SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1544

Polunadzirana klasifikacija rukom pisanih znakova generativnim suparničkim modelima

Marko Jelavić

Zagreb, srpanj 2017.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem se mentoru, izv. prof. dr. sc. Siniši Šegviću na podršci i vodstvu kroz diplomski studij te na savjetima i pomoći prilikom izrade diplomskog rada.

SADRŽAJ

1. Uvod

Klasifikacija rukom pisanih znakova se često koristi kao uvodni problem u područje računalnog vida. Najpoznatiji takav problem je klasifikacija rukom pisanih znamenki ([27]) gdje se svakoj slici mora pridijeliti pripadajuća znamenka kao klasa. Pošto se radi o maloj bazi podataka slika niskih rezolucija, rezultati se mogu brzo dobiti te se novi modeli najčešće provjeravaju nad njom. Modeli koji uspješno obavljaju klasifikaciju rukom pisanih znakova najčešće primjenu nalaze kao dio sustava za pretraživanje informacija iz dokumenata.

Duboki diskriminativni modeli se smatraju najsuvremenijom tehnologijom iz razloga što pružaju najbolje rezultate u širokom spektru primjena te su najzaslužniji za ekspanziju primjene umjetne inteligencije u industriji. Mana takvih modela je ta što zahtijevaju ogroman broj ručno označenih podataka da bi postigli zadovoljavajuće rezultate. Označavanje podataka je izrazito skup proces te ga se nastoji zaobići.

S druge strane, duboki generativni modeli ne zahtijevaju veliku količinu označenih podataka. No, problemi s dosadašnjim modelima se javljaju pri aproksimacijama vjerojatnosnih izračuna pri modeliranju vjerojatnosti ulaznih podataka što usporava proces treniranja modela ([22]). U navedenom radu predlaže se model koji se zasniva na suparničkim neuronskim mrežama.

Model generativnih suparničkih mreža se sastoji od dviju neuronskih mreža, generatorske i diskriminatorske. Spada pod familiju modela bez učitelja, odnosno nisu mu potrebni označeni primjeri za učenje, već samo uči distribuciju ulaznih podataka. Mreža generator ima zadatak naučiti distribuciju ulaznih podataka, a to čini provođenjem latentnih varijabli na ulazu kroz svoje slojeve te kao izlaz daje sliku poput primjera za učenje. Mreža diskriminator ima zadatak odrediti da li je primjer na njenom ulazu generiran ili stvaran. Kroz postupke optimizacije težina, istovremeno se poboljšavaju generator i diskriminator.

Nije dugo trebalo da se ovaj model, kao i većina generativnih, pokuša preformulirati u polunadzirani model, te da u konačnici služi kao klasifikator. Pokazalo se da za manji broj ulaznih podataka model funkcionira bolje od izoliranog klasifikatora ([32]).

Upravo to je i cilj ovog rada koji ima zadatak dati uvid u najpoznatije inačice generativnih suparničkih modela, te isprobati model u polunadziranom okruženju na novoj bazi primjera za učenje.

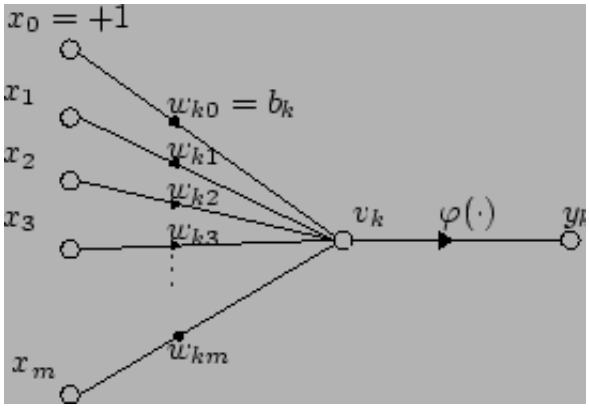
Poglavlje 2 daje uvod u neuronske mreže te obrazloženje odabranih pojmova u tom polju, generativne suparničke mreže s teoretske strane su obrađene u poglavlju 3, dok poglavlje 4 prezentira rezultate eksperimenata sa slikama i mjerenjima uz popratne komentare. Zaključak rada se nalazi u poglavlju 5, a literatura je prikazana na samom kraju prije sažetka.

2. Neuronske mreže

Umjetne neuronske mreže nastale su kao simulacija rada ljudskog mozga. Skupina umjetnih neurona koji su međusobno povezani zajedno čine neuronsku mrežu. Svaki od neurona procesira informaciju na ulazu te daje izlaz. Grupiranje više neurona omogućava modeliranje složenijih relacija koji postoje u podacima na ulazu. Arhitektura neuronske mreže određuje način na koji su umjetni neuroni u istoj spojeni. Također, svaki od neurona može na više načina procesirati ulaze. Neuronske mreže stoga predstavljaju izrazito konfigurabilan model strojnog učenja, koji nalazi široku primjenu u polju umjetne inteligencije.

2.1. Umjetni neuron

Osnovni model umjetnog neurona postavljen je 1940-ih [30]. Sastoji se od ulaza u neuron koji su težinski povezani s istim, zatim se otežani ulazi zbrajaju, te se primjenjuje aktivacijska funkcija neurona, kao što je prikazano na slici 2.1.



Slika 2.1: Umjetni neuron [12]

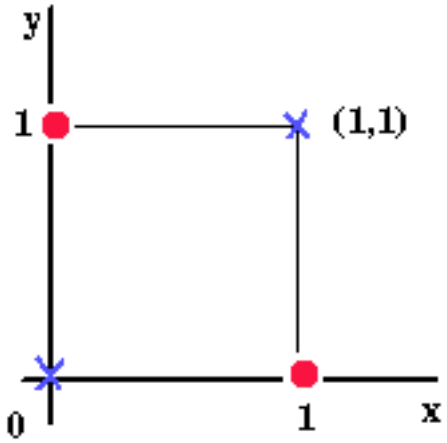
Pošto je svim neuronima zajedničko otežano zbrajanje ulaza, svojstva pojedinog neurona zapravo najviše ovise o njegovoj aktivacijskoj funkciji. O njima više u idućoj sekciji.

2.1.1. Aktivacijske funkcije

Aktivacijske funkcije su najznačajnija karakteristika svakog neurona, stoga ne čudi da napredak u razumijevanju pojedinih aktivacijskih funkcija, te predlaganje novih često dovodi do napretka u neuronskim mrežama.

Prag

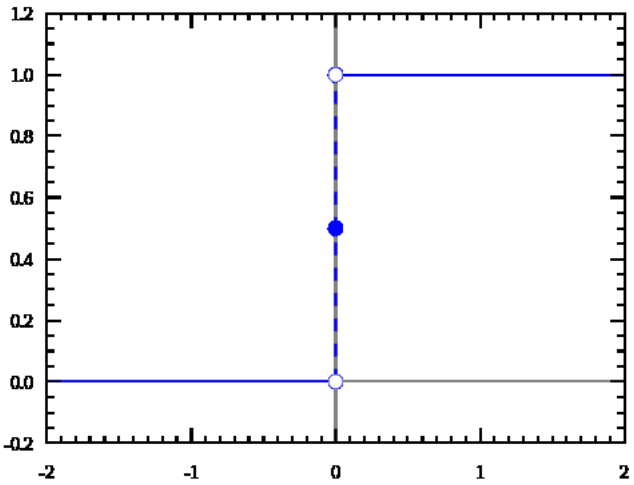
Funkcija praga predložena je u algoritmu Perceptrona [36]. Koristi se kao binarni klasifikator. Model Perceptrona je linearan, što znači da algoritam nije u mogućnosti ispravno klasificirati ni najjednostavnije linearno neseparabilne probleme poput prikazanog na slici 2.2. Ta tvrdnja je i dokazana [31].



Slika 2.2: XOR problem [19]

Nakon sumiranja otežanih ulaza, funkcija praga se primjenjuje po izrazu 2.1, a graf funkcije je vidljiv na slici 2.3.

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.1}$$



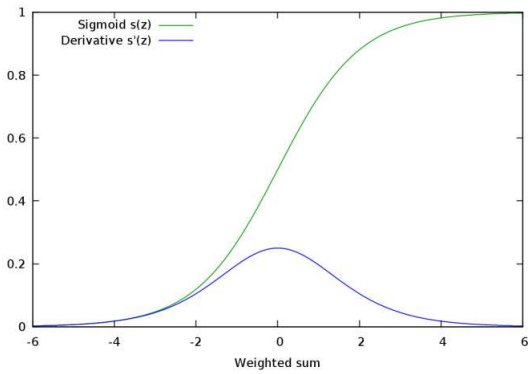
Slika 2.3: Funkcija praga [16]

Sigmoidalna funkcija

Sigmoidalna funkcija se prvotno koristila u logističkoj regresiji, te sama logistička regresija može bit interpretirana kao neuron kojem je aktivacijska funkcija sigmoidalna funkcija. Logistička regresija je također linearan klasifikator, no svoju značajnost za razliku od funkcije praga poprima u neuronskim mrežama.

Sigmoidalna funkcija je nelinearna aktivacijska funkcija. U neuronskim mrežama izlaz jednog neurona je ulaz drugome. Uvođenjem nelinearnosti preko aktivacijske funkcije omogućava se modeliranje kompleksnijih relacija od linearnih, te širenje primjene modela i na linearno neseparabilne probleme. Zadana je jednadžbom 2.2, a slika 2.4 prikazuje graf funkcije te graf njene derivacije.

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



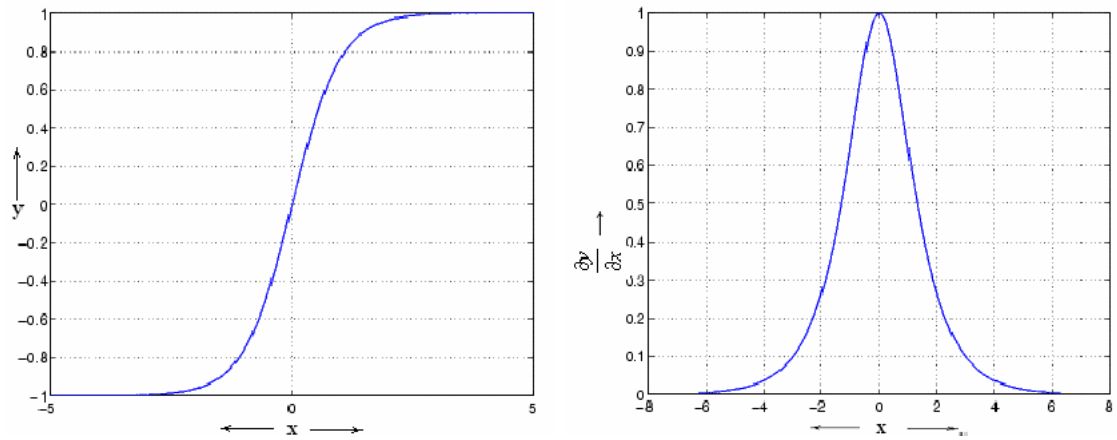
Slika 2.4: Sigmoidalna funkcija [15]

Iz grafa funkcije je vidljivo da se sigmoidalnom funkcijom dobije pouzdanost klasifikacije neurona za određeni primjer, što je korisna informacija. No, prva derivacija otkriva manu, a to je problem nestajućeg gradijenta. Ukoliko je izlaz neurona primjerice 0.99, odnosno klasa 1, a zapravo bi trebao biti klasa 0, gradijent je izrazito mali, što znači da će za korekciju klasifikacije trebati velik broj iteracija algoritma gradijentnog spusta.

Tangens hiperbolni

Tangens hiperbolni kao aktivacijska funkcija se počeo češće koristiti kao zamjena za sigmoidalnu funkciju iz razloga što ona daje isključivo pozitivne izlaze u intervalu $[0, 1]$ te je centrirana oko 0.5. To ne predstavlja problem samo jednom neuronu, dok u višeslojnoj neuronskoj mreži koči brzinu konvergencije u postupku učenja. Stoga se predlažu funkcije centrirane oko 0, a upravo jedna od takvih je tangens hiperbolni [28]. Funkcija je zadana jednadžbom 2.3, a graf funkcije i njene derivacije je vidljiv na slici 2.5.

$$f(x) = tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.3}$$



Slika 2.5: Tangens hiperbolni [17]

Predložena je i modifikacija aktivacijske funkcije u izraz [28]:

$$f(x) = 1.7159tanh\left(\frac{2}{3}x\right). \tag{2.4}$$

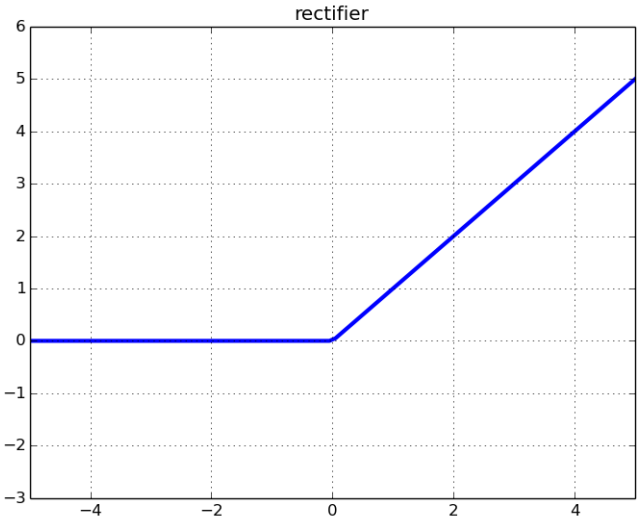
Modificirana funkcija za ulaze koji su centrirani na 0, sa varijancom 1, daje izlaze koji su također centrirani na 0 i varijanca bi im trebala biti blizu 1.

