy paraméterezzük, hogy közel optimális legyen (ami egy jól felépített háló esetén nem túl nagy elvárás).



3.5. ábra. Két Gauss eloszlás között a Wasserstein szerinti gradiens több információt rejt magában, sokkal simább gradienst ad, mint a GAN, és még akkor is jó irányba mutat, amikor a generátor rossz képeket állít elő [32]

Így ha veszünk egy kompakt W téren paraméterezett f diszkriminátort, ami minden $w \in W$ paraméterre K-Lipschitz, akkor mindössze csak annyi a dolgunk, hogy $\mathbb{E}_{x \sim p(z)}[\nabla f_w(g_\theta)]$ szerint gradienscsökkentést végzünk a θ-val paraméterezett diszkriminátorunkon.

A 3.5 ábrán jól látszik a Wassertsein távolságra áttérésnek az előnye a sima GAN veszteségfüggvényhez képest.

3.3.2. WGAN algoritmus

De hogyan érjük el, hogy a f_w K Lipsczitz legyen? [32] első ötlete az volt, hogy a neuális háló súlyait egy adott intervallumra kell korlátozni (pl [-0.01,0.01] közé), így a háló valamilyen K konstanssal Lipschitzes lesz.

De egy ilyen korlátozás nagyon radikális, amivel több probléma is adódhat. Hiszen ha a korlát túl nagy, a diszkriminátor túl lassan éri el az optimumát. Ellenben ha túl picire vesszük a korlátokat, a gradiens könnyen eltűnhet, ha nem használunk Batchnormalizációt, vagy túl sok rétege van a hálónak.

A ([32]) által javasolt algoritmus:

Algorithm 2 WGAN Algorithm

Javasolt paraméterek: $\alpha = 0.00005, c = 0.01, m = 64, n_{critic} = 5$

Result: g_θ generátor, és f_w diszkriminátor

while θ nem konvergál **do**
  for i=0..n_{critic} **do**
    $\{x^i\}_{i=1}^m \sim \mathbb{P}_r$ ▷ Vegyünk egy batch-nyit a tanuló adathalmazból:
    $\{z^i\}_{i=1}^m \sim p(z)$ ▷ Generáljuk egy batch-nyit a látens tér random eloszlásából:

    Számoljuk ki a stochasztikus gradiensét a diszkriminátornak:
    $\xi = \nabla_w[\frac{1}{m}\sum_{i=1}^m f_w(x^i) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^i))]$
    Frissítsük a diszkriminátor paramétereit gradienscsökkentéssel:
    $w \leftarrow w + \alpha\text{RMSProp}(w, \xi)$
    $w = clip(w, -c, c)$ ▷ A K-Lipschitzesség elérése érdekében vágjuk le a súlyokat
  $\xi = -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_{theta}(z^i))$ ▷ Számítsuk ki a gradiensét a generátornak
  $\theta = \theta - \alpha\text{RMSProp}(w, \xi)$ ▷ Frissítjük a generátor paramétereit

3.4. Improved Wasserstein GAN

Az eredeti WGAN cikk alapján javasolt súlyok "levágása" nagyon radikális lépés, és bár ez általában stabilabb tanulást eredményez, a háló nem várt viselkedéseket produkálhat.

Ezért [33] szerzői új módszet javasolnak. Az 1-Lipschitz tulajdonságot a loss függvénybe bevezetett büntetéssel ajánlják elérni. Ezt az alábbi célfüggvény minimalizációjával javasolják:

$$L = \mathbb{E}_{\hat{x}\sim P_\theta}[D(\hat{x})] - \mathbb{E}_{x\sim P_r}[D(x)] + \lambda\mathbb{E}_{\tilde{x}\sim P_{\tilde{x}}}[(||\nabla_{\tilde{x}}D(\tilde{x})||_2 - 1)^2] \tag{26}$$

Ahol $\tilde{x}$ pontokat véletlenszerűen olyan pontpárok által meghatározott egyenesből generáljuk, ahol az egyik pont $\mathbb{P}_r$, a másik $\mathbb{P}_g$ eloszlásból jön. Ez a feltétel intuitíven onnan jön, hogy egy optimális diszkriminátor (ami *1-Lipschitz*) gradiense az ilyen pontok között éppen 1.

A GAN implementációk kezdben mind használtak Batchnormalizációt mind a generátor, mind a diszkriminátor hálójában. Azonban a gradiens normalizálása azt is eredményezi, hogy erre a diszkriminátorban nincs feltétlen szükség.

Valójában az elmélet szerint elég lenne csak annyit feltenni, hogy egy ilyen interpolált pontban a gradiens legyen ≤ 1, de a gyakorlatban a kétoldali büntető feltétel bizonyult jobbnak.

4. Generatív modellek kiértékelése

A gépi tanulás sok modell sikerességének kiértékelése meglehetősen egyszerű (főleg címkézett, teszt résszel is rendelkező adathalmazok esetén), például:

1. a klasszifikáló modelleket a teszt adathalmazon a klasszifikálás pontosságával (accuracy)
2. objektumfelismerés esetén a teszt-adathalmazon elért sikerességgel

jellemezhetjük. Ilyen címkézett teszthalmaz, jóldefiniált kiértékelőfüggvény a generatív modellek esetén sajnos nem áll rendelkezésre.

4.1. Alapprobléma

A generatív modellek legfontosabb kiértékelője maga az ember, hiszen a generatív modellünkkel általában az ember számára értelmezhető dolgokat szoktunk előállítani (mint hang, kép, vagy videó). Azonban a képgenerálás már eljutott arra a szintre, hogy már nem az a kérdés, melyik modell generátumai kevésbé torzak. Az elmúlt 5 évben a GAN-ok komoly fejlődésen mentek keresztül, ahogyan ezt a 4.6 ábra is mutatja. A legjobb modellek nagy felbontású generátumairól már nehéz megmondani, igazi képek-e, vagy generátumok.

4.6. ábra. Ian Goodfellow: az utóbbi 5 év GAN fejlődése [34]

Azon kívül, hogy egy modell emberi értékelése nagyon szubjektív, az ember csak kevés számú mintát tud megnézni a tanító, és generált elemekből. Így nem várható el, hogy 10 000 -es nagyságrendben nézzen át generátumokat. Az ember csak azt tudja eldönteni, hogy valósághű képet lát-e, de hogy a generátor mennyire tanulta meg a tanító eloszlást, arról meglehetősen keveset tud mondani.

Így tehát attól, hogy egy modell már át tud verni egy embert, még nem jelenti azt, hogy valóban meg is tanulta a tanító eloszlást megfelelően: nincsenek-e olyan

hibái, mint a mode-collapse (alacsony változatosságú képek generálása). Könnyen előfordulhat például, hogy a modell az adathalmaznak bizonyos tulajdonságait nem tanulja meg (például emberi arcok esetén nem generál szemüveges embereket, vagy bizonyos embercsoportok alul/felülreprezentáltak).

De a háló tanítása során, vagy esetleg paramétereinek finomhangolásánál sincs ott az ember, hogy eldöntse, mennyire jó a generátor. A fentiek miatt tehát szükség van egy matematikai alapokon nyugvó kiértékelő függgvényre, amely relevánsan értékelni tudja a generátor modellt. Egy ilyen, megbízhatóan működő kiértékelő függvényt sajnos nem létezik eddig. Az irodalomban leggyakrabban említettek az "Inception Score" és a "Frechet Score" (lásd 4.2 fejezet).

Egy jól működő kiértékelő függvénynek nagy segítsége lenne egy olyan metrika, ami két kép között (az emberi szem szerinti) hasonlóságot jól írja le. Egy ilyen metrikának rezisztens kellene lennie mozgatásra, torzulásra, háttér változásra, stb. A bevett l_2 metrika fekete-fehér, és kis felbontású képekre még nem működik olyan rosszul, ám nagy felbontású és színes képeknél szinte hasznavehetetlen.

4.2. Az irodalomban felmerülő kiértékelő metrikák

Az irodalomban számos próbálkozás született, hogy a generatív modellek teljesítményét megfelelően mérjék, de mindeddig nem alakult ki egy általánosan elfogadott mérőszám, ami a modellek teljesítményét jól mérje. Ezekre azért is szükség lenne, hogy a generatív modellek fejlődését megfelelően elemezhessük.

Átlagos Log-Likelihood

Az általános GAN-ok a generált eloszlás, és a tanító-eloszlás közötti Kullback-Leibner divergenciát minimalizálja, mert azt feltételezi, hogy egy jó generátor eloszlása közel lesz a tanító-eloszláshoz. Így ezen feltételezés szerint ezen távolság kiszámolása jó metrika egy generátor teljesítményének mérésére. A baj csak az, hogy egyik eloszlás sem ismert, így nehézkes azt (magasabb dimenzióban) egzakt módon kiszámolni.

