Generativne kontradikcijske neuronske mreže u sintezi originalnih rješenja

Homolak, Sandi

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Sandi Homolak

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Doc. Dr. sc. Petar Ćurković, dipl. ing.

Student:

Sandi Homolak

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru, doc. dr. sc. Petru Ćurkoviću na stručnoj pomoći i savjetima te na susretljivosti i ugodnoj atmosferi koju je pružao tijekom izrade ovog rada.

Također, zahvaljujem se svojoj obitelji i bliskim prijateljima koji su mi pružali podršku tijekom cijelog preddiplomskog studija.

Sandi Homolak

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **Sandi Homolak** Mat. br.: 0035199603

Naslov rada na hrvatskom jeziku: **Generativne kontradikcijske neuronske mreže u sintezi originalnih rješenja**

Naslov rada na engleskom jeziku: **Generative adversarial neural networks for synthesis of original solutions**

Opis zadatka:

Generativne kontradikcijske neuronske mreže (GEKON) su složeni algoritmi strojnog učenja koji se temelje na natjecanju dviju neuronskih mreža u hipotetskoj igri s nultom sumom. Nasuprot klasičnom pristupu kod neuronskih mreža kod kojih se nakon učenja novi podaci klasificiraju u kategorije, kod GEKON mreža se nakon učenja stvara novi ulaz, koji je svojom kvalitetom barem ekvivalentan kvaliteti ulaznog seta podataka korištenog za učenje.
Paradigma rada neuronske mreže je dakle promijenjena na način da mreža sada ne klasificira, već stvara nove vrijednosti.

Primjenom ovih mreža moguće je stvoriti vrlo realistične modele ljudi, životinja, ali i predmeta koji ne postoje, već su u potpunosti proizvod GEKON mreže.

U radu je potrebno napraviti sljedeće:

- upoznati se s teorijom i radom GEKON mreža
- identificirati mogućnosti primjene ovih mreža u području umjetne inteligencije s naglaskom na primjenu za rješavanje tehničkih problema u robotici
- opisati postojeće etičke probleme nastale uslijed iznimne realističnosti generiranih modela ljudi
- ispitati mogućnost primjene i implementacije GEKON mreže na opremi dostupnoj u Laboratoriju za projektiranje izradbenih i montažnih sustava

U radu je potrebno navesti literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

30. studenoga 2020.

Zadatak zadao:

Doc. dr. sc. Petar Ćurković

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
POPIS OZNAKA IV
SAŽETAK VI
SUMMARY VII
1. UVOD 1
1.1. Biološki neuron 2
1.2. Umjetni neuron 2
1.3. Aktivacijske funkcije 2
1.3.1. Step funkcija 4
1.3.2. Linearna funkcija 5
1.3.3. Sigmoidalna funkcija 6
1.3.4. ReLu funkcija 7
1.4. Funkcije troška 8
1.5. Postupak učenja 9
1.6. Algoritam sa širenjem pogreške unazad 11
1.6.1. Neuronska mreža s jednim neuronom u svakom sloju 11
1.6.2. Neuronska mreža s više neurona u svakom sloju 17
2. GENERATIVNE KONTRADIKCIJSKE NEURONSKE MREŽE 20
2.1. Primjena generativnih kontradikcijskih neuronskih mreža 21
2.1.1. Primjena GEKON modela u robotici 24
2.2. Etički problemi pri korištenju GEKON modela 26
2.3. Struktura GEKON modela 27
2.3.1. Generator 28
2.3.2. Diskriminator 30
2.3.3. Funkcija troška GEKON modela 32
2.3.4. Treniranje cijelog modela 35
2.4. Optimizacija GEKON modela 36
2.4.1. Normalizacija serije podataka 36
2.4.2. Konvolucijski GEKON model 39
2.4.3. Wasserstein GEKON model 44
2.4.4. Uvjetna i kontrolirana generacija 47
2.5. Primjeri korištenja GEKON modela 49
3. ZAKLJUČAK 53
LITERATURA 54

POPIS SLIKA

Slika 1.1. Građa biološkog neurona ... 2
Slika 1.2. Struktura umjetnog neurona ... 3
Slika 1.3. Step funkcija ... 4
Slika 1.4. Linearna funkcija ... 5
Slika 1.5. Sigmoidalna funkcija ... 6
Slika 1.11. Neuronska mreža s više neurona u svakom sloju ... 17
Slika 2.1. Podjela generativnih modela ... 20
Slika 2.2. Napredak u korištenju GEKON modela pri generiranju realističnih fotografija ljudskog lica ... 21
Slika 2.3. Uporaba GEKON modela u programu *Photoshop* ... 22
Slika 2.4. Uporaba GEKON modela za izradu filtera unutar aplikacije *FaceApp* ... 22
Slika 2.5. Usporedba GEKON modela s drugim tehnikama korištenim za super-rezoluciju ... 23
Slika 2.6. Korištenje GEKON mreža pri izradi 3D modela ... 23
Slika 2.7. Prikaz slike u pokretu korištenjem GEKON modela ... 24
Slika 2.8. Ulaz i izlaz generatora ... 29
Slika 2.9. Proces učenja generatora ... 29
Slika 2.10. Vizualni prikaz procesa učenja klasifikatora ... 31
Slika 2.11. Proces učenja diskriminatora ... 32
Slika 2.12. Ovisnost funkcije troška o predikciji kada je željena vrijednost diskriminatora jednaka 1 ... 34
Slika 2.13. Ovisnost funkcije troška o predikciji kada je željena vrijednost diskriminatora jednaka 0 ... 35
Slika 2.14. Normalizacija serije podataka ... 37
Slika 2.15. Kovarijantni pomak ... 37
Slika 2.16. Primjer izvođenja konvolucije ... 40
Slika 2.17. Primjer sažimanja korištenjem max funkcije ... 42
Slika 2.18. Primjer naduzorkovanja ponavljanjem ... 42
Slika 2.19. Primjer transponirane konvolucije ... 43
Slika 2.20. Vizualni prikaz Lipschitz kontinuiteta ... 46
Slika 2.21. Primjer tranzicije između dva generirana primjera ... 48
Slika 2.22. Korištenje klasifikatora za generiranje traženih svojstava ... 49
Slika 2.23. Primjer podataka za treniranje ... 50
Slika 2.24. Rezultati klasičnog gekon modela ... 50
Slika 2.25. Rezultati dubokog konvolucijskog GEKON modela ... 51
Slika 2.26. Rezultati Wasserstein GEKON modela ... 52

POPIS TABLICA

Tablica 1. Utjecaj predikcije modela na prvi dio funkcije troška 33
Tablica 2. Utjecaj predikcije modela na drugi dio funkcije troška 34
Tablica 3. Razlike između uvjetne i kontrolirane generacije 48

POPIS OZNAKA

$a^{(L)}$ – aktivacijska funkcija neurona u posljednjem sloju

$a_j^{(L)}$ – aktivacijska funkcija *j*-tog neurona u posljednjem sloju

$a_k^{(L-1)}$ – aktivacijska funkcija *k*-tog neurona u prethodnom sloju

b – bias neurona

$b^{(L)}$ – bias neurona u posljednjem sloju

$b_j^{(L)}$ – bias *j*-tog neurona u posljednjem sloju

$b_k^{(L-1)}$ – bias *k*-tog neurona u prethodnom sloju

c – koeficijent smjera linearne funkcije

$c(x)$ – predikcija kritičara za originalne primjere

$c(\hat{x})$ – predikcija kritičara za generirane primjere

$d(x)$ – predikcija diskriminatora za originalne primjere

$d(\hat{x})$ – predikcija diskriminatora za generirane primjere

$E(x)$ – pretpostavljena izlazna vrijednost za danu ulaznu vrijednost x

$E(z_i^{(L)})$ – pretpostavljena srednja vrijednost od $z_i^{(L)}$

$h(x_i, \theta)$ – predikcija modela

J – trošak neuronske mreže

J_k – trošak neuronske mreže za *k*-ti primjer

m – broj koraka učenja mreže

n – broj korištenih primjera u jednoj epohi

$n^{(L)}$ – broj neurona u posljednjem sloju

$Var\left(z_i^{(L)}\right)$ – varijanca vrijednosti $z_i^{(L)}$

w – težinski faktor neurona

$w^{(L)}$ – težinski faktor koji povezuje neuron posljednjeg sloja s neuronom prethodnog sloja

$w_{jk}^{(L)}$ – težinski faktor koji povezuje *j-ti* neuron posljednjeg sloja s *k*-tim neuronom prethodnog sloja

x – ulazna vrijednost

x_i – i-ta ulazna vrijednost

$\hat{x}'$ – vrijednosti novog primjera kod gradijentne kazne

z – ulazna vrijednost neurona u aktivacijsku funkciju

$z^{(L)}$ – ulazna vrijednost neurona u aktivacijsku funkciju u posljednjem sloju

$z_i^{(L)}$ – ulazna vrijednost neurona u aktivacijsku funkciju u posljednjem sloju za *i*-ti primjer

$\hat{z}_i^{(L)}$ – normalizirana ulazna vrijednost neurona u aktivacijsku funkciju u posljednjem sloju za *i*-ti primjer

$z_j^{(L)}$ – ulazna vrijednost *j*-tog neurona u aktivacijsku funkciju u posljednjem sloju

y_i – željeni izlaz mreže u *i*-tom koraku učenja

$y_i^{(L)}$ – ulazna vrijednost neurona u aktivacijsku funkciju u posljednjem sloju za *i*-ti primjer nakon provođenja normalizacije podataka

$\hat{y}_i$ – dobiveni izlaz mreže u *i*-tom koraku učenja

β – faktor pomaka

γ – faktor skaliranja

ε – konstanta standardne devijacije

ε' - faktor gradijentne kazne

θ – parametri neuronske mreže (težinski faktori i bias-i svih neurona unutar mreže)

λ – regularizacijski faktor gradijentne kazne

$\mu_{z_i^{(L)}}$ – aritmetička sredina svih $z_i^{(L)}$ unutar promatrane iteracije grupnog učenja

σ – sigmoidalna aktivacijska funkcija

$\sigma^2_{z_i^{(L)}}$ – varijanca $z_i^{(L)}$ unutar promatrane iteracije grupnog učenja

∇J – vektor gradijenta funkcije troška

SAŽETAK

Generativne kontradikcijske neuronske mreže jedan su od najuspješnijih generativnih modela strojnog učenja. Njihovo izvođenje se temelji na natjecanju dviju neuronskih mreža u hipotetskoj igri s nultom sumom. Za razliku od mnogih drugih generativnih modela, GEKON mreže ne koriste funkcije gustoće vjerojatnosti kako bi opisale distribuciju vjerojatnosti već generiraju nove vrijednosti isključivo na temelju proučavanja značajki na velikim bazama podataka. U ovom radu prikazana je temeljna teorijska i matematička pozadina neuronskih mreža, nakon čega je detaljnije opisana struktura GEKON modela i mogućnost njegove optimizacije za specifične zadatke. U radu su također prikazani primjeri uporabe takvog modela s naglaskom na primjenu u robotici, ali i etička pitanja koja nastaju zbog iznimno realističnih rezultata takvih mreža. Na kraju rada prikazana je i usporedba implementacije triju vrsta GEKON modela za generiranje realističnih „rukom napisanih“ znamenki.