Zglobnica

U kontekstu neuronskih mreža, zglobnica je aktivacijska funkcija čija formula glasi:

$$f(x) = max(0, x) \tag{2.5}$$

Trenutno je najpopularnija aktivacijska funkcija u arhitekturama dubokih neuronskih mreža ([13]). Izračunski je vrlo jednostavna, te njen gradijent također (0 za $x < 0$, 1 za $x > 0$, nediferencijabilan u 0), dovodi i do rjeđih aktivacija neurona. Potencijalni problemi su: nije centrirana oko 0, nediferencijabilna u 0, neograničena aktivacija te se zna dogodit fenomen umirućih neurona, odnosno neuroni se dovedu u mrtvo stanje iz kojeg ne mogu više izaći, efektivno smanjujući kapacitet neuronske mreže [13]. Diferencijabilnost u 0 se rješava tako što se definira da gradijent u 0 bude ili 0 ili 1, neograničenu aktivaciju pod kontrolom drži grupna normalizacija koja najčešće prethodi aktivacijskoj funkciji, a fenomen umirućih neurona rješavamo parametriziranom zglobnicom (2.6) gdje se za odabir a često uzima vrijednost u intervalu $< 0, 1 >$. Slika 2.6 prikazuje zglobnicu.



Slika 2.6: Zglobnica [14]

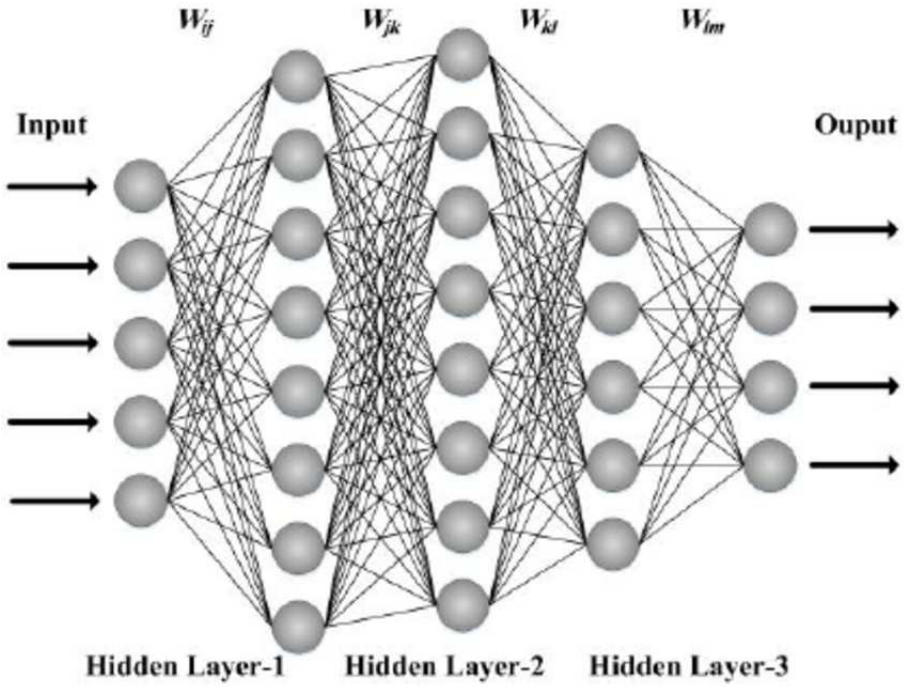
ReLU je lesto korištena skraćenica za zglobnicu. Poznate varijante su Parametric ReLU (2.6) i ELU (2.7) [13].

$$f(x) = \begin{cases} x, & x > 0 \\ ax, & otherwise \end{cases} \tag{2.6}$$

$$f(x) = \begin{cases} x, & x \geq 0 \\ a(e^x - 1), & otherwise \end{cases} \tag{2.7}$$

2.2. Višeslojni perceptron

Općenita unaprijedna neuronska mreža najčešće se naziva i višeslojnim perceptronom. Sadrži 3 vrste slojeva: ulazni, skriveni i izlazni. Ima 1 ulazni i 1 izlazni sloj, a skrivenih može biti po volji mnogo. Ilustracija jednog višeslojnog perceptrona dana je na slici 2.7.



Slika 2.7: Višeslojni perceptron [10]

Ulazni sloj predstavlja značajke podataka koje obrađujemo, na primjer za sliku, svaki pojedini neuron bi bio jedan piksel slike. U njima se ne vrši nikakva obrada već se samo proslijede prvom idućem skrivenom sloju. Najčešće su slojevi potpuno povezani, odnosno svaki neuron iz prethodnog sloja je povezan sa svakim neuronom u idućem sloju.

Skriveni slojevi vrše obradu sumacije otežanih ulaza te primjenu aktivacijske funkcije. Ključ leži u nelinearnosti aktivacijskih funkcija, u protivnom se proizvoljno dubok višeslojni perceptron svodi na najobičniji linearni perceptron. Broj skrivenih slojeva kao i broj neurona u skrivenim slojevima ovisi o problemu koji zadan, no opće-

nito se može reći da neuronska mreža u skrivenim slojevima uči različite reprezentacije ulaznih podataka sa ciljem što uspješnijeg rješavanja problema. Dokazano je da neuronska mreža s barem jednim skrivenim slojem služi kao univerzalni funkcijski aproksimator, neovisno o izboru aktivacijske funkcije [23].

Izlazni sloj predstavlja izlaz iz mreže, pri klasifikacijskim problemima to bude vektor pouzdanosti pripadnosti ulaznog primjera određenom broju klasa. Za učenje parametara neuronske mreže koristi se algoritam unazadne propagacije.

2.2.1. Algoritam unazadne propagacije

Algoritam unazadne propagacije korigira parametre neuronske mreže s obzirom na gradijent funkcije pogreške. Gradijentni postupci u optimizaciji se koriste već stoljećima, pravilo ulančavanja gradijenata je također otprije poznato stoga je teško navesti sam početak primjene ovog postupka, pa čak i kod neuronskih mreža, no većina se slaže da je metoda popularizirana u radu [37].

Algoritam zahtijeva da su aktivacijske funkcije diferencijabilne, te da za zadane ulaze imamo i odgovarajuće izlaze, odnosno primjenjiv je samo u nadziranom učenju. Učenje se odvija u dvije faze:

1) Unaprijedni prolaz odnosno računanje aktivacija neurona za zadani ulaz sve do izlaza mreže, te posljedično računanje izlaza mreže za zadani ulaz.

2) Unazadni prolaz gdje se računa odstupanje izlaza mreže od traženog izlaza zadanog primjera, propagira se unazad kroz mrežu računajući grešku pri svakom od neurona, te naposlijetku se obavlja korekcija težina mreže jednom od gradijentnih metoda.

Ulančavanje gradijenata pokazat ćemo na jednostavnom primjeru mreže sa ulaznim, jednim skrivenim, te izlaznim slojem. Neka je zadano N primjera za učenje oblika $(x_\mathrm{n}, y_\mathrm{n})$. Neka mreža koristi sigmoidalnu aktivacijsku funkciju u skrivenom i izlaznom sloju, te minimizira funkciju srednje kvadratne pogreške

$$E = \frac{1}{2N}\sum_{n=1}^{N}(y_\mathrm{n} - o_\mathrm{n})^2 \tag{2.8}$$

gdje je o_n izlaz mreže za ulaz x_n. Za početak ćemo pokazati izraz za podešavanje parametara izlaznog sloja. Neka $\mathbf{W^{(i)}}$ označava matricu parametara izlaznog sloja, a $\mathbf{h_n}$ vektor izlaza srednjeg sloja za n-ti primjer, odnosno ulaza u izlazni sloj, kojemu je dodan član pristranosti 1. Tada derivacija funkcije greške po parametrima izlaznog sloja izgleda:

$$\frac{\partial E}{\partial \mathbf{W^{(i)}}} = \frac{\partial E}{\partial o_{\mathrm{n}}} \frac{\partial o_{\mathrm{n}}}{\partial net_{\mathrm{n}}{}^{(\mathrm{i})}} \frac{\partial net_{\mathrm{n}}{}^{(\mathrm{i})}}{\partial \mathbf{W^{(i)}}} \tag{2.9}$$

$$\frac{\partial E}{\partial o_{\mathrm{n}}} = -\frac{1}{N} \sum_{n=1}^{N} (y_{\mathrm{n}} - o_{\mathrm{n}}) \tag{2.10}$$

Pošto je o_{n} sigmoidalna funkcija, njena derivacija iznosi:

$$\frac{\partial o_{\mathrm{n}}}{\partial net_{\mathrm{n}}{}^{(\mathrm{i})}} = o_{\mathrm{n}}(1 - o_{\mathrm{n}}) \tag{2.11}$$

net označava ulazni sloj (u ovom slučaju skriveni sloj) pomnožen s matricom težina odnosno parametara, u ovom slučaju izlaznog sloja:

$$net_{\mathrm{n}}{}^{(\mathrm{i})} = \mathbf{W^{(i)}} h_{\mathrm{n}} \tag{2.12}$$

$$\frac{\partial net_{\mathrm{n}}{}^{(\mathrm{i})}}{\partial \mathbf{W^{(i)}}} = h_{\mathrm{n}} \tag{2.13}$$

Izračunavši sve potrebne parcijalne derivacije konačan izraz za gradijent izlaznog sloja glasi:

$$\frac{\partial E}{\partial \mathbf{W^{(i)}}} = -\frac{1}{N} \sum_{n=1}^{N} (y_{\mathrm{n}} - o_{\mathrm{n}}) o_{\mathrm{n}}(1 - o_{\mathrm{n}}) h_{\mathrm{n}} \tag{2.14}$$

Često se koristi i skraćena verzija zapisa:

$$\frac{\partial E}{\partial \mathbf{W^{(i)}}} = -\frac{1}{N} \sum_{n=1}^{N} \delta_{\mathrm{n}}{}^{(\mathrm{i})} h_{\mathrm{n}} \tag{2.15}$$

gdje je

$$\delta_{\mathrm{n}}{}^{(\mathrm{i})} = (y_{\mathrm{n}} - o_{\mathrm{n}}) o_{\mathrm{n}}(1 - o_{\mathrm{n}}) \tag{2.16}$$

Neka $\mathbf{W^{(h)}}$ predstavlja matricu parametara skrivenog sloja, a neka x_{n} predstavlja ulazni vektor u skriveni sloj, također proširen s članom pristranosti 1. Da bi dobili gradijent skrivenog sloja moramo se vratit korak unatrag:

$$\frac{\partial net_{\mathrm{n}}{}^{(\mathrm{i})}}{\partial \mathbf{W^{(h)}}} = \mathbf{W^{(i)}} \frac{\partial h_{\mathrm{n}}}{\partial \mathbf{W^{(h)}}} \tag{2.17}$$

Odnosno nastavljamo s parcijalnim deriviranjem unatrag kroz mrežu. Trenutno gradijent izgleda:

$$\frac{\partial E}{\partial \mathbf{W^{(i)}}} = -\frac{1}{N} \sum_{n=1}^{N} \delta_{\mathrm{n}}{}^{(\mathrm{i})} \mathbf{W^{(i)}} \frac{\partial h_{\mathrm{n}}}{\partial \mathbf{W^{(h)}}} \tag{2.18}$$

$$\frac{\partial h_{\mathrm{n}}}{\partial \mathbf{W^{(h)}}} = \frac{\partial h_{\mathrm{n}}}{\partial net_{\mathrm{n}}{}^{(\mathrm{h})}} \frac{\partial net_{\mathrm{n}}{}^{(\mathrm{h})}}{\partial \mathbf{W^{(h)}}} \tag{2.19}$$

Pošto je h_{n} izlaz sigmoidalne aktivacijske funkcija, derivacija glasi:

$$\frac{\partial h_{\mathrm{n}}}{\partial net_{\mathrm{n}}{}^{(\mathrm{h})}} = h_{\mathrm{n}}(1 - h_{\mathrm{n}}) \tag{2.20}$$

$$net_{\mathrm{n}}{}^{(\mathrm{h})} = \mathbf{W^{(h)}} x_{\mathrm{n}} \tag{2.21}$$

$$\frac{\partial net_{\mathrm{n}}{}^{(\mathrm{h})}}{\mathbf{W^{(h)}}} = x_{\mathrm{n}} \tag{2.22}$$

Izračunom svih potrebnih parcijalnih derivacija dobije se:

$$\frac{\partial h_{\mathrm{n}}}{\partial \mathbf{W^{(h)}}} = h_{\mathrm{n}}(1 - h_{\mathrm{n}}) x_{\mathrm{n}} \tag{2.23}$$

Konačan gradijent parametara skrivenog sloja glasi:

$$\frac{\partial E}{\partial \mathbf{W^{(h)}}} = -\frac{1}{N} \sum_{n=1}^{N} \delta_{\mathrm{n}}{}^{(\mathrm{i})} \mathbf{W^{(i)}} h_{\mathrm{n}}(1 - h_{\mathrm{n}}) x_{\mathrm{n}} \tag{2.24}$$

ili skraćeno:

$$\frac{\partial E}{\partial \mathbf{W^{(h)}}} = -\frac{1}{N} \sum_{n=1}^{N} \delta_{\mathrm{n}}{}^{(\mathrm{h})} x_{\mathrm{n}} \tag{2.25}$$

gdje je

$$\delta_{\mathrm{n}}{}^{(\mathrm{h})} = \delta_{\mathrm{n}}{}^{(\mathrm{i})} \mathbf{W^{(i)}} h_{\mathrm{n}}(1 - h_{\mathrm{n}}) \tag{2.26}$$

Ulančavanje se nastavlja po istom principu i za više slojeva. Učenje se vrši optimizacijskim postupcima koji se temelje na gradijentima. Nakon što smo pokazali kako se računaju gradijenti s obzirom na funkciju pogreške neuronske mreže, slijedi pregled odabranih optimizacijskih metoda.