Azonban a Parzen ablak becslés minták segítségével approximációt ad ezekre az eloszlásokra:

7. Definíció (Parzen ablak avagy Kernel sűrűségfüggvény becslés). *Adott $\{x_1, \ldots, x_n\}$ minta adott p eloszlásból. p Kernel sűrűségfüggvény becslése adott K Kernelfüggvénnyel (ami általában Gauss):*
$P(x) \approx \frac{1}{z} \sum_{i=1}^{n} \boldsymbol{K}(x - x_i)$, *ahol z egy normalizáló konstans.*

A generátumokat mint összetett Gauss eloszlás középpontjait vesszük, és a kevert eloszlást ennek segítségével számoljuk ki.

Így tehát kiszámíthatjuk a log-likelihoodját a tanító adtahalmaznak a generátor eloszlásra nézve:

$$L = \frac{1}{N} \sum_i \log P(x_i)$$

Számos hátulütője van ennek a módszernek:

- A Parzen ablakkal közelíteni az igazi loglikelihood eloszlást magasabb dimenzióban nem kielégítő. Még kis képekre is (6*6) rengeteg mintára van szükség, hogy a log-likelihoodoz jó közelítést kapjunk.
- Theis ([35]) megmutatta, hogy a log-likelihood és a képminőség csak gyenge kapcsolatban van egymással: Előfordulhat, hogy egy model gyenge log-likelihood mellett kíváló generátumokat gyárt, míg más modell közeli log-likelihood esetén nagyon rossz generátumokat. Az utóbbira egy példa, ha egy jó modellt ($\alpha \approx 0.01$), és egy rossz modell ($1 - \alpha$ súllyal) súlyozott összegét vesszük. Ennek magas lesz a loglikelihoodja, de mégis gyenge generátumokat ad.

Inception Score (IS)

GAN-ok kiértékelésére talán az egyik leggyakrabban használt mértéke a Salimani ([36]) által javasolt "Inception Score". Egy előretanított neurális hálót (Inception Net [37] klasszifikáló háló, amit az ImageNet [38] adathalmazon tanítottak) használ, hogy ennek segítségével a generátumok kívánt tulajdonságait kapja meg: magas változatosság, magas képminőség.

Ezt úgy éri el, hogy a mintát átengedi a klasszifikáló hálón, és annak a címke eloszlásából következtet az előző kettőre:

1. a könnyű klasszifikálhatóság: magas képminőséget,
2. a címkék magas változatossága pedig generátumok változatosságát

prognosztizálja. Formálisan, legyen $p(y|x)$ címkék feltételes eloszlása adott x generátumra. Ennek entrópiája könnyen klasszifikálható elemekre alacsony kell legyen. $p(y)$ pedig jelölje a címkék marginális eloszlását. Ez egyenletes címke eloszlás esetén az entrópia pedig magas lesz.

8. Definíció (Inception Score).

$$IS = \exp\left(\mathbb{E}[\mathbb{KL}(p(y|x)||p(y))]\right) = \exp H(y) - \mathbb{E}_x[H(y|x)]$$

ahol $p(y)$-t

$$p(y) \approx \frac{1}{N} \sum_{n=1}^{N} p(y|x_n = G(z_n))$$

-vel közelíthetjük.

Tehát az Inception Score olyan generatív modelleket részesít előnyben, ahol $p(y|x)$ alacsony, és $p(y)$ magas. Az Inception score az általa támasztott két elvárásnak: a képminőség és diverzitásnak többé kevésbé megfelel. Azonban az alkalmazhatósága több ok miatt is megkérdőjelezhető:

- Mint a legtöbb GAN-metrika, az Inception Score is a "memorizáló GAN"-okat (amely a tanító adathalmazhoz túl közel eső képeket memorizálja, a generálás során ebből mintavételez) részesíti előnyben, és így nem képes a túltanulást büntetni.
- Mivel az InceptionNet rengeteg címkével rendelkező adathalmazon volt tanítva, így olyan képeket részesít előnyben, amelyeken jó tárgyak vannak, de a kép egészéről: azaz hogy a kép egyben mennyire valósághű, már kevesebbet mond.
- A kép csak a generált eloszlással foglalkozik, és teljesen ignorálja a tanító eloszlást. Bár nagyon valószínűtlen, hogy arcképekkel tanítunk egy hálót és a generátor nagyon élethű állatképeket generál, az IS ennek szintén magas pontszámot ad, hiába nem tanult meg semmit az adathalmazból a modell. Így az IS nem jutalmazza azt a modellt, amelyik értékes tulajdonságait tanulta meg a tanító adathalmaznak, csupán csak azokat a modelleket, amelyek éles, és változatos képeket generálnak.
- Asszimetrikus: egy ideális kiértékelő pontszámtól azt várjuk el, hogy ha a tanító mintákat felcseréljük a generátor mintáival, akkor a pontszám ne változzon. Az előző pont szerint ez nagyon sérül az Inception Score esetén, hiszen az IS figyelembe sem veszi a tanító mintákat a pontszámítás során.
- Érzékeny a kép minőségére/felbontására

Az Inception Score definíciójában a két tagot érdemes külön is megvizsgálni. Zhou és szerzőtársai [39] azt találták, hogy $H(y|x)$ az elvártaknak megfelelően működik, azonban $H(y)$-nél anomáliákat találtak.

Általában egy mode-collapse generátor (kevés fajta kép generálása) alacsony Inception Score-t kap. Sőt elméletben az a modell, amely csak egyfajta képet generál, ott ugye $p(y) = p(y|x)$, az IS minimális lesz, hiszen a KL távolságuk 0, és így $IS_{model} = exp(0) = 1.0$.

Ennek ellenére nem tartják megbízható indikátornak az IS-t mode-collapse jelzésére, ugyanis egy modell, amely minden ImageNet osztályból csupán egy elemet jegyez meg, magas IS pontot kap.

Variációk Inception Score-ra

Módosított Inception Score

Számos javaslat született az IS javítására. A minden osztályból egy elemet memorizáló generátorok büntetésére Gurumurthy [40] nem csak az összes generátum a címkék szerinti feltételes eloszlását bünteti, hanem az egyes osztályokon belüli magas változatosságot is megköveteli. Ezt egy cross-entrópia típusú pontszámmal teszi meg: $-p(y|x_i)log(p(y|x_j))$ ahol x_i adott generátum, és ehhez a genrátumhoz tartozó címke szerint választunk az inception modellből x_j pontokat. Kompakt formában tehát:

$$\text{m-IS} = \exp\left(\mathbb{E}_{x_i}[\mathbb{E}_{x_j}[\mathbb{KL}(P(y|x_i)||P(y|x_j))]]\right)$$

Mode Score

A tanító adathalmaz ignorálásának kiküszöbölésére [41]-től az alábbi ötlet született:

$$\text{Mode Score} = \exp\left(\mathbb{E}[\mathbb{KL}(p(y|x)||p(y^{train})) - \mathbb{KL}(p(y)||p(y^{train}))]\right)$$

ahol $p(y^{train})$ a tanító adathalmazból számolt empírikus eloszlása a címkéknek. Azonban [42] megmutatta, hogy az Inception Score, és a most definiált Mode Score valójában ekvivalensek.

Fréchet Inception Távolság (FID)

Talán a legmegbízhatóbb, és leggyakrabban használt kiértékelési metrika a Heusel ([43]) által beveetett Fréchet Inception Score. Az FID leképezi a generátumokat egy látens térbe, amit az InceptionNet (vagy bármely előre betanított klasszifikáló háló) jól megválasztott rétege definiál. A rétegnek, mint többváltozós Gauss eloszlásnak kiszámolhatjuk mind a generátor, mint a tanító eloszlásra az így definiált Gauss várható értékét, és szórását (μ_g, Σ_g). Az így definiált két eloszlás közti (itt: Wasserstein-2) távolság lesz a Fréchet pontszám. Az Inception scoreral ellentétben a kissebb Fréchet pontszám jelzi a jobb modellt:

9. Definíció (Fréchet Inception Távolság).

$$FID(r, g) = \|\mu_r - \mu_g\| + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$$

Annak ellenére, hogy a két Gauss eloszlásnak csak két momentumát veszi figyelembe a pontszám számítása során, valamint, hogy nem is garantált, hogy a látens térben a két eloszlás Gauss típusú, a Fréchet Score meglehetősen jól teljesít: robusztusság és számítási kapacitás szempontjából is.