Ključne riječi: Generativne kontradikcijske neuronske mreže, neuronske mreže, umjetna inteligencija, strojno učenje, generativni model

SUMMARY

Generative adversarial networks are one of the most successful generative models of machine learning. Their performance is based on the competition of two neural networks in a hypothetical zero-sum game. Unlike many other generative models, GANs do not use probability density functions to describe the probability distribution, but rather generate new values solely based on the study of features on large databases. This paper presents the basic theoretical and mathematical background of neural networks, after which the structure of GANs and the possibilities of its optimization for specific tasks are described in more detail. The paper also presents examples of using such a model with an emphasis on the applications in robotics, but also ethical issues that arise due to the extremely realistic results of such networks. A comparison of the implementation of three types of GANs for generating realistic "handwritten" digits is presented at the end of the paper.

Key words: Generative adversarial networks, neural networks, artificial intelligence, machine learning, generative models

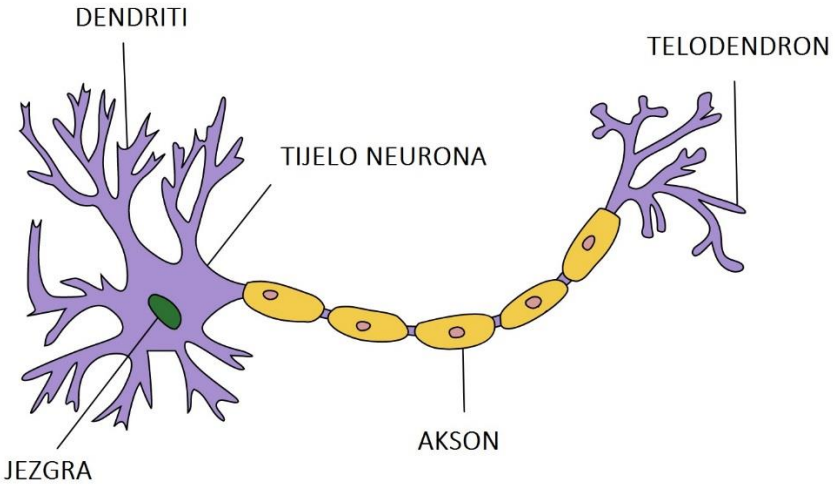
1. UVOD

U današnjem svijetu velika količina podataka obrađuje se digitalnim računalima. To je prvenstveno tako zato što računala posjeduju sposobnost vrlo brze i precizne obrade podataka kada se radi o podacima čija se obrada može jednostavno automatizirati. No, još uvijek postoje brojni zadaci kod čijih se izvođenja ne možemo osloniti isključivo na automatizaciju samog procesa. Kod takvih zadataka ljudski mozak još uvijek prednjači računalima. Kako bi osposobili računala da izvode i takve zadatke, bilo je potrebno osmisliti drugačiji koncept njihovog rada koji bi im omogućio imitaciju rada ljudskog mozga. To je područje kojim se bavi umjetna inteligencija. Ispravno je ovdje postaviti pitanje što uopće znači inteligencija. Možemo li sustav koji oponaša rad ljudskog mozga uopće nazvati inteligentnim? Po čemu se točno rad ljudskog mozga razlikuje od rada računala i mogu li računala uistinu imitirati takav proces uz davanje zadovoljavajućih rezultata? Čovjek posjeduje sposobnost iznošenja ispravnih hipoteza na temelju iskustva te donošenja ispravnih zaključaka na temelju logičkih pravila, a bitno je istaknuti i sposobnost učenja, koji u aspektu umjetne inteligencije shvaćamo kao mogućnost obavljanja promjene nad samim sobom.

Ideja umjetnih neuronskih mreža potiče iz 1943. godine, kada Warren S. McCulloch i Walter H. Pitts objavljuju rad pod nazivom „A logical calculus of the ideas immanent in nervous activity“ [1] u kojem izlažu prvi matematički model neuronske mreže. U to vrijeme procesna moć računala nije bila dosegla potrebnu razinu za implementaciju predložene strukture umjetne neuronske mreže. Međutim, već nešto manje od desetljeća kasnije se pojavljuju i prva praktična rješenja. Današnja računala posjeduju znatno veću procesnu moć od ondašnjih, a sam razvoj hardware-a i software-a, ali i same strukture današnjih računala omogućio je osnutak i daljnje usavršavanje brojnih algoritama strojnog učenja koji se temelje na radu umjetnih neuronskih mreža.

1.1 Biološki neuron

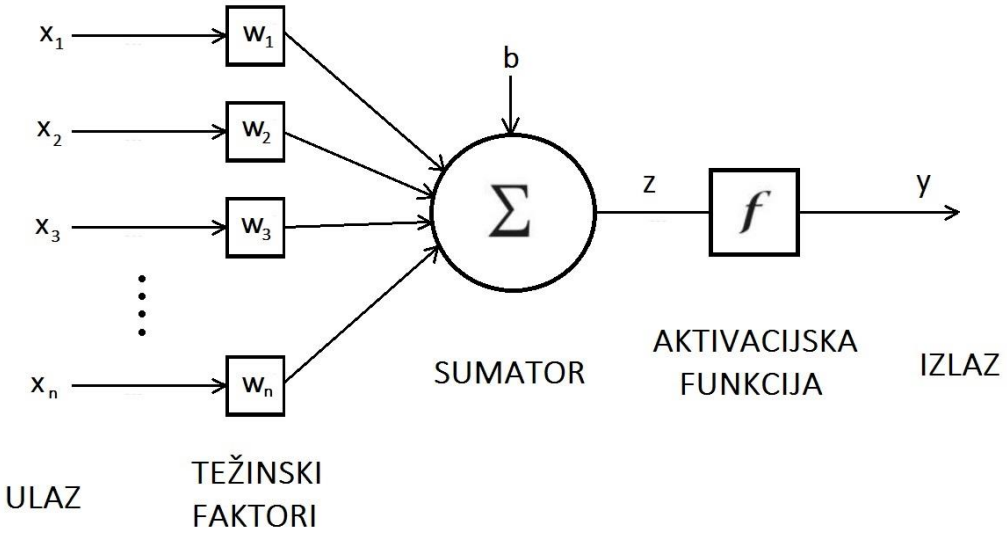
Neuron (živčana stanica) predstavlja sastavni dio živčanog sustava ljudskog tijela. U mozgu ih se nalazi oko 100 milijardi te je svaki neuron u prosjeku povezan sa 10^4 susjednih neurona. Također postoje brojne vrste bioloških neurona s obzirom na specifičnu funkciju koju obavljaju. Građa svakog neurona može se podijeliti na tri osnovna dijela: tijelo stanice, dendrite i akson. Informacija koju neuron sadrži pohranjena je u tijelu stanice (soma) u obliku električnog potencijala između unutrašnjeg i vanjskog dijela stanice. Dendriti su kraći produžeci na stanici na koje putem sinaptičkog kontakta dolazi informacija od druge stanice, a akson je obično duži produžetak na stanici te je njegova funkcija prenositi živčane impulse na druge živčane stanice ili izvršne organe. Slika 1.1 prikazuje građu neurona s naznačenim dijelovima stanice.



Slika 1.1. Građa biološkog neurona [2]

1.2. Umjetni neuron

McCulloch-Pitts model umjetnog neurona, koji se još naziva i *Treshold Logic Unit* (TLU), pokušava imitirati funkcionalnost biološkog neurona prateći istu analogiju. Informacija u svakom neuronu spremljena je u obliku numeričkog iznosa koji se dobiva sumacijom signala svih neurona iz prethodnog sloja pomnoženih s težinskim faktorom. Svaki neuron također posjeduje aktivacijsku funkciju u koju ulazi dobivena suma.



Slika 1.2. Struktura umjetnog neurona

Na slici 1.2 prikazana je struktura jednog umjetnog neurona. Vrijednosti x_1, x_2, ... x_n predstavljaju ulaze u neuron koji mogu biti početni ulazi u neuronsku mrežu ako se radi o nultom sloju ili izlazi prethodnog sloja ako se radi o nekom drugom sloju unutar mreže. Ulazni signali općenito su realni brojevi u intervalu [-1,1], [0,1] ili elementi iz {0,1} ako se radi o Booleovom ulazu. Težinski faktori w_1, w_2, ... w_n koji množe ulaze vezani su za prikazani umjetni neuron i njihova vrijednost određuje osjetljivost neurona na svaki od zasebnih ulaza. Težinski faktori mogu biti pozitivni i negativni brojevi što znači da ulazni signali mogu imati pozitivan i negativan učinak na izlaz umjetnog neurona. Svaki neuron posjeduje i svoj *bias* kojeg označavamo s b i dodajemo na već spomenutu sumu. Izlaz iz sumatora za dani neuron se određuje sljedećim izrazom:

$$z = \sum_{i=0}^{n} (w_i\, x_i) + b\,. \tag{1.1}$$

Dobivena suma, koju označavamo sa z, ulazi u aktivacijsku funkciju koja opisuje prag osjetljivosti neurona. Izlaz iz aktivacijske funkcije označavamo s y i od predstavlja izlaznu vrijednost promatranog neurona, ali i ulaznu vrijednost neurona sljedećeg sloja ako se ne radi o posljednjem sloju.

$$y = f(z)\,. \tag{1.2}$$

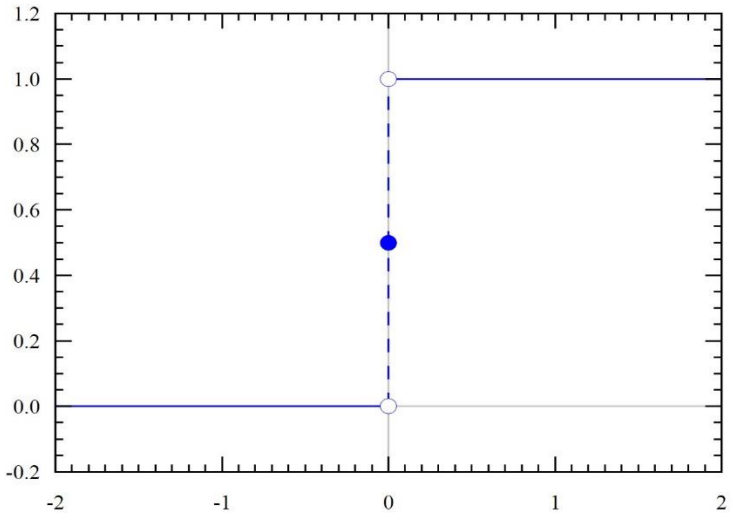
1.3. Aktivacijske funkcije

Aktivacijske funkcije mogu poprimiti razne oblike. U nastavku će biti prikazane: step, linearna, sigmoidalna i ReLu funkcija.