Gradijentni spust

Gradijentni spust je osnovna numerička metoda minimizacije funkcije pogreške u strojnom učenju. Promatramo funkciju pogreške kao funkciju parametara algoritma, te optimiziramo parametre algoritma u smjeru minimalne greške te funkcije. Gornji primjer pokazuje kako izračunati gradijente parametara, dok se podešavanje parametara odvija po idućoj formuli:

$$\mathbf{W^{(n+1)}} = \mathbf{W^{(n)}} - \eta \frac{\partial E}{\partial \mathbf{W^{(n)}}} \tag{2.27}$$

gdje je $n + 1$ vremenska oznaka buduće vrijednosti, n trenutne, a η nazivamo stopom učenja, ona određuje koliki će biti pomak u smjeru gradijenta. Kao i svi numerički postupci, stabilnost ovog postupka ovisi o odabiru vrijednosti η.

Postoje tri različita načina osvježavanja parametara. Stohastički gradijentni spust se temelji na tome da se za svaki primjer izračunaju gradijenti i odma osvježe parametri modela. Grupni gradijentni spust zbroji gradijente svih primjera, uzme prosjek tako da zbroj podijeli s brojem primjera te onda osvježi parametre modela. Treći način je kompromisni, ujedno i najkorišteniji, u kojem se odredi nekakav n koji je veličina grupe nakon koje će model osvježavati parametre. Stohastički gradijentni spust je vremenski skup radi čestog osvježavanja parametara, no otporniji je na lokalne optimume. Grupni gradijentni spust radi izrazito dobro pri konveksnim problemima, no ne i u onim gdje krivulje funkcije pogreške imaju mnogo lokalnih optimuma. Također je memorijski zahtjevno koristiti sve primjere za učenje odjednom. Upravo zato je i najkorišteniji kompromisni način, jer veličinom grupe kontroliramo prednosti i mane oba načina. Postupak optimizacije se vrši do zadovoljavanja zadanih kriterija konvergencije ili do isteka postavljenog maksimalnog broja iteracija.

Moment

Moment metoda osim gradijenta uzima u obzir i gradijent u prethodnom koraku. Cilj je dobiti konzistentniji smjer gradijenta, dajući mu "širu sliku" kretnje po hiperravnini funkcije pogreške, a ne samo trenutno stanje. Predložena je kao poboljšanje nad stohastičkim gradijentnim spustom u radu [37].

$$\Delta W^{(\mathrm{n})} = \eta \frac{\partial E}{\partial \mathbf{W^{(n)}}} + \alpha \Delta W^{(\mathrm{n\text{-}1})} \tag{2.28}$$

$$W^{(\mathrm{n+1})} = W^{(\mathrm{n})} - \Delta W^{(\mathrm{n})} \tag{2.29}$$

RMSProp

RMSProp (engl. *Root Mean Square Propagation*) je varijanta gradijentnog spusta s dodanim parametrom [41]. Uz stopu učenja η, postoji i stopa zaboravljanja γ. Osnovna ideja je da model na početku brzo podešava težine, no što je bliži konvergenciji gradijenti su manji stoga je i proces učenja sporiji. RMSProp to mijenja tako što modificira stopu učenja obrnuto proporcionalno magnitudi gradijenta.

$$v^{(\mathrm{n})} = \gamma v^{(\mathrm{n\text{-}1})} + (1 - \gamma) \left(\frac{\partial E}{\partial \mathbf{W^{(n)}}} \right)^2 \tag{2.30}$$

Po izrazu 2.28 modificiramo stopu učenja. $v^{(\mathrm{n})}$ označava magnitudu gradijenta u n-tom koraku, a pri tom koristi i magnitudu gradijenta u $n-1$ koraku, sve zajedno parametrizirano s γ. Sada postupak osvježavanja parametara izgleda:

$$\mathbf{W^{(n+1)}} = \mathbf{W^{(n)}} - \frac{\eta}{\sqrt{v^{(n)}}} \frac{\partial E}{\partial \mathbf{W^{(n)}}} \tag{2.31}$$

Adam

Adam (engl. *Adaptive Moment Estimation*) je kombinacija Moment i RMSProp metoda [25]. Eksperimentalno je dokazan kao učinkovita metoda optimizacije te su ponuđeni parametri za koje se smatra da su široko primjenjivi i robusni.

$$m_{\mathrm{w}}{}^{(\mathrm{n+1})} = \beta_1 m_{\mathrm{w}}{}^{(\mathrm{n})} + (1 - \beta_1) \frac{\partial E}{\partial \mathbf{W^{(n)}}} \tag{2.32}$$

$$v_{\mathrm{w}}{}^{(\mathrm{n+1})} = \beta_2 v_{\mathrm{w}}{}^{(\mathrm{n})} + (1 - \beta_2) \left(\frac{\partial E}{\partial \mathbf{W^{(n)}}} \right)^2 \tag{2.33}$$

$$\hat{m}_{\mathrm{w}}{}^{(\mathrm{n})} = \frac{m_{\mathrm{w}}{}^{(\mathrm{n})}}{1 - \beta_1^n} \tag{2.34}$$

$$\hat{v}_{\mathrm{w}}{}^{(\mathrm{n})} = \frac{v_{\mathrm{w}}{}^{(\mathrm{n})}}{1 - \beta_2^n} \tag{2.35}$$

$$\mathbf{W^{(n+1)}} = \mathbf{W^{(n)}} - \eta \frac{\hat{m}_{\mathrm{w}}{}^{(\mathrm{n})}}{\sqrt{\hat{v}^{(\mathrm{n})}} + \epsilon} \tag{2.36}$$

Iz gornjih jednadžbi očigledno je da postupak ima 4 parametra. Parametri β_1 i β_2 su faktori zaboravljanja za prvi, odnosno drugi moment gradijenata, ϵ je mali korekcijski faktor za slučaj izbjegavanja dijeljenja s 0 i η koja je stopa učenja kao i u prethodnim gradijentnim postupcima. Relacije 2.32 i 2.33 daju pristrane procjenitelje prvog i drugog momenta gradijenta stoga se korigiraju za faktor pristranosti. Pristranost se dobije inicijalizacijom $v_{\mathrm{w}}{}^{(0)} = 0$ i $m_{\mathrm{w}}{}^{(0)} = 0$, a na primjeru za v_{w} ćemo pokazati kako [25]. Pošto vrijedi $v_{\mathrm{w}}{}^{(0)} = 0$, to znači da:

$$v_{\mathrm{w}}{}^{(\mathrm{n})} = (1 - \beta_2) \sum_{i=1}^{n} \beta_2^{n-i} g_i^2 \tag{2.37}$$

Zanima nas ponašanje $\mathbb{E}[v_{\mathrm{w}}{}^{(\mathrm{n})}]$ s obzirom na $\mathbb{E}[g_n^2]$, tako da možemo korigirat pristranost.

$$\mathbb{E}[v_{\mathrm{w}}{}^{(\mathrm{n})}] = \mathbb{E}\left[(1 - \beta_2) \sum_{i=1}^{n} \beta_2^{n-i} g_i^2 \right] \tag{2.38}$$

$$\mathbb{E}[v_{\mathrm{w}}{}^{(\mathrm{n})}] = \mathbb{E}[g_n^2](1 - \beta_2) \sum_{i=1}^{n} \beta_2^{n-i} + \zeta \tag{2.39}$$

$$\mathbb{E}[v_{\mathrm{w}}{}^{(\mathrm{n})}] = \mathbb{E}[g_n^2](1 - \beta_2^n) + \zeta \tag{2.40}$$

gdje je $\zeta = 0$ ukoliko je $\mathbb{E}[g_i^2]$ stacionaran proces. U jednadžbi 2.40 ostaje $(1 - \beta_2^n)$ što je upravo korigirajući faktor u jednadžbi 2.35.

2.3. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže, za razliku od višeslojnih perceptrona, koriste konvolucijske slojeve o kojima će biti govora kasnije u ovoj sekciji. Prva takva zabilježena mreža zove se Neocognitron [21], no za popularizaciju istih uglavnom se navodi rad iz 1996. godine [26]. Najveći napredak su postigle u polju računalnog vida, gdje su pri nekim problemima postigle i nadljudske rezultate.

2.3.1. Konvolucija

U obradi slike koristimo 2D konvolucijski operator nad diskretnim vrijednostima. Stoga operator 2D diskretne konvolucije se definira preko idućeg izraza:

$$f(x,y) * g(x,y) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f(n_1,n_2) g(x-n_1, y-n_2) \tag{2.41}$$

Na prvi pogled jednostavan izraz, oko kojeg je teško razviti intuiciju [4]. No, gornji izraz je općenit slučaj za konvoluciju dva dvodimenzionalna signala, na primjerima ćemo vidjeti da se u obradi slike radi o mnogo jednostavnijem slučaju. Slika 2.8 prikazuje 2 često korištena Laplace filtra.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Slika 2.8: Laplace filter [7]

Slika 2.9: Primjer djelovanja Laplace filtra na sliku [8]

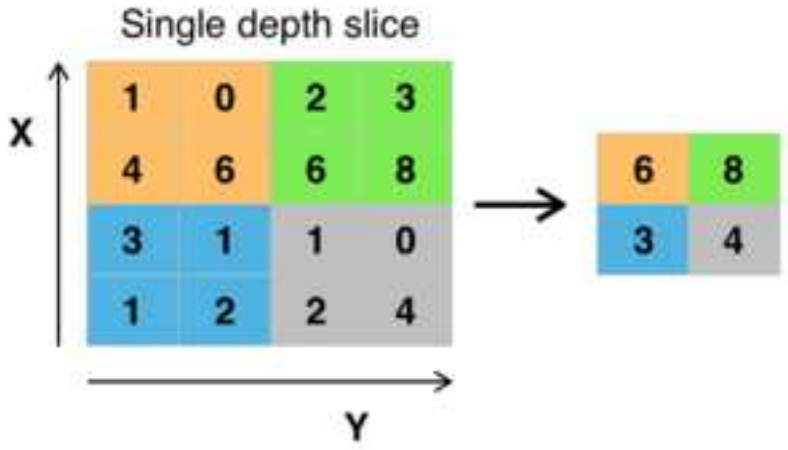
Konvolucijskim prozorom se prolazi kroz cijelu sliku, sumirajući težinski pikselne vrijednosti, te na izlazu dobijemo obrađenu sliku. Primjer djelovanja Laplace filtra dan je na slici 2.9.

Upravo ideja da se koriste konvolucijski operatori s filtrima koji su naučeni, a ne predodređeni, omogućile su neuronskim mrežama izrazit uspjeh u polju računalnog vida. Pri obradi slike to je od iznimne važnosti jer se postiže lokalna osjetljivost i dijeljenje parametara, odnosno model sam nauči mapiranje iz slike u prostor značajki koji je kasnije povezan na potpuno povezani sloj najčešće, ili nekakav drugi klasifikator.

Korištenje konvolucijskih slojeva omogućava slojevitiju mrežu, pošto su parametri dijeljeni, problem naglog rasta broja parametara sa slojevitošću je manji nego kod potpuno povezanih slojeva višeslojnog perceptrona. Stoga se konvolucijski slojevi bliže ulaznoj slici najčešće fokusiraju na detekciju značajki niske razine poput rubova i linija, dok kasniji konvolucijski slojevi modeliraju značajke viših razina poput razumijevanja scene i objekata.

