

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1760

Stvaranje slika pomoću dubokog učenja

Juraj Šušnjara

Zagreb, lipanj 2018.

Umjesto ove stranice umetnите izvornik Vašeg rada.

Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.

Zahvaljujem se doc. dr. sc. Marinu Šiliću na mentorstvu i savjetima pruženim pri izradi ovog rada. Posebno se zahvaljujem svojoj obitelji na velikoj podršci tijekom svih ovih godina.

SADRŽAJ

1. Uvod	1
2. Duboko Učenje	3
2.1. Konvolucijske neuronske mreže	4
2.2. Generativni modeli	5
2.2.1. Varijacijski autoenkoder	6
2.2.2. Generativne suparničke mreže	9
2.2.3. Usporedba GAN i VAE	11
3. Radni okviri za duboko učenje	13
3.1. TensorFlow	13
3.2. CUDA	16
4. Postojeći modeli za stvaranje slika	17
4.1. Duboke konvolucijske generativne suparničke mreže	17
4.2. Napredne metode treniranja generativnih suparničkih mreža	19
4.3. Uvjetne generativne suparničke mreže	20
4.3.1. Skup podataka MNIST	20
4.3.2. Automatsko stvaranje oznaka za slike	21
5. Programsко ostvarenje i vrednovanje	23
5.1. Skup podataka MNIST	24
5.1.1. Varijacijski autoenkoder	25
5.1.2. Uvjetni varijacijski autoenkoder	27
5.1.3. Generativne suparničke mreže	27
5.2. Skup podataka CIFAR-10	30
5.2.1. Varijacijski autoenkoder	31
5.2.2. Generativne suparničke mreže	31

5.3.	Skup podataka LFW	32
5.3.1.	Varijacijski autoenkoder	32
5.3.2.	Generativne suparničke mreže	34
6.	Zaključak	35
	Literatura	37

1. Uvod

Umjetna neuronska mreža je model strojnog učenja nadahnut ljudskim mozgom. Razvojem računalnog sklopolja, pokazala se kao dobar model za izvršavanje raznih zadataka koje je teško izravno riješiti. Najčešći način korištenja neuronske mreže je treniranje takve mreže s označenim podatcima. To se zove nadzirano učenje jer je izravno navedeno što je izlaz za svaki ulaz. Nedostatak takvog učenja je što podatci za treniranje trebaju imati označene izlaze, koje najčešće označavaju ljudi. Nenadzirano učenje, s druge strane, ne zahtijeva oznake kako bi se istrenirao model.

Generativni modeli pripadaju nenadziranom učenju. Takvi modeli mogu naučiti razdiobu podataka za treniranje te stvarati slične podatke. Jedna od primjena generativnih modela je stvaranje slika. Moguće je napraviti neuronsku mrežu, koja će, na temelju slika s kojima je trenirana, naučiti stvarati slične slike. Takav zadatak često rezultira slikama koje izgledaju nestvarno i umjetno. To je očekivano jer se mreže uobičajeno treniraju kako bi se maksimizirala prepoznatljivost predmeta. Koristeći statističke modele za učenje oblika i uzorka na slikama omogućuje mreži da razumije duboke strukture slika. Google-ov *Deep Dream* generator [12] je primjer uporabe generativnih modela. Na slici 1.1 je prikazan jedan od rezultata. Treniranje mreže za stvaranje slika koje izgledaju prirodno predstavlja težak problem za klasične modele neuronskih mreža.

Postoji više modela koji mogu naučiti stvarati slike. Najpoznatiji su u varijacijski autoenkoder (engl. *Variational Autoencoder*, VAE) i generativne suparničke mreže (engl. *Generative Adversarial Networks*, GAN). VAE predstavlja varijaciju klasičnog autoenkodera koji je prikladan za stvaranje podataka. GAN modeli doživjeli su popularnost kroz zadnje dvije godine i predstavljaju jedan od najkreativnijih načina treniranja neuronskih mreža. Pokazali su dobre empirijske rezultate na generativnim zadatcima.

Cilj ovog rada je proučiti postojeće generativne modele za stvaranje slika i uspješno programski ostvariti vlastite modele. Potrebno je proučiti VAE i GAN modele. Kako bi se detaljno istražili i ocijenili iskorišteno je nekoliko različitih skupova podataka:

MNIST [19], LFW [14] i CIFAR-10 [5]. Potrebno je objasniti dobivene rezultate, usporediti dva modela, zaključiti nad kakvim su skupovima podataka primjenjivi i procijeniti kolike su njihove mogućnosti u zadatku stvaranja slika.



Slika 1.1: Slika stvorena Google-ovim *Deep Dream* generatorom

2. Duboko Učenje

Strojno učenje koristi računalnu moć za izgradnju modela koji uče iz postojećih podataka kako bi predviđeli buduće ponašanje i ishode. Duboko učenje pripada u skupinu algoritama strojnog učenja gdje su modeli, koji su nadahnuti načinom na koji mozak funkcioniра, izraženi matematički. Parametri koji definiraju takve modele uče se iz podataka. Duboko učenje je ključan faktor tehnologija umjetne inteligencije.

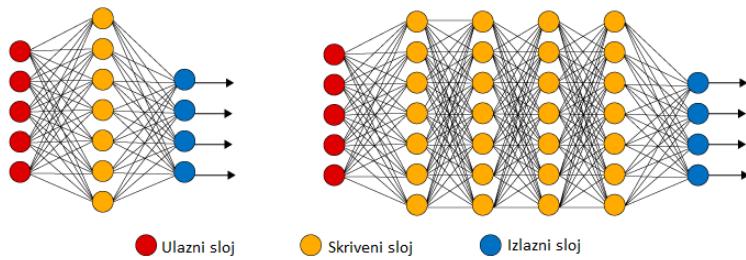
Algoritmi dubokog učenja transformiraju ulaze kroz mnogo slojeva pa se zbog toga i zove duboko učenje. U svakom sloju, osnovna jedinica, umjetni neuron čiji se parametri mogu naučiti treniranjem, transformira ulazne signale. Dubina neuronske mreže je broj skrivenih slojeva. Ne postoji strogo definirana granica broja slojeva koje bi učenje svrstalo u duboko učenje ali najčešće se to smatra barem dva nelinearna sloja.

Algoritmi dubokog učenja zasnivaju se na raspodijeljenim prikazima. Osnovna pretpostavka iza raspodijeljenih prikaza je da se promatrani podatci stvaraju interakcijama faktora organiziranih u slojeve. Svaki sloj odgovara određenoj razini apstrakcije. Mijenjanje broja i veličine slojeva može se koristiti kako bi se pružili drugačiji oblici apstrakcije. Kod dubokog učenja viši se slojevi apstrakcije uče od nižih. Na temelju tih apstrakcija može se odabratkoje su značajke korisne za učenje.

Postoji mnogo različitih arhitektura dubokih mreža te je svaka od njih predviđena za rješavanje određene skupine problema. Duboke neuronske mreže (engl. *Deep Neural Networks*, DNN) pripada u skupinu umjetnih mreža (engl. *Artificial Neural Networks*, ANN) s više skrivenih slojeva. DNN su takozvane unaprijedne mreže što znači da se ulazni podatci kroz svaki sloj provuku samo jednom i propagiraju dalje naprijed. Postoje i povratne neuronske mreže (engl. *Recurrent Neural Networks*, RNN) kod kojih izlazi iz jednog sloja mogu biti opet ulazi u taj isti ili čak neki prethodni sloj. Takva arhitektura pogodna je za modeliranje jezika, prepoznavanje govora, pisanih znakova itd. Za analizu slike i probleme računalnog vida najpogodnije su konvolucijske neuronske mreže (engl. *Convolutional Neural Networks*, CNN) o kojima će se više govoriti u ovom radu.

Na slici 2.1 pojmovni je prikaz neuronske mreže. Ulazna informacija (crveno)

mijenja se kroz skrivene slojeve (žuto) kako bi se dobila izlazna informacija (plavo). Mreža zapravo predstavlja sastav mnogo različitih funkcija. Model sa slike predstavlja usmjereni aciklički graf koji opisuje kako su funkcije povezane. Na primjer, funkcije $f^{(1)}$, $f^{(2)}$ i $f^{(3)}$ povezane u lanac čine funkciju $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Ovakvi lanci najčešće su strukture neuronskih mreža. U ovom slučaju $f^{(1)}$ je prvi sloj, $f^{(2)}$ drugi itd. Tijekom treniranja neuronskih mreža cilj je postići da funkcija mreže odgovara $f^*(\mathbf{x})$ koja je vrednovana u različitim točkama za treniranje. Svaki ulaz \mathbf{x} povezan je s oznakom $y \approx f^*(\mathbf{x})$. Podatci za treniranje govore kakvu vrijednost izlazni sloj mora poprimiti za svaki \mathbf{x} . Ponašanje ostalih slojeva nije izravno povezano s podacima za treniranje. Algoritam učenja treba odlučiti kako koristiti te slojeve da bi se dobila najbolja moguća procjena izlaza.



Slika 2.1: Neuronska mreža

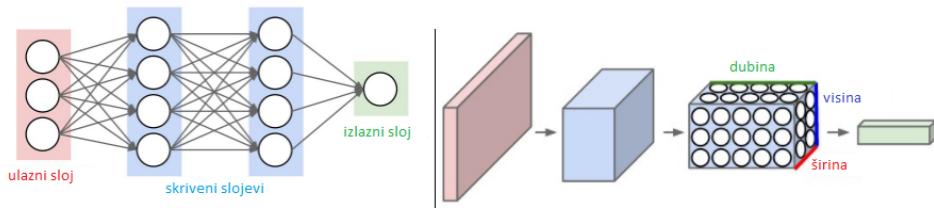
2.1. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže su veoma slične klasičnim neuronskim mrežama. Sastoje se od slojeva od kojih svaki ima određen broj neurona koji imaju parametre koji se mogu naučiti. Svaki neuron prima neki ulaz, računa skalarni produkt i provlači rezultat kroz neku nelinearnu funkciju. Razlika od klasičnih mreža je to što se sada radi izravna pretpostavka da su ulazi u mrežu slike. Zbog toga se u arhitekturu mreže mogu ubaciti određena svojstva prikladna za obradu slike. Tako se može smanjiti broj parametara te ubrzati proces provlačenja podataka kroz mrežu.

Slojevi konvolucijske neuronske mreže organizirani su u 3 dimenzije: širina, visina, dubina. Na primjer, slike iz CIFAR-10 [5] skupa podataka imaju dimenzije 32x32x3 (visina, širina, RGB komponenta). Na slici 2.2 uspoređene su klasična i konvolucijska neuronska mreža.