1.3.1. Step funkcija

Step funkcija predstavlja najjednostavniju aktivacijsku funkciju. Ako je izlazna vrijednost *z* veća od zadane vrijednosti *t* step funkcije, tada izlaz step funkcije postaje jednak 1, a ako je vrijednost *z* manja od *t* će izlaz biti jednak 0.
Ako uzmemo da je $t = 0$, dobivamo sljedeće:

$$y = f(z) = \begin{cases} 1, \text{ako je } z \geq 0 \\ 0, \text{ako je } z < 0. \end{cases} \tag{1.3}$$



Slika 1.3. Step funkcija [3]

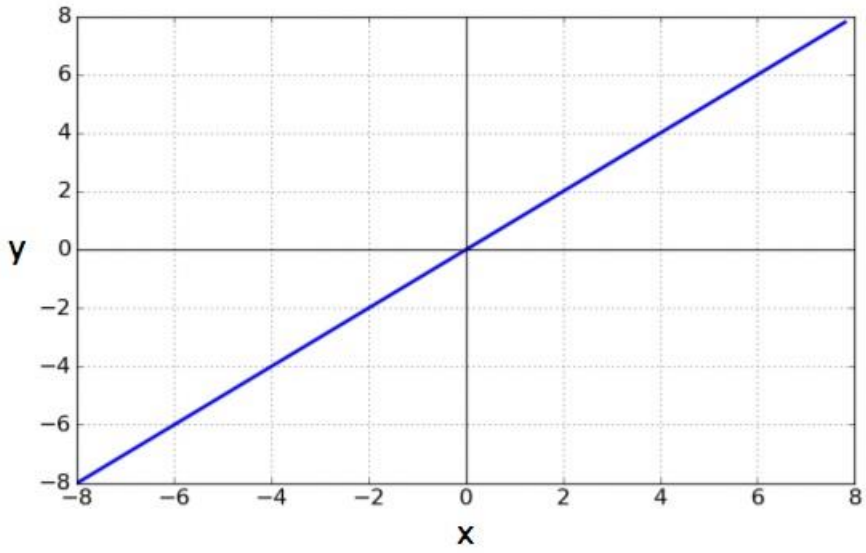
Negativna strana korištenja step funkcije je što ne dozvoljava više različitih izlaznih vrijednosti zbog čega se kod korištenja takve funkcije ne može klasificirati ulaze u više različitih klasa.

1.3.2. Linearna funkcija

Linearna aktivacijska funkcija ima oblik:

$$A(x) = cx. \tag{1.4}$$

Gdje je ulazna varijabla funkcije $x=z$.



Slika 1.4. Linearna funkcija

Linearna aktivacijska funkcija daje izlazni signal proporcionalan ulaznom. Za razliku od step funkcije koja za izlaz daje samo 0 ili 1, linearna funkcija može imati više različitih izlaznih vrijednosti. Međutim, kod linearne aktivacijske funkcije postoje dva problema:

1. Linearna aktivacijska funkcija ne omogućava korištenje algoritma sa širenjem pogreške unazad (*engl. backpropagation)* kod treniranja modela. Razlog tome je što je derivacija linearne funkcije konstanta iz čega se ne može izvući nikakva informacija o tome kakvi bi parametri davali bolje rješenje.

2. Bez obzira koliko slojeva neuronska mreža posjeduje, zadnji će sloj biti linearna funkcija prvog sloja. To je zato što je linearna kombinacija linearnih funkcija još uvijek samo – linearna funkcija. Dakle, linearna aktivacijska funkcija pretvara svaku neuronsku mrežu u jednoslojnu.

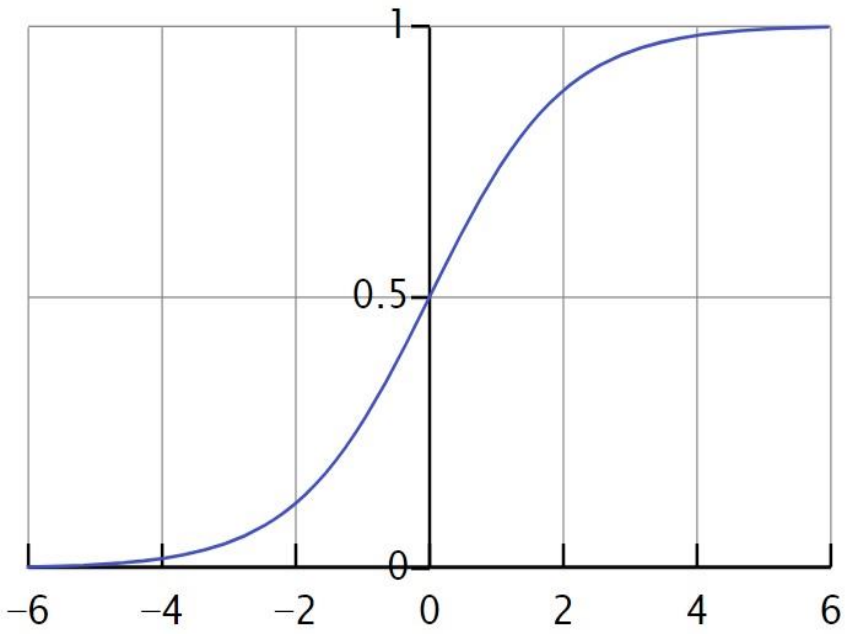
Za razliku od prethodno navedenih linearnih funkcija, nelinearne funkcije kao što su sigmoidalna i ReLu funkcija, dopuštaju korištenje algoritma sa širenjem pogreške unazad i korištenje neuronskih mreža s više slojeva.

1.3.3. Sigmoidalna funkcija

Sigmoidalna funkcija je definirana izrazom:

$$A(x) = \frac{1}{1 + e^{-x}} . \tag{1.5}$$

Gdje je ulazna varijabla funkcije x=z. Sigmoidalna funkcija se također često označava sa σ.



Slika 1.5. Sigmoidalna funkcija [4]

Prednosti korištenja sigmoidalne aktivacijske funkcije su:

1. Gradijent sigmoidalne funkcije je takav da ne dopušta velike "skokove" u izlaznim vrijednostima.
2. Izlazne vrijednosti se nalaze unutar skupa [0,1].
3. Ulazne vrijednosti iznad 2 i ispod -2 su kao izlaz vrlo bliske vrijednostima 1 i 0 što omogućuje jednostavne predikcije.

Nedostaci korištenja sigmoidalne funkcije su:

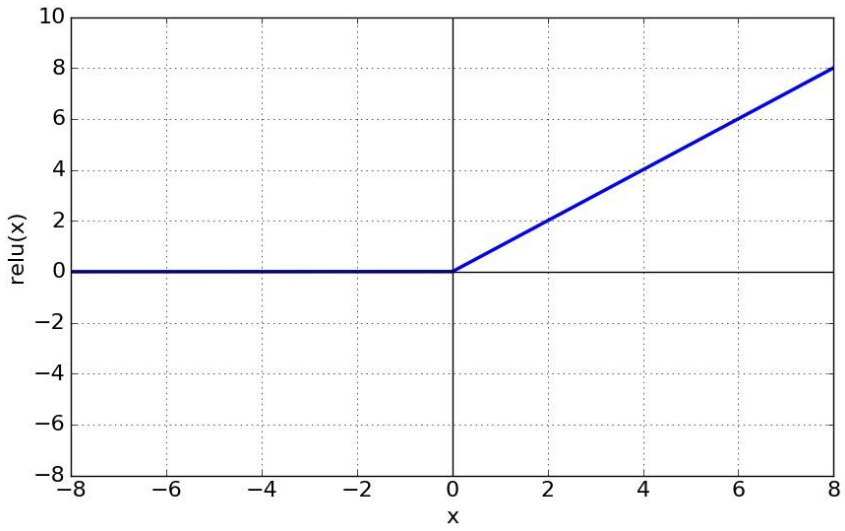
1. Kod vrlo visokih i niskih ulaznih vrijednosti nema gotovo nikakve razlike kod izlaza što prouzročuje gubitak značajnog gradijenta. Zbog ovoga neuronskoj mreži može trebati previše vremena da ispravi predikcije.
2. Daje izlaze koji nisu centrirani oko nule. To stvara probleme kod optimizacije mreže korištenjem algoritma sa širenjem pogreške unazad jer gradijent uvijek ispadne istog predznaka.
3. Računski je kompleksnija od ostalih funkcija koje se koriste što može usporiti izvođenje modela u praksi.

Funkcija tangens hiperbolni se često koristi umjesto sigmoidalne jer preuzima sličan oblik. Jedina bitna razlika između sigmoidalne funkcije i tangensa hiperbolnog kada se koriste kao aktivacijske funkcije je što tangens hiperbolni daje izlaz koji se nalazi unutar skupa [-1,1].

1.3.4. ReLu funkcija

ReLu (engl. *Rectified linear unit*) aktivacijska funkcija je definirana izrazom:

$$y = f(z) = \begin{cases} z, \text{ako je } z \geq 0 \\ 0, \text{ako je } z < 0. \end{cases} \tag{1.6}$$



Slika 1.6. ReLu funkcija [5]

Prednosti korištenja ReLu aktivacijske funkcije:

1. Računski je jednostavna što ubrzava i olakšava izvođenje neuronskoj mreži. Zbog ovoga se u praksi često koristi umjesto sigmoidalne.