Előnyei az IS-el szemben:

1. A tapasztalat szerint robusztusabb a zajra, és modellek sorbarendezésében jobban megegyezik az ember megítélésével, mint az Inception Score.
2. Sőt a Módosított Inception Score-hoz hasonlóan rossz pontszámot ad olyan modelleknek, amelynek az egyes osztályokon belül alacsony a változatossága.
3. Az FID rosszabb lesz, ha egy képhez véletlen tárgyakat teszünk
4. Figyelembe veszi a tanító eloszlást

Generativ Ellenséges Metrika

GA-ok közötti jobbik kiválasztására egy természetes ötlet, hogy a két modell generátor-diszkriminátor részét a másikkal is versenyeztessük, és megnézzük melyik veri át nagyobb valószínűséggel a másikat. Ezt végezhetjük például hipotézisvizsgálattal: Legyen $M_1 = (D_1, G_1)$, $M_2 = (D_2, G_2)$ a két modell a megfelelő generátor-diszkriminátor hálókkal. A hipotézis $\mathbb{H}_0$, hogy a két modell ekvivalens, míg $\mathbb{H}_1$ szerint G_1 jobban átveri D_2-t, mint G_2 átveri D_1-et, vagy fordítva. Formálisan:

$$\mathbb{H}_0 : p(x|y = 1; M_1') = p(x|y = 1; M_2') \iff \tag{27}$$

$$p(y = 1|x; D_1)p(x; G_2) = p(y = 1|x; D_2)p(x; G_1) \tag{28}$$

hol $M_1' = (D_1, G_2)$, $M_2' = (D_2, G_1)$ a kicserélt modellek. Így nem kell mást tenni, mint $\frac{D_1(x_2)}{D_2(x_1)}$ hányadost megfigyelni (ahol $x \sim G_1$, $x_2 \sim G_2$).

A metrikának azonban előfeltétele, hogy a modelleket azonos adathalmazon tanítsuk, ami nagyban bekorlátozza különböző adathalmazokon tanított teljesítményének összehasonlítását. A metrika azonban ha sok modellt szeretnénk összehasonlítani, számításilag költséges lehet, hiszen minden egyes párra ki kell számítani ezt az értéket. Sőt, még akkor sem vagyunk előrébb, ha egy sorrendet akarunk felállítani, hiszen körbeverések lehetségesek. Ezt azonban lehet orvosolni az alábbi metrikával.

Tournament Pontszám

Az egyes generátorok, diszkriminátorok mint versenyzők összehasonlítása Olsson ([44])-tól származik. Az ötletet a sakkban ("Elo" pontszám, Élő Árpád magyar

születésű amerikai fizikus/sakkozó után [45]), vagy a teniszben (UTR = Universal Tennis Rating) a világranglista pontszámítása során alkalmazott pontszámításhoz hasonló. Ezek a pontszámok természetesen figyelembe veszik, hogy az adott győzelem/vereség milyen erős játékosok között történt (Így egy erős játékos megverése jobban növeli a pontszámot, és fordítva is, egy gyenge játékostól a kikapás jobban rontja). A játékosokat egy Gauss változóval írjuk le, ahol a várható érték adja meg a játékos jelenlegi tudását, a szórás pedig egy felívelés/lejtmenet gyorsaságát jelzi.

A pontszám hasznos információt nyújthat egy tanulási folyamat során, az egyes fázisok tanulási mértékére, valamint különböző modellek rangsorolására (akár ugyanazon modellek, más paraméterekkel). A tournamentekben minden játékos vagy egy generátor (akinek a célja a verseny során a diszkriminátorok átverése), vagy egy diszkriminátor (akiknek célja igazi és generátumok minél biztosabb megkülönböztetése). Példa kétféle pontszámításra:

1. Nyerési arány: Egy G generátort és egy D diszkriminátort két adott méretű batchen (ez a méret függ attól, mekkora a tanító adathalmaz változatossága) hasonlítunk össze. Az egyik batch tanító-beli adatokat tartalmaz, a másik pedig G generátumait. Ezeket D címkézi $\{$*Igazi, Generátum* $\}$ címkékkel. Ha legalább a felét jól címkézi, akkor a meccset D nyeri, különben pedig G nyer.

   A tournament végén mind a generátorokra, mind a diszkriminátorokra kialakul egy arányszám: hogy a meccsei hány százalékát nyerte meg; ez adja a rangsort.

2. Rutin pontszám: Ez a pontszám figyelembe beszi a torna során az egymás után levő meccsek eredményeit, és ez alapján számítja (online) a generátorok/diszkriminátorok pontszámát.

A Fréchet score-ral szemben ez nincs bizonyos tulajdonság halmazhoz kötve, például ha egy játékos az ellenfelek valamely hibájára jól ráérez , akkor akár nagyon sikeres is lehet. Vegyünk egy szélsőséges példát: a tanító képeink emberi arcképek. Az egyik modell generátora pedig olyan képeket generál, amelyeken az embereknek hiányzik a fülük. A Fréchet score csak akkor képes az ilyen modelleket hátrébb sorolni, ha az általa használt klasszifikáló háló képes felismerni, hogy az arcképnek van-e füle. Míg a Tournament pontszámítás során nincsenek ezek a tulajdonságok előre lefixálva: ha valamely diszkriminátor jól ráérez egy ilyen tulajdonságra, akkor azt jutalmazza a rendszer.

Valamint az ember is beszállhat a tournamentbe diszkriminátor játékosként, (vagy egy teszt adathalmazzal generátorként) és így a emberi értékelés is számszerűsíthető.

5. Leimplementált kiértékelési metrikák

5.1. Wasserstein távolság alkalmazása a kiértékelésben, és az alkalmazás nehézségei

1. Jelölés. *Jelölje* $[\![n, m]\!]$ *az* $\{n, n+1, \ldots, m\}$ *halmazt*

A 3.3 fejezetben bemutattam, hogy a Wasserstein távolság alkalmazása GAN-okra jobb gradienst ad, mint a Kullback-Leibner divergencia. Így a tanulás gyorsabb, és megbízhatóbb lesz. A WGAN-ban tehát az igazi eloszlás (p_r, amiből a tanító adathalmazt mintavételeztük), és a generált eloszlás (p_g amit a neurális hálónk kódol el) közötti Wasserstein távolságot minimalizáljuk. Így tehát ha tudnánk ennek a pontos értékét, akkor azzal jellemezhetnénk a generatív modellünket, mennyire közel van az optimumhoz. Továbbá a tanulást is lehetne így jellemezni, mennyire csökken a Wasserstein távolság egy epochon belül. Azonban a Wasserstein távolság direkt kiszámolása meglehetősen reménytelen. Még úgy is, ha a képeket nem mint folytonos térben ($[0, a] \times [0, b] \to [\![0{,}255]\!]^3$), hanem mint diszkretizált térben ($[\![0, n]\!] \times [\![0, m]\!] \to [\![0{,}255]\!]^3$) értelmezzük.

$64x64$ méretű színes képek esetén is $2 * 256^{64*64*3}$ pontra (ez kb egy 29593 jegyű szám) kellene kiszámolni egy szállítási feladatot. Még akkor is, ha a feladatot minimum költségű maximum folyam feladatként írjuk fel, aminek egy $log(n)$ approximációját közel lineáris időben kiszámolhatjuk [46]. Még ennek a közelítésnek a kiszámolása is egy szuperszámítógéppel 200 petaflop számítási kapacitással ($= 200 * 10^{15}$ művelet másodpercenként), 10^{29571} évbe telne.

Ez csak a számítási kapacitással való problémák kezdete. A nagyobb probléma inkább az, hogy sem p_r, sem p_g eloszlást nem tudjuk explicit kiszámolni. Ugyanis:

- p_r esetében az ebből mintavételezett pár pont áll rendelkezésre. Ha ezt tudnánk, akkor megoldottuk volna a generatív modellezési feladatot.
- p_g esetén annyi a könnyebbség, ha nagyon akarjuk, kiszámolhatjuk: ugyanis ha a látens tér minden pontjából indítva a generátor modellt az összes lehetséges generátumot megkapjuk. Az eloszlást egy adott pontban pedig nyílván az előfordulás számossága határozza meg.

Ezen okok miatt egészen reménytelennek tűnik kiszámítani a Wasserstein távolságot direktben két (általános) eloszlás között.

Ezért az egyik megközelítés egy approximációra, ha p_r illetve p_g eloszlást a mintavételezésük által meghatározott eloszlással közelítjük. Azaz a mintákon az eloszlás $\frac{1}{\text{mintaméret}}$ értéket vesz fel, míg a mintában nem szereplő elemeken 0.