Konvolucijski sloj osnovna je jedinica ovakvih mreža. Taj sloj sastoji se od niza filtra koje treba naučiti. Svaki filter je prostorno malen ali se produži kroz punu du-

binu podataka. Uobičajeni filter prvog sloja konvolucijske mreže ima dimenzije 5x5x3. Tijekom računanja idućeg sloja, svaki filter klizi (konvoluiru) preko širine i visine podataka te računa skalarni produkt između vrijednosti filtra i podataka na trenutnoj poziciji. Tako će mreža naučiti filtre koji će se aktivirati kada vide neko slikovito svojstvo poput određenog ruba ili boje u prvom sloju ili neke složenije uzorku u sljedećim slojevima mreže. Svaki filter stvara 2D aktivacijsku mapu koje se poslože po dimenziji dubine da se dobiju vrijednosti novog sloja kojemu je dubina jednaka broju filtra prethodnog sloja. Konvolucijske mreže koriste tri osnovne ideje: rijetku povezanost, dijeljenje parametara i ekvivariantnost prikaza. Osim konvolucijskog, u ovakve mreže često se umeće takozvani *pooling* sloj koji smanjuje prostorne dimenzije pa tako i broj parametara i računanja u mreži. Tako se sprečava i pretreniranje. Normalizacijski sloj služi kako bi se ubrzalo učenje jer se smanjuje kovarijacijski pomak. Konvolucijske mreže često imaju i klasične potpuno povezane slojeve.



Slika 2.2: Lijevo: obična troslojna neuronska mreža. Desno: CNN, neuroni su posloženi u 3 dimenzije, svaki sloj transformira 3D ulaz u 3D izlaz aktivacija neurona. U ovom primjeru crveni ulazni soj predstavlja sliku tako da su širina i visina dimenzije slike, a dubina su RGB komponente.

2.2. Generativni modeli

Generativni model moćan je način za naučiti bilo kakvu razdiobu podataka koristeći nenadzirano učenje. Doživjeli su uspjeh u posljednjih nekoliko godina. Svim tipovima generativnih modela cilj je naučiti pravu razdiobu podataka za treniranje kako bi mogli stvarati nove podatke uz neke promjene. Nije uvijek moguće naučiti točnu razdiobu podataka tako da je dovoljno pokušati modelirati razdiobu koja je što sličnija pravoj. Zbog toga se može iskoristiti moć neuronskih mreža kako bi se naučila funkcija koja će približno odgovarati stvarnoj. Dva najčešće korištena i najučinkovitija pristupa su Varijacijski Autoenkoder (engl. *Variational Autoencoder*, VAE) i Generativne Suparničke Mreže (engl. *Generative Adversarial Networks*, GAN). Cilj VAE je mak-

simizirati donju granicu log-izglednosti podataka dok je cilj GAN-a postići ravnotežu između generatora i diskriminadora. Oba pristupa detaljnije su opisana u [15].

2.2.1. Varijacijski autoenkoder

Autoenkoder se koristi kako bi se podatci šifrirali na prikaz s puno manje dimenzija. Takav prikaz može pohraniti skrivene informacije o razdiobi ulaznih podataka. Kod klasičnog autoenkodera, šifrirani vektor samo može biti mapiran na odgovarajući ulaz koristeći dekoder. Ne može se iskoristiti za stvaranje novih, sličnih podataka s nekim promjenama.

Kako bi se to postiglo, model treba naučiti razdiobu podataka za treniranje. VAE je jedan od poznatijih pristupa za učenje složene razdiobe podataka, kao što su slike, koristeći neuronske mreže i nenadzirano učenje. To je vjerojatnosni grafički model kojem je cilj naučiti osnovnu razdiobu podataka za treniranje kako bi mogao uzorkovati nove podatke iz naučene razdiobe. Ideja je naučiti nisko-dimenzionalni skriveni prikaz podataka, takozvane latentne ili skrivene varijable. To su varijable koje nisu izravno promatrane nego su zaključene preko matematičkog modela. Može se pretpostaviti da su podatci za treniranje stvoreni iz tih varijabli. Skrivene varijable mogu čuvati korisne informacije o vrsti izlaza kojeg model treba stvoriti. Razdioba skrivene varijable z označena je s $P(z)$. Normalna razdioba je odabrana kao početna pretpostavka (a priori) kako bi se naučila razdioba $P(z)$. Odabrana je jer se tako s lakoćom mogu uzorkovati novi podatci tijekom vremena zaključivanja.

Osnovni cilj sada je oblikovati podatke s parametrima koji maksimiziraju izglednost podataka za treniranje X . Prepostavlja se da je nisko-dimenzionalni skriveni vektor stvorio podatke $x \in X$ i da se skriveni vektor može mapirati na podatak x koristeći determinističku funkciju $f(z; \Theta)$ parametriziranu s Θ koji treba odrediti. Tijekom ovog postupka potrebno je maksimizirati vjerojatnost svakog podatka u X kao što je prikazano u izrazu 2.1:

$$P_\Theta(X) = \int P_\Theta(X, z) dz = \int P_\Theta(X|z) P_\Theta(z) dz \quad (2.1)$$

U izrazu 2.1 $f(z; \Theta)$ je zamijenjen razdiobom $P_\Theta(X|z)$. Intuicija iza maksimalne izglednosti je da ako model može stvarati uzorke za treniranje iz skrivenih varijabli onda može i stvarati slične uzorke s nekim promjenama. Drugim riječima, ako se uzorkuje velik broj skrivenih varijabli iz $P(z)$ i stvori x iz tih varijabli, onda stvoreni x treba odgovarati razdiobi $P_{data}(x)$. Ostala su još dva pitanja za odgovoriti. Kako uhvatiti razdiobu skrivenih varijabli i kako uklopiti izraz 2.1 za sve dimenzije z ?

Teško je ručno zadati bitne informacije za šifriranje u skrivene varijable za stvaranje izlaznih slika. Bolje se osloniti na neuronske mreže za računanje z , s prepostavkom da skriveni vektor može približno odgovarati normalnoj razdiobi kako bi se kasnije moglo lako uzorkovati. Ako postoji normalna razdioba od z u n-dimenzionalnom prostoru onda je uvijek moguće stvoriti bilo kakvu razdiobu koristeći dovoljno složenu funkciju dok inverz takve funkcije može biti korišten kao sama skrivena varijabla.

U izrazu 2.1 integral je preko svih dimenzija z i teško ga je izračunati. Moguće ga je izračunati koristeći metode Monte-Carlo integracije što nije jednostavno programski ostvariti. Zato je potrebno osmisliti novi, bolji pristup za približnu maksimizaciju $P_\Theta(X)$ iz izraza 2.1. Ideja VAE je zaključiti $P(z)$ koristeći $P(z|X)$ koji nije poznat. $P(z|X)$ se zaključuje koristeći varijacijsko zaključivanje što je u osnovi optimizacijski problem u Bayesovoj statistici. Prvo se oblikuje $P(z|X)$ koristeći jednostavniju razdiobu $Q(z|X)$ koju je lako pronaći. Nakon toga pokuša se minimizirati razlika između $P(z|X)$ i $Q(z|X)$ pomoću KL-divergencije [6] da bi prepostavka bila što bliže pravoj razdiobi. Konačna ciljna funkcija VAE navedena je u izrazu 2.2.

$$\log P(X) - D_{KL}[Q(z|X)||P(z|X)] = E[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)] \quad (2.2)$$

Iraz 2.2 ima sljedeće značenje. $Q(z|X)$ je u osnovi mreža enkoder, z je šifrirani prikaz podataka $x \in X$ i $P(X|z)$ je mreža dekoder. Cilj izraza je maksimizirati log-izglednost razdiobe podataka uz neku pogrešku danu s $D_{KL}[Q(z|X)||P(z|X)]$. Vidi se da VAE pokušava minimizirati donju granicu od $\log P(X)$ dok $P(z|X)$ nije izračunljiv, ali je izraz KL-divergencije ≥ 0 . To je isto kao i maksimiziranje $E[\log P(X|z)]$ i minimiziranje $D_{KL}[Q(z|X)||P(z|X)]$. Poznato je da je maksimiziranje $E[\log P(X|z)]$ procjena najveće izglednosti i da je oblikovano pomoću mreže dekodera. Poželjno je da skriveni prikaz bude blizu normalnoj razdiobi pa se $P(z)$ prepostavlja kao $N(0, 1)$. S tom prepostavkom $Q(z|X)$ također treba biti blizak toj razdiobi. Ako prepostavimo da je to normalna razdioba s parametrima $\mu(X)$ i $\Sigma(X)$, pogreška s obzirom na razliku između te dvije razdiobe ($P(z)$ i $Q(z|X)$) se može napisati izrazom 2.3.

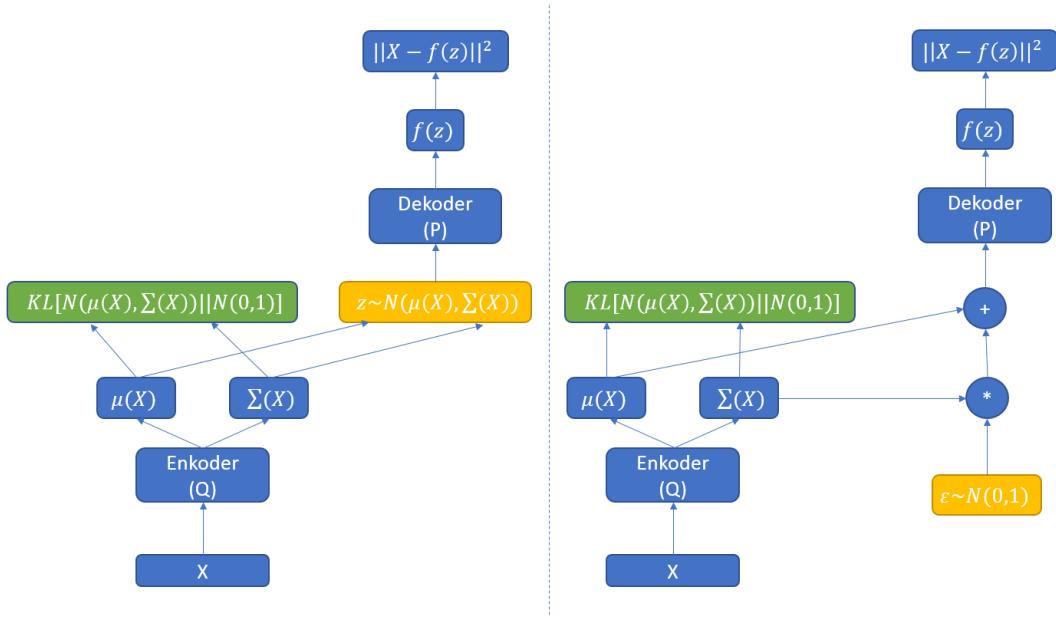
$$D_{KL}[N(\mu(X), \Sigma(X))||N(0, 1)] = \frac{1}{2} \sum_k (\exp(\Sigma(X)) + \mu^2(X) - 1 - \Sigma(X)) \quad (2.3)$$

S obzirom na to da se optimizira donja varijacijska granica, optimizacijska funkcija je:

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

gdje je rješenje druge funkcije u izrazu 2.3.

Stoga funkcija gubitka sadrži dva izraza. Prvi je rekonstrukcija gubitka ulaza prema izlazu i drugi je gubitak u obliku KL-divergencije. Sada je moguće trenirati mrežu algoritmom propagacije greške unatrag (engl. *backpropagation*). Problem je što prvi izraz ne ovisi samo o parametrima razdiobe P nego i o parametrima razdiobe Q . Ta se ovisnost ne pojavljuje u gornjoj jednadžbi. Dakle, kako propagirati unatrag kroz sloj gdje se z slučajno uzorkuje iz razdioba $Q(z|X)$ ili $N[\mu(X), \Sigma(X)]$ tako da P može dešifrirati. Gradijenti se ne mogu računati u slučajnim čvorovima. Za to se koristi reparametrizacijski trik kako bi se mreža učinila diferencijabilnom. Uzorkuje se iz $N[\mu(X), \Sigma(X)]$ tako da se prvo uzorkuje $\varepsilon \sim N(0, 1)$ nakon čega se računa $z = \mu(X) + \Sigma^{\frac{1}{2}}(X) * \varepsilon$. To je opisano i na slici 2.3. Prolazak unaprijed je jednak za obje mreže sa slike (lijeva i desna) ali gradijenti se mogu propagirati unatrag samo kroz desnu mrežu.