2. Bez obzira na to što izgleda kao linearna funkcija, ReLu aktivacijska funkcija posjeduje derivacijske funkcije što omogućava korištenje algoritma sa širenjem pogreške unazad.

Jedini pravi nedostatak korištenja ReLu funkcije je što gradijent funkcije postane nula kada se ulazna vrijednost približava nuli ili je negativna. Međutim, postoji varijanta ove funkcije pod nazivom *Leaky ReLu* koja poprima blagi nagib i za negativne ulazne vrijednosti.

1.4. Funkcije troška

Neuronske mreže posjeduju svojstvo nelinearnosti, što znači da kod njih nije moguće precizno izračunati parametre za rješavanje zadanog problema. Da bi se uklonila greška na izlazu neuronske mreže, koriste se iterativne metode za postepeno smanjenje funkcije troška. Spomenute metode koriste izračun gradijenta koji govori o potrebnim promjenama nad

parametrima neuronske mreže s ciljem minimiziranja greške na izlazu. S obzirom na to da su parametri neuronske mreže na samom početku procesa treniranja uzeti kao nasumične vrijednosti, ne može se znati hoće li gradijent dovesti do konvergencije prema globalnom ili lokalnom minimumu. Najčešće se kod izračuna pogreške koristi određivanje srednje kvadratne pogreške:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2. \tag{1.7}$$

Gdje su:

θ – parametri neuronske mreže (težinski faktori i bias-i svih neurona unutar mreže)

m – broj koraka učenja mreže

$\hat{y}_i$ – dobiveni izlaz mreže u i-tom koraku učenja

y_i – željeni izlaz mreže u i-tom koraku učenja

Često se kod računanja pogreške koristi i gubitak unakrsne entropije kako slijedi:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]. \tag{1.8}$$

1.5. Postupak učenja

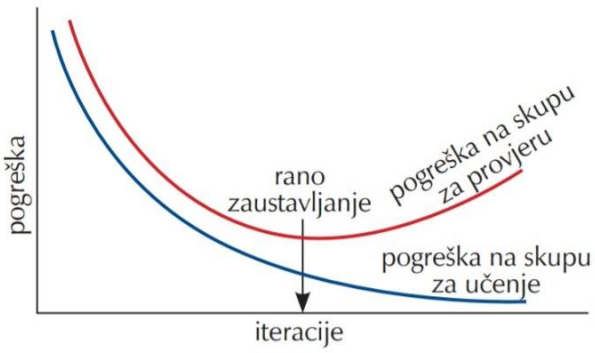
U slučaju kada se koriste složenije aktivacijske funkcije, kao što je sigmoidalna funkcija, ili kada se dopušta rad s realnim brojevima, često se gubi mogućnost nadziranja obrade podataka unutar mreže. Iz tog se razloga prvo izvodi postupak učenja, odnosno treniranja mreže, u kojem mreža pronalazi potrebnu kombinaciju parametara koji će iz zadanog ulaza u mrežu dati zadovoljavajući izlaz. U tom slučaju je prije učenja mreže potrebno precizno definirati arhitekturu mreže. Trajanje postupka učenja mreže varira zavisno o složenosti zadataka koji mreža mora izvršiti, arhitekturi mreže i željenoj kvaliteti izlaznih podataka. Dakle, pod postupkom učenja mreže podrazumijeva se da se radi o iterativnom postupku smanjivanja

pogreške izlaznih podataka u odnosu na očekivane rezultate. Ovisno o tome je li točan izlaz iz mreže za vrijeme postupka učenja poznat, pa se pri učenju koristi uz svaki ulaz, ili je izlaz iz mreže nepoznat, razlikuju se dvije vrste učenja mreža:

1. Nadzirano učenje (*engl. supervised learning)* – kod učenja mreže se daju ulazi u mrežu s točno definiranim očekivanim izlazima. Kod ovakvog učenja se koristi baza s označenim podatcima (*engl. labeled data*).

2. Nenadzirano učenje (*engl. unsupervised learning*) – kod učenja model sam pronalazi poveznice unutar neoznačenih podataka (*eng. unlabeled data*). Algoritmi koji se koriste kod nenadziranog učenja su uglavnom kompleksniji, obzirom da neuronska mreža raspolaže s manje ulaznih podataka i očekivani rezultat mreže nije poznat.

Baza podataka za učenje mreže se često dijeli na: skup podataka za učenje, skup podataka za testiranje i skup podataka za provjeru. Skup podataka za učenje se koristi tijekom samog procesa učenja, dakle tijekom iterativnog postupka podešavanja parametara mreže (težinskih faktora i biasa). Skup za testiranje se koristi za vrijeme učenja kako bi se ustanovilo stanje izlaza neuronske mreže u ovisnosti o trenutnim parametrima. Ponekad je potrebno rano zaustaviti (*engl. early stopping*) postupak učenja neuronske mreže u određenom trenutku, kako se model ne bi pretrenirao (*engl. overfitting*). Naime, ponekad neuronska mreža izgubi svojstvo generalizacije nakon određenog broja iteracija u procesu učenja. To znači da neuronska mreža nauči obrađivati isključivo podatke iz skupa podataka za učenje te izgubi sposobnost izvršavanja kvalitetne obrade nad podacima izvan te skupine. Pomoću skupine za testiranje mreže se može odrediti iteracija u kojoj dobiveni izlaz najmanje odstupa od željenog (slika 1.7). Naposljetku se skupom podataka za provjeru provjerava točnost obrade podataka.



Slika 1.7. Rano zaustavljanje (*engl. early stopping*) [6]

Postupak učenja neuronskih mreža može se podijeliti ovisno o broju predočenih primjera unutar jedne iteracije učenja pa tako razlikujemo:

1. Pojedinačno učenje (*engl. on-line training*) – kod ovakvog tipa učenja se prilagodba parametara neuronske mreže vrši nakon svakog pojedinačnog primjera za učenje. Odnosno, jedna iteracija ovdje predstavlja samo jedan uzeti primjer iz baze podataka.

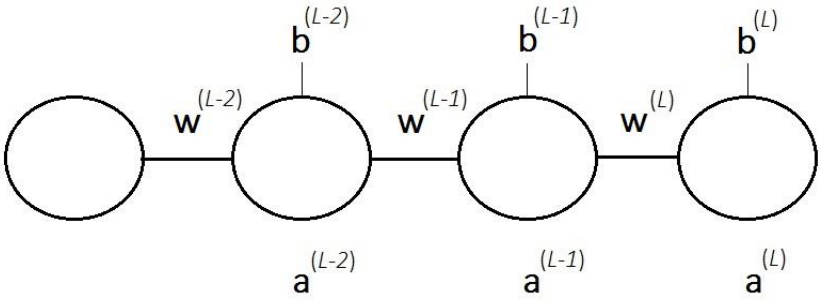
2. Grupno učenje (*engl. batch training*) – kod ovakvog tipa učenja se prilagodba parametara neuronske mreže vrši nakon svake epohe. Epoha predstavlja jedan cjelokupni skup za učenje. Dakle, kod grupnog učenja se iteracija podudara s epohom i neuronska mreža prilagođava parametre tek nakon nekoliko izvedenih primjera.

1.6. Algoritam sa širenjem pogreške unazad

Kao što je već ranije spomenuto, učinkovita i popularna metoda učenja višeslojnih mreža se naziva algoritam sa širenjem pogreške unazad. U ovom će se poglavlju prikazati matematička pozadina algoritma sa širenjem pogreške unazad na neuronskim mrežama koje posjeduju samo jedan neuron u svakom sloju, a zatim analogno tome na neuronskim mrežama koje posjeduju više neurona u svakom sloju. Kako bi neuronske mreže mogle rješavati probleme iza kojih stoje visoko nelinearne funkcije, potrebno je da i same aktivacijske funkcije neurona također budu nelinearne, ali i da posjeduju derivabilne funkcije kako bi se mogla primijeniti gradijentna metoda kod postupka učenja mreže. Iz tog razloga će se u sljedećim primjerima za aktivacijske funkcije uzeti sigmoidalna funkcija.

1.6.1. Neuronska mreža s jednim neuronom u svakom sloju

Kako bi mogli stvoriti poveznicu između svih prethodno navedenih pojmova i shvatiti matematičku pozadinu na kojoj se temelje neuronske mreže, uzet ćemo neuronsku mrežu u kojoj se u svakom sloju nalazi jedan neuron i prikazati na koji način svaki od parametara takve mreže utječe na funkciju troška. Označimo li aktivacijsku funkciju neurona u posljednjem sloju s $a^{(L)}$, a u prethodnim slojevima s $a^{(L\text{-}1)}$ i $a^{(L\text{-}2)}$ dobit ćemo prikazanu strukturu (slika 1.8).



Slika 1.8. Neuronska mreža s jednim neuronom u svakom sloju

Kao što se može vidjeti sa slike, ova neuronska mreža je definirana sa 6 parametara. Od tih 6 parametara 3 su težinski faktori: $w^{(L)}$, $w^{(L-1)}$ i $w^{(L-2)}$, a preostala 3 parametra bias-i svakog zasebnog neurona: $b^{(L)}$, $b^{(L-1)}$ i $b^{(L-2)}$.