2.3.2. Slojevi sažimanja

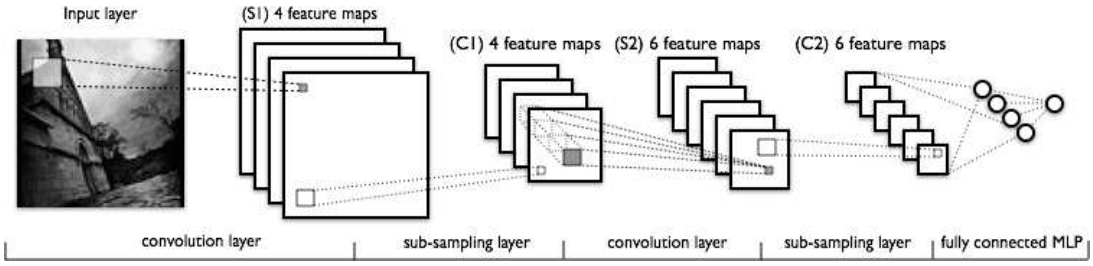
Uz konvolucijske slojeve često se u arhitekturama konvolucijskih neuronskih mreža mogu pronaći i slojevi sažimanja. Najčešće korišten operator je sažimanje po maksimalnom odzivu, no postoji sažimanje po minimalnom i prosječnom odzivu. Slojevi sažimanja općenito primaju sliku te zatim po regijama slike primjenjuju odabrani operator. Postižu smanjenje dimenzionalnosti, posljedično i smanjenje broja parametara modela što spriječava prenaučenost.



Slika 2.10: Primjer sažimanja po maksimalnom odzivu [9]

2.3.3. Arhitektura konvolucijskih neuronskih mreža

Konvolucijske neuronske mreže primaju 2D ulaze poput slike, provlače ih kroz prvi konvolucijski sloj, nakon što se dobije prva mapa značajki, neuroni konvolucijskog sloja primjenjuju aktivacijske funkcije te se nakon toga vrši sloj sažimanja. Taj proces se ponavlja ovisno o tome koliko arhitektura mreže ima konvolucijskih slojeva. Izlaz zadnjeg konvolucijskog sloja je također mapa značajki koja se zatim "izravna" te se predaje potpuno povezanim slojevima.



Slika 2.11: Primjer arhitekture konvolucijske neuronske mreže [11]

Predložena su i pojednostavljenja u arhitekturama konvolucijskih mreža [40]. Predloženo je izbacivanje slojeva sažimanja te proširenje koraka konvolucijskih filtara. Moderni duboki modeli za analizu slike izbjegavaju korištenje potpuno povezanih slojeva [29].

U zadnje vrijeme počeo se koristi pojam prijenosa učenja (engl. *transfer learning*). Odnosi se na to da se mreža naučena nad jednim skupom podataka primjenjuje i nad drugim skupovima podataka. Često samo zahtijeva modifikacije nad posljednjim klasifikacijskim slojem jer se skupovi podataka razlikuju u broju klasa. Proces treniranja je isti, no praksa je pokazala da značajke naučene na problemu s velikim brojem primjera za učenje služe bolje od učenja nove mreže s incijaliziranim težinama. Stoga nije ni čudo da do sada već postoji par standardnih modela koji su unaprijed naučeni za klasifikaciju na ImageNetu [38], koje se dalje prilagođavaju nad ostalim skupovima podataka. Najčešće se koristi kada za ciljani zadatak postoji puno manje primjera za učenje nego što je to slučaj od osnovnog zadatka za koji je mreža naučena.

2.4. Grupna normalizacija

Grupna (engl. *batch*) normalizacija je suvremena metoda za efektivno učenje neuronskih mreža [24]. Motivacija je parametrizirano normalizirati ulaze u svaki sloj neuronske mreže pošto se distribucije ulaza u slojeve mijenjaju kroz treniranje. Gore navedeni problem povlači i druge probleme poput opreznog postavljanja stope učenja te inicijalizacije parametara mreže. Grupna normalizacija do neke mjere eliminira te probleme. Grupe se normaliziraju po sljedećim izrazima:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.42}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{2.43}$$

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.44}$$

$$y_i = \gamma\hat{x_i} + \beta \tag{2.45}$$

Jednadžba 2.38 računa prosječnu vrijednost grupe, jednadžba 2.39 računa varijancu grupe. Koristeći oboje, podaci iz grupe se normaliziraju na prosječnu vrijednost 0 s varijancom 1 s izrazom 2.40. Izraz 2.41 nudi 2 parametra, γ i β koji mogu skalirati i pomaknuti distribuciju. To su optimizirajući parametri, odnosno optimizacijski postupci traže idealne γ i β za svaki sloj. y_i predstavlja izlaz iz grupe normalizacije, odnosno transformirani ulaz.

Metoda se ispostavila od značajne važnosti jer ulazi kasnijih slojeva više ne ovise o magnitudama prijašnjih slojeva pošto se vrši parametrizirana normalizacija. Model sam nauči parametre grupne normalizacije, odnosno skaliranje i pomak normalno distribuiranih ulaza. Autori su eksperimentima pokazali da postupak dovodi do brže konvergencije i boljih rezultata, te se grupna normalizacija smatra neizostavnim dijelom većine modernih arhitektura dubokih neuronskih mreža.

Grupna normalizacija se primjenjuje na poseban način kod konvolucijskih slojeva neuronskih mreža. Za razliku od potpuno povezanih slojeva, gdje cijeli sloj dijeli parametre γ i β, kod konvolucijskih slojeva oni su dijeljeni po mapama značajki [3]. Dakle, neuroni unutar jedne mape značajki ne dijele samo težinske parametre, već i normalizacijske. To je potrebno radi očuvanja glavnog svojstva konvolucijskih neuronskih mreža, invarijantnosti na translaciju.

3. Generativne suparničke mreže

Generativne suparničke mreže predstavljaju modele u kojima su suprotstavljene dvije neuronske mreže, generator i diskriminator [22]. Zadatak generatora je modelirati distribuciju ulaznih podataka, a zadatak diskriminatora je odrediti da li podatak potječe iz stvarnog skupa ili je generiran generatorom. Diskriminator minimizira binarnu klasifikacijsku funkciju pogreške (da li je podatak na ulazu stvaran ili generiran), dok generator maksimizira grešku diskriminatora za generirane podatke. Model se uči nenadziranim učenje, odnosno nisu mu potrebni označeni podaci. Pošto su generator i diskriminator neuronske mreže, uče se algoritmom unazadne propagacije te nisu potrebne vjerojatnosne estimacije podataka kao kod drugih generativnih modela.

Generator je neuronska mreža koja na ulazu prima šum (najčešće uniformne ili normalne distribucije), provodi šum kroz skrivene slojeve, te na izlazu daje sliku. Vektor uzorkovan iz šuma najčešće označavamo s z, dok vjerojatnosnu distribuciju šuma označavamo s p_z. Zadatak generatora je modelirati vjerojatnosnu distribuciju p_g, da bude što sličnija distribuciji stvarnih podataka p_{data}. Globalno, generator se može interpretirati kao diferencijabilna funkcija $G(z; \theta_g)$, koja prima vektor šuma z, te parametrima θ_g definira preslikavanje šuma u podatkovni prostor.

Diskriminator je neuronska mreža koja na ulazu prima stvaran ili generiran podatak, provodi ga kroz skrivene slojeve te na izlazu daje skalarnu vrijednost, odnosno vjerojatnost pripadanja ulaza stvarnim podacima. Diskriminator predstavlja diferencijabilnu funkciju $D(x, \theta_d)$, koja prima vektor podatka x te parametrima θ_d definira klasifikaciju, odnosno da li je podatak stvaran ili generiran.

Generator i diskriminator igraju minimax igru sa sljedećom kriterijskom funkcijom:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))] \tag{3.1}$$

Kroz jednadžbu (3.1) zadan je maksimizacijski izraz za diskriminator, iz izraza je također očigledan i minimizacijski izraz generatora, odnosno generator minimizira $\log(1-D(G(z))$. Praksa je pokazala da minimiziranje tog izraza daje slabije gradijente na početku, te se savjetuje maksimizacija $\log D(G(z)$ [22]. Algoritam generativne suparničke mreže glasi:

for *number of training iterations* **do**

  for *ksteps* **do**

    Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from $p_z(z)$

    Sample minibatch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from real data

    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))\right] \tag{3.2}$$

  end

  Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from $p_z(z)$

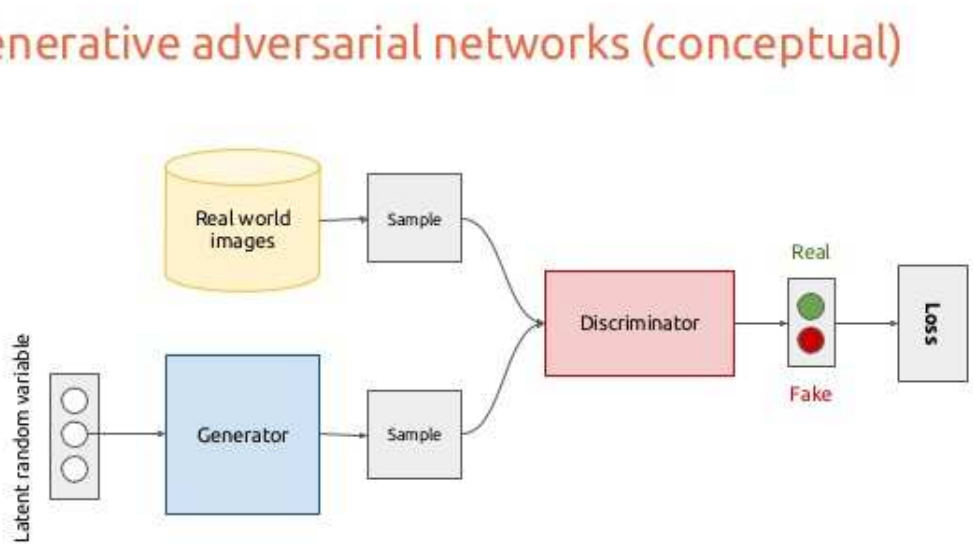
  Update the generator by ascending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log D(G(z^{(i)})) \tag{3.3}$$

end

Algorithm 1: GAN

Model generativnih suparničkih mreža podrazumijeva treniranje diskriminatora do optimalnosti. Algoritam radi s numeričkom aproksimacijom gdje se diskriminator trenira određen broj koraka, jer bi inače model bio vremenski preskup. Praksa je pokazala da je dovoljno trenirati diskriminator jednom, a nekad generator više puta jer model pati od nestabilnosti pošto diskriminator prebrzo nauči razlikovati podatke, no o tome više u sljedećem poglavlju. Slika 3.1 ilustrira generativnu suparničku mrežu.



Slika 3.1: Generativna suparnička mreža [6]

3.1. Globalni optimum modela

Globalni optimum modela se dobije kada $p_g = p_{data}$. Slijedi i dokaz za tu tvrdnju. Prvo izvodimo optimalan diskriminator za bilo koji generator.

Propozicija 1. *Za fiksiran generator, optimalan diskriminator glasi:*

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{3.4}$$

Dokaz. Jednadžba (3.1) daje kriterij koji diskriminator optimizira bez obzira na generator.

$$\begin{aligned} V(G,D) &= \int_x p_{data}(x) \log D(x) dx + \int_z p_z(z) \log(1 - D(G(z)) dz \\ &= \int_x (p_{data}(x) \log D(x) + p_g(x) \log(1 - D(x))) dx \end{aligned} \tag{3.5}$$

Cilj je pronaći optimalan D(x). Deriviranjem funkcije pod integralom te pronalaženjem optimalnog D(x) optimizirali smo i izraz V pošto optimalni D maksimizira V u svakoj točki. Radi jednostavnosti notacije, gornji podintegralni izraz svest ćemo na:

$$f(y) = a \log y + b \log(1 - y) \tag{3.6}$$

gdje je $a = p_{data}(x)$, $b = p_g(x)$, $y = D(x)$.

$$f'(y) = \frac{a}{y} - \frac{b}{1 - y} \tag{3.7}$$

Za dobit ekstrem funkcije prvu derivaciju izjednačimo s 0.

$$\begin{aligned} \frac{a}{y} - \frac{b}{1-y} &= 0 \\ \frac{a}{y} &= \frac{b}{1-y} \\ a - ay &= by \\ a &= ay + by \\ a &= y(a+b) \\ y &= \frac{a}{a+b} \end{aligned} \tag{3.8}$$

Gornji izraz je ekstrem funkcije, sada treba provjeriti vrijednost druge derivacije funkcije u toj točki, ukoliko je negativna radi se o maksimumu, a maksimum je u ovom slučaju optimum.

$$\begin{aligned} f''(y) &= -\frac{a}{y^2} - \frac{b}{(1-y)^2} \\ f''\left(\frac{a}{a+b}\right) &= -\frac{a}{\left(\frac{a}{a+b}\right)^2} - \frac{b}{(1-\frac{a}{a+b})^2} \end{aligned} \tag{3.9}$$

Pošto se radi s vjerojatnostima koje su definirane na intervalu [0, 1], gornji izraz je uvijek negativan (osim kada $a = b = 0$, ali taj slučaj nema smisla razmatrati), te smo dobili da je $\frac{a}{a+b}$ maksimum tražene funkcije, odnosno da vrijedi:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{3.10}$$

□

Nakon pronalaska optimalnog diskriminatora, slijedi teorem koji potvrđuje da se globalni optimum minimax igre, odnosno kriterijske funkcije nalazi u $p_g = p_{data}$

Teorem 1. *Globalni minimum kriterija $C(G)$ je postignut ako i samo ako $p_g = p_{data}$ te u toj točki on iznosi* $\log \frac{1}{4}$.