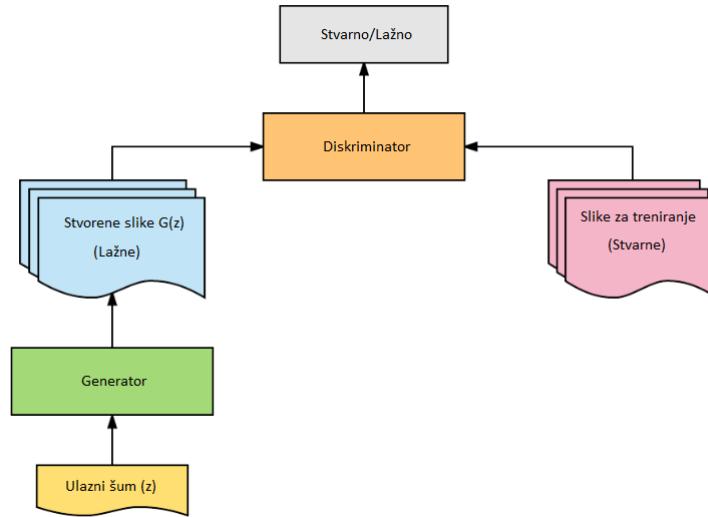
Slika 2.3: Reparametrizacijski trik korišten kako bi se omogućila propagacija greške unatrag kroz slučajne čvorove.

Tijekom konačnog uzorkovanja može se jednostavno uzorkovati $z \sim N(0, 1)$ i taj z provući kroz dekoder kako bi se dobio novi podatak. S obzirom na to da se optimizira donja varijacijska granica, kvaliteta stvorene slike će biti nešto lošija u usporedbi s *state-of-the art* pristupu poput GAN-ova (generativne suparničke mreže).

2.2.2. Generativne suparničke mreže

Generativne suparničke mreže jedan su od najzanimljivijih načina za treniranje mreže te su u zadnje vrijeme dosegle veliku popularnost. Proizvode izvrsne rezultate. Suparničko treniranje je promijenilo način na koji se uče neuronske mreže za izvršavanje nekog zadatka. GAN-ovi ne rade izravnu pretpostavku razdiobe kao VAE. Umjesto toga, osnova je uzeta iz teorije igara s ciljem pronašlaska *Nash* ravnoteže [16] između dvije mreže, generatora i diskriminadora. Ideja je uzorkovati iz jednostavne razdiobe poput normalne i naučiti preobraziti šum u razdiobu podataka za treniranje koristeći funkciju procjene kao što je neuronska mreža.

To se postiže suparničkim treniranje dviju mreža. Model generator G uči kako prepoznati razdiobu podataka dok model diskriminator D procjenjuje vjerojatnost da je neki uzorak došao iz razdiobe podataka umjesto iz razdiobe modela G . U osnovi, zadatak generatora je stvarati slike koje izgledaju prirodno, dok je zadatak diskriminadora odlučiti je li slika lažna ili stvarna. Na to se može gledati kao *min-max* igra između dva igrača gdje se učinkovitost oba poboljšava tijekom vremena. U ovoj igri, generator pokušava prevariti diskriminator tako da stvara što stvarnije slike dok ga diskriminator pokušava razotkriti tako da poboljšava svoju mogućnost diskriminacije stvarnih i lažnih slika. Na slici 2.4 prikazana je osnovna arhitektura GAN-ova.



Slika 2.4: Generativan suparnički model

U početku se definiraju varijable ulaznog šuma $P(z)$. Generator to mapira na razdiobu podataka koristeći složene diferencijabilne funkcije s parametrima Θ_g . Nadalje, druga mreža, diskriminator na ulaz uzima x te koristi drugu diferencijabilnu funkciju s

parametrima Θ_d . Izlaz joj je skalarna vrijednost koja označava vjerojatnost da x dolazi iz stvarne razdiobe podataka $P_{data}(x)$. Ciljna funkcija GAN-a je navedena u izrazu 2.4.

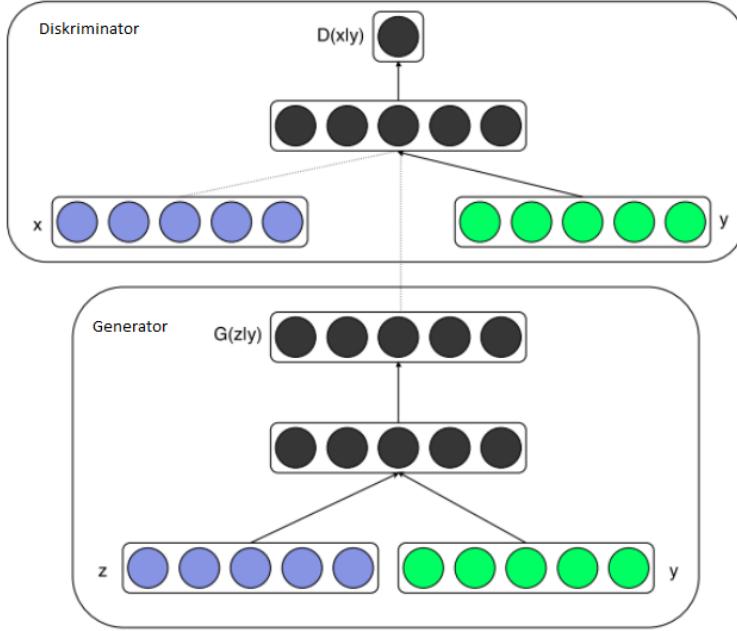
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (2.4)$$

U izrazu 2.4., ako ulaz diskriminatora dolazi iz stvarne razdiobe podataka onda izlaz $D(x)$ treba biti 1 kako bi se maksimizirala funkcija D . S druge strane, ako je ulaz stvorio generator onda $D(G(z))$ treba biti 1 kako bi se minimizirala funkcija G . Problem kod optimiziranja generatora je što na početku treniranja generator još ne zna ništa pa su gradijenti najčešće veoma mali, dok su kasnije, kada je dobro naučen, preveliki. Bilo bi bolje da je suprotno. Stoga je bolje maksimizirati $\mathbb{E}[\log(D(G(z)))]$ umjesto minimizirati $\mathbb{E}[\log(1 - D(G(z)))]$. Postupak treniranja sastoji se od izmjenične primjene stohastičkog gradijentnog spusta na diskriminatoru i generatoru. Treniranje prestaje kada diskriminator više ne može razlikovati stvarne od lažnih podataka ($D(x, \Theta_d) = \frac{1}{2}$).

Jedan od najranijih GAN modela s konvolucijskom mrežom je duboka konvolucijska generativna neuronska mreža (engl. *Deep Convolutional Generative Adversarial Networks*, DCGAN). Ovakva mreža na ulaz prima n slučajnih brojeva iz uniformne razdiobe dok je izlaz slika željenih dimenzija. Mreža se sastoji od mnogo konvolucijskih, dekonvolucijskih i potpuno povezanih slojeva. Normalizacija grupe (engl. *batch normalization*) koristi se kako bi se stabiliziralo treniranje. Aktivacijska funkcija zglobnica se koristi u generatoru za sve slojeve osim za izlazni gdje se koristi tanh sloj. Mreža se trenira s mini-grupnim stohastičkim gradijentnim spustom. Pomoću Adam optimizatora [8] ubrzava se učenje. Nadalje, pokazano je da se i aritmetikom vektora z može na zanimljive načine upravljati slikama koje se stvaraju.

Najčešće korištena varijacija GAN-a je uvjetni GAN kojemu je uz vektor šuma dodan još uvjetni vektor. Uvjetni GAN omogućuje stvaranje slika sa željenim svojstvima. Pitanje je postoji li način za ubaciti dodatne informacije o tipu slike u model. Uvjetni GAN (engl. *Conditional GAN*, cGAN) to omogućuje. Uvjetovanjem modela s dodatnom informacijom koja se daje generatoru i diskriminatoru moguće je usmjeriti postupak stvaranja podataka. cGAN se koristi za razne zadatke poput stvaranje slika iz teksta, pretvaranje slike u drugu sliku, automatsko označavanje slika itd. Struktura obje mreže cGAN-a prikazana je na slici 2.5.

Jedna od dobrih strana GAN-a je što se može trenirati i s malim skupom podataka. Iako su rezultati obećavajući, postupak treniranja nije trivijalan zbog podešavanja hi-



Slika 2.5: Uvjetni GAN (*cGAN*)

perparametara mreže. Teško ih je optimizirati jer ne konvergiraju lako. Postoje trikovi koji pomažu u tom zadatku, navedeni su u [7]. Nadalje, ne postoji kriterij za kvantitativnu procjenu rezultata osim da ljudi ručno provjeravaju izgledaju li slike stvarno ili ne.

2.2.3. Usporedba GAN i VAE

Problem kod VAE modela je to što je potrebno pamtitи sve detalje u skrivenom prostoru z kako bi se dobila dobra procjena razdiobe izvornih slika $P(X)$. Nadalje, procjena $P(z|X)$ je najčešće previše pojednostavljena jer je nemoguće parametrizirati složenu razdiobu. Zbog toga je teško naučiti stvarati slike s više detalja. Uvođenje razdiobe $Q(z|X)$ podrazumijeva da model nije u stanju naučiti stvarnu razdiobu. Cilj VAE modela je oblikovanje skrivene varijable z . Takvi modeli mogu rezultirati dobrim poopćavanjem iako su najbolji kada je z u prvom planu. Mogućnost izravnog postavljanja bilo kakvih razdioba $Q(z|X)$ prije treniranja doprinosi treniranju u slučajevima kada je problem bolje poznat te se zna kako bi z trebao izgledati.

GAN modeli na pate od takvih problema zbog toga što se generator izravno tjera da stvara uzorke koji izgledaju stvarno. Cijena takvog pristupa je da generator ponekad nauči stvarati sve uzorke jednake, koji dolaze iz samo jednog dijela stvarne razdiobe. Drugi problem GAN modela je vrednovanje. Ne postoji jedinstvena objektivna mjera s kojom bi se ocijenio takav model. Postoje neke naprednije metode treniranja koje

mogu pomoći u tome. Spomenute su u 4.2. Dok VAE modeli ponekad stavljaju sredinu razdiobe z na mesta koja nemaju smisla, GAN modeli ponekad u potpunosti promaše tu razdiobu. To je vidljivo u MNIST rezultatima u 5.1.3. Bez obzira na to GAN modeli su pokazali izuzetno dobre empirijske rezultate.

3. Radni okviri za duboko učenje

U ovom poglavlju opisani su radni okviri i tehnologije korištene za izradu programskog dijela rada.