Izlazna vrijednost ove neuronske mreže označena je s $a^{(L)}$. Ako željenu vrijednost označimo s y i ubacimo obje vrijednosti u funkciju troška (1.7) dobit će se sljedeće:

$$J_0(\theta) = (a^{(L)} - y)^2. \tag{1.9}$$

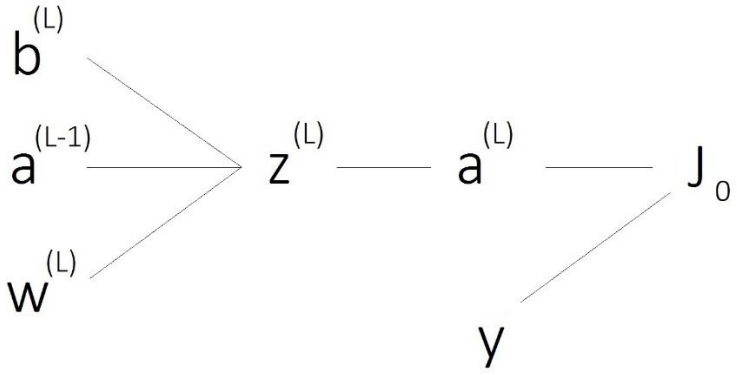
Obzirom da je za aktivacijsku funkciju odabrana sigmoidalna aktivacijska funkcija, do izlazne vrijednosti neurona u posljednjem sloju se dolazi ubacivanjem vrijednosti $z^{(L)}$ u sigmoidalnu funkciju:

$$a^{(L)} = \sigma\left(z^{(L)}\right). \tag{1.10}$$

Vrijednost $z^{(L)}$ dobiva se iz aktivacije prethodnog sloja, težinskog faktora koji povezuje izlazni neuron s neuronom iz prethodnog sloja $w^{(L)}$ i bias-om kojeg dodaje neuron posljednjeg sloja $b^{(L)}$:

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}. \tag{1.11}$$

Zavisnost J_0 o prethodnim parametrima unutar mreže može se vizualizirati na sljedeći način:



Slika 1.9. Vizualni prikaz zavisnosti J_0 o parametrima mreže

Kao što se vidi na slici 1.9, promjene na parametrima mreže $w^{(L)}$ i $b^{(L)}$, ali i aktivacijske funkcije $a^{(L-1)}$ koja u sebi sadrži težinske faktore i bias-e prethodnih slojeva indirektno utječu na promjenu J_0.

Utjecaj promjene težinskog faktora $w^{(L)}$ na J_0 određuje se kako slijedi:

$$\frac{\delta J_0}{\delta w^{(L)}} = \frac{\delta z^{(L)}}{\delta w^{(L)}} \frac{\delta a^{(L)}}{\delta z^{(L)}} \frac{\delta J_0}{\delta a^{(L)}}. \tag{1.12}$$

Kako bi se došlo do konačnog izraza je potrebno parcijalno derivirati izraze (1.9), (1.10) i (1.11).

Parcijalnom derivacijom funkcije (1.9) po $a^{(L)}$ se dobiva:

$$\frac{\delta J_0}{\delta a^{(L)}} = 2(a^{(L)} - y). \tag{1.13}$$

Parcijalnom derivacijom funkcije (1.10) po $z^{(L)}$ se dobiva:

$$\frac{\delta a^{(L)}}{\delta z^{(L)}} = \sigma'(z^{(L)}). \tag{1.14}$$

Parcijalnom derivacijom funkcije (1.11) po $w^{(L)}$ se dobiva:

$$\frac{\delta z^{(L)}}{\delta w^{(L)}} = a^{(L-1)}. \tag{1.15}$$

Nakon uvrštavanja (1.13), (1.14) i (1.15) u (1.12) dobiva se konačan izraz za utjecaj promjene težinskog faktora $w^{(L)}$ na J_0:

$$\frac{\delta J_0}{\delta w^{(L)}} = a^{(L-1)}\sigma'(z^{(L)})2(a^{(L)} - y). \tag{1.16}$$

Važno je napomenuti da izraz (1.16) vrijedi za slučaj provođenja samo jednog primjera kroz neuronsku mrežu. Prava funkcija troška predstavljala bi prosjek svih primjera koji su korišteni za učenje mreže pa bi prema tome utjecaj promjene težinskog faktora $w^{(L)}$ na ukupni J bio definiran kao:

$$\frac{\delta J}{\delta w^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1} \frac{\delta J_k}{\delta w^{(L)}}. \tag{1.17}$$

Utjecaj promjene bias-a $b^{(L)}$ na J_0 određuje se kako slijedi:

$$\frac{\delta J_0}{\delta b^{(L)}} = \frac{\delta z^{(L)}}{\delta b^{(L)}}\frac{\delta a^{(L)}}{\delta z^{(L)}}\frac{\delta J_0}{\delta a^{(L)}}. \tag{1.18}$$

Parcijalnom derivacijom funkcije (1.11) po $b^{(L)}$ se dobiva:

$$\frac{\delta z^{(L)}}{\delta b^{(L)}} = 1. \tag{1.19}$$

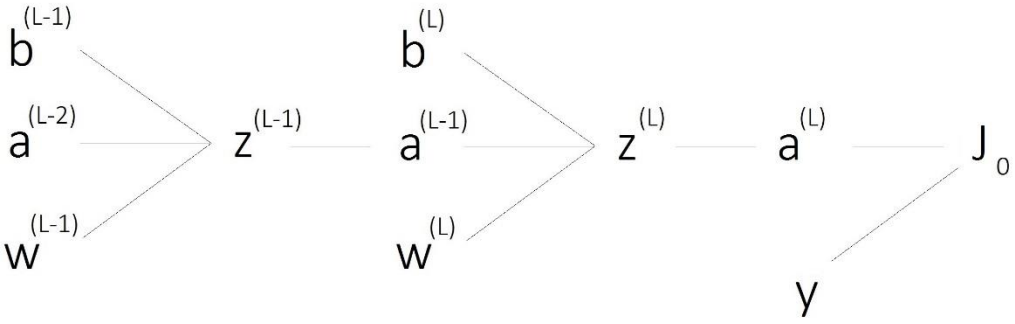
Nakon uvrštavanja (1.13), (1.14) i (1.19) u (1.18) dobiva se konačan izraz za utjecaj promjene bias-a $b^{(L)}$ na J_0:

$$\frac{\delta J_0}{\delta w^{(L)}} = \sigma'(z^{(L)})2(a^{(L)} - y). \tag{1.20}$$

Analogno utjecaju promjene težinskog faktora $w^{(L)}$ na sveukupni *J*, kod utjecaja promjene bias-a na sveukupni *J* je također potrebno promatrati sve primjere koji se koriste za učenje mreže pa je tako:

$$\frac{\delta J}{\delta b^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1} \frac{\delta J_k}{\delta b^{(L)}}. \tag{1.21}$$

Dobiveni izrazi (1.17) i (1.21) predstavljaju samo neke od članova vektora gradijenta koji se koristi za iterativno podešavanje parametara neuronske mreže. Kako bi se dobili utjecaji promjene parametara neurona koji se nalaze u prethodnim slojevima je potrebno proširiti sliku 1.9 na:



Slika 1.10. Vizualni prikaz zavisnosti J_0 o parametrima mreže (prošireno)

Da bi se dobili utjecaji težinskih faktora i bias-a prethodnih slojeva, ponavlja se isti postupak opisan s parametrima $w^{(L)}$ i $b^{(L)}$, ali se izrazi proširuju dublje u strukturu neuronske mreže. Otuda je algoritam sa širenjem pogreške unazad i dobio svoj naziv. Nakon izračuna svih utjecaja parametara na ishod mreže, vektor gradijenta poprima konačan oblik:

$$\nabla J = \begin{bmatrix} \frac{\delta J}{\delta w^{(1)}} \\ \frac{\delta J}{\delta b^{(1)}} \\ \vdots \\ \frac{\delta J}{\delta w^{(L)}} \\ \frac{\delta J}{\delta b^{(L)}} \end{bmatrix}. \tag{1.22}$$

Nekada je također korisno provjeriti utjecaj aktivacije prethodnog sloja:

$$\frac{\delta J_0}{\delta a^{(L-1)}} = \frac{\delta z^{(L)}}{\delta a^{(L-1)}} \frac{\delta a^{(L)}}{\delta z^{(L)}} \frac{\delta J_0}{\delta a^{(L)}}. \tag{1.23}$$

Parcijalnom derivacijom funkcije (1.11) po $a^{(L-1)}$ se dobiva:

$$\frac{\delta z^{(L)}}{\delta a^{(L-1)}} = w^{(L)}. \tag{1.24}$$

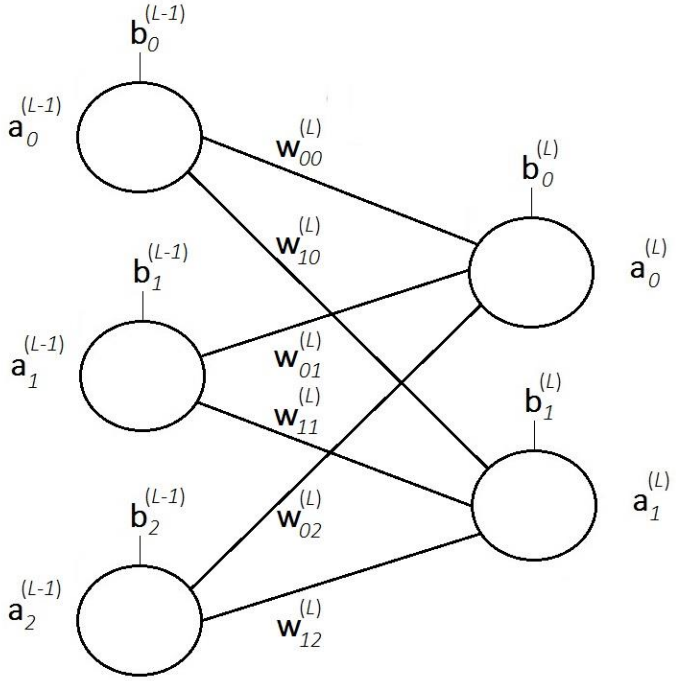
Nakon uvrštavanja (1.13), (1.14) i (1.24) u (1.23) dobiva se konačan izraz za utjecaj promjene aktivacije prethodnog sloja $a^{(L-1)}$ na J_0:

$$\frac{\delta J_0}{\delta a^{(L-1)}} = w^{(L)} \sigma'(z^{(L)}) 2(a^{(L)} - \hat{y}). \tag{1.25}$$

Analogno tome, može se dobiti utjecaj svakog parametra mreže na ishod. Naravno, složenost izraza će zavisiti o sloju mreže unutar kojeg se promatrani parametar nalazi. Drugim riječima, parametri koji se nalaze u prvim slojevima mreže imat će veći utjecaj na funkciju cijele mreže u odnosu na parametre iz zadnjih slojeva mreže.