5.2. Wasserstein távolság közelítése teljes párosítással

Emlékeztetőül a Wasserstein távolságot $P = \{x_1, ..., x_n\}$ és $Q = \{y_1, ..., y_n\}$ mintahalmazok között a 2.3 fejezetben így írtuk fel:

$$\begin{aligned} \min_{\gamma} \quad & d\gamma \\ \text{s.t.} \quad & \\ \sum_i \gamma(i,j) = \quad & y_j \qquad \forall j \in N \\ \sum_j \gamma(i,j) = \quad & x_i \qquad \forall i \in N \end{aligned} \tag{29}$$

Mivel a GAN tanítása során sem a tanító adat eloszlását, sem a generátor eloszlását nem tudjuk, ezért egy nagyon egyszerű közelítéshez folyamodunk: Egy adott n méretű mintán egy minta valószínűségét $\frac{1}{n}$-nek vesszük, mindenhol máshol pedig nullának. Így tehát a Wasserstein távolság:

$$\begin{aligned} \min_{\gamma} \quad & d\gamma \\ \text{s.t.} \quad & \\ \sum_i \gamma(i,j) = \quad & \tfrac{1}{n} \qquad \forall j \in N \\ \sum_j \gamma(i,j) = \quad & \tfrac{1}{n} \qquad \forall i \in N \end{aligned} \tag{30}$$

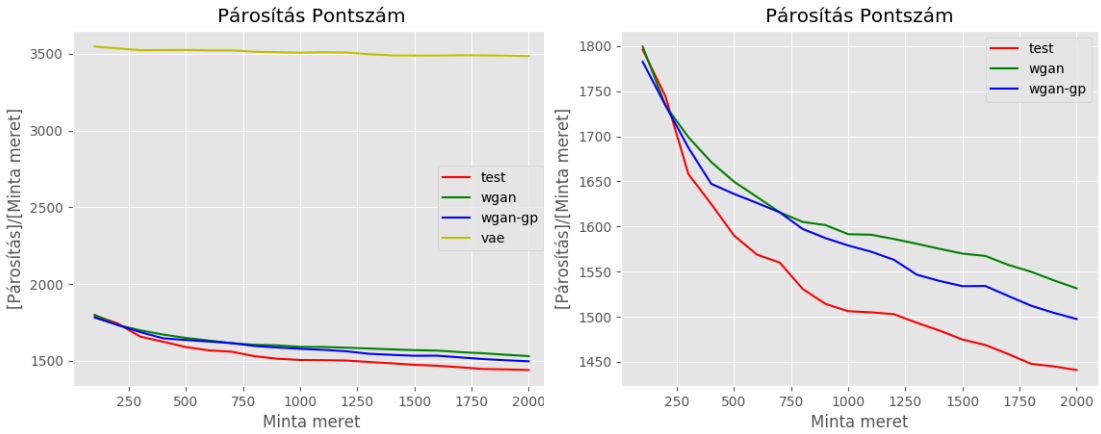
Mivel a folyamkorlátok egészértékűek (ez általános esetben nem teljesül, csak ebben a diszkretizált formában). De ez nem más, mint a minimális költségű teljes párosítás feladat, ahol γ {0,1} értékű, és pont azokon az éleken 1, amelyek a párosítást alkotják.

Tehát elegendő egy minimális költségű teljes párosítást keresnünk a két halmaz között. Ez a párosítás abból a szempontból is előnyös, hogy az ember számára is vizualizálható, mennyire jó a párosítás, azaz mennyire van közel a két eloszlás egymáshoz:

- a legjobban illeszkedő n pontpár vizualizációja
- véletlen n pont vizualizációja

Itt a már említett "emberi értékeléssel" vethetjük össze a fent definiált értékelő függvényt.

Így jó eszközt nyújt arra, hogy felismerjük a generatív modellünknek milyen gyenge pontjai vannak. Például előfordulhat, hogy egyes pontpárok túlságosan is jól illeszkednek, azaz a modell "túltanult". Vagy esetleg csak bizonyos típusú képek illeszkednek jól, és ezek viszik le a párosítás értékét.



(a) (b)

5.7. ábra. MNIST-en a minimális költségű teljes párosítás által meghatározott pontszám

Azt is érdemes megnézni, a mintavételezés méretével hogyan változik az átlagos párosítás költsége. Kezdetben nyílván egészen magas lesz, hiszen a tanító adathalmaz változatossága nagy, így kis mintában kevés hasonló elem lesz. Azonban ahogy megyünk a mintavételezés méretével a végtelenbe, annál valószínűbb lesz, (egy jó generátor esetén) hogy egy generált elem közel lesz egy tanító-adatbeli ponthoz. Így tehát az átlagos párosítás értékétől azt várjuk, hogy közelítsen a nullához, ahogy a minta méretével közelítünk a végtelenbe.

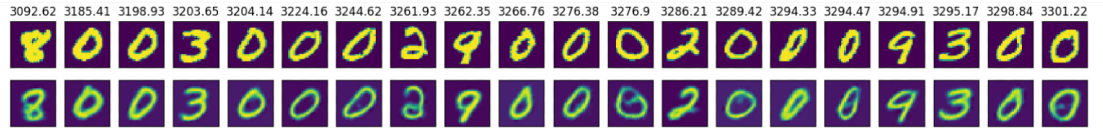
A módszer egyik hátulütője, ahogyan sok kiértékelő metrikánál az irodalomban, hogy hogyan mérjük két kép közötti távolságot, ugyanis a Wasserstein távolság két eloszlás között különböző metrikákkal különböző értékeket ad. Ennek a metrikának a megválasztása kulcsfontosságú, ahogyan ez a következő fejezetben kiderül.

2. Jelölés (Párosítás Pontszám). *Adott G generatív modell által generált S minta, és legyen T a tanító minta egy részhalmaza ($|S| = |T|$). M legyen az S és T halmaz által meghatározott páros gráf m metrika szerinti (mint költség) minimális költségű teljes párosítás értéke.Ekkor G Párosítás Pontszáma M és $|S|$ hányadosa.*

5.3. Eredmények MNIST adathalmazon

Az MNIST adathalmaz 28x28-as méretű, fekete-fehér képeket tartalmaz, melyeken kézzel írt számjegyek vannak 0-tól 9-ig. Mivel nagyon népszerű adathalmaz, és szinte minden ML könyvtár része, kézenfekvő volt ezzel az adathalmazzal kezdeni.

Ezen az adathalmazon 3 általam implementált, és fent bemutatott generatív



5.8. ábra. VAE Párosítás Pontszámának legolcsóbb 20 élének megjelenítése

modellt hasonlítottam össze: WGAN, WGAN-GP, VAE.

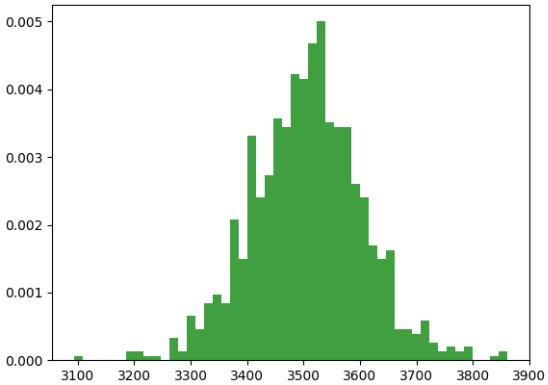
A WGAN és WGAN-GP implementálásánál a generátorra, és a diszkriminátorra a 5.12 ábrán található modelleket alkalmaztam. A generátor lényegében csak 3 konvolúciós blokkból áll, kiegészítve egy batchnormalizáló réteggel, a stabilabb tanulás érdekében. Az első két konvolúciós blokk után pedig megduplázom az első két csatorna dimenzióját, így nagyítva fel a kezdetben 7x7-es méretet 28x28 méretre. Csupán egy *Dense* réteg kerül a bemenet, és a blokkok közé, ami a bemenet átméretezése miatt szükséges (lásd 5.13b ábra).

A diszkriminátor négy konvolúciós blokkból áll, azonban itt nem szükséges a batchnormalizáció: a WGAN esetén paraméterek korlátozása, a WGAN-GP esetén pedig a gradiens korlátozása végzi el helyette a normalizációt. A generátorral ellentétben a konvolúciós blokkok után itt kicsinyítjük az első két dimenziót, mivel itt a 28x28-as bemenetet az 1x1-re kell csökkentenünk (lásd 5.13a ábra).