3.1. TensorFlow

TensorFlow je biblioteka otvorenog koda za numeričke izračune koristeći grafove protoka podataka. Čvorovi grafa predstavljaju matematičke operacije, dok rubovi grafa predstavljaju multidimenzionalne podatke (tenzore) koji teku između čvorova. Fleksibilna arhitektura omogućuje razvijanje programa koji se mogu pokretati na CPU ili GPU na desktop računalima, poslužiteljima ili mobilnim uređajima bez mijenjanja koda. TensorFlow uključuje i TensorBoard, alat za slikoviti prikaz podataka. TensorFlow je izvorno razvio *Google Brain* unutar Google-ove organizacije *Machine Intelligence Research* u svrhu istraživanja strojnog učenja i dubokih neuronskih mreža. Sustav je dovoljno opširan za primjenu nad brojnim drugim domenama također. TensorFlow pruža API za Python, C i C++ od kojih je onaj za Python najbolje dokumentiran i najčešće korišten. Za potrebe ovog rada korišten je TensorFlow-ov Python API.

Kod naveden u ovom poglavlju prikazuje izgradnju jednostavne troslojne neuronske mreže kojoj je cilj naučiti klasificirati slike znamenaka iz MNIST skupa podataka. Arhitektura mreže je 784x512x1024x10. Ulaz u mrežu su 28x28x1 slike preoblikovane u 784-dimenzionalan vektor dok je izlaz 10-dimenzionalan vektor vjerojatnosti. Svaki element izlaznog vektora pokazuje kolika je vjerojatnost da je ulazni podatak slika određene znamenke. Nulti indeks odgovara znamenki 0, prvi znamenci 1 itd.

Program koji koristi TensorFlow uvijek se sastoji od dvije odvojene faze:

- oblikovanje računskog grafa
- zadavanje računskih upita

Čvorovi grafa mogu biti konstante (*tf.constant*), ulazni podatci (*tf.placeholder*), varijable (*tf.Variable*) i tensorski izrazi. Time se oblikuje računski graf. Računski upiti

se zadaju pozivom metode *run* izvedbenog konteksta *session*. Prvi argument metode *run* predstavlja listu čvorova koje treba izračunati. Drugi argument predstavlja rječnik s vrijednostima ulaznih podataka koje će biti iskorištene tijekom izračuna. Povratna vrijednost metode *run* je n-torka vrijednosti čvorova koji su navedeni u prvom argumentu.

Učitavanje TensorFlow biblioteke i MNIST skupa podataka obavlja se slijedećim kodom.

```
1 import tensorflow as tf
2 from tensorflow.examples.tutorials.mnist
3     import input_data
4
5 mnist = input_data.read_data_sets(
6     "MNIST_data/", one_hot=True)
```

Na početku je potrebno definirati čvorove varijable mreže i ulazne podatke. Sljedećim kodom definiraju se inicijalizatori za W i b matrice neuronske mreže. W su težine koje povezuju neurone susjednih slojeva i inicijaliziraju se prema normalnoj razdiobi sa sredinom 0 i devijacijom 0.02. b je varijabla pristrandosti (engl. *bias*) i inicijalizira se na 0.

```
1 w_init = tf.truncated_normal_initializer(
2     mean=0, stddev=0.02)
3 b_init = tf.constant_initializer(0)
```

x i y predstavljaju ulazne podatke gdje *None* u dimenzijama predstavlja proizvoljnu veličinu mini-grupa kojim će se trenirati mreža.

```
1 x = tf.placeholder(tf.float32, shape=(None, 784))
2 y = tf.placeholder(tf.int32, shape=(None, 10))
```

Idućim računskim upitima definiraju se operacije koje predstavljaju neuronsku mrežu. Zglobnica se koristi kao aktivacijska funkcija na svakom od skrivenih slojeva. Aktivacija izlaznog sloja je *softmax* funkcija zbog toga što se radi o klasifikaciji za više od dvije kategorije.

```

1   w1 = tf.get_variable(
2           'w1', [784, 512], initializer=w_init)
3   b1 = tf.get_variable('b1', [512], initializer=b_init)
4   h1 = tf.nn.relu(tf.matmul(x, w1) + b1)
5
6   w2 = tf.get_variable(
7           'w2', [512, 1024], initializer=w_init)
8   b2 = tf.get_variable('b2', [1024], initializer=b_init)
9   h2 = tf.nn.relu(tf.matmul(h1, w2) + b2)
10
11  w3 = tf.get_variable(
12      'w3', [1024, 10], initializer=w_init)
13  b3 = tf.get_variable('b3', [10], initializer=b_init)
14  logits = tf.matmul(h2, w3) + b3
15  o = tf.nn.softmax(logits)

```

Sada je potrebno definirati funkciju gubitka i njen optimizacijski postupak. Funkcija gubitka u ovom slučaju bit će unakrsna entropija između izlaznih kategorija. Za optimizacijski postupak odabran je gradijentni spust.

```

1   loss = tf.nn.softmax_cross_entropy_with_logits(
2           logits=logits, labels=y)
3   optim = tf.train.GradientDescentOptimizer(0.01).
4           minimize(loss)

```

Slijedi stvaranje konteksta i inicijalizacija varijabli. Nakon toga ostalo je još iterirati po skupinama podataka te u svakoj iteraciji pokretati optimizacijski postupak koji optimizira varijable mreže.

```

1   sess = tf.Session()
2   tf.global_variables_initializer().run()
3
4   for i in range(100000):
5       batch, labels = mnist.train.next_batch(batch_size=100)
6       sess.run(optim, feed_dict={x: batch, y: labels})

```

3.2. CUDA

CUDA je platforma i API za paralelno izvršavanje zadataka koristeći GPU. Radni okvir omogućuje slanje koda izravno na GPU. CUDA je proizvod kojeg je stvorila NVIDIA te zahtjeva njihov GPU sa specijaliziranim hardverom s CUDA-jezgrama. CUDA jezgre su procesorske jedinice koje se izvršavaju u paraleli te se njihov broj na grafičkim karticama najčešće kreće od 500 do nekoliko tisuća. Radni okviri namijenjeni za strojno učenje mogu iskoristiti funkcionalnosti CUDA-e kako bi se ubrzalo treniranje modela. cuDNN je radni okvir posebno namijenjen za neuronske mreže. Sadrži osnovna svojstva potrebna za treniranje neuronskih mreža kako bi se poboljšala učinkovitost.

Za potrebe ovog rada korištena je *NVIDIA Tesla K80* grafička kartica na *Google Cloud*-u kako bi se istrenirali modeli. Tako se znatno ubrzao postupak treniranja.

4. Postojeći modeli za stvaranje slika

U svrhu ovog rada obavljeno je istraživanje postojećih generativnih modela i njihovih rezultata. Ti radovi pokazuju prednosti i nedostatke takvih modela kao i različite metode treniranja i vrednovanja. Poslužili su kao motivacija za ovaj rad te je stoga ovo poglavlje posvećeno kratkom opisu svakog od tih radova.

4.1. Duboke konvolucijske generativne suparničke mreže

Duboke konvolucijske generativne neuronske mreže ili skraćeno DCGAN (engl. *Deep Convolutional Generative Neural Networks*) je arhitektura GAN-ova koje je namijenjena za stvaranje slika korištenjem prednosti konvolucijskih mreža. U [3] pokazani su uspješni rezultati i zanimljivi detalji treniranja DCGAN-ova. Osnovni rezultati na MNIST skupu podataka [19] prikazani su na slici 4.1 gdje je očito kako su uzorci stvoreni DCGAN-om jasniji i glađi od onih stvorenih običnim GAN-om iako i GAN proizvodi rezultate u kojima je vidljivo o kojem se broju radi.



Slika 4.1: Usporedba između izvornog MNIST skupa podataka, slika stvorenih GAN-om i slika stvorenih DCGAN-om.

DCGAN omogućuje nenadzirano učenje nad slikama veće razlučivosti od običnih

GAN-ova. U [3] trenirana je mreža nad 3 milijuna slika spavačih soba. Mreža je uspješno naučila opće uzorke te iz toga stvoriti nove slike koje se ne pojavljuju u izvornom skupu podataka. Rezultati nakon 5 epoha prikazani su na slici 4.2. Unatoč tome što su slike niske razlučivosti, očito je da se radi o sobama.



Slika 4.2: Slike spavačih soba stvorene DCGAN-om.

DCGAN je treniran i nad CelebA skupom podataka [20]. U tom skupu nalaze se slike lica poznatih osoba u visokoj razlučivosti. Rezultati treniranja prikazani su na slici 4.3 gdje se vidi da su slike oštore, da nema puno šuma i da su boje ispravne.



Slika 4.3: Slike lica stvorenih DCGAN-om

Nad navedenim ulaznim skupovima podataka nije izvršena predobrada osim skaliranja na željene dimenzije i na vrijednosti $[-1, 1]$ što odgovara $tanh$ funkciji. Svi modeli trenirani su stohastičkim gradijentnim spustom nad mini skupom podataka veličine 128. Sve težine inicijalizirane su normalnom razdiobom sa sredinom 0 i standardnom devijacijom 0.02. Kao aktivacija u skrivenim slojevima korištena je propusna zglobnica gdje nagib postavljen na 0.2. Korišten je Adam optimizator uz stopu učenja 0.0002.

Svojstva DCGAN mreža čine ih prilagodljivima za razne vrste ulaza što je očito

iz prethodno opisanih rezultata. Ista mreža sposobna je stvoriti slike lica, znamenki i spavačih soba bez mijenjanja parametara za učenje. U radu se još spominju 3 glavne metode korištene u DCGAN-ovima koje omogućuju dobre rezultate:

- **Normalizacija grupe** mijenja normalizaciju od zadatka koji se izvršava jednom prije treniranja na cijelom skupu podataka u nešto što se pojavljuje u svakoj iteraciji nad malim grupama podataka. Normalizacija čini mreže manje podložnima neoptimiziranim parametrima i omogućuje koritšenje većih stopa učenja.
- **Cijela konvolucijska mreža** je pristup kojim se, takozvani *pooling* slojevi koji smanjuju dimenzije slika, mijenjaju čistim konvolucijskim slojem. Obični *pooling* slojevi imaju predefinirano ponašanje kao što na primjer *max-pooling* smanjuje dimenzije tako da uzima najveću vrijednost. Cijela konvolucijska mreža može sama naučiti način smanjivanja dimenzija tako da nije ograničena na početnu zadanu arhitekturu. Ovakav pristup može se koristiti i na generatoru i diskriminatoru.
- **Rijetka povezanost** je ideja smanjivanja broja veza u mreži uklanjanjem potpuno povezanih slojeva. Prednost ovog pristupa je manji broj operacija i smanjeno zauzeće memorije. Objasnjenje rijetke povezanosti je da nije svaki piksel slike važan sam za sebe, već s okolnim pikselima čini oblike i uzorke.