1.6.2. Neuronska mreža s više neurona u svakom sloju

Neuronska mreža samo s jednim neuronom u svakom sloju pokazala se kao dobar primjer za bazično razumijevanje funkcije neuronskih mreža. Međutim, realniji primjer bile bi neuronske mreže koje u svakom sloju sadrže više od jednog neurona. U ovom će se poglavlju promotriti utjecaj parametara na ishod takve mreže i usporediti sličnosti i razlike u odnosu na neuronsku mrežu koja sadrži samo jedan neuron u svakom sloju obrađenu u prethodnom poglavlju. Ako analogno slici 1.8 prikažemo strukturu neuronske mreže s više neurona u svakom sloju, dobit ćemo sljedeći prikaz:



Slika 1.11. Neuronska mreža s više neurona u svakom sloju

S obzirom na to da se sada u svakom sloju nalazi više neurona, potrebno je koristiti dodatne indekse kako bi svaki parametar mreže bio jedinstveno označen. Kao i kod neuronske mreže s jednim neuronom u svakom sloju, posljednji sloj označen je s (L), dok su neuroni unutar posljednjeg sloja označeni indeksom j. Tako se aktivacije neurona posljednjeg sloja označavaju s $a_j^{(L)}$ i bias-i s $b_j^{(L)}$, gdje je u ovom slučaju $j=0,1$. Neuroni prethodnog sloja označeni su indeksom k pa se aktivacije prethodnog sloja označavaju s $a_k^{(L-1)}$ i bias-i s $b_j^{(L-1)}$, gdje je u ovom

slučaju *k=0,1,2.* Težinski faktori koji spajaju neurone između posljednjeg i predzadnjeg sloja, označeni su kao $w_{jk}^{(L)}$.

Izlazne vrijednosti ove neuronske mreže označene su s $a_j^{(L)}$. Ako željene vrijednosti označimo s y_j i ubacimo te vrijednosti u funkciju troška (1.7) dobit će se sljedeće:

$$J_0(\theta) = \sum_{j=0}^{n^{(L)}-1} (a_j^{(L)} - y_j)^2. \tag{1.26}$$

Gdje je:

$n^{(L)}$ – broj neurona u posljednjem, odnosno (L) sloju mreže

Do aktivacije neurona u posljednjem sloju $a_j^{(L)}$, dolazi se korištenjem sigmoidalne aktivacijske funkcije na vrijednosti $z_j^{(L)}$:

$$a_j^{(L)} = \sigma\left(z_j^{(L)}\right). \tag{1.27}$$

Vrijednost $z_j^{(L)}$ dobiva se iz aktivacija prethodnog sloja, težinskih faktora koji povezuju izlazni neuron s neuronima iz prethodnog sloja $w_{jk}^{(L)}$ i bias-om kojeg dodaje neuron posljednjeg sloja $b_j^{(L)}$:

$$z_j^{(L)} = w_{j0}^{(L)} a_0^{(L-1)} + w_{j1}^{(L)} a_1^{(L-1)} + w_{j2}^{(L)} a_2^{(L-1)} + b_j^{(L)}. \tag{1.28}$$

Ili ako se napiše općeniti izraz:

$$z_j^{(L)} = \sum_{k=0}^{n^{(L-1)}-1} (w_{jk}^{(L)} a_k^{(L-1)}) + b_j^{(L)}. \tag{1.29}$$

Analogno izrazu (1.12), utjecaj promjene težinskog faktora $w_{jk}^{(L)}$ na J_0 određuje se kako slijedi:

$$\frac{\delta J_0}{\delta w_{jk}^{(L)}} = \frac{\delta z_j^{(L)}}{\delta w_{jk}^{(L)}} \frac{\delta a_j^{(L)}}{\delta z_j^{(L)}} \frac{\delta J_0}{\delta a_j^{(L)}}. \tag{1.30}$$

Isto tako se analogno izrazu (1.18), utjecaj promjene bias-a $b_j^{(L)}$ na J_0 određuje se kako slijedi:

$$\frac{\delta J_0}{\delta b_j^{(L)}} = \frac{\delta z_j^{(L)}}{\delta b_j^{(L)}} \frac{\delta a_j^{(L)}}{\delta z_j^{(L)}} \frac{\delta J_0}{\delta a_j^{(L)}}. \tag{1.31}$$

Razlika između neuronske mreže s jednim neuronom u svakom sloju i neuronske mreže s više neurona u svakom sloju, najbolje se očituje kada se pogleda utjecaj aktivacije prethodnog sloja $a_k^{(L-1)}$ na J_0:

$$\frac{\delta J_0}{\delta a_k^{(L-1)}} = \sum_{j=0}^{n^{(L)}-1} \frac{\delta z_j^{(L)}}{\delta a_k^{(L-1)}} \frac{\delta a_j^{(L)}}{\delta z_j^{(L)}} \frac{\delta J_0}{\delta a_j^{(L)}}. \tag{1.32}$$

Do razlike dolazi zato što aktivacija prethodnog sloja $a_k^{(L-1)}$ utječe na obje aktivacije u posljednjem sloju, dakle $a_0^{(L)}$ i $a_1^{(L)}$, a one dalje obje imaju utjecaj na težinsku funkciju. Iz tog je razloga potrebno koristiti sumu kod promatranja utjecaja aktivacije prethodnog sloja na J_0.

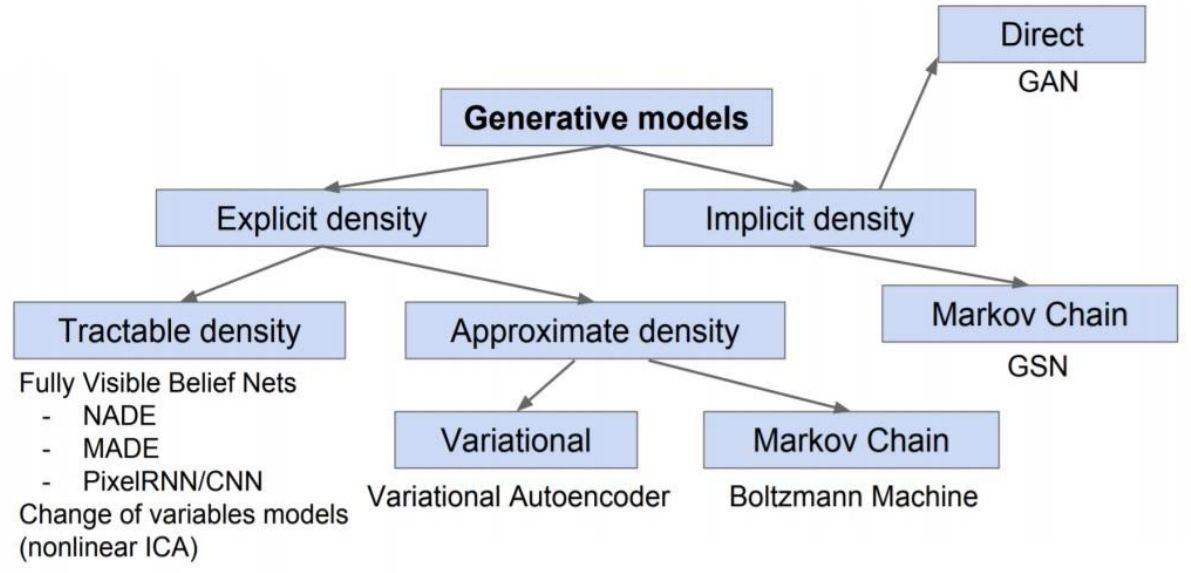
Nakon izračuna svih utjecaja parametara na ishod mreže, vektor gradijenta za neuronsku mrežu s više neurona u svakom sloju poprima konačan oblik:

$$\nabla J = \begin{bmatrix} \frac{\delta J}{\delta w_{00}^{(1)}} \\ \vdots \\ \frac{\delta J}{\delta w_{0k}^{(1)}} \\ \frac{\delta J}{\delta b_0^{(1)}} \\ \vdots \\ \frac{\delta J}{\delta w_{00}^{(L)}} \\ \vdots \\ \frac{\delta J}{\delta w_{jk}^{(L)}} \\ \frac{\delta J}{\delta b_j^{(1)}} \end{bmatrix}. \tag{1.33}$$

2. GENERATIVNE KONTRADIKCIJSKE NEURONSKE MREŽE

Kao što je već prethodno spomenuto, postupak učenja neuronskih mreža dijelimo na nadzirano učenje i nenadzirano učenje. Kod nadziranog učenja podaci preuzimaju oblik (ulaz, izlaz) = (x,y), tj. potrebno je pronaći preslikavanje $\hat{y} = f(x)$. Ako je izlazna vrijednost y diskretna/nebrojčana vrijednost radi se o postupku klasifikacije, a ako je y kontinuirana/brojčana vrijednost radi se o postupku regresije. Kod nenadziranog učenja su podaci dani bez ciljane vrijednosti. Kod takvog učenja potrebno je pronaći pravilnost u podacima, a tipično se koristi za: grupiranje (*engl. clustering*), procjenu gustoće (*engl. density estimation*) i smanjenje dimenzionalnosti (*engl. dimensionality reduction*). Generativni modeli koriste nenadzirani postupak učenja kako bi generirali nove primjere podataka koji se po određenim značajkama mogu klasificirati u istu skupinu podataka korištenu pri treniranju modela. Postoje dvije metode koje se koriste kod generiranja novih podataka:

1. Metoda generativnih modela koja koristi funkcije gustoće vjerojatnosti (*engl. probability density function*) kako bi se opisala distribucija vjerojatnosti (*engl. probability distribution*) generiranih podataka.
2. Metoda generativnih modela kod koje model ne definira funkciju gustoće vjerojatnosti već generira nove primjere isključivo na temelju proučavanja značajki na velikim bazama podataka.



Slika 2.1. Podjela generativnih modela [7]

Generativne kontradikcijske neuronske mreže (*engl. Generative adversarial networks*) su generativni model strojnog učenja kojeg 2014. godine predlaže Ian Goodfellow u radu pod naslovom „*Generative Adversarial Nets*“ [8]. Taj tip generativnog modela koristi drugu spomenutu metodu kod generiranja novih podataka. Dakle, umjesto definiranja funkcije gustoće vjerojatnosti, model se trenira korištenjem velikih baza podataka. Generativne kontradikcijske neuronske mreže (ili skraćeno GEKON model) koriste takvu metodu jer njihova funkcija nije opisati značajke koje objedinjuju navedeni skup primjera, već isključivo generirati nove primjere podataka jednake kvalitete onima koji su korišteni pri treniranju modela.