Dokaz.

$$
\begin{aligned}
C(G) &= \max_{\mathbf{D}} V(D,G) \qquad (3.11)\\
&= V(D^*,G)\\
&= \mathbb{E}_{x\sim p_{data}(x)}[\log D^*(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D^*(x)]\\
&= \int_x p_{data}(x)\log D^*(x)dx + \int_x p_g(x)\log(1-D^*(x))dx\\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right)dx + \int_x p_g(x)\log\left(1-\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right)dx\\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right)dx + \int_x p_g(x)\log\left(\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right)dx\\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)dx - \int_x p_{data}(x)\log 2dx\\
&+ \int_x p_g(x)\log\left(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)dx - \int_x p_g(x)\log 2dx\\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)dx - \log 2\\
&+ \int_x p_g(x)\log\left(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)dx - \log 2\\
&= KL\left(p_{data}\|\frac{p_{data}+p_g}{2}\right) + KL\left(p_g\|\frac{p_{data}+p_g}{2}\right) - 2\log 2\\
&= -\log 4 + 2JSD(p_{data}\|p_g)\\
&= \log\frac{1}{4} + 2JSD(p_{data}\|p_g)
\end{aligned}
$$

□

Kada vrijedi $p_{data} = p_g$ tada je Jensen-Shannon divergencija jednaka nuli, te se dobije globalni minimum $\log\frac{1}{4}$. Tada generator savršeno replicira distribuciju stvarnih podataka. U svim ostalim slučajevima, Jensen-Shannon divergencija je veća od 0 stoga se distribucija p_g ne nalazi u globalnom minimumu.

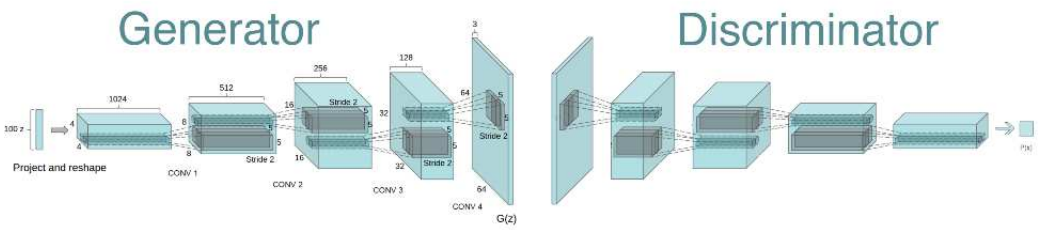
3.2. Duboke konvolucijske generativne suparničke mreže

U generativnim suparničkim mrežama na početku poglavlja predviđeni su višeslojni perceptroni kao odabran model za generator i diskriminator. Duboke konvolucijske generativne suparničke mreže koriste konvolucijske arhitekture [35]. Primjena već poznatih i često korištenih dubokih arhitektura ne garantira uspješnost treniranja modela jer generativne suparničke mreže pate od degradacije treniranja u propast (pojava pri kojoj generator daje istu sliku za proizvoljan vektor šuma, ne učeći pritom modeliranje stvarnih slika) za velik broj arhitektura, te je pronalazak stabilnih arhitektura možda i najveći izazov pri njihovoj primjeni. Autori u gore spomenutom radu navode tri važna faktora pri modeliranju dubokih konvolucijskih generativnih suparničkih mreža.

Prvi faktor je korištenje konvolucijskih mreža bez slojeva sažimanja, već korištenje konvolucija s korakom preko kojeg mreže uče prostorno povećanje ili sažimanje. Generator tako uči prostorno povećanje preko dekonvolucija s korakom, a diskriminator prostorno sažimanje preko konvolucija s korakom. Korak je hiperparametar modela, najčešći odabir je korak veličine 2 u obe dimenzije konvolucije.

Drugi faktor je izostavljanje potpuno povezanih slojeva. Generator uzima vektor šuma iz latentnog prostora, zatim koristi potpuno povezan sloj za postići određene dimenzije u skrivenom sloju, no taj sloj se preoblikuje u 4D tensor gdje jedna dimenzija odgovara veličini grupe, dvije dimenzije predstavljaju matricu veličine slike, a zadnja dimenzija je broj kanala u slici. Slojevi dekonvolucije se ponavljaju dok se ne dođe do traženih dimenzija. Kod diskriminatora, na ulaz se direktno primjenjuju konvolucijski slojevi do posljednjeg, koji se zatim "izravna" te predaje sigmoidalnom neuronu.

Treći faktor je korištenje grupne normalizacije što pridonosi stabilnijem treniranju ukupnog modela, te izbjegavanju degradacije treniranja u propast. Grupna normalizacija se ne primjenjuje na ulaz u diskriminator, te na izlaz generatora jer se eksperimentalno pokazalo da to vodi nestabilnijem treniranju [35]. Također se savjetuje korištenje ReLU aktivacijska funkcija, te Leaky ReLU što predstavlja parametriziranu ReLU funkciju gdje je parametar $a \in < 0, 1 >$.

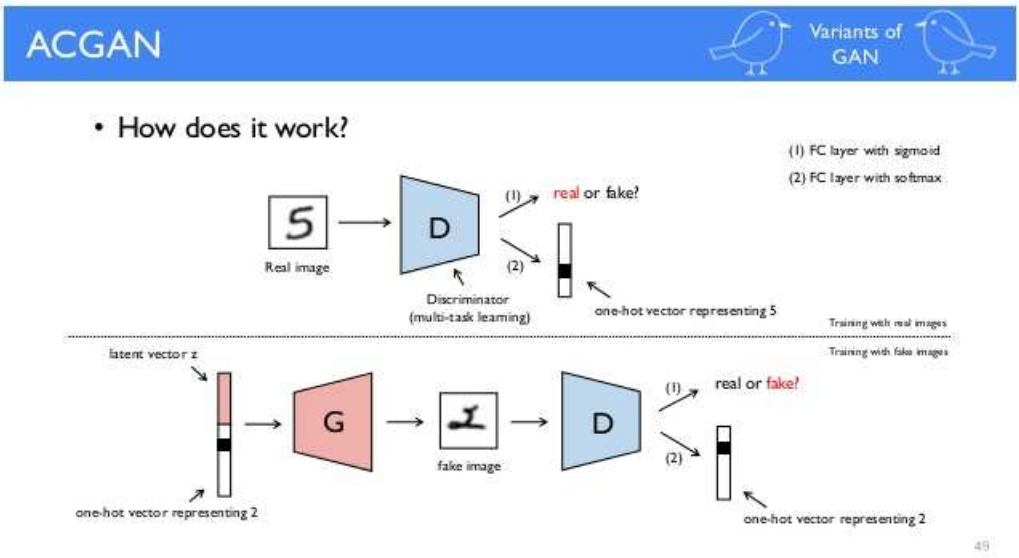


Slika 3.2: Duboka konvolucijska generativna suparnička mreža [5]

3.3. Klasifikatorska generativna suparnička mreža

Gore spomenuti modeli uče isključivo nenadzirano, odnosno nisu im potrebni označeni primjeri. Klasifikatorska generativna suparnička mreža zahtijeva označene primjere za učenje, te istovremeno diskriminator uči klasifikaciju primjera po danim oznamaka, kao i binarnu klasifikaciju da li je podatak stvaran ili generiran [34].

Generatoru generira vektor šuma na koji doda oznaku koja mu je proslijeđena, stoga generator uči generirati podatke po klasi što mu znatno olakšava posao. Diskriminator ima 2 gubitka, gubitak po uzorkovanju te po klasifikaciji, model je usporediv s pravim nadziranim klasifikatorima, no u praksi se nije pokazao boljim. Obe tvrdnje će biti pokazane u sljedećem poglavlju.



Slika 3.3: Klasifikatorska generativna suparnička mreža [1]

Funkcije izglednosti izgledaju:

$$L_S = \mathbb{E}[\log P(S = real|X_{real})] + \mathbb{E}[\log P(S = fake|X_{fake})] \tag{3.12}$$
$$L_C = \mathbb{E}[\log P(C = c|X_{real})] + \mathbb{E}[\log P(C = c|X_{fake})]$$

gdje L_S predstavlja izglednost uzorkovanja diskriminatora, a L_C klasifikacijsku izglednost. Diskriminator maksimizira vrijednost $L_S + L_C$ dok generator maksimizira $L_C - L_S$.

4. Implementacija i rezultati

Eksperimenti su provedeni na računalu s operacijskim sustavom Ubuntu 16.04 LTS, procesorom Intel i5-6600, grafičkom karticom GTX1060 6GB te 16GB RAMa. Za programsku implementaciju korišten je Tensorflow [20] te Python3. Za usporedbu i pomoć autor se poslužio implementacijama dostupnim na webu [18][2].

4.1. Skupovi podataka

U ovoj sekciji opisani su skupovi podataka na kojima se model trenirao.

4.1.1. MNIST

Za većinu eksperimenata korištena je baza rukom pisanih znakova MNIST [27]. Sastoji se od 55000 primjera za učenje, 5000 za validaciju, te 10000 za testiranje. Slike su dimenzija 28x28, a oznake dolaze kao vektor dimenzija 1x10 gdje broj jedan označava znamenku koju slika predstavlja. Slika 4.1 ilustrira neke od primjera iz skupa podataka. Slika 4.2 prikazuje primjere iz baze.

Slika 4.1: Primjeri MNIST znakova

4.1.2. FM3

FM3 je baza slika nadograđena na FM2 koja je pobliže opisana u radu [39]. Sastoji se od 6442 slike u boji iz prometa podijeljene u 8 klasa. Za učenje je predviđeno 3222 primjera, a za testiranje 3220 primjera. Slike su rezolucije 640x480x3 nastale u prometu, iz postavljene kamere na autu dok se vozio cestama. Pretprocesiranje se sastojalo od svođenja slike na kvadratni format 480x480x3 tako što je oduzeto 80 piksela s lijeve i desne strane, zatim preoblikovanje u sliku dimenzija 32x32x3 koja se dalje prosljeđivala modelu.

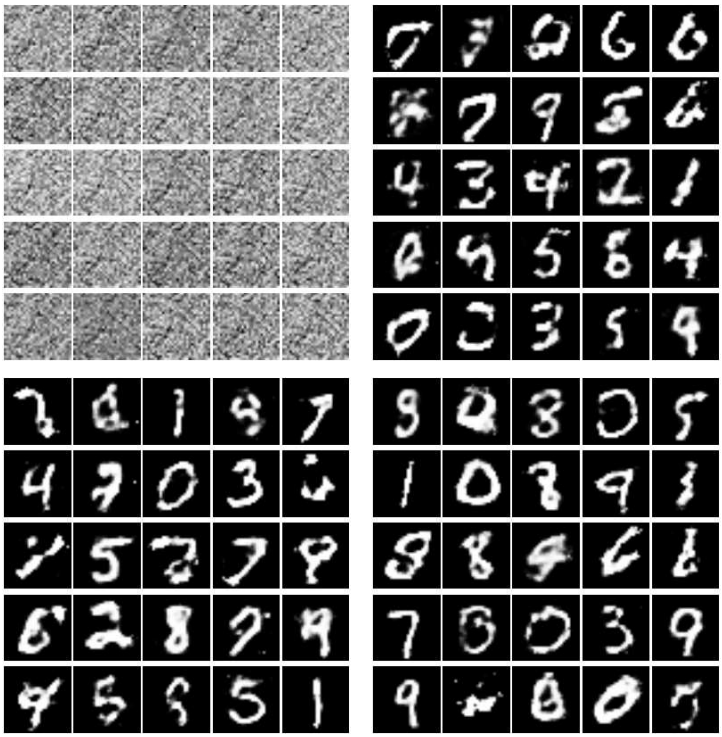
Slika 4.2: Primjeri iz baze slika FM3

4.2. Generativna suparnička mreža

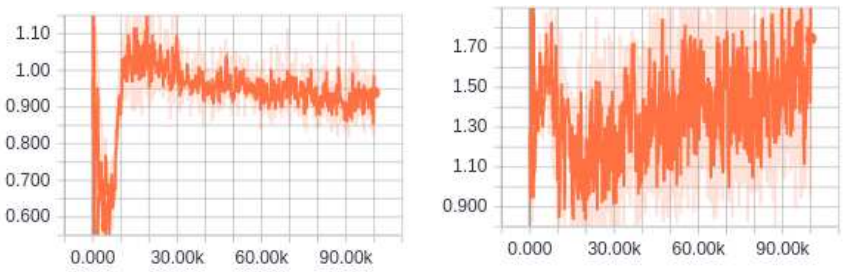
Generativna suparnička mreža zamišljena je pomoću dvaju višeslojnih perceptrona, jedan u ulozi diskriminatora, drugi u ulozi generatora. Početak modela predstavlja uzorkovanje iz šuma. Za veličinu vektora šuma odredili smo 100, odnosno vektor je veličine 1x100. Uzorkujemo ga iz distribucije $\mathcal{N}(0, 1)$. Prvi potpuno povezani sloj je veličine 100x200, te on preslikava šum na ulazu u skriveni sloj. Izlazni sloj je potpuno povezani sloj veličine 200x784 (odgovara dimenzijama slike 28x28). Skriveni sloj koristi ReLU aktivacijsku funkciju, izlazni sloj koristi sigmoidu. Diskriminator dobije ulaz veličine 784 (kada se "izravna" slika). Prvi skriveni sloj ima dimenzije 784x200 te koristi ReLU aktivacijsku funkciju, izlazni sloj je veličine 200x1 te koristi sigmoidalni neuron koji određuje vjerojatnost da je ulazni podatak stvaran. Za optimizaciju parametara diskriminatora koristi se stohastički gradijentni spust s stopom učenja 0.01, za optimizaciju parametara generatora koristi se Adam optimizator s predefiniranim parametrima. Težine se inicijaliziraju normalnom distribucijom s srednjom vrijednosti 0, a standardnom devijacijom 0.01, te se odbijaju sve generirane težine koje odstupaju za više od dvije standardne devijacije (*tensorflow.truncated_normal*), težine pristranosti se postavljaju na 0.1.