4.2. Napredne metode treniranja generativnih suparničkih mreža

U [17] opisane su nove arhitekture i svojstva koja poboljšavaju treniranje GAN-ova i primjenjuju se u nenadziranom i polu-nadziranom učenju:

- **Poklapanje svojstava** mijenja način na koji generator uči nova svojstva. U običnom GAN-u, generator se trenira uzimajući u obzir samo izlaz diskriminadora. Tako GAN-ovi mogu biti nestabilni i može doći do pretreniranja diskriminadora. Novi cilj treniranja je da generator stvara podatke koji odgovaraju statistici pravih podataka, gdje se diskriminator koristi samo kako bi se pokazalo na statistiku za koju se smatra da treba naučiti. Točnije, generator se trenira kako bi poklopio svojstva skrivenog sloja sa skrivenim slojem diskriminadora. Izraz 4.1 predstavlja novu funkciju pogreške generatora. $f(x)$ označava aktivacije jednog od skrivenih slojeva diskriminadora. Ne može se garantirati

da će ovaj način treniranja konvergirati ali empirijski rezultati pokazuju da je učinkovit u situacijama gdje su obični GAN-ovi nestabilni.

$$\|\mathbb{E}_{x \sim p_{data}} f(x) - \mathbb{E}_{z \sim p_z(z)} f(G(z))\|_2^2 \quad (4.1)$$

- **Diskriminacija mini grupe** pomaže stabilizaciji diskriminatora tako da obrađuje slike u grupama umjesto jednu po jednu. To umanjuje urušavanje modela jer se generator kažnjava za stvaranje sličnih slika. Ovaj pristup uspoređen je s *poklapanjem svojstava* i zaključeno je da diskriminacija mini grupe konvergira brže i pruža bolje rezultate.
- **Ublažavanje oznaka** ublažava izlaze diskriminatora kako bi se uravnotežile kvalitete izvođenja oba modela. Tijekom treniranja, diskriminatoru se izravno zada da stvarne slike trebaju biti 1, a lažne 0. U ovom pristupu ocjene 1 i 0 ublažavaju se na 0.9 i 0.1.

4.3. Uvjetne generativne suparničke mreže

Kao što je već opisano, jedna od poznatijih inaćica GAN-ova su takozvane uvjetne generativne suparničke mreže. Takve mreže uče stvarati slike uz određeni uvjet. U [11] pokazano je kako se uvjetni GAN može istrenirati na MNIST skupu podataka uz zadanu znamenku kao uvjet. Na takav način moguće je stvoriti sliku željene znamenke. Osim toga pokazano je i kako se može naučiti višemodalni model na zadatku stvaranja tekstualnih oznaka za dane slike.

4.3.1. Skup podataka MNIST

Kako bi se istrenirao uvjetni GAN na MNIST skupu podataka napravljena je sljedeća konfiguracija. Ulaz generatora su 100 dimenzionalni šum z , koji je stvoren iz jednolike razdiobe, i uvjet y , *one-hot* vektor koji označava znamenku na slici. z i y mapirani su u skrivene ReLU slojeve s dimenzijama redom 200 i 1000. Ta dva sloja spojena su u skriveni 1200-dimenzionalni ReLU sloj. Na kraju se nalazi *sigmoid* sloj koji predstavlja 784-dimenzionalne MNIST uzorke. Diskriminator mapira ulaznu sliku x na maxout [10] sloj s 240 jedinica i 5 dijelova, i uvjet y na maxout sloj s 50 jedinica i 5 dijelova. Oba sloja spojena su u maxout sloj s 240 jedinica i 4 dijela prije nego što ulaze u posljednji *sigmoid* sloj.

Model se trenirao stohastičkim gradijentnim spustom s mini grupama veličine 100 i stopom učenja 0.1 koja je eksponencijalno padala do 0.000001 s faktorom raspada 1.00004. Korištena je *dropout* tehnika s vjerojatnošću 0.5 i na generatoru i na diskriminatoru. Na slici 4.4 prikazane su stvorene slike za svaku znamenku. Rezultati su dobri ali postoje pristupi koji su se pokazali bolji u zadatku stvaranja slika, jedan od takvih su obični GAN-ovi bez uvjeta. Glavni cilj ovih rezultata je pokazatelj da je moguće učinkovito stvarati slike s uvjetom koristeći generativne suparničke mreže. Autori rada vjeruju da daljnje istraživanje hiperparametara i arhitekture uvjetnih GAN-ova može nadmašiti učinkovitost mreža bez zadanih uvjeta.



Slika 4.4: Slike stvorene uvjetnim generativnim suparničkim mrežama.

4.3.2. Automatsko stvaranje oznaka za slike

Druga stvar pokazana u ovom radu je stvaranje tekstualnih oznaka za dane slike. Flickr stranica je dobar izvor označenih slika. Tu se nalazi mnoštvo slika kojima su korisnici dali svoj opis i oznake. Svaki korisnik može drugaćije označiti istu sliku i svaki korisnik ima drugačiji vokabular. Zbog toga je potrebno modelirati učinkovit način za normalizaciju tih oznaka. Rezultat ovog istraživanja je automatsko označavanje slika s mogućnošću dodavanja više oznaka za istu sliku.

Prvi korak je izračun svojstava slika. Trenira se konvolucijski model na cijelom ImageNet skupu s 21000 oznaka, sličan kao u [4]. Izlaz posljednjeg potpuno povezанog sloja uzet je kao 4096-dimenzionalni prikaz slike.

Za izračun prikaza tekstualnih oznaka korištena je zbirka korisničkih oznaka, naslova i opisa iz YFCC100M [2] skupa podataka. Nakon predobrade i čišćenja teksta treniran je skip-gram model [18] s 200-dimenzionalnim vektorom koji predstavlja riječi. Korišten vokabular sastojao se od 247465 riječi.

Za treniranje mreža korišten je MIR Flickr 25000 [1] skup podataka te su izračunati prikazi slika i oznaka pomoću prethodno opisanih konvolucijskog i riječnog modela. Slike bez oznaka su izbačene dok su slike s više oznaka ponavljanje više puta u skupu za treniranje, po jedan put za svaku oznaku.

Za procjenu rezultata stvoreno je 100 uzoraka za svaku sliku te pronađeno 20 najблиžih riječi koristeći kosinusovu udaljenost u vektorskom prikazu za svaki uzorak. Nakon toga odabire se 10 najčešćih riječi od svih 100 uzoraka. Na slici 4.5 prikazano je nekoliko slika, oznake koje su korisnici stavili te oznake koje je model stvorio. Najbolji model sastojao se od generatora koji za ulaz prima 100-dimenzionalan Gaussov šum kojeg mapira na 500-dimenzionalan ReLU skriveni sloj. 4096-dimenzionalan prikaz slika mapiran je na 2000-dimenzionalan ReLU skriveni sloj. Oba sloja mapirani su na združeni 200-dimenzionalan linearni sloj koji predstavlja stvorene vektore riječi (oznaka). Diskriminaciju se sastoji od 500-dimenzionalnog ReLU skrivenog sloja za vektor oznaka, 1200-dimenzionalan ReLU skriveni sloj za prikaz slika i maxout sloj od 1000 jedinica i 3 dijela koji združuje prva dva navedena sloja. Maxout sloj je povezan s izlaznom sigmoid jedinicom.

Model je treniran stohastičkim gradijentnim spustom s mini grupama veličine 100 i početnom stopom učenja 0.1 koja se eksponencijalno smanjuje do 0.000001 s faktorom raspada 1.00004. Korišten je dropout s vjerojatnošću 0.5 i na generatoru i na diskriminatoru.

	montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails
	food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
	water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
	people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life

Slika 4.5: Uzorci oznaka koje su za pojedine slike dodali korisnici (lijevo) i koje je stvorio model (desno).

5. Programsко ostvarenje i vrednovanje

Ostvareno je nekoliko generativnih modela za stvaranje slika iz različitih skupova podataka. Cilj je uspješno ponoviti neke rezultate iz radova navedenih u poglavlju 4, primijeniti ostvarene modele na više skupova podataka te ih usporediti i vrednovati. Generativne modele je teško objektivno vrednovati, ne postoji jedinstvena mjera koja govori koliko model dobar, stoga je korištena subjektivna procjena autora rada. Cilj modela za stvaranje slika je postići stanje u kojem model stvara slike koje je nemoguće razlučiti od izvornih podataka.

Početno vrednovanje provedeno je nad MNIST skupom podataka. Nakon toga modeli su ocijenjeni nad složenijim CIFAR10 i LFW skupovima. Varijacijski autoenkoder i generativne suparničke mreže su dvije različite arhitekture dubokih neuronskih mreža koje su korištene kao modeli za stvaranje slika. Oba modela opisana su i teorijski uspoređena u poglavlju 2. U ovom poglavlju uspoređena su empirijskim putem.

TensorFlow je odabran kao glavni radni okvir za programsko ostvarenje potrebnih modela. TensorFlow je ostvaren na nižoj razini od ostalih poznatih radnih okvira za duboko učenje (Keras, Caffe, PyTorch, Chainer itd.). Tako se posjeduje veća kontrola nad ostvarenim sustavom, posvećuje veća pažnja detaljima i moguće je više naučiti o samim modelima. Nadalje, TensorFlow podržava pokretanje programa na grafičkim karticama tako da su modeli trenirani na *Google Cloud-u* na Tesla K80 grafičkoj kartici. Treniranje na procesorima bi trajalo znatno dulje.

Svaki model sastoji se od brojnih parametara koji utječu na brzinu i kvalitetu treninga. Cilj ovog rada nije utvrđivanje njihovih utjecaja te su stoga za sve istrenirane modele uzeti parametri koji su u drugim radovima pokazani kao najbolji za određenu skupinu problema. Zajednički parametri koji su se koristili u većini ostvarenih modela objašnjeni su u sljedećoj listi:

- **Konvolucijski slojevi** koriste se u modelima koji obrađuju slike.

- **Slojevi transponirane konvolucije** predstavljaju istu operaciju kao i konvolucijski slojevi. Razlika je u tome što se konvolucija koristi kako bi se podatci većih prostornih dimenzija i manje dubine pretvorili u podatke male prostorne dimenzije i velike dubine dok se transponirana konvolucija koristi u suprotnom smjeru. Cilj je iz dubokih svojstava dobiti oblik koji odgovara slikama. Takvi se slojevi koriste u dekoderu kod VAE i generatoru kod GAN.
- **Propusna zglobnica** s faktorom propusnosti 0.2 korištena je kao aktivacijska funkcija u skrivenim slojevima. Preporučena je u [7].
- **Isključivanje čvorova** (engl. *dropout*) korišteno je u skrivenim slojevima kako bi se ubrzalo treniranje i spriječilo pretreniranje modela. Vjerojatnost ispadanja postavljena je na 0.5. Učinkovitost *dropout* slojeva opisana je u [13].
- **Adam** sa stopom učenja 0.0005 je odabrana optimizacijska funkcija. Opisana je u [8] kao nazučinkovitija i preporučena u [7] za treniranje generativnih modela.
- **Normalizacija grupe** smanjuje kovarijacijski pomak u skrivenim slojevima. Koristi se i u ulaznom sloju kako bi se podatci skalirali unutar određenog intervala i tako ubrzalo treniranje. Ulazni podatci u ostvarenim modelima normalizirani su u intervalu $[0, 1]$ dok su skriveni slojevi normalizirani s obzirom na srednju vrijednost i devijaciju trenutne grupe podataka. Detaljnije objašnjeno u [9].