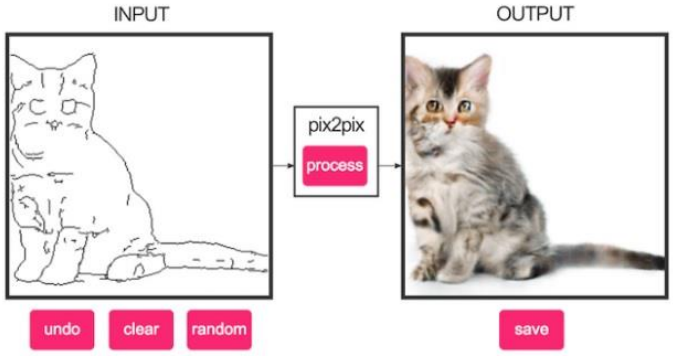
2.1. Primjena generativnih kontradikcijskih neuronskih mreža

GEKON model pokazao se kao jedan od najboljih modela za generiranje novih vrijednosti s vrlo širokim područjem primjene. Od 2014. godine kada nastaju pa sve do danas, još se uvijek usavršavaju algoritmi koji se koriste pri primjeni GEKON modela. U nastavku je prikazan napredak GEKON modela za generiranje realističnih fotografija ljudskog lica (slika 2.2).

Slika 2.2. Napredak u korištenju GEKON modela pri generiranju realističnih fotografija ljudskog lica [9]

Brojne poznate tvrtke već koriste generativne kontradikcijske neuronske mreže unutar vlastitih programa. Neki od primjera uporabe GEKON mreža su:

1. *Adobe* koji je implementirao GEKON model u novoj generaciji poznatog grafičkog računalnog programa *Photoshop.* GEKON model se ovdje koristi kako bi pretvorio skice u realistične fotografije (slika 2.3).



Slika 2.3. Uporaba GEKON modela u programu *Photoshop* [10]

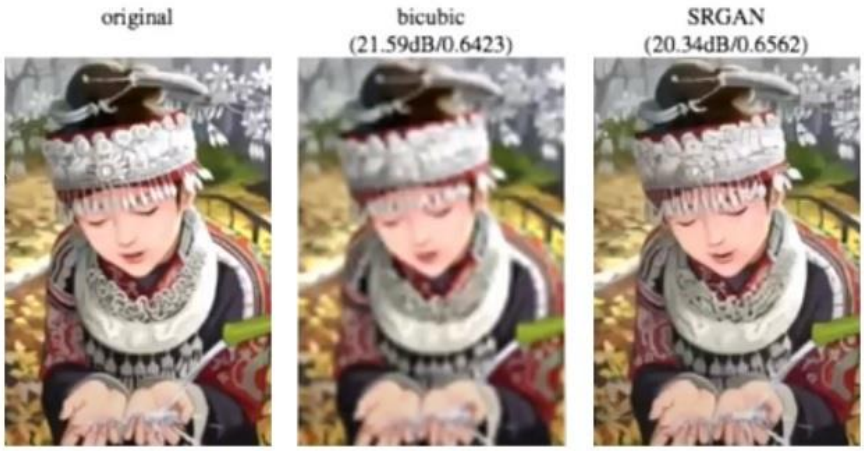
2. *Google* koristi GEKON model za generiranje teksta i slika.

3. *IBM* koristi GEKON model za povećanje podataka (*engl. data augmentation*). Dakle, GEKON model se ovdje koristi kako bi povećao bazu podataka koja se dalje može koristiti na drugim modelima, npr. za treniranje klasifikatora.

4. *SnapChat*, *TikTok* i *FaceApp* koriste GEKON model za izradu novih filtera.

Slika 2.4. Uporaba GEKON modela za izradu filtera unutar aplikacije *FaceApp*

5. *Disney* koristi GEKON model za super-rezoluciju slika i animacija.



Slika 2.5. Usporedba GEKON modela s drugim tehnikama korištenim za super-rezoluciju [7]

Slika 2.4 prikazuje usporedbu korištenja GEKON modela s alternativnim rješenjima koje se koriste za super-rezoluciju. Postupak se provodi tako da se originalnoj slici namjerno upola smanji rezolucija, nakon čega se različitim metodama pokuša rekonstruirati slika originalne rezolucije. Druga slika s lijeve strane prikazuje rekonstrukciju korištenjem bikubične interpolacije. Međutim, kao što se može vidjeti, rezultat koji se dobije ovim postupkom izgleda puno mutnije od originalne slike. GEKON model je s druge strane sposoban rekonstruirati sliku s minimalno izgubljenih detalja i oštrim prikazom.

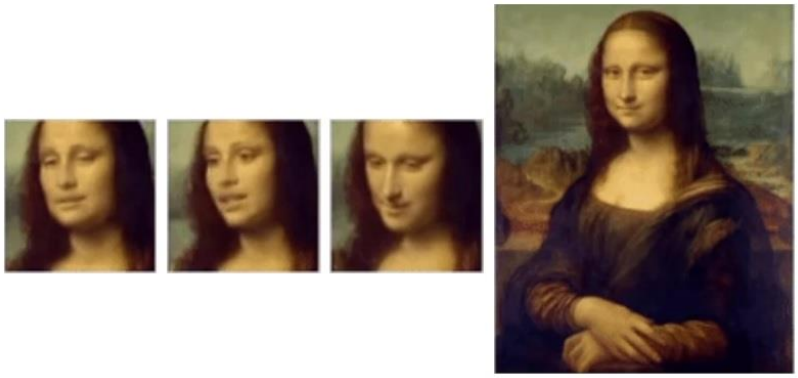
Osim rekonstrukcije, generiranja i primjene filtera na slike i fotografije, GEKON model može se koristiti i za generiranje 3D modela (slika 2.5).

Slika 2.6. Korištenje GEKON mreža pri izradi 3D modela [11]

Uporaba GEKON modela za generiranje 3D modela predstavlja mogućnost pronalaska brojnih novih konstrukcijskih rješenja i uštedu vremena u samom procesu konstruiranja novih proizvoda.

Bitno je za istaknuti i mogućnosti primjene GEKON modela u području medicine gdje se kao jedna od mogućih primjena navodi otkrivanje abnormalnosti na rendgenskim snimkama [12].

GEKON model se također koristi za rekonstrukciju izgubljenih isječaka video prikaza ili generiranje potpuno novih video prikaza korištenjem slika ili fotografija. Slika 2.6 prikazuje korištenje spomenute metode za prikaz poznate umjetničke slike *Mona Lisa* u pokretu.



Slika 2.7. Prikaz slike u pokretu korištenjem GEKON modela [13]

2.1.1. Primjena GEKON modela u robotici

Zbog velike fleksibilnosti pri obavljanju raznih vrsta zadataka, GEKON model pronašao je uporabu i u robotici. Jedan od primjera takve uporabe naglašava i sam tvorac GEKON modela, Ian Goodfellow, isticanjem rada "Unsupervised Learning for Physical Interaction through Video Prediction" [14]. U radu se opisuje korištenje generativnih modela za izradu simulacija mogućih budućih ishoda za pojačano učenje (*engl. reinforcement learning*). Postoje dva načina na koji se spomenuta metoda može koristiti u praksi. Prva mogućnost je korištenje GEKON modela za stvaranje simulacija unutar kojih se treniraju drugi modeli umjetnih neuronskih mreža koji se kasnije implementiraju u robotima. Prednost takvog postupka treniranja novih modela je brz, jeftin i jednostavan način izrade simuliranih prostora koji se mogu koristiti za

paralelno učenje više različitih modela istovremeno. Također je bitno napomenuti da je treniranje modela u simulacijama sigurnija i jeftinija metoda od obavljanja postupka treniranja u stvarnom radnom prostoru robota. Druga mogućnost je korištenje generativnih modela za predviđanje mogućih budućih ishoda u stvarnom vremenu i odabir najpovoljnije odluke unaprijed. Ova se metoda temelji na sposobnosti GEKON modela da generiraju nastavak video zapisa na temelju analize prethodnih isječaka snimljenog videa. Dakle, u ovom radu se predlaže fizička interakcija robota i okoline bez prethodno definiranih svojstava okoline i predmeta koji se u njoj nalaze.

Još jedna od uporaba GEKON modela u robotici je prepoznavanje tvrdoće materijala korištenjem taktilnih senzora na robotskoj ruci. U radu pod nazivom "Hardness Recognition of Robotic Forearm Based on Semi-supervised Generative Adversarial Networks" [15] se predlaže korištenje GEKON modela za treniranje na neoznačenim podacima. Metode prepoznavanja tvrdoće materijala koje se temelje na dubokom učenju pokazale su se dobre u praksi, međutim, za treniranje takvih modela je bila potrebna velika baza označenih podataka do koje se teško dolazi na brz i jeftin način. U radu se predlaže GEKON model koji kao vrsta generativnog modela neuronskih mreža ima sposobnost rada na neoznačenim podacima.

GEKON model može se koristiti i pri pronalasku inverzne kinematike i dinamike robotskih manipulatora kao što je opisano u radu pod nazivom "Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks" [16]. Tipično se kod aproksimacije robotskih sustava koriste analitičke metode. U zadnje vrijeme se za kompenzaciju pogreške nastale takvim postupkom koristi i strojno učenje, međutim kombinacija analitičke metode i strojnog učenja se u praksi pokazala redundantnom i računski skupom. Također se, kao i kod prepoznavanja tvrdoće materijala, pokazalo da korištenje drugih modela strojnog učenja zahtijeva veliku bazu označenih podataka. U ovom se radu predlaže korištenje GEKON modela koji imaju sposobnost rješavanja složenih zadataka bez velikih baza označenih podataka.

Socijalni roboti gestama mogu izražavati svoja unutarnja stanja, ali se njihove interaktivne mogućnosti često zasnivaju na unaprijed programiranim kretnjama. Iz tog se razloga pokreti socijalnih robota često ponavljaju što ih čini vrlo predvidljivim. U radu pod nazivom

"Affective Robot Movement Generation Using CycleGANs" [17] predlaže se korištenje GEKON modela za generiranje originalnih kretnji socijalnih robota u pravom vremenu na temelju učenja modela na već ručno programiranim gestama.

GEKON model može se koristiti i kod učenja mobilnih robota. U radu pod nazivom "Towards adversarial training for mobile robots" [18] opisuje se eksperiment proveden na robotu koji je morao naučiti lopticom pogoditi traženu metu. Eksperiment je isprva proveden tako da se učenje zasnivalo isključivo na prethodnim neuspjelim pokušajima, nakon čega se učenje provelo na još generiranih primjera korištenjem GEKON modela. Pokazalo se da je nakon učenja na dodatno generiranim primjerima robot postizao bolje rezultate. Bitno je napomenuti da su i druge vrste neuronskih mreža također pronašle uporabu u području planiranja trajektorija mobilnih robota [19].