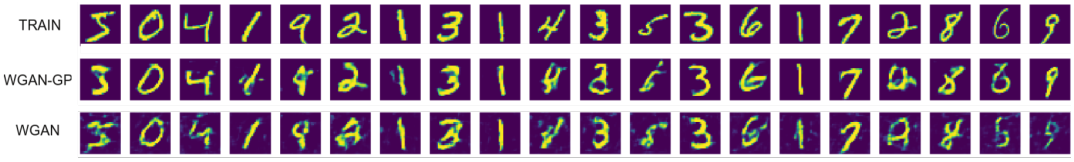
Amint a 3.1.2 fejezetben említettem, a három modell közül a VAE számít a leggyengébb generatív modellnek, utána a WGAN, és továbbfejlesztése a WGAN-GP. Ezt a sorrendet szemrevételezve (lásd 5.10 ábra) az én implementációmban is sikerült megőrizni, annak ellenére, hogy nagyon nehéz azonos feltételeket biztosítani a különböző modelleknek, mert azok különböző optimalizátort, aktivációs függvényeket, normalizációt, stb. preferálnak. A **Párosítás Pontszám** is megerősíti ezt az intuíciót, ahogyan ez a 5.7a ábrán is látszik. Első ránézésre az látszik, hogy a VAE nagyon lemarad a másik két modelltől. Vajon miért? Ennek megértésében segít a 5.8 ábra. A párosítás legjobb 20 élét megvizsgálva az látszik, hogy a VAE valójában szép számjegyeket rajzol, azonban nagyon homályosakat, nincsenek is teljesen fekete pontok. Ez az, ami azt eredményezi, hogy a pontszám többszöröse a WGAN, és WGAN-GP pontszámainak. A homályosság miatt így nincs is kimagaslóan jó vagy rossz kép, ahogyan ez a párosítás éleinek költségéből készült hisztogramon is látszik (5.9 ábra).

Ezek után a VAE-t kiveszem a további vizsgálódásból, ugyanis túl gyenge modellnek mutatkozik a másik kettőhöz viszonyítva.

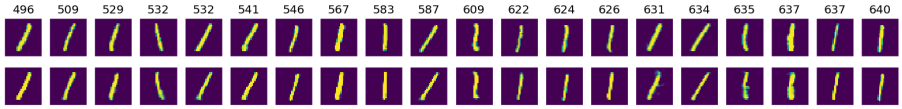
Térjünk tehát vissza a 5.7b ábrához, immár a VAE nélkül. Az összehasonlításhoz hozzávettem plusz egy "modellt": a teszt adathalmazt, ami egy ideális



5.9. ábra. Az MINST adathalmazon tanított VAE Párosítás Pontszám élei histogramon ábrázolva



5.10. ábra. Tanító adathalmaz - WGAN - WGAN-GP párosítás random élei ábrázolva



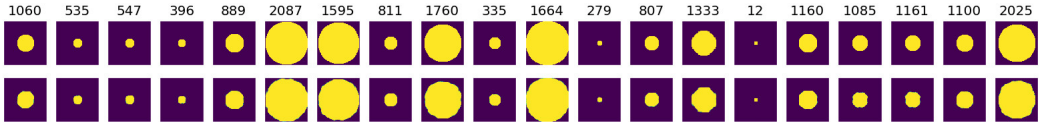
5.11. ábra. Tanító adathalmaz - WGAN-GP párosítás súly szerinti legjobb élei ábrázolva

generátor generátumainak felel meg. Erre az adathalmazra is ugyanúgy kiszámolhatjuk a pontszámot. Ezzel egy elméleti korlátot kapunk az ideális generátor pontszámára. Ha valamelyik modell generátorának pontszáma ez alá menne, akkor az már túltanulásra (overfitting) utalhat.

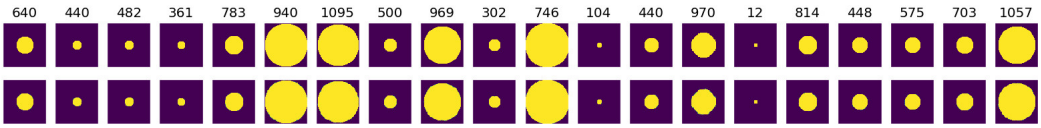
Végül pedig érdemes szemmel is összehasonlítani a 5.10 ábrán, a definiált pontszám egybevág-e az emberi intuícióval.

A Párosítós Pontszám számítása során az l_2 távolságot használtunk, aminek az egyik hátrányát a 5.11 ábrán láthatjuk; hogy a párosításban a legolcsóbb élek csupa 1-esek. Tovább vizsgálva a párosításban szereplő éleket; azokat sorba rendezve, és abból egyenletesen mintavételezve jutunk a 5.13 ábrához. Itt az látszik, hogy az egyesek, és hetesek szerepelnek a legalacsonyabb értékkel a párosításban. Ez azzal magyarázható, hogy ennél a két számjegynél a legkisebb a fekete-fehér határfelületek aránya, így egy másik számjegynél azonos arányú torzítás esetén nagyobb lesz a hiba.

5.4. Eredmények Syn-circles adathalmaz



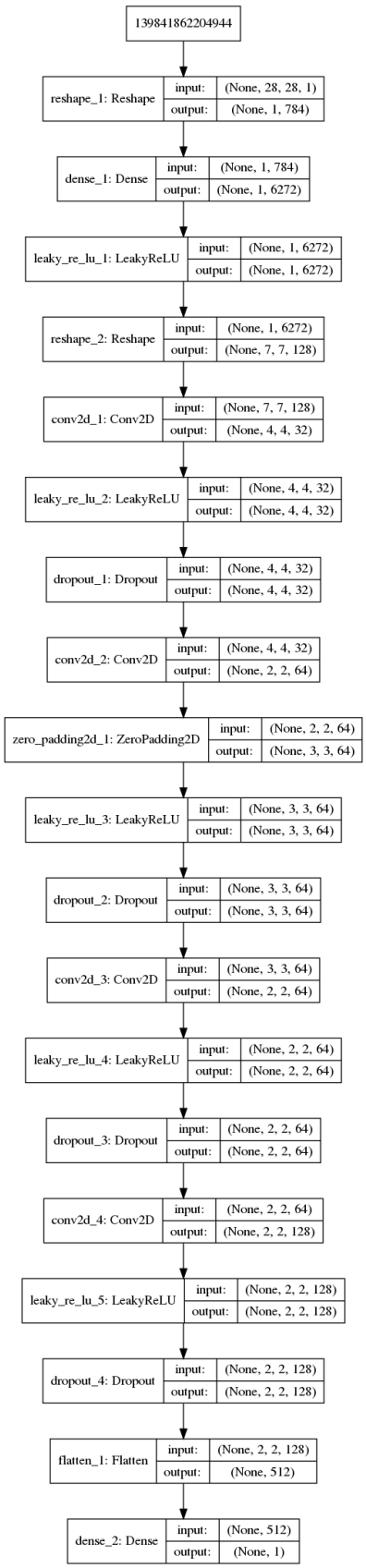
5.14. ábra. WGAN Párosítás véletlenszerű élei



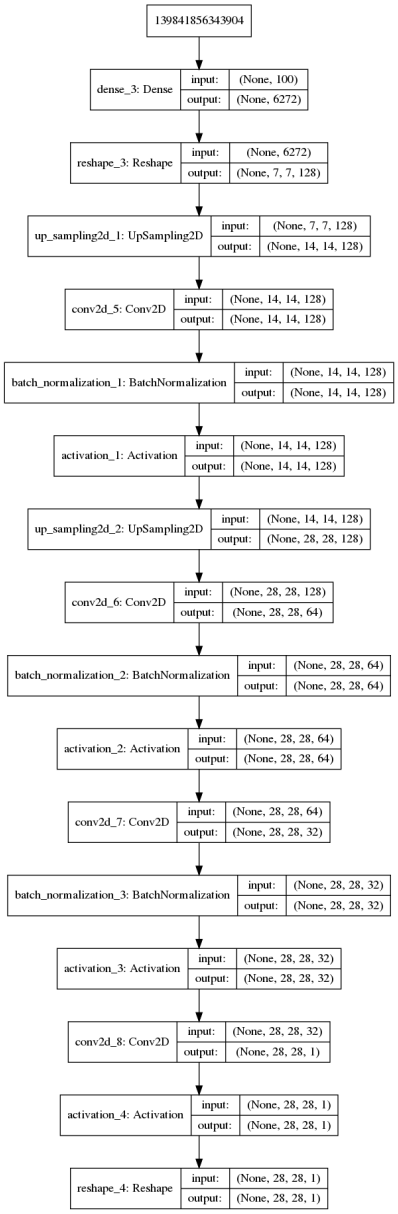
5.15. ábra. WGAN-GP Párosítás véletlenszerű élei

Az MNIST esetében sajnos az l_2 távolság sokszor két hasonlóan kinéző számjegyre is adhat nagy távolságot, és így a metrika nem annyira megbízható. Érdemes kipróbálni tehát egy olyan adathalmazon, ahol az l_2 távolság az ember által vélt hasonlósággal azonos.