5.1. Skup podataka MNIST

MNIST [19] je skup slika koje predstavljaju ručno napisane znamenke od 0 do 9. Vrlo je jednostavan pa se često koristi kao početni skup kojim se treniraju modeli čiji je cilj obrada slikovnih podataka. Koristi se za treniranje i testiranje mnogih algoritama strojnog učenja. Sastoji se od 60000 crno-bijelih slika dimenzija 28x28 od kojih su neke prikazane na slici 4.1. Takvi podatci mogu poslužiti za treniranje klasifikatora koji na ulaz primaju sliku, a na izlaz daju znamenku prikazanu na slici. Nadalje, može poslužiti i testiranje konvolucijskih mreža koje su posebno namijenjene za obradu slika. U svrhu ovog rada poslužit će za treniranje početnog modela koji će naučiti što vjernije stvarati slike iz tog skupa.

5.1.1. Varijacijski autoenkoder

Varijacijski autoenkoder korišten je kao prvi model za treniranje nad MNIST skupom podataka. Model se sastoji od enkodera i dekodera.

Ulaz enkodera je izvorna $28 \times 28 \times 1$ slika čije su vrijednosti piksela u intervalu $[0, 1]$. Nakon toga slijede 3 konvolucijska sloja dubine 64, veličine jezgre 4 i s pomacima redom 2, 2 i 1. Aktivacijska funkcija je propusna zglobnica s faktorom propusnosti 0.2. Nakon svakog konvolucijskog sloja nalazi se dropout sloj s vjerojatnošću ispadanja 0.5. Izlaz iz enkodera je skriveni n-dimenzionalni vektor.

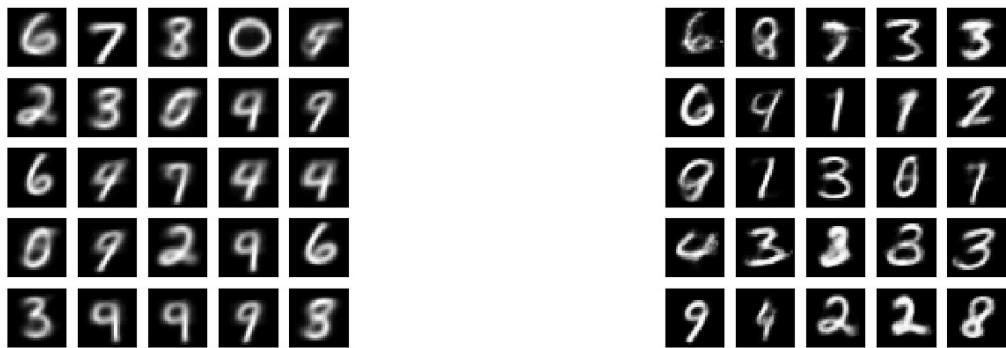
Ulaz dekodera je n-dimenzionalni vektor (izlaz enkodera). Prva dva sloja su 49-dimenzionalni i potpuno povezani. Nakon toga 49-dimenzionalni izlaz drugog potpuno povezanog sloja preoblikovan je u 7×7 oblik kako bi se mogao provući kroz slojeve transponirane konvolucije. Nakon toga slijede 3 takva sloja dubine 64, veličine jezgre 4 i pomacima 2, 1 i 1. Aktivacijska funkcija nad potpuno povezanim i slojevima transponirane konvolucije je propusna zglobnica s faktorom propusnosti 0.2. Nad tim slojevima koristi se *dropout* s vjerojatnosti ispadanja neurona 0.5. Posljednji sloj je potpuno povezan veličine 784 sa *sigmoid* aktivacijom. Konačno, taj vektor se preoblikuje u 28×28 s vrijednostima u intervalu $[0, 1]$, što je zapravo oblik izvornih podataka.

Adam sa stopom učenja 0.0005 je odabrana optimizacijska funkcija. Veličine grupe za korak treniranja je 100. Broj epoha treniranja je 20.

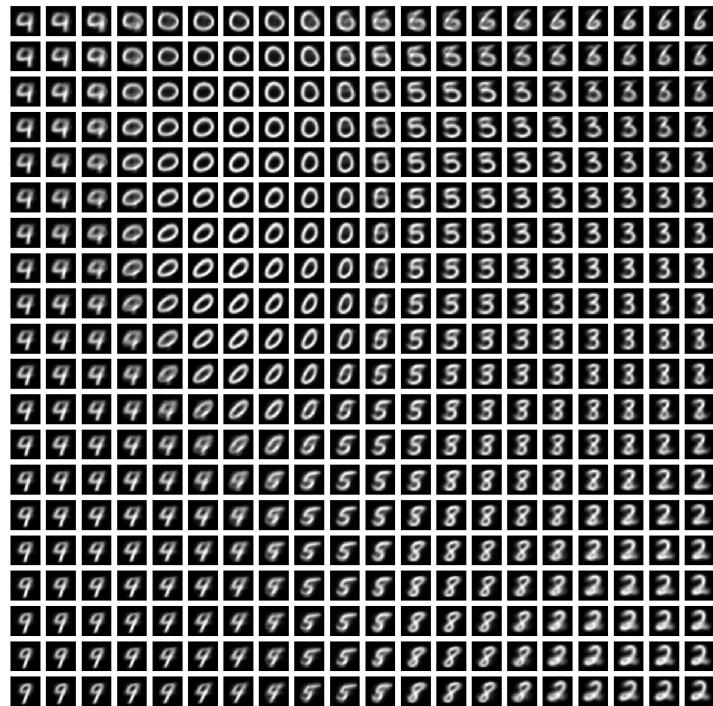
Na slici 5.1 prikazani su rezultati prethodno opisane mreže koja ulazne podatke šifrira u 2-dimenzionalni, odnosno 8-dimenzionalni skriveni vektor z koji se trenira da odgovara normalnoj razdiobi. Sada je moguće uzorkovati n-dimenzionalne podatke iz normalne razdiobe te iz njih stvoriti slike. Rezultati su slični izvornim slikama. U rezultatima s 8-dimenzionalnim vektorom z vidljivo je više šuma.

Nadalje, na slici 5.2 prikazani su rezultati interpolacije dvodimenzionalnog vektora z . Redci i stupci predstavljaju interpolaciju vektora z u intervalu $[-1.0, 1.0]$ s korakom 0.1. Prva sličica je stvorena iz $z = [-1.0, -1.0]$, druga iz $z = [-1.0, -0.9]$, 11-ta iz $z = [-0.9, -1.0]$ itd. Slika 5.2 pokazuje da se prilikom treniranja z uzorkuje iz normalne razdiobe. Vidi se kako određene vrijednosti pripadaju određenim znamenkama i glatki prijelazi između njih. Na slici 5.2 nisu prikazane sve znamenke zbog toga što navedeni raspon nije dovoljan kako bi se sve pokrilo.

Zaključak je da je moguće uspješno istrenirati varijacijski autoenkoder nad MNIST skupom podataka. Dvodimenzionalni skriveni vektor je dao bolje rezultate ali manje različitosti. Takav model može stvoriti 10-ak različitih slika određene znamenke zbog toga što se ulazne slike šifriraju u samo dvije dimenzije (bolje prikazano u 5.1.2).



Slika 5.1: Uzorci stvorenji varijacijskim autoenkoderom s 2-dimenzionalnim (lijevo) i 8-dimenzionalnim (desno) skrivenim slojem.



Slika 5.2: Interpolacija uzorka stvorenih varijacijskim autoenkoderom s dvodiomenzionalnim skrivenim slojem.

MNIST skup podataka sadrži mnogo različitih inaćica pojedine znamenke te bi bilo bolje da je model izražajniji. 8-dimenzionalni skriveni vektor pokazuje više izražaj-

nosti ali više šuma. Takav model može stvoriti na desetke različitih slika određene znamenke. Treniranjem takvog modela s više epoha moglo bi proizvesti vjernije rezultate s manje šuma.

5.1.2. Uvjetni varijacijski autoenkoder

Uvjetni varijacijski autoenkoder razlikuje se od običnog u tome što je ulaz enkodera i dekodera proširen dodatnim uvjetom. Kod MNIST skupa podataka dodatan uvjet je znamenka na slici. To se može iskoristiti kako bi se istrenirao generativni model kojem je moguće zadati znamenku čiju sliku treba stvoriti.

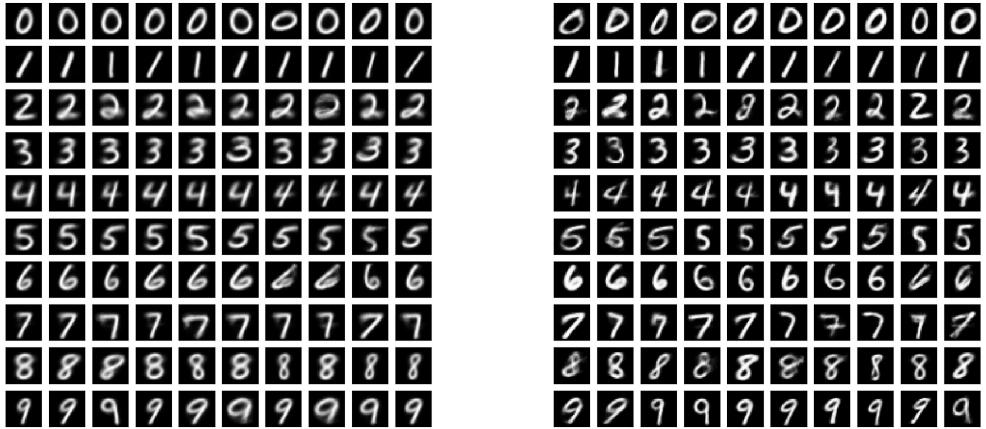
Arhitektura ostvarenog modela ista je kao u prethodnom poglavlju. Jedina razlika su ulazi enkodera i dekodera. Ulaz dekodera je 28x28x1 izvorna slika na koju je pridodan 57-dimenzionalan *one-hot* vektor koji govori o kojoj se znamenici radi. S obzirom na to da postoji 10 znamenaka, posljednjih 47 dimenzija uvjetnog vektora uvijek će biti 0. Vektor ima 57 dimenzija zbog toga što je model konvolucijska mreža koja obrađuje podatke u obliku slike (visina, širina, dubina). Tako je ulazni podatak proširen na 29x29x1. Kako bi se to omogućilo potrebno je dodati još 57 dodatnih polja na izvorni ulazni podatak. Dekoder na ulaz prima n-dimenzionalni skriveni vektor na koji je spojen 57-dimenzionalan uvjetni vektor.

Na slici 5.3 prikazano je po 10 slika za svaku znamenku koje je stvorio model s 2-dimenzionalnim i 8-dimenzionalnim skrivenim slojem. S 2-dimenzionalnim skrivenim slojem vidi se kako ne postoji dovoljno različitosti unutar jedne znamenke. Sve stvorene slike za pojedinu znamenku gotovo su identične. Unatoč tome rezultati su zadovoljavajući i slični izvornim podatcima. S 8-dimenzionalnim skrivenim slojem vidi se više različitosti unutar jedne znamenke ali više šuma.

Iz priloženih rezultata vidi se kako uvjetni VAE rezultira čišćim i boljim slikama od običnog VAE. Razlog tome je što svaka znamenka ima svoju razdiobu u skrivenom prostoru. Zadajući oznaku pri treniranju, model može lakše naučiti razdiobu za svaku znamenku. Kod običnog VAE postoji mnogo rubnih slučajeva kada model nije siguran radi li se npr. o 3 ili 8 pa je rezultat nešto između. Kod uvjetnog VAE, modelu je izravno zadano o kojoj se znamenici radi, stoga rijetko dolazi do preklapanju dviju ili više znamenaka.