Odabrana veličina grupe primjera je 500, te postupak treniranja traje 100000 iteracija, crtajući nasumične uzorke koji se preslikavaju iz latentnog u podatkovni prostor svakih 5000 iteracija. Slika 4.3 ilustrira proces učenja generatora. Slika 4.4 prikazuje

gubitke diskriminatora i generatora. U idealnom slučaju gubitak generatora bi trebao iznositi $log2 \approx 0.3$, tada bi gubitak diskriminatora bio $log4 \approx 0.6$. No, radi se o jednostavnom modelu, te ne možemo očekivati da on u potpunosti modelira stvarnu distribuciju podataka. Diskriminator i generator se treniraju jednom po iteraciji. Eksperimenti s više parametara i slojeva višeslojnog perceptrona vrlo lako mogu propasti radi nestabilnosti arhitektura.



Slika 4.3: Uzorci nakon 0 (gore-lijevo), 25000 (gore-desno), 50000 (dolje-lijevo), 100000 (dolje-desno) iteracija



Slika 4.4: Gubitak diskriminatora (lijevo) i generatora (desno)

4.3. Duboke konvolucijske generativne suparničke mreže

U svrhu dokaza da [35] vodi ka stabilnijem treniranju različitih arhitektura, implementirane su 4 modela s različitim generatorima. Prva 3 modela se razlikuju po broju potpuno povezanih slojeva na početku (broj potpuno povezanih slojeva ide od 1 do 3), te sva 3 imaju 2 dekonvolucijska, dok četvrti ima samo 1 potpuno povezan sloj i 4 dekonvolucijska sloja. Fokus će biti na arhitekturama 1 i 4.

Sve varijable su inicijalizirane s *tensorflow.truncated_normal* i standardnom devijacijom 0.02, težine pristranosti se postavljaju na 0.1. Za treniranje se koristi Adam optimizator kojem je stopa učenja postavljena na 0.0002, te β_1 parametar na 0.5. Veličina grupe iznosi 64, a postupak treniranja traje 100000 iteracija. Vektor šuma se ovdje uzorkuje iz $\mathcal{U}(-1, 1)$. Važna napomena je da se za svako treniranje diskriminatora, generator trenira dvaput.

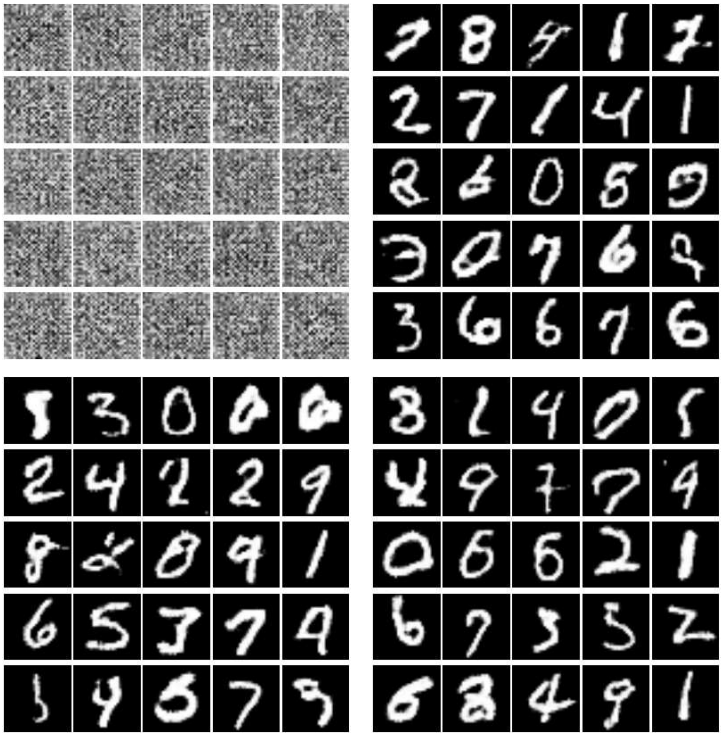
Generator prvog modela uzorkuje vektor šuma veličine 1x100 te je spojen na potpuno povezan sloj veličine 100x6272. Taj sloj se zatim preoblikuje u 7x7x128 gdje 7x7 predstavlja sliku koju generator povećava, a 128 broj kanala te slike. Sada slijedi dekonvolucijski sloj s pomakom 2 u obe dimenzije te dozvoljenim nadopunjavanjem. Dekonvolucijski sloj se sastoji od 7x7x128 filtara koji sliku na ulazu veličine 7x7x128, pretvore u sliku veličine 14x14x128. Idući dekonvolucijski sloj rezultira slikom dimenzija 28x28, njegov dekonvolucijski filtar je veličine 14x14x1. Grupna normalizacija se primjenjuje svugdje osim u zadnjem dekonvolucijskom sloju, aktivacijske funkcije su u svim slojevima ReLU, osim u zadnjem gdje se primjenjuje simgoidalna funkcija.

Generator četvrtog modela uzorkuje šum na isti način. Njegov potpuno povezani sloj je veličine 100x2048 koji se zatim preoblikuje u tenzor dimenzija 2x2x512. Dekonvolucijski filtri koriste nadopunjavanje te korak veličine 2. Prvi dekonvolucijski filtar je veličine 2x2x256, što znači da rezultira tenzorom 4x4x256. Drugi dekonvolucijski filtar je veličine 4x4x128, njegov izlaz je tenzor veličine 7x7x128. Treći dekonvolucijski filtar je veličine 7x7x64, njegov rezultat je tenzor veličine 14x14x64. Posljednji dekonvolucijski filtar je veličine 14x14x1 te je njegov rezultat slika dimenzija 28x28. Svi slojevi osim posljednjeg primjenjuju grupnu normalizaciju i ReLU, posljednji sloj koristi sigmoidalnu funkciju i nema grupnu normalizaciju. Generator četvrtog modela je nešto većeg kapaciteta od generatora prvog modela, te koristi više dekonvolucijskih slojeva stoga će bit zanimljivo usporediti dinamiku učenja oba modela.

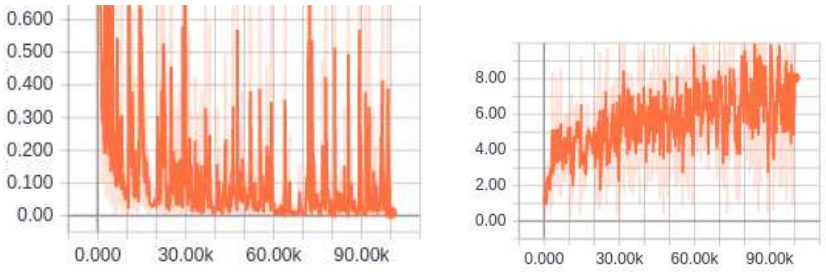
Diskriminator sva 4 modela je isti. Ulaz mu je slika dimenzija 28x28. Prvi konvolucijski sloj je veličine 5x5x16, s korakom 2 i nadopunjavanjem, što znači da mu je izlazni tenzor veličine 14x14x16. Drugi konvolucijski sloj je veličine 5x5x32, s korakom

2 i nadopunjavanjem što znači da mu rezultira tenzorom 7x7x32. Treći konvolucijski sloj je veličine 5x5x64, s korakom 2 i nadopunjavanjem, izlaz mu je veličine 4x4x64. Posljednji konvolucijski sloj je veličine 5x5x128, s korakom 2 i nadopunjavanjem što znači da je njegov izlaz tenzor dimenzija 2x2x128. Taj tenzor se zatim izravna da bude sloj veličine 1x512 te se proslijedi sigmoidalnom neuronu koji određuje pripadnost podatka. Svi konvolucijski slojevi koriste *leakyReLU* aktivacijsku funkciju, to je parametrizirana ReLU funkcija s parametrom $a = 0.2$ u ovom slučaju. Također koriste grupnu normalizaciju.

Slika 4.5 prikazuje generirane slike u određenim iteracijama, slika 4.6 prikazuje gubitke generatora i diskriminatora kroz iteracije za prvi model. Ponašanjem prikazanih gubitaka, da se naslutiti da generator nije dovoljnog kapaciteta naspram diskriminatora, no svejedno nauči modelirati distribuciju stvarnih podataka do neke mjere.

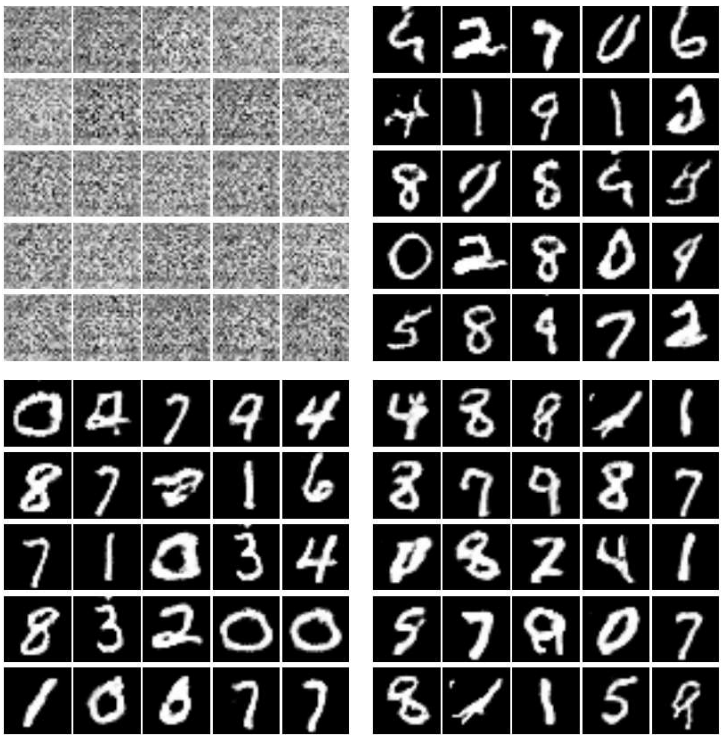


Slika 4.5: Uzorci nakon 0 (gore-lijevo), 25000 (gore-desno), 50000 (dolje-lijevo), 100000 (dolje-desno) iteracija (prvi model)

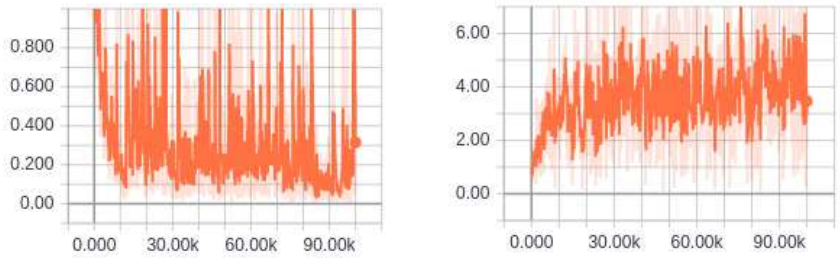


Slika 4.6: Gubitak diskriminatora (lijevo) i generatora (desno) (prvi model)

Slika 4.7 prikazuje generirane slike nakon broja iteracija, slika 4.8 prikazuje gubitke generatora i diskriminatora kroz treniranje za četvrti model. Zaključak je da oba modela nemaju dovoljno velik kapacitet generatora, no da ipak donekle uspješno modeliraju distribuciju stvarnih podataka. Četvrti model ima generator malo većeg kapaciteta te se to odražava na gubitku generatora koji je u prosjeku manji nego kod prvog modela. Također, veza između gubitaka diskriminatora i generatora je inverzna, odnosno rastom jednog pada drugi, što je očekivano ponašanje.