A "syn-circles" pont egy ilyen adathalmaz, amely különböző átmérőjű, koncentrikus körlapokat tartalmaz, amelyek fekete-fehér, 64x64 pixeles képen vannak ábrázolva. Bár valójában ez egy egydimenziós adathalmaz (a kör átmérője az egyetlen változó), mégse triviális ezt egy generatív modellnek megtanulnia. A generatív modellnek 64x64-es fekete-fehér képekből kellene leképeznie egy egydimenziós térbe, és ennek az eloszlását kellene megjegyeznie, majd ezt visszaképeznie a 64x64 dimenziós térbe.

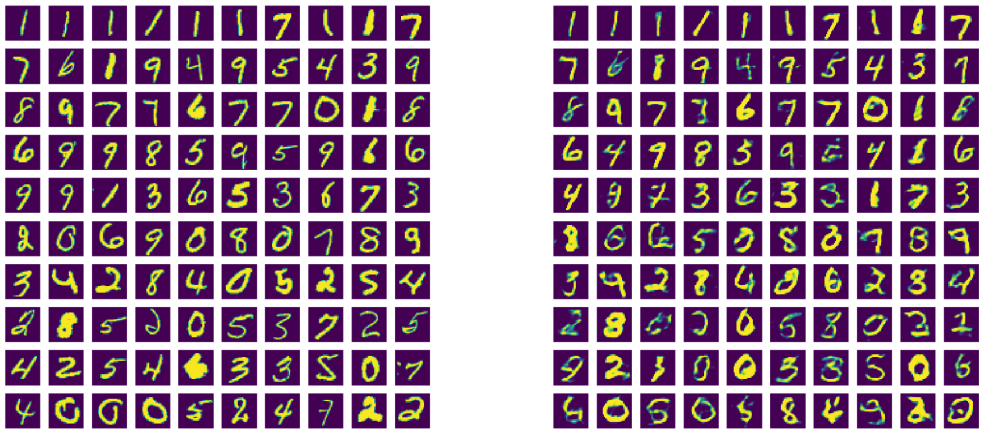


(a) A diszkriminátor neurális hálója



(b) A generátor neurális hálója

5.12. ábra. A WGAN és a WGAN-GP által használt modell



(a) Tanító adat

(b) A hozzá párosított WGAN-GP

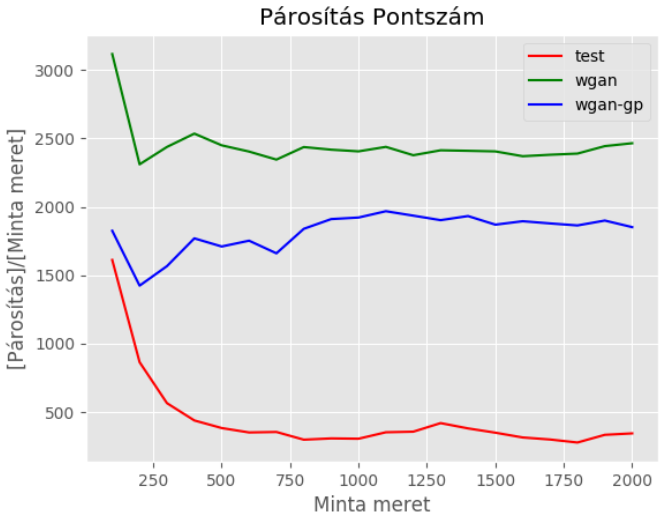
5.13. ábra. A párosítás sorbarendezett éleiből mintavételezve (minden huszadikat kiemelve) sorfolytonosan: legjobban illeszkedőek a bal felső sarokban, a legrosszabbul pedig a jobb alsóban vannak

Az 5.16 ábrán az látszik, hogy habár a WGAN és WGAN-GP közötti különbség számottevő, mégsem tudják megközelíteni a teszt adathalmaz értékét. Érdemes ezért megnézni, hogyan is néznek ki a párosított pontok, és a párosítás súlyai hogyan oszlanak meg.

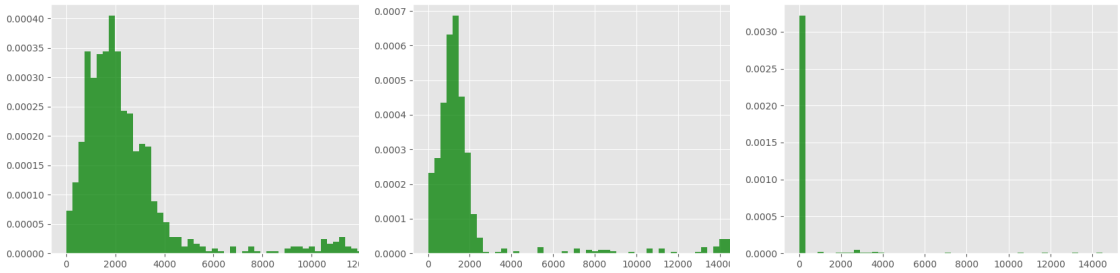
A 5.14 és 5.15 ábrán az látszik, hogy bár távolról nézve jó generátumokat ad mind a WGAN mind a WGAN-GP, de nem teljesen tökéletes köröket adnak vissza, ezért az átlagos hiba beragad 1800-2500 körülire. Ha alaposan megnézzük 5.14 ábrát, a WGAN körei tényleg nagyon szabálytalanok, így jogos a 5.16 ábrán látott különbség a két modell között. Ez nagyjából azt jelenti, hogy a pixelek $1.2 - 2.3\%$-át hibázzák el a generátorok átlagosan.

A TESZT modell elérése azonban a syn-circles adathalmazon elérhetetlennek tűnik (lásd 5.17 ábra), míg az MNIST esetén egészen elérhető volt. Elsőre ez meglepőnek tűnik, hogy a modellek rosszabbul teljesítenek egy könnyebb adathalmazon. Valójában azonban a syn-circles generálás szempontjából nehezebb, hiszen 32 osztályból áll, de az osztályon belül minden kép egyforma, ezért egy generátornak tökéletesen meg kell tanulnia (memorizálnia) az egyes osztályokat.

Ez tehát a *Párosítás Pontszám* újabb hátránya, hogy favorizálja a memory-GAN modelleket.



5.16. ábra. Syn-circles adathalmazra átlagos teljes párosítás értéke a batchméret szerint különböző modellekre



(a) WGAN (b) WGAN-GP (c) TEST

5.17. ábra. Syn-circles train és generátumok közötti legolcsóbb teljes párosítás éleinek súlya hisztogramon ábrázolva.

5.5. Eredmények Celeba adathalmaz

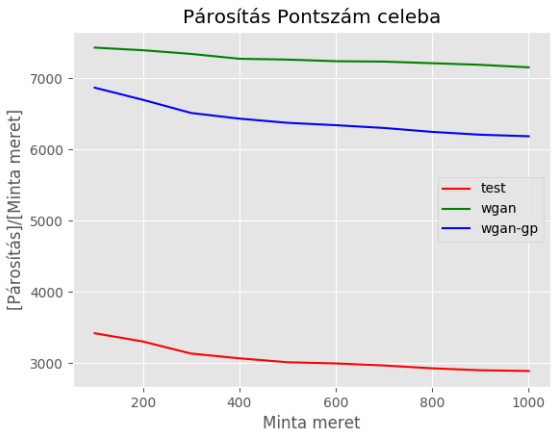
A Celeba adathalmaz híres emberekről készített színes képeket tartalmaz. A GAN-ok tanítására használt egyik kedvelt adathalmaz volt az elmúlt években, mert viszonylag nehéz volt élethű arcokat generálni, és ha mégis, az rendkívül felkapott lett a ML világon kívül is.

Az új eredményeket az ember már nem is tudja szemmel megkülönböztetni, hogy melyik modell jobb. Ezért lenne nagy szükség olyan kiértékelő metrikákra, amely olyan apró dolgokat is észrevesznek, amit az ember nem, lásd a fenti syn-circles adathalmaznál.

Sajnos az átlagos párosítás ezen az adathalmazon azért nem képes ilyen eredményekre, mert a a Celeba-n nem ismert olyan metrika, amely két kép közötti

(az ember által vélt) hasonlóságot jól mérné.

Így marad az l_2, amivel olyan modellek között tudunk különbséget tenni, amelyek a képélességben, vagy a színhasználatban egy bizonyos szint alatt vannak. A 5.18 modell mutatja, hogy ez a szükséges rossz metrika is meg tud különböztetni két közepes modellt.