Iz navedenih istraživanja može se zaključiti da GEKON model posjeduje velik potencijal za rješavanje tehničkih problema u robotici, kao i općenito potencijal za uporabu u ostalim tehničkim, ali i netehničkim strukama.

2.2. Etički problemi pri korištenju GEKON modela

GEKON modeli još se uvijek usavršavaju, međutim njihovi su se rezultati pokazali toliko realističnima u zadnjih nekoliko godina da je za ljude postalo gotovo nemoguće procijeniti što je originalno a što generirano. Njihova sposobnost da generiraju slike, videe, zvučne zapise itd. koji nalikuju na realne primjere predstavlja veliki etički problem koji će svake sljedeće godine biti sve više prisutan. Sposobnost generiranja slika koje nalikuju fotografijama pravih ljudi u samo jednom kliku otvara mogućnost generiranja beskonačno lažnih profila na socijalnim mrežama koji će izgledati kao pravi. Naime, kada se u tu svrhu koriste fotografije pronađene na internetu, uvijek postoji opcija pronalaska originalne fotografije korištenjem raznih algoritama. Kada netko iskoristi GEKON model kako bi generirao takvu sliku, spomenuti algoritmi neće ništa pronaći. Osoba koja tako želi zlouporabiti GEKON modele u kombinaciji sa socijalnim mrežama ne mora čak ni trenirati vlastiti model jer već postoji velik broj takvih modela koji su obavili proces učenja i svima su dostupni putem interneta. Jedan od takvih modela može se pronaći na internetskoj stranici pod nazivom „thispersondoesnotexist.com"

[20]. Osim sposobnosti generiranja fotografija koje nalikuju ljudima, GEKON modeli mogu imati velik utjecaj i u umjetničkim sferama. Na internetskoj

stranici pod nazivom „noartist“ [21] već se prodaju slike koje model generira nakon učenja na slikama poznatih slikara. Netko bi tako mogao prodavati slike ili pjesme i predstavljati ih kao vlastiti rad. Postavlja se pitanje prave umjetničke vrijednosti djela koje nije osmislio čovjek. Pripisuje li se zasluga za sve generirane primjere osobi koja je napravila prvotni model? Hoće li se u budućnosti smanjiti broj umjetnika koji više neće moći konkurirati generiranim umjetničkim djelima ili ćemo uvijek pronalaziti neku skrivenu vrijednost u djelima koje je napravio netko kao što smo mi? Još jedan od velikih problema koji bi mogla prouzročiti zlouporaba GEKON modela je širenje lažnih informacija. Kao što je već spomenuto, GEKON modeli mogu se trenirati na video zapisima tako da ostvare sposobnost generiranja nastavka video zapisa ili čak izmjene određene dijelove video zapisa. Aplikacije kao što su „deepfacelab“ [22] već su se dokazale vrlo sposobnima u generiranju tzv. deepfake video zapisa. Iz navedenih primjera može se zaključiti da će se uporaba GEKON modela morati strogo regulirati.

2.3. Struktura GEKON modela

Učenje GEKON modela temelji se na natjecanju dviju neuronskih mreža koje nazivamo generator i diskriminator. U ovom će se poglavlju objasniti ciljevi svake od mreža zasebno te prikazati kako njihova hipotetska igra s nultom sumom utječe na sveukupni model.
Kao što je već objašnjeno u prethodnom poglavlju, GEKON model jedan je od vrsta generativnih modela strojnog učenja pa je tako i smisao natjecanja između dviju neuronskih mreža dobiti generativnu neuronsku mrežu koja generira zadovoljavajuće rezultate. To znači da se neuronska mreža koja predstavlja diskriminator isključivo koristi samo za treniranje neuronske mreže koja predstavlja generator te da se kao takva nakon postupka učenja odbacuje.

Cilj generatora je generirati nove primjere koji su dovoljno slični originalnim ulaznim primjerima tako da ih diskriminator ne može razlikovati, dok je cilj diskriminatora naučiti razlikovati generirane od originalnih primjera. Kako bi se objasnilo natjecanje između generatora i diskriminatora često se koristi analogija krivotvoritelja slika i inspektora za umjetnine. Krivotvoritelja slika ovdje predstavlja generator koji pokušava napraviti dovoljno

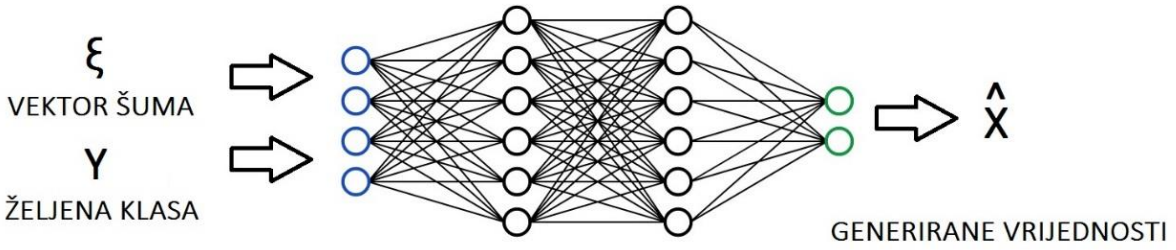
realnu sliku kako bi prevario inspektora za umjetnine kojeg ovdje predstavlja diskriminator. Na samom početku igre izlaz generatora nije ni približno sličan originalnim primjerima jer se generatoru na samom početku igre uopće ne prikazuju originalni primjeri kakve bi trebao

generirati. Diskriminator na početku nema nikakvu sposobnost raspoznavanja originalnih od generiranih primjera tako da on započinje s potpuno nasumičnim nagađanjem za svaki dani primjer. Za razliku od generatora, diskriminatoru se na samom početku igre prikazuju originalni primjeri, međutim, diskriminator ih još ne može prepoznati kao originalne jer ulaze u njega s generiranim primjerima.

Generativne kontradikcijske mreže koriste prethodno spomenutu metodu grupnog učenja gdje se prilagodba parametara mreže vrši nakon cjelokupnog skupa primjera za učenje. Nakon svakog izvršenog skupa primjera na diskriminatoru, rezultati diskriminatora se uspoređuju sa željenim izlazom koji ovdje predstavlja vektor nula i jedinica. Nule u željenom izlaznom vektoru predstavljaju stopostotnu sigurnost da je gledani primjer generiran od strane generatora, dok jedinice predstavljaju stopostotnu sigurnost da je gledani primjer uzet iz skupa originalnih primjera. Realni izlaz diskriminatora bit će vektor čije se vrijednosti članova nalaze između nule i jedinice s obzirom na to da diskriminator neće nikada u potpunosti biti siguran u svoju odluku. Tako se nakon svakog skupa primjera za učenje uspoređuje izlaz diskriminatora sa željenim izlazom i odlučuje koja mreža je bolje odradila svoj zadatak.

2.3.1. Generator

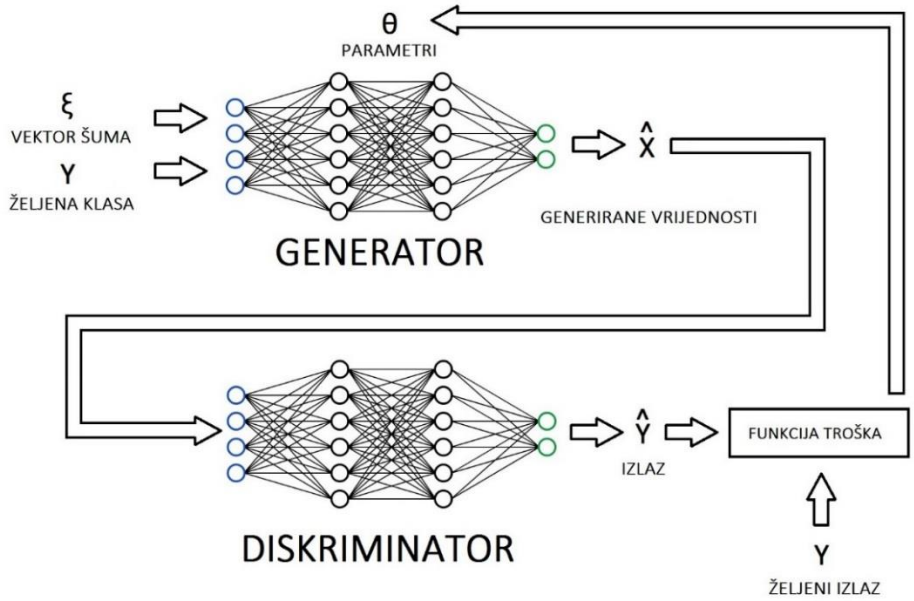
Generator je tip neuronske mreže unutar GEKON modela koji se koristi za generiranje novih primjera. Kako bi se osiguralo da će se generirani primjeri uvijek razlikovati od prethodnih kao ulaz u generator uzimaju se različiti setovi nasumično generiranih brojeva poznati kao vektor šuma (engl. noise vector). Ako je generator treniran tako da može generirati primjere iz više različitih klasa uz vektor šuma ponekad je potrebno dati i klasu Y za koju se traži novi generirani .primjer.



Slika 2.8. Ulaz i izlaz generatora

Generirane vrijednosti koje se dobivaju na izlazu iz generatora mogu poprimiti formu: slike, teksta, videa, itd. Broj neurona koji se nalaze u izlaznom sloju ovisit će o traženoj formi izlaza. U slučaju da se radi o slici rezolucije 28x28 piksela, u izlaznom sloju bi se moralo nalaziti barem 784 neurona. Generirana vrijednost svakog neurona u izlaznom sloju bi tada odgovarala nijansi pripadajućeg piksela.

Proces učenja generatora može se prikazati na sljedeći način:



Slika 2.9. Proces učenja generatora

Kao što se vidi na slici 2.9, za treniranje generatora je potreban diskriminator jer se za prilagođavanje parametara koristi ista funkcija troška. U trenutku kada rezultati generatora budu zadovoljavajući postoji opcija spremanja parametara generatora kako ne bi došlo do pretreniranja mreže. U tom se trenutku diskriminator može odbaciti jer se on kod GEKON modela koristi isključivo za treniranje generatora.