5.1.3. Generativne suparničke mreže

Arhitektura generativnih suparničkih mreža nad MNIST podatcima gotovo je ista kao arhitektura enkodera i dekodera kod VAE. Tako se mogu usporediti VAE i GAN s



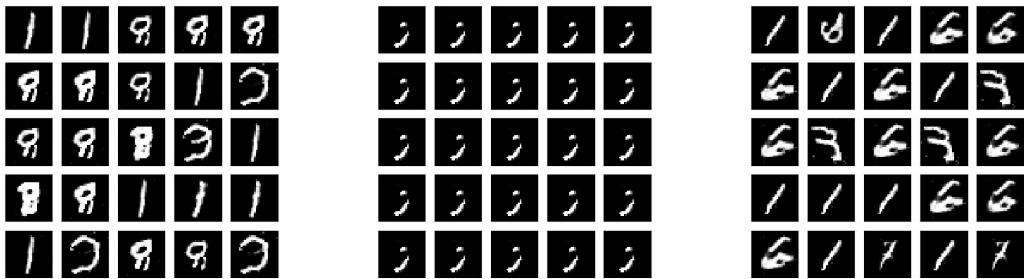
Slika 5.3: Uzorci stvorenim uvjetnim varijacijskim autoenkoderom s 2-dimenzionalnim (lijevo) i 8-dimenzionalnim (desno) skrivenim slojem.

jednako izražajnim modelima.

Arhitektura generatora jednaka je dekoderu kod VAE modela. Diskriminatator na ulaz prima $28 \times 28 \times 1$ podatak koji može biti stvoren s generatorom ili izvorni podatak. Nakon toga slijede 3 konvolucijska sloja sa 64 filtera, veličine jezgre 4×4 , aktivacijom propusnom zglobnicom i pomacima 2, 2 i 1. Nad konvolucijskim slojevima koristi se *dropout* s vjerojatnosti ispadanja neurona 0.5. Izlaz zadnjeg konvolucijskog sloja se izravna u $n \times 1$ vektor nakon čega slijedi potpuno povezani sloj i konačno izlazna jedinica sa *sigmoid* aktivacijom. Izlaz je vrijednost u intervalu $[0, 1]$. Adam sa stopom učenja 0.0005 je odabrana optimizacijska funkcija. Veličina grupe za korak treniranja je 100. Broj epoha treniranja je 20.

Model je istreniran s vektorom z veličine 2, 8 i 100. Umjesto *dropout* sloja korištena je i normalizacija skupine za skrivene slojeve. Nijedna od navedenih postavki nije proizvela zadovoljavajuće rezultate. Kao što se vidi na slici 5.4, znamenke su neprepoznatljive. Nadalje, svi stvoreni uzorci za pojedine postavke modela su gotovo isti. To je vjerojatno zbog toga što je treniranje zapelo u lokalnom minimumu. Treniranje GAN modela često je nezadovoljavajuće. Teško je otkriti zašto je uspješno ili neuspješno zbog složenosti sustava. Generator i diskriminatator nadmeću se i pokušava se uspostaviti ravnoteža. Često se dogodi da jedan od njih prevagne i time model divergira. U ovom slučaju pogreška diskriminatora smanjila se gotovo na 0 i generator više nije mogao pobijediti.

Nakon neuspješnog treniranja GAN modela čija je arhitektura gotovo ista kao i



Slika 5.4: Slike uzorkovane iz GAN modela s veličinom vektora z redom 2, 8 i 100.

ona kod prethodno opisanog VAE, napravljen je izražajniji GAN model. Generator i diskriminatore novog modela sastoje se isključivo od konvolucijskih slojeva. Uzani podatci preoblikovani su iz $28 \times 28 \times 1$ u $64 \times 64 \times 1$ i skalirani iz intervala $[0, 1]$ u $[-1, 1]$.

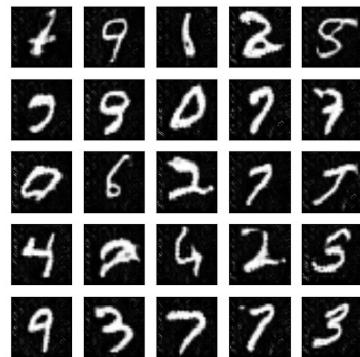
Dakle, ulaz diskriminatora je $64 \times 64 \times 1$ slika. Nakon toga slijedi 5 konvolucijskih slojeva veličine jezgre 4×4 , pomacima 2, 2, 2, 2 i 1 te redom sa 128, 256, 512, 1024 i 1 filtera. Izlaz svakog konvolucijskog sloja osim zadnjeg je normaliziran i provučen kroz propusnu zglobnicu s faktorom propusnosti 0.2. Izlaz zadnjeg konvolucijskog sloja provučen je kroz *sigmoid* aktivaciju kako bi izlazna vrijednost bila u intervalu $[0, 1]$.

Uzaz generatora je 100-dimenzionalni vektor z . Nakon toga slijedi 5 slojeva transponirane konvolucije veličine jezgre 4×4 , pomacima 1, 2, 2, 2 i 2 te redom s 1024, 512, 256, 128 i 1 filtera. Izlaz svakog konvolucijskog sloja osim zadnjeg je normaliziran i provučen kroz propusnu zglobnicu s faktorom propusnosti 0.2. Izlaz zadnjeg konvolucijskog sloja provučen je kroz *tanh* aktivaciju kako bi izlazne vrijednosti bile u intervalu $[-1, 1]$.

Adam sa stopom učenja 0.0005 je odabrana optimizacijska funkcija. Veličina grupe za korak treniranja je 100. Broj epoha treniranja je 20.

Rezultat treniranja prikazan je na slici 5.5. Rezultati su bolji od prethodnog, manje izražajnog GAN modela. Iz većine slika očito je o kojoj se znamenci radi. Nadalje, vidi se da rezultati ne dolaze iz izvornih podataka.

Mnogo izražajniji GAN model nije uspio nadmašiti VAE na MNIST skupu podataka. VAE stvara bolje slike. Kako bi se poboljšali rezultati pokušane su i neke naprednije metode treniranja koje su opisane u 4.2. Rezultati su sive slike te su modeli neuspješni. Na osnovi isprobanih modela na MNIST skupu podataka pokazano je da je VAE jednostavniji model te ga je lako natjerati da radi. GAN model je složeniji i takvi modeli iskustveno rade na mnogo skupova podataka te su se općenito pokazali uspješ-

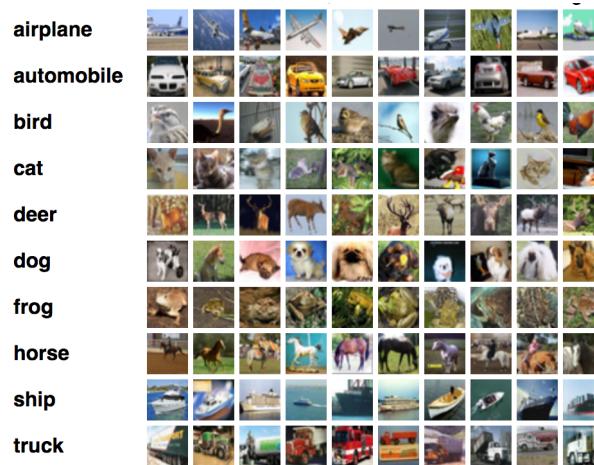


Slika 5.5: MNIST uzorci stvorenji GAN modelom.

nim rješenjem. U slučajevima kada ne radi teško je otkriti razlog te koje parametre treba promijeniti kako bi postao učinkovit.

5.2. Skup podataka CIFAR-10

Skup podataka CIFAR-10 je skup 32x32x3 slika koje se često koriste za treniranje algoritama strojnog učenja. Sastoji se od 60000 slika u boji raspoređenih u 10 različitih kategorija: avioni, automobili, ptice, mačke, jeleni, psi, žabe, konji, brodovi i kamioni. Podatci su jednostavni ali složeniji od MNIST-a jer sadrže stvarne prizore iz prirode i boje u obliku RGB komponenti. Stoga CIFAR-10 predstavlja novi izazov za VAE i GAN modele nakon MNIST-a. Primjer izvornih podataka prikazan je na slici 5.6.



Slika 5.6: Izvorni CIFAR-10 podatci preuzeti s [5].

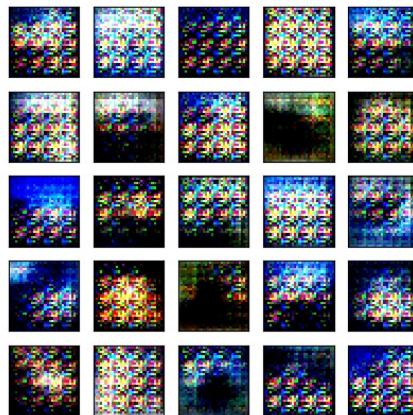
5.2.1. Varijacijski autoenkoder

Ulag enkodera je izvorna $32 \times 32 \times 3$ slika. Slika se provlači kroz 5 konvolucijskih slojeva veličine jezgre 4×4 , pomacima $2, 2, 2, 2$ i 2 te redom sa $128, 256, 512, 1024$ i 100 filtera. Izlaz svakog konvolucijskog sloja normaliziran je i provučen kroz propusnu zglobnicu s faktorom propusnosti 0.2 . Izlaz iz enkodera je 100 -dimenzionalni vektor z .

Dekoder na ulaz prima 100 -dimenzionalni vektor z . Nakon toga slijede 4 sloja transponirane konvolucije veličine jezgre 4×4 , pomacima $1, 2, 2, 1$ te redom s $1024, 512, 256$ i 3 filtera. Izlaz svakog konvolucijskog sloja osim zadnjeg normaliziran je i provučen kroz propusnu zglobnicu s faktorom propusnosti 0.2 . Izlaz dekodera odgovara izvornoj slici ($32 \times 32 \times 3$).

Adam sa stopom učenja 0.0005 je odabrana optimizacijska funkcija. Veličina grupe za korak treniranja je 100 . Broj epoha treniranja je 50 .

Rezultati su prikazani na slici 5.7. Rezultati nisu uspješni jer se na slikama ne prepoznaju podatci iz CIFAR-10 skupa. Iz ovog primjera vidljivo je kako VAE nije u stanju naučiti složeniju razdiobu kao što je CIFAR-10 skup podataka.

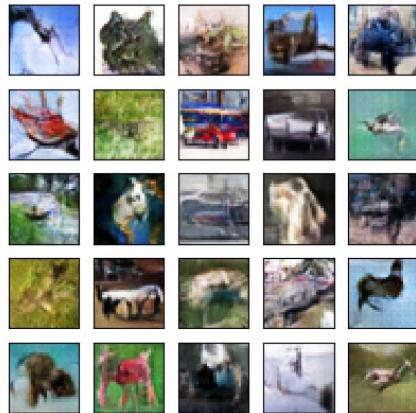


Slika 5.7: CIFAR-10 uzorci stvoreni VAE modelom.

5.2.2. Generativne suparničke mreže

Arhitektura generatora ista je kao i ona kod dekodera u prethodno opisanom VAE modelu. Diskriminatore se od enkodera razlikuje samo u zadnjem sloju gdje se nalazi konvolucijski sloj s jednim filtrom i *sigmoid* aktivacijom. Ostali parametri za učenje su isti.