5.18. ábra. Átlagos párosítás értéke batchméret szerint; WGAN és WGAN-GP modellek Celeba adathalmazon.

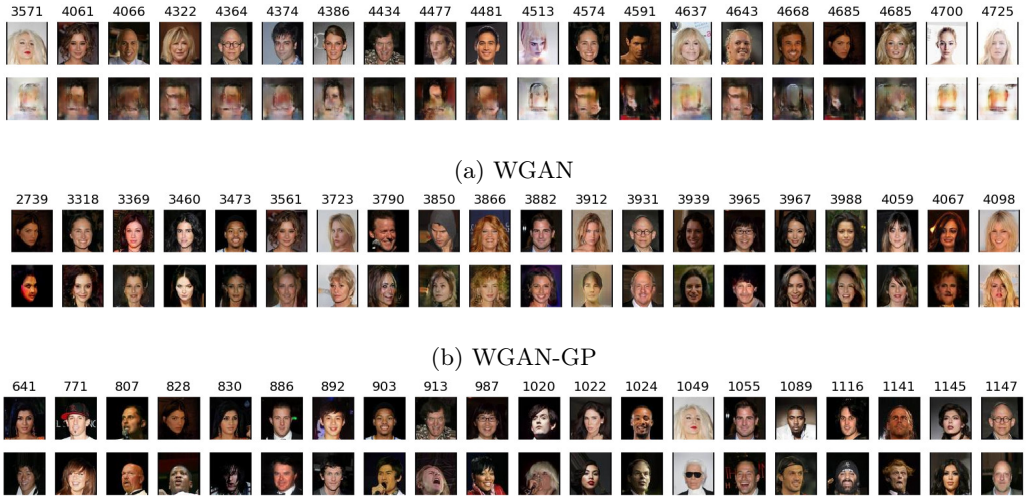
Azonban ahogyan a 5.19 ábrán látjuk, a párosítás nem talál hasonló kinézetű embereket párosít össze. A párosítás sokkal inkább a háttér, arcszín, hajszín alapján történik, a kisebb tulajdonságok egyezését teljesen ignorálja (mint például a szemszín).

5.6. Wasserstein távolság közelítése maximális párosítással, ritkított gráfon

Az előző metrikával lehetnek olyan ellenvetéseink, hogy előfordulhat, hogy pár generált pontnak nem lesz a train adathalmazban közeli párja, mert túlságosan kicsi mintaméretet vettünk. Így ezeket a pontokat bárhogyan is párosítjuk, a pontszámot hamis irányba fogják vinni.

Valamint jó lenne, ha a metrikánk lebuktátná az olyan modelleket, amelyek túl sok hasonló képet generálnak (lásd korábban: mode-collapse). Vagy az olyan modelleket próbáljuk meg lebuktatni, amelyek bizonyos vonásait az adathalmaznak nem tanulják meg (például Celeba adathalmaz esetén nem generál szemüveges embereket, MINST esetén nem generál valamely számjegyet, stb).

Ezekre a felvetésekre azt próbáltam ki, mi történik, ha a G=(Tanító minta, Generált minta) páros gráfban (l_2 távolságot véve a képek között) a generált minta pontjainak csak a legjobb k szomszédját tartjuk meg. Így ebben a ritkított



(a) WGAN

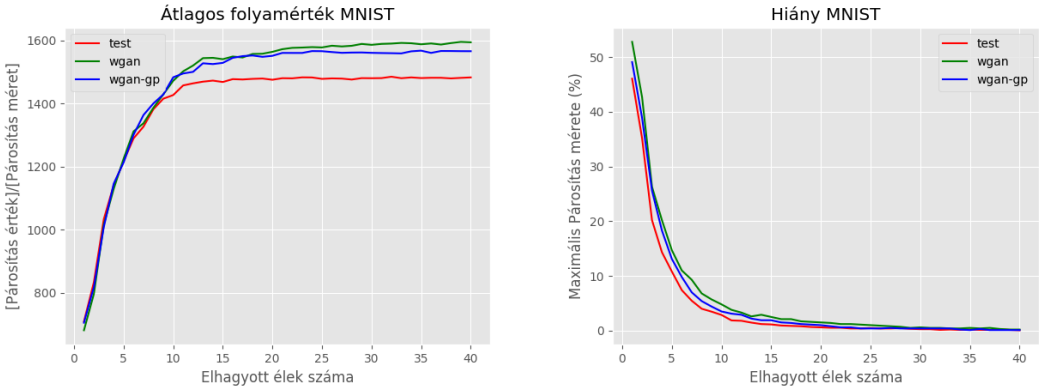
(b) WGAN-GP

(c) TESZT

5.19. ábra. Párosított képek Celeba adathalmazon

G' gráfban keresnénk egy maximális méretű minimális költségű párosítást.

Ennek kiszámítása többféle módon is történhet. Én egy minimális költségű maximális folyamfeladattal oldottam meg: a szokásos módon a $G = (S, T)$ gráfhoz hozzávettem egy s, t forrás-, illetve nyelő-pontot. (s, S), (t, T), valamint az (S, T) élekre a kapacitást 1-nek, valamint a kitörölt éleken 0-nak vesszük. A költségek az (S, T) éleken pedig maradnak az adott képek közötti l_2 távolság, mindenhol máshol pedig 0. Ezen G' gráfon kell minimális költségű maximális folyamot keresni, és mivel a kapacitások egészek voltak, a folyam is választható egésznek, ami így az eredeti gráfban egy maximális méretű minimális költségű párosítást ad.



(a) Az x tengelyen a meghagyott élek száma szerint az y tengelyen az átlagos párosítás értéke

(b) Az x tengelyen a meghagyott élek száma szerint az y tengelyen a hiány mérete

5.20. ábra. Maximális párosítás élelhagyással az MNIST adathalmazon, 1000-1000 mintával

A 5.20 ábrán az látszik, hogy alacsony k esetén a maximális párosítás sok

pontot kihagy, azaz a ritka gráfnak a "hiánya" nagyon nagy, de a párosított pontok átlagos költsége nagyon alacsony.

Ahogyan növeljük k-t a hiány egészen gyorsan csökken, elegendő csak $10-15$ élet meghagyni ahhoz, hogy a maximális párosításból kimaradó pontok száma kevesebb legyen mint 5 százalék. Valamint az is látszik, hogy ennél több élet meghagyva már állandósul a sorrend, és az átlagos párosítás is, és gyorsan konvergál az élelhagyás nélküli pontszámhoz, mint amit fent láttunk.

6. Implementáció

Keras, Tensorflow

A Keras egy open-source Python könyvtár, ami képes Tensorflow, Theano, és Microsoft Cognitive Toolkit-en is futni. Egy felhasználóbarát interfészt ad az alatta levő ML könyvtárakhoz, így átláthatóbbá, és könnyebben implementálhatóbbá teszi bizonyos modellek felépítését.

Előnye, hogy csekély kódmódosítással átültethető ugyanaz a kód az egyik kernelről a másikra.

Hátránya, hogy ugyan felhasználóbarát, és így a nagyobb "építőkockákkal" sokszor könnyebben lehet haladni, de egy kutatás során típikusan a keras által használt egyszerűsítések gyakran nagyobb bonyodalmakat okoznak, mint amennyi előny származik a használatával. Egy ilyen probléma volt pl.: a WGAN-GP implementálásánál a gradiens büntetése beépítése a loss függvénybe. Ezért egy kutatás jellegű, vagy újfajta modell felépítését érdemes alacsonyabb szinten implementálni.

A Tensorflow ebből a szempontból egy nagyon felhasználóbarát kernel. A Tensorflow alapja egy számítási gráf, ahol minden egyes csúcs egy műveletet jelöl. Több előnye is van, ha egy ilyen számítási gráffal számoljuk ki az adott ML feladatunkat: ha a számítási gráfunkat olyan műveletekből raktuk össze, amelyeknek mind megadtuk a deriváltját is, akkor a back-propagation segítségével a Tensorflow minden műveleti csúcsban a számítási gráf bármely bemenete szerint megadja a gradienst. Ez a gradiens számítás szimbolikusan történik, egy másik számítási gráfban, így lehetséges magasabb rendű deriváltak kiszámítása is. A Tensorflow természetesen tartalmazza a modern deep-learning eszközöket is (optimalizátorok, aktivációs függvények, neurális rétegek, stb.), de új eszközök fejlesztésére is nagyon hasznos, hiszen lenyúlhatunk egészen a számítási gráfig (a Keras-sal ellentétben).