Slika 4.7: Uzorci nakon 0 (gore-lijevo), 25000 (gore-desno), 50000 (dolje-lijevo), 100000 (dolje-desno) iteracija (četvrti model)



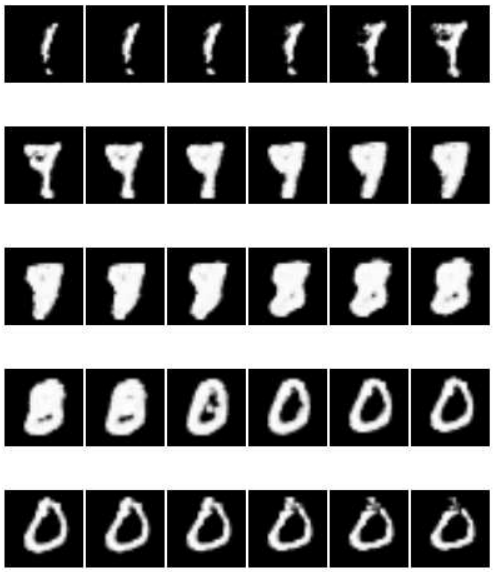
Slika 4.8: Gubitak diskriminatora (lijevo) i generatora (desno) (četvrti model)

4.3.1. Šetnja skrivenim prostorom

Skriveni prostor je onaj prostor iz kojeg uzorkujemo šum. Proces učenja generatora je takav da mapira područja iz prostora šuma u prostor stvarnih podataka. Kvaliteta mapiranja ovisi o parametrima generatora. Duboki modeli općenito imaju velik kapacitet te su sposobni prenaučiti primjere preslikavanja. Stoga jedna od metoda validacije da naše učenje uči distribuciju pravih podataka i preslikavanje, a da ne pamti naslijepo primjere koje je vidio, zove se šetnja skrivenim prostorom.

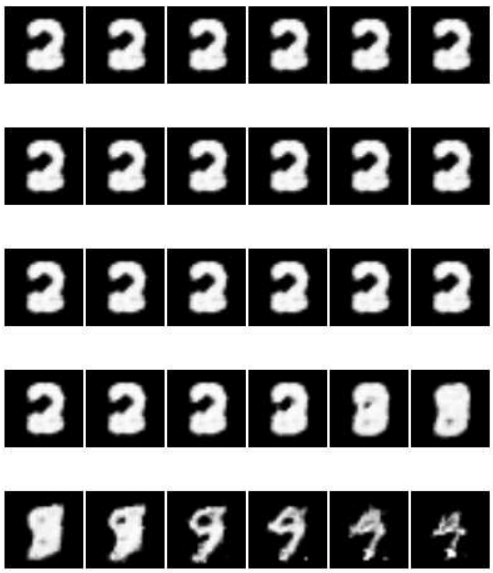
Nasumično uzorkovanje vektora šuma kao što to radimo trenirajući generator ne daje nam dovoljnu validacijsku informaciju, pošto ti primjeri mogu biti zapamćeni. Stoga, odabire se skup kolinearnih točaka u latentnom prostoru te se nacrta preslikavanje koje se dobije kada ih se mapira u podatkovni prostor s generatorom. Očekuju se prijelazi između podataka, te doze neodređenosti između gdje se oblici mijenjaju.

Prvi način pronalaženja kolinearnih točaka je inicijalizacija jednog vektora šuma na [0 0 ... 0] te zatim stvaranje novih s pomakom, npr 0.05, u oba smjera. Ilustracije radi drugi primjer bi izgledao [0.05 0.05 ... 0.05], a treći [-0.05 -0.05 ... -0.05]. Sve točke leže na istoj hiperravnini pošto su kolinearne, odnosno razlikuju se za faktor $k*pomak$. Slika 4.9 ilustrira ovakav način šetnje skrivenim prostorom. Iako se ne možemo sa sigurnošću reći koju točno regiju preslikavanja smo pogodili, da se naslutiti da smo zahvatili dio između broja 9 i 0, s potencijalno brojem 8 između, te se vidi prijelaz u oblicima i neodređen dio u sredini.



Slika 4.9: Primjer šetnje skrivenim prostorom

Drugi način pronalaženja kolinearnih točaka je inicijalizacija jednog vektora šuma tako da mu svih 100 vrijednosti bude u sitnom intervalu, primjerice [-0.05, 0.05]. Tada taj vektor uzastopno dodajemo i oduzimamo na vektor [0 0 ... 0], odnosno on postaje pomak. Nasumičnošću postižemo pogađanje različitih regija mapiranja, za razliku od točnog određivanja svakog elementa vektora kojima šetamo. Svi ti vektori su kolinearni jer se razlikuju za faktor pomnožen s vektorom koji služi kao pomak. Slika 4.10 prikazuje rezultat ovakve šetnje. Pretpostavka je da je odabran dio preslikavanja broja 2 te prijelaz u broj 9, te između se nalazi regija neodređenosti.



Slika 4.10: Primjer druge šetnje skrivenim prostorom

4.3.2. Izvlačenje značajki

Izvlačenje značajki je interesantno područje za provjeru kvalitete reprezentacija koje pronalaze nenadzirani modeli. Ideja je da se model trenira nenadzirano, da se zatim iz modela izvlače značajke koje je naučio nenadzirano i stave kao ulaz linearnom klasifikatoru, te se tako različiti nenadzirani modeli uspoređuju po kvaliteti naučenih reprezentacija. Linearni klasifikatori se koriste radi svog limitiranog kapaciteta, stoga rezultati će dati pravu sliku o značajkama, dok kad bi koristili modele visokog kapaciteta nikad ne bi znali da li su za dobar rezultat zaslužne značajke ili sam klasifikator.

Model koji je korišten u ovom eksperimentu je model dva, koji je identičan prvom modelu osim što ima jedan potpuno povezan sloj više. Stoga njegovo preslikavanje vektora šuma ne ide u skriveni sloj dimenzija 100x6272, već u prvi sloj 100x1024, a zatim u drugi sloj veličine 1024x6272, te se taj sloj preoblikuje u tenzor na isti način kao što je opisan prvi model. Svi ostali hiperparametri te inicijalizacije su identični. Nakon što je model naučen, zadnji sloj pred klasifikaciju diskriminatora je izvučen te se na temelju njega učilo linearne klasifikatore. Model je u potpunosti naučen nenadzirano.

Eksperimenti su obavljeni nad cijelim skupom za treniranje baze MNIST, nad 100 primjera po klasi (1000 primjera sveukupno), te na 10 primjera po klasi (100 primjera sveukupno). Svaki eksperiment je ponovljen 10 puta, da bi se dobila srednja vrijednost i varijanca samog postupka. Mjerena metrika je točnost klasifikacije. Mjereni su i rezultati linearnih klasifikatora nad pikselima slike, odnosno bez izvlačenja značajki.

Osim usporedbe s linearnom klasifikacijom nad pikselima slike, usporedili smo model s još 2 modela. Za prvi model smo uzeli slučajno inicijalizirani diskriminator arhitekture koju koristimo i u DCGANu, te bez treniranja smo izvlačili njegovih 512 značajki i mjerili rezultate s linearnim klasifikatorima. Taj model bit će labeliran s RND. Za drugi model uzeli smo konvolucijsku neuronsku mrežu arhitekture diskriminatora te smo ju nadzirano trenirali s skupovima za učenje navedenima gore. Taj model nazvali smo Diskriminator.

Tablica 4.1: Rezultati mjerenja za Linearni SVM

Klasifikator	Cijeli MNIST	1000 primjera	100 primjera
LSVM (pikseli)	$91.75\% \pm 0\%$	$83.13\% \pm 0\%$	$72.2\% \pm 0\%$
LSVM (RND značajke)	$93.86\% \pm 0.48\%$	$83.55\% \pm 1.15\%$	$66.84\% \pm 1.7\%$
LSVM (DCGAN značajke)	$97.68\% \pm 0.21\%$	$92.99\% \pm 0.45\%$	$75.36\% \pm 2\%$
Diskriminator	$99.21\% \pm 0.07\%$	$91.14\% \pm 2.6\%$	$71.27\% \pm 1.64\%$

Tablica 4.2: Rezultati mjerenja za Logističku regresiju

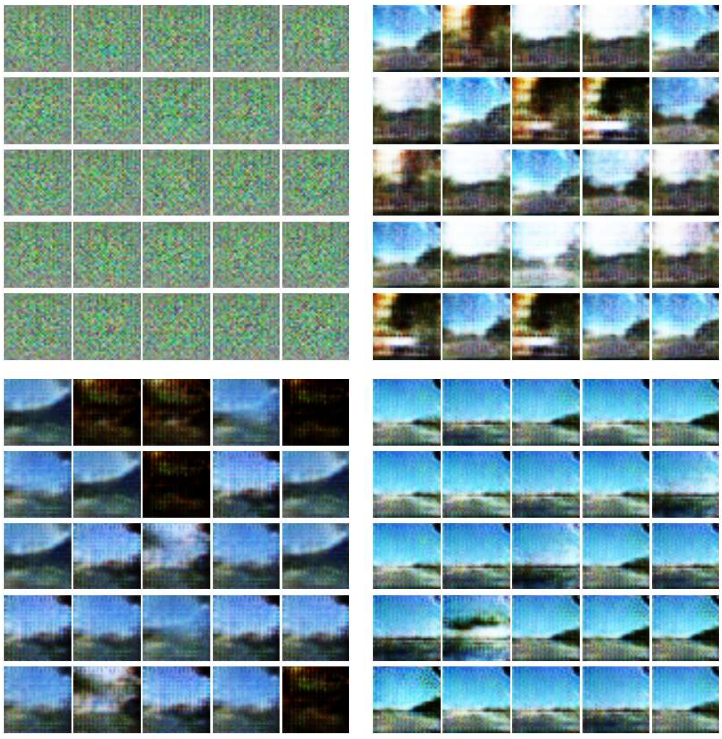
Klasifikator	Cijeli MNIST	1000 primjera	100 primjera
LR (pikseli)	$91.98\% \pm 0\%$	$85.85\% \pm 0\%$	$73.15\% \pm 0\%$
LR (RND značajke)	$93.98\% \pm 0.51\%$	$85.11\% \pm 1\%$	$67.69\% \pm 1.86\%$
LR (DCGAN značajke)	$97.93\% \pm 0.21\%$	$93.74\% \pm 0.44\%$	$74.16\% \pm 2.24\%$
Diskriminator	$99.21\% \pm 0.07\%$	$91.14\% \pm 2.6\%$	$71.27\% \pm 1.64\%$

Tablica 4.1 prikazuje rezultate mjerenja gdje je za linearni klasifikator odabran linearni SVM, a tablica 4.2 prikazuje rezultate mjerenja gdje je linearan klasifikator logistička regresija. Korištene su zadane implementacije logističke regresije i linearnog klasifikatora iz scikit-learn python paketa.

Rezultati govore da ukoliko imamo velik broj označenih primjera, da su standardne nadzirane metode ispravan put u rješavanju klasifikacijskih problema. No, za manje označene skupove za učenje vidljiv je napredak u klasifikaciji naspram strogo nadziranog modela. Stoga, kombinacija nadziranog i nenadziranog učenja je idealna za situacije kada baza ima mnogo primjera za učenje, no kad ih je mali dio označen.

4.3.3. Primjena na FM3 bazi slika

Četvrti model je učen na FM3 bazi slika. Promjene u arhitekturi su nezamjetne, jedan dekonvolucijski sloj je promijenjen s dekonvolucijskog prozora veličine 7x7 na 8x8, te posljedično i onaj poslije njega s 14x14x1 na 16x16x3 da bi se dobila rezultirajuća slika 32x32x3. Prvi sloj diskriminatora ovog puta prima 3 ulazna kanala umjesto jednog (slike su u boji, a ne u sivoj skali poput MNISTa). Korišteno je 3222 primjera za učenje, veličina grupe 64, te se model trenirao 2000 epoha, približno 100000 iteracija.



Slika 4.11: Uzorci nakon 0 (gore-lijevo), 25000 (gore-desno), 50000 (dolje-lijevo), 100000 (dolje-desno) iteracija

Na slici 4.11 su vidljivi artefakti, pojava pri preklapanju u dekonvoluciji [33]. Jedan od savjeta autora jest da se koriste dekonvolucijski prozori koji su višekratnici veličine koraka konvolucijskog prozora u generatoru, što je i istina za model u pitanju, no kako i oni kažu to nije garancija uklanjanja artefakata. Uklanjanje istih može biti smjernica za budući rad.

Model se koristio za izvlačenje značajki i usporedio pri klasifikaciji s običnim linearnim modelom. Klasifikacija je obavljena u *jedan protiv ostalih* shemi u kojoj odabranu klasu uzimamo kao pozitivne primjere, a sve ostale za negativne te vršimo binarnu klasifikaciju po svih 8 klasa. Tablica 4.3 prikazuje metriku točnosti, gdje stupci odgovaraju klasi koja je uzeta kao pozitivan primjer. Za linearni model smo koristili linearni SVM u oba slučaja, jednom nad pikselima slike, a drugi put nad izvučenim značajkama.