Rezultati su prikazani na slici 5.8. Rezultati su bolji od prethodnog VAE modela. Slike su slične izvornom skupu i na nekima se može vidjeti o kojem se predmetu radi. Na ovom primjeru pokazano je kako je GAN model u stanju naučiti stvarati složenije slike od VAE modela. Za razliku od treniranja nad MNIST podatcima, ovaj model se nije raspao te se ravnoteža generatora i diskriminadora održavala do kraja treniranja.



Slika 5.8: CIFAR-10 uzorci stvoreni GAN modelom.

5.3. Skup podataka LFW

Skup podataka LFW [14] je skup slika ljudskih lica. Skup je stvoren kako bi se pro- učio problem prepoznavanja lica. Skup se sastoji od nešto više od 13000 slika lica prikupljenih s Interneta. Svaka slika označena je imenom osobe na slici. Slike su veličine 250x250x3 te su poravnate tako da se na svakoj slici nalazi samo lice i pozadina. Primjeri podataka iz LFW skupa prikazani su na slici 5.9. Za potrebu treniranja slike su preoblikovane u 64x64x3.

5.3.1. Varijacijski autoenkoder

Ulaz enkodera je 64x64x3 slika. Slika se provlači kroz 6 konvolucijskih slojeva veličine jezgre 4x4, pomacima 2 te redom sa 128, 256, 512, 512, 1024 i 100 filtera. Izlaz svakog konvolucijskog sloja normaliziran je i provučen kroz propusnu zglobnicu s faktorom propusnosti 0.2. Izlaz iz enkodera je 100-dimenzionalni vektor z .

Dekoder na ulaz prima 100-dimenzionalni vektor z . Nakon toga slijede 5 sloja transponirane konvolucije veličine jezgre 4x4, pomacima 1, 2, 2, 2, i 2 te redom s 1024,

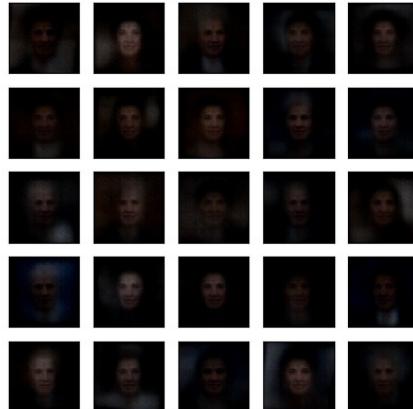


Slika 5.9: Izvorni LFW podatci.

512, 256, 128 i 3 filtera. Izlaz svakog konvolucijskog sloja osim zadnjeg normaliziran je i provučen kroz propusnu zglobnicu s faktorom propusnosti 0.2. Izlaz dekodera odgovora ulaznoj slici (64x64x3).

Adam sa stopom učenja 0.0005 je odabrana optimizacijska funkcija. Veličina grupe za korak treniranja je 50. Broj epoha treniranja je 20.

Rezultati su prikazani na slici 5.10. Iako su slike tamne moguće je razaznati da se radi o licima. Model je naučio osnovne crte lica ali nije uspješno stvorio detalje.



Slika 5.10: LFW uzorci stvoreni VAE modelom.

5.3.2. Generativne suparničke mreže

Arhitektura generatora ista je kao i ona kod dekodera u prethodno opisanom VAE modelu. Diskriminatator se od enkodera razlikuje samo u zadnjem sloju gdje se nalazi konvolucijski sloj s jednim filtrom i *sigmoid* aktivacijom. Ostali parametri za učenje su isti.

Rezultati su prikazani na slici 5.11. Rezultati su bolji od prethodnog VAE modela. Jasno je da se radi o licima te se vide detalji poput odjeće i frizure kakvi se nalaze i u izvornom skupu. Bez obzira na to i dalje je lako prepoznati da se radi o slikama koje je stvorio model. Na ovim rezultatima, isto kao i kod CIFAR-10, pokazano je kako su GAN modeli moćniji od VAE i u stanju naučiti složenije razdiobe.



Slika 5.11: LFW uzorci stvorenji GAN modelom.

6. Zaključak

Generativni modeli su se pokazali uspješnim za brojne zadatke. U ovom radu istraženo je stvaranje slika. Pokazano je da je moguće uspješno ostvariti modele za stvaranje slika koje dolaze iz različitih razdioba. Postoji više različitih arhitektura i načina treniranja kojim se taj zadatak može ostvariti. Složenije slike s više detalja zahtijevaju složenije modele. Takve modele potrebno je prilagoditi podatcima. Nadalje, pokazano je i uspješno treniranje uvjetnih generativnih modela. Pomoću takvih modela moguće je upravljati stvaranjem slika. Moguće je zadati neke uvjete koje stvorena slika mora zadovoljavati.

Uspoređujući VAE i GAN modela zaključak je da su oba modela u stanju stvarati slike slične izvornim podatcima. Zbog svoje jednostavnosti i intuicije koja stoji iza modela, VAE je uspješno istreniran nad jednostavnim MNIST podatcima. Procjenom razdiobe $P(z)$ s normalnom, i uvođenjem KL-divergencije kao mjeru sličnosti između razdioba $P(z)$ i $Q(z|X)$ pokazalo se uspješnom metodom. Ipak, to nije bilo dovoljno dobro za složenije podatke poput CIFAR-10 i LFW. KL-divergencija optimizira donju varijacijsku granicu te znatno pojednostavljuje razdiobu $P(z|X)$ što onemogućuje učenje složenijih razdioba pa tako i stvaranje složenijih podataka. Možda bi se dobili bolji rezultati kada bi se napravila statistička analiza CIFAR-10 i LFW podataka te modelirala primjenjivija razdioba $P(z)$. S druge strane GAN se pokazao kao moćan ali i nepredvidljiv model. Tijekom treniranja nad MNIST-om diskriminator je nadjačao te je model ispaio iz ravnoteže. Time je generator stvarao loše uzorke. Ipak, nad CIFAR-10 i LFW stvorio je zadovoljavajuće uzorka i pokazao se uspješnim.

VAE može biti prikladan za stvaranje jednostavnih slika s manje detalja. Uz pažljivo oblikovanje mogao bi biti uspješan i na CIFAR-10 i LFW podatcima. GAN je uspješniji i bolje ga je koristiti za složenije slike.

Za budući rad ostavlja se ostvarenje modela koji može stvarati složenije slike, veće razlučivosti od svih navedenih u radu. Treniranje takvog modela zahtjeva detaljniju analizu razdiobe skupa za treniranje i prilagođavanje modela u skladu s tim. Nadalje, ostavlja se i treniranje uvjetnog generativnog modela nad slikama lica iz skupa CelebA

[20]. Taj skup, osim slika lica, sadrži i oznake koje opisuju svako lice. Neke od tih oznaka su: brkovi, naočale, boja kože, spol, brada itd. Postoji 40 oznaka. Tako se može istrenirati uvjetni model koji bi iz zadanih oznaka stvorio prikladno lice.

LITERATURA

- [1] The mirflickr retrieval evaluation.
- [2] Yahoo flickr creative commons 100 million dataset.
- [3] Soumith Chintala Alec Radford, Luke Metz. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016.
- [4] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. 2012.
- [5] Geoffrey Hinton Alex Krizhevsky, Vinod Nair. The cifar-10 dataset.
- [6] Count Bayesie. Kullback-leibler divergence explained. 2017.
- [7] Soumith Chintala. How to train a gan? tips and tricks to make gans work. *GitHub*, 2016.
- [8] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. 2017.
- [9] Firdaouss Doukkali. Batch normalization in neural networks. *Towards Data Science*, 2017.
- [10] Mehdi Mirza Aaron Courville Yoshua Bengio Ian Goodfellow, David Warde-Farley. Maxout networks. 2013.
- [11] Simon Osindero Mehdi Mirza. Conditional generative adversarial nets. 2014.
- [12] Alexander Mordvintsev. Deep dream generator. *Google*, 2015.
- [13] Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov Nitish Srivastava, Geoffrey Hinton. Dropout: A simple way to prevent neural networks from overfitting. 2014.

- [14] University of Massachusetts. Labeled faces in the wild.
- [15] Prakash Pandey. Deep generative models. *Towards Data Science*, 2018.
- [16] Bernhard von Stengel Theodore L. Turocy. Game theory. 2001.
- [17] Alec Radford Xi Chen Wojciech Zaremba Vicki Cheung Tim Salimans, Ian Goodfellow. Improved techniques for training gans. 2016.
- [18] Greg Corrado Jeffrey Dean Tomas Mikolov, Kai Chen. Efficient estimation of word representations in vector space. 2013.
- [19] Christopher Burges Yann LeCun, Corinna Cortes. The mnist database.
- [20] Xiaogang Wang Xiaoou Tang Ziwei Liu, Ping Luo. Large-scale celebfaces attributes (celeba) dataset.

Stvaranje slika pomoću dubokog učenja

Sažetak

Generativni modeli su modeli umjetnih neuronskih mreža koji su namijenjeni za stvaranje podataka iz naučene razdiobe. Pomoću takvih modela moguće je stvarati slike. Neuronske mreže za obradu slikovnih podataka koriste konvolucijske slojeve pa su i generativni modeli za stvaranje slika sastavljeni od njih. Obradene su dvije vrste takvih modela: varijacijski autoenkoder (VAE) i generativne suparničke mreže (GAN). VAE je jednostavnija arhitektura koja je autoenkoder, prilagođen za stvaranje novih podataka. GAN je složena arhitektura od dvije mreže: generator i diskriminat. Mreže su vrednovane nad MNIST, LFW i CIFAR-10 skupovima podataka. VAE i GAN su uspješno istrenirane za stvaranje slika iz svih skupova. VAE je uspješniji nad jednostavnim MNIST skupom dok je GAN bolji nad složenijim LFW i CIFAR-10 skupovima. Nadalje, ostvareni su uvjetni VAE i GAN. Takvi modeli stvaraju slike sa zadanim uvjetom. Ostvareni su nad MNIST skupom podataka tako da je moguće zadati znamenku čiju sliku treba stvoriti. Uvjetni generativni modeli otvaraju mogućnosti za stvaranje složenijih slika s uvjetom, kao što su slike lica sa željenim svojstvima.

Ključne riječi: duboko učenje, neuronska mreža, slika, generativni model, uvjet, varijacijski autoenkoder, generativne suparničke mreže

Generating Images Using Deep Learning

Abstract

Generative models are neural network model made for generating data from learned distribution. It is possible to generate images with that kind of models. Neural networks that process images use convolutional networks so generative models for images contain them as well. Variational autoencoder (VAE) and generative adversarial networks (GAN) are two types of generative models explained in this paper. VAE has simpler architecture that is autoencoder adjusted for generating new data. GAN has complex architecture that consists of two networks: generator and discriminator. Networks are evaluated on MNIST, LFW and CIFAR-10 datasets. VAE and GAN are successfully trained for generating images on all of mentioned datasets. VAE was better on simple MNIST dataset while GAN performed better on LFW and CIFAR-10 datasets. Furthermore, conditional VAE and GAN are implemented as well. Those models generate data with given condition. They are trained on MNIST dataset so it is possible to set digit which image will be generated. Conditional generative models create opportunities for generating complex images with condition, such as face images with desired attributes.

Keywords: deep learning, neural network, image, generative model, condition, variational autoencoder, generative adversarial networks