Implementálási nehézségek

Az WGAN, WGAN-GP modelleket a [47] repository-ban implementáltam. Ez a repository az őszi Deep learning kutatóműhely eredménye, ami a különböző miniprojektek együttese, ahol egy közös platformon implementáltuk a generatív modelleket, hogy azok könnyen összehasonlíthatóak legyenek. Minden modellt egy inicializációs fájllal kell elindítani, amiben a mérésekben állítgatott paraméterek (mint a tanulási ráta, bemenő adathalmaz, optimalizátor stb.) szerepelnek, így ugyan azon modell többféle konfigurációval elindítható, és összehasonlítható.

A kiértékelési metrikákat pedig a [48] repository-ban implementáltam. A program nagy része *C++* nyelven van írva, a gyors futás érdekében. Azonban a gördülékeny működéshez egy *python* interfészt kellett írjak, hiszen a kiértékelő modelleket általában egy *.h*5 fájl kódolta, vagy a generátumokat *.npy* formában menti ki a generatív modell. Ezen modellek beolvasása *C++* nyelven nehézkes lett volna (python-ban a megfelelő könyvtárak segítségével ez egyetlen parancs), így ez elkerülhetetlen volt. Továbbá a *python* nagy segítséget nyújt a megjelenítéshez, így kézenfekvő volt, hogy minden kiértékelő futáseredményt rögtön fel is dolgozzam, és megfelelő könyvtárstruktúrába rendezzem. Így a megjelenítés teljesen automatizálva volt a fejlesztés során.

7. Összefoglaló

A szakdolgozatomban bemutattam pár generatív modellt, mint a Variational Autoenkódert (VAE), és általánosan a Generatív ellenséges hálók (GAN) működését. Az Generatív Ellenséges Hálók tanítása általánosan nem bizonyul túl stabilnak. Ennek a javítására számos GAN modell létezik, ebből a Wasserstein GAN volt az, ami elméletben és gyakorlatban is ígéretes eredményeket ígért, ezért választottam ennek részletes bemutatását, és implementálását is.

A szakdolgozat második felében pedig a pedig bemutattam a generatív modellek kiértékelésének nehézségeit, az irodalomban használt legtöbbet használt, és legjobbnak vélt kiértékelő metrikákat, és pár olyat is, ami bár nem túl gyakran alkalmazott, de az ötlet mögötte figyelemre méltó.

Majd pedig bemutattam két saját kiértékelési metrikát, három különböző adathalmazon: a "syn-circles", "MNIST" és a "Celeba"-n. Ezeken az alkalmazhatóság csak az l_2 távolságon múlott, hogy az mennyire méri le jól az adathalmazban levő képek hasonlóságát.

Hivatkozások

[1] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, „Decision tree regression for soft classification of remote sensing data," *Remote Sensing of Environment*, vol. 97, no. 3, pp. 322 – 336, 2005.

[2] A. Gidudu, G. Hulley, and T. Marwala, „Classification of Images Using Support Vector Machines," *CoRR*, vol. abs/0709.3967, 2007.

[3] K. He, X. Zhang, S. Ren, and J. Sun, „Deep Residual Learning for Image Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, „TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. Software available from tensorflow.org.

[5] Theano Development Team, „Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[6] J. Ker, L. Wang, J. Rao, and T. Lim, „Deep Learning Applications in Medical Image Analysis," *IEEE Access*, vol. 6, pp. 9375–9389, 2018.

[7] T. Afouras, J. S. Chung, A. W. Senior, O. Vinyals, and A. Zisserman, „Deep Audio-Visual Speech Recognition," *CoRR*, vol. abs/1809.02108, 2018.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, „A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[9] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, „Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[10] J. Redmon and A. Farhadi, „YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[11] B. Dolhansky and C. Canton-Ferrer, „Eye In-Painting with Exemplar Generative Adversarial Networks," *CoRR*, vol. abs/1712.03999, 2017.

[12] G. Cohn, „AI Art at Christie's Sells for $432,500," *The New York Times*, 2018.

[13] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, „LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop," *arXiv preprint arXiv:1506.03365*, 2015.

[14] A. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. V. Arthur, P. A. Merolla, P. Datta, M. Gonzalez Tallada, B. Taba, A. Andreopoulos, A. Amir, S. Esser, J. Kusnitz, R. Appuswamy, C. Haymes, B. Brezzo, R. Moussalli, R. Bellofatto, C. Baks, and D. S. Modha, „Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with 100x Speedup in Time-to-Solution and 100,000x Reduction in Energy-to-Solution," vol. 2015, 11 2014.

[15] H. Cheng, W. Wen, C. Wu, S. Li, H. H. Li, and Y. Chen, „Understanding the design of IBM neurosynaptic system and its tradeoffs: A user perspective," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 139–144, March 2017.

[16] K. T. Frank András, „Operációkutatás," pp. 42–52, Typotex, 2013.

[17] H. W. Kuhn, „The Hungarian method for the assignment problem," in *Naval Research Logistics Quarterly, 2(1-2)*, pp. 83–97, 1955.

[18] E. Jenő, „Mátrixok kombinatorikus tulajdonságairól," pp. 16–28, Mat. Fiz. Lapok, 1931.

[19] K. Dénes, „Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre.," pp. 453–465, Math. Ann. 77, 1916.

[20] A. Jüttner, „On the efficiency of Egerváry's perfect matching algorithm," *Matematikai Lapok. (New Series)*, vol. 1, 01 2004.

[21] „Network Flow Algorithms," 1989 Apr.

[22] J. Edmonds and R. M. Karp, „Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM*, vol. 19, pp. 248–264, Apr. 1972.

[23] L. R. Ford and D. R. Fulkerson, „Maximal Flow Through a Network," *Canadian Journal of Mathematics*, vol. 8, p. 399–404, 1956.

[24] „Richard Feynmann idézet," `https://archives.caltech.edu/pictures/1.10−29.jpg`.

[25] M. W. Diederik P Kingma, „Auto-Encoding Variational Bayes," 1 May 2014.

[26] D. W. Danilo Jimenez Rezende, Shakir Mohamed, „Stochastic Backpropagation and Approximate Inference in Deep Generative Models," 30 May 2014.

[27] D. J. R. Shakir Mohamed, „Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning," 29 Sep 2015.

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, „Extracting and Composing Robust Features with Denoising Autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), pp. 1096–1103, ACM, 2008.

[29] C. Doersch, „Tutorial on Variational Autoencoders," *arXiv e-prints*, p. arXiv:1606.05908, Jun 2016.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, „Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[31] G. Cohn, „Generative Adversarial Nets," in *AI Art at Christie's Sells for $432,500*, The New York Times, 2018-10-25.

[32] L. B. Martin Arjovsky, Soumith Chintala, „Wasserstein GAN," 2017.

[33] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, „Improved Training of Wasserstein GANs," *CoRR*, vol. abs/1704.00028, 2017.

[34] I. Goodfellow, „4.5 years of GAN progress on face generation.," 1956. `https://twitter.com/goodfellow_ian/status/1084973596236144640`.

[35] L. Theis, A. van den Oord, and M. Bethge, „A note on the evaluation of generative models," *arXiv e-prints*, p. arXiv:1511.01844, Nov 2015.

[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, „Improved Techniques for Training GANs," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V.

Luxburg, I. Guyon, and R. Garnett, eds.), pp. 2234–2242, Curran Associates, Inc., 2016.

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, „Rethinking the Inception Architecture for Computer Vision," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[38] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, „Imagenet: A large-scale hierarchical image database," in *In CVPR*, 2009.

[39] Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, and Y. Yu, „Activation Maximization Generative Adversarial Nets," in *International Conference on Learning Representations*, 2018.

[40] S. Gurumurthy, S. R. K. Sarvadevabhatla, and R. Babu, „DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data," 06 2017.

[41] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, „Mode Regularized Generative Adversarial Networks," *CoRR*, vol. abs/1612.02136, 2016.

[42] „Inception Score, Label Smoothing, Gradient Vanishing and -log(D(x)) Alternative," *CoRR*, vol. abs/1708.01729, 2017. Withdrawn.

[43] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, „GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium," *CoRR*, vol. abs/1706.08500, 2017.

[44] C. Olsson, S. Bhupatiraju, T. Brown, A. Odena, and I. Goodfellow, „Skill Rating for Generative Models," 2018.

[45] D. Aldous, „Elo Ratings and the Sports Model: A Neglected Topic in Applied Probability?," *Statist. Sci.*, vol. 32, pp. 616–629, 11 2017.

[46] A. Mądry, „Fast Approximation Algorithms for Cut-based Problems inUndirected Graphs," 6 Nov 2010.

[47] „Generativ Modelling Repository," `https://github.com/zsoltzombori/generative_modeling/tree/doma945`.

[48] „GAN evaluation metrics," `https://github.com/doma945/GAN_evaluations`.