Tablica 4.3: Rezultati mjerenja točnosti

	0	1	2	3	4	5	6	7
pikseli	93.51%	94.94%	99.32%	99.44%	95.71%	97.80%	98.60%	98.70%
značajke	95.16%	94.63%	99.41%	99.81%	96.40%	98.85%	98.85%	98.66%

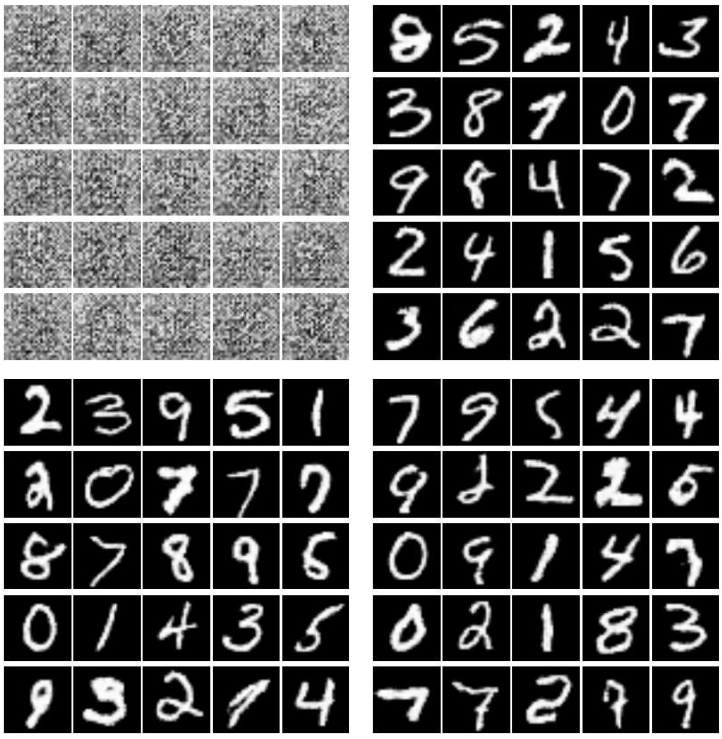
Rad sa značajkama je marginalno bolji, u nekim slučajevima možda i gori. Pretpostavljamo da je to do metrike točnosti, pošto je u ovom slučaju nesrazmjer u broju pozitivnih i negativnih primjera, stoga je istražena metrika prosječne preciznosti u Tablici 4.4.

Tablica 4.4: Rezultati mjerenja prosječne preciznosti

	0	1	2	3	4	5	6	7
pikseli	95.79%	69.55%	96.62%	80.65%	73.40%	29.62%	60.12%	61.37%
značajke	97.33%	68.90%	97.07%	92.35%	77.39%	64.50%	66.67%	61.89%

4.4. Klasifikatorska generativna suparnička mreža

Klasifikatorska generativna suparnička mreža koristi arhitekturu identičnu drugom modelu, osim što diskriminator ima 2 izlaza, za nadzirano i nenadzirano učenje. Cijeli model je strogo nadziran, odnosno bez modifikacija ne može raditi s neoznačenim primjerima. Za nadzirano učenje izlaz ima 10 *softmax* (praktički, višeklasna logistička regresija) neurona, za nenadzirano ima standardni sigmoidalni neuron. Slika 4.12 prikazuje generirane primjere. Naočigled, primjeri su puno više nalik stvarnima, što i ima smisla pošto generatorski prostor dijelimo i po oznakama, stoga je zadatak generatora uvelike olakšan.



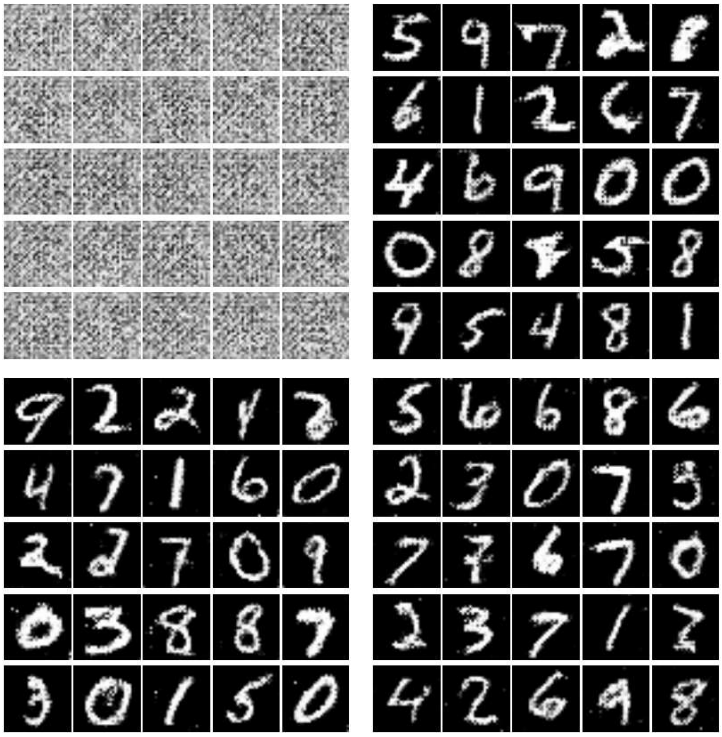
Slika 4.12: Uzorci nakon 0 (gore-lijevo), 25000 (gore-desno), 50000 (dolje-lijevo), 100000 (dolje-desno) iteracija

Provedeni su i klasifikacijski eksperimenti te su uspoređeni s zasebnom mrežom diskriminatora koja je učena označenim primjerima. Kako klasifikatorska generativna suparnička mreža ima nadzirani i nenadzirani gubitak, pretpostavka je da će pri optimizaciji nenadzirani gubitak štetiti nadziranom i obratno, stoga je proveden eksperiment gdje se težine diskriminatora uče otežano, odnosno faktor 0.9 se nalazi uz nadzirani gubitak, a faktor 0.1, uz nenadzirani, što bi optimizacijski postupak nagnalo da težine gura ka smanjivanju klasifikacijske greške. Rezultati su prikazani u tablici 4.5. Klasifikatorska generativna suparnička mreža se referira kao AC-GAN, model s otežanim klasifikacijskim gubitkom je Weighted AC-GAN, a Diskriminator označava zasebnu mrežu arhitekture diskriminatora.

Tablica 4.5: Rezultati mjerenja za klasifikatore

Klasifikator	Cijeli MNIST	1000 primjera	100 primjera
AC-GAN	98.56%	75.48%	16.4%
Weighted AC-GAN	99.21%	88.65%	22.13%
Diskriminator	99.2%	91.89%	75.4%

Pretpostavke su se ispostavile ispravnima, nenadzirani gubitak šteti klasifikaciji modela. Podrobniji eksperimenti nisu potrebni pošto se model nije pokazao superiornijim običnom klasifikatoru stoga je jedina prednost ovog modela bolji generirani uzorci te mogućnost uvjetovanja uzorkovanja, odnosno može se od modela zatražiti primjere odabrane klase. Slika 4.13 prikazuje generirane slike Weighted AC-GANa, očekivano su slabije kvalitete pošto je manji faktor uz nenadzirani gubitak.



Slika 4.13: Uzorci nakon 0 (gore-lijevo), 25000 (gore-desno), 50000 (dolje-lijevo), 100000 (dolje-desno) iteracija

5. Zaključak

Ubrzani razvoj na području dubokog učenja luči skoro pa svakodnevne napretke i sve širu primjenu. U ovom radu opisane su neuronske mreže kroz povijest te je objašnjena pojava dubokih modela neuronskih mreža, kao i svih najvažnijih pojmova vezanih za suvremene duboke neuronske mreže.

U sklopu rada implementirano je više inačica generativnih suparničkih mreža u radnom okviru *Tensorflow* s podrškom za CUDA procesiranje koji je otvoren za korištenje svima te se kontinuirano nadograđuje.

Generativne suparničke mreže su se pokazale kao jedan od najpopularnijih modernih modela čije nove inačice izlaze gotovo na tjednoj bazi. Ovaj rad pruža uvid u nekoliko najbitnijih takvih inačica, njihove arhitekture, hiperparametre i rezultate. Prikazana je i njegova primjena kao nenadziranog modela u nadziranoj klasifikaciji, gdje potencijalno pronalazi primjenu u polunadziranom učenju, kada na raspolaganju imamo mnogo primjera, ali malo označenih.

Odabir arhitektura generatora i diskriminatora se pokazao ključnim u uspješnosti učenja modela, stoga u budućnosti očekujemo stabilnije modifikacije modela koji su manje osjetljivi na arhitekturu te napredak po pitanju kriterija konvergencije modela.

LITERATURA

[1] Ilustracija klasifikatorske generativne suparničke mreže. https://www.slideshare.net/YunjeyChoi/generative-adversarial-networks-75916964. Accessed: 2017-07-10.

[2] Dcgan implementacije. https://github.com/carpedm20/DCGAN-tensorflow. Accessed: 2017-05-25.

[3] Opis primjene grupne normalizacije. https://wiki.tum.de/display/lfdv/Batch+Normalization,. Accessed: 2017-06-26.

[4] Konvolucija. https://graphics.stanford.edu/courses/cs178/applets/convolution.html,. Accessed: 2017-05-25.

[5] Ilustracija duboke konvolucijske generativne suparničke mreže. https://github.com/dmonn/GAN-face-generator. Accessed: 2017-07-10.

[6] Ilustracija generativne suparničke mreže. https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models- Accessed: 2017-07-10.

[7] Laplace filter. http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm, . Accessed: 2017-05-25.

[8] Primjer laplace filtra. https://docs.gimp.org/en/plug-in-laplace.html,. Accessed: 2017-05-25.

[9] Primjer sažimanja po maksimalnom odzivu. https://en.wikipedia.org/wiki/Convolutional_neural_network. Accessed: 2017-05-25.

[10] Višeslojni perceptron. https://www.researchgate.net/figure/287209604_fig4_Figure-4-Multilayer-perceptron-neural-network-mode Accessed: 2017-05-20.

[11] Primjer arhitekture konvolucijske neuronske mreže. http://deeplearning.net/tutorial/lenet.html. Accessed: 2017-05-25.

[12] Artificial neuron. https://en.wikipedia.org/wiki/Artificial_neuron. Accessed: 2017-05-15.

[13] Zglobnica. https://en.wikipedia.org/wiki/Rectifier_(neural_networks),. Accessed: 2017-05-16.

[14] Graf zglobnice. https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks, . Accessed: 2017-05-16.

[15] Sigmoidalna funkcija. http://whiteboard.ping.se/MachineLearning/BackProp. Accessed: 2017-05-16.

[16] Funkcija praga. https://en.wikipedia.org/wiki/Heaviside_step_function. Accessed: 2017-05-16.

[17] Tangens hiperbolni. https://www.researchgate.net/figure/260677487_fig12_Figure-13-a-ytanhx-b-Derivative-of-y. Accessed: 2017-05-16.

[18] Gan implementacije. https://github.com/wiseodd/generative-models/tree/master/GAN. Accessed: 2017-05-25.

[19] Xor problem. https://faculty.iiit.ac.in/~vikram/nn_intro.html. Accessed: 2017-05-16.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, i Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[21] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, i Yoshua Bengio. Generative adversarial nets. U *Advances in neural information processing systems*, stranice 2672–2680, 2014.

[23] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[24] Sergey Ioffe i Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] Diederik Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Yann LeCun i Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[27] Yann LeCun, Corinna Cortes, i Christopher JC Burges. The mnist database of handwritten digits, 1998.

[28] Yann A LeCun, Léon Bottou, Genevieve B Orr, i Klaus-Robert Müller. Efficient backprop. U *Neural networks: Tricks of the trade*, stranice 9–48. Springer, 2012.

[29] Jonathan Long, Evan Shelhamer, i Trevor Darrell. Fully convolutional networks for semantic segmentation. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, stranice 3431–3440, 2015.

[30] Warren S McCulloch i Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[31] Marvin Minsky i Seymour Papert. Perceptrons. 1969.

[32] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[33] Augustus Odena, Vincent Dumoulin, i Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[34] Augustus Odena, Christopher Olah, i Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[35] Alec Radford, Luke Metz, i Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[36] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[37] David E Rumelhart, Geoffrey E Hinton, i Ronald J Williams. Learning internal representations by error-propagation. U *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1*, svezak 1, stranice 318–362. MIT Press, Cambridge, MA, 1986.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, i Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[39] Ivan Sikiric, Karla Brkic, Josip Krapac, i Sinisa Segvic. Image representations on a budget: Traffic scene classification in a restricted bandwidth scenario. U *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, stranice 845–852. IEEE, 2014.

[40] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, i Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[41] Tijmen Tieleman i Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

Polunadzirana klasifikacija rukom pisanih znakova generativnim suparničkim modelima

Sažetak

U radu su opisane neuronske mreže, metode učenja neuronskih mreža te je dan uvid u najpoznatije moderne arhitekture, s naglaskom na generativnim suparničkim mrežama. Implementirano je više inačica spomenutog modela u *Tensorflow* radnom okviru te su modeli testirani na 2 skupa podataka, MNIST i FM3. Tablicama i slikama su prikazani klasifikacijski i generativni rezultati. Model se pokazao korisnim u polunadziranom okruženju, kada služi za izvlačenje značajki u nenadziranom okruženju, te prosljeđivanju istih nadziranom modelu.

Ključne riječi: Neuronske mreže, generativne suparničke mreže, polunadzirana klasifikacija

Semisupervised classification of handwritten characters using generative adversarial models

Abstract

This paper describes neural networks, methods on training neural networks and it provides an insight to the most famous modern architectures, with emphasis on generative adversarial networks. A few versions of the aforementioned model have been implemented in *Tensorflow* framework, with models being trained on 2 datasets, MNIST and FM3. Tables and figures provide insight to classification and generative results. Model has proven useful in a semisupervised environment, when a large dataset is available but only a fraction of it has been labeled.

Keywords: Neural networks, generative adversarial networks, semisupervised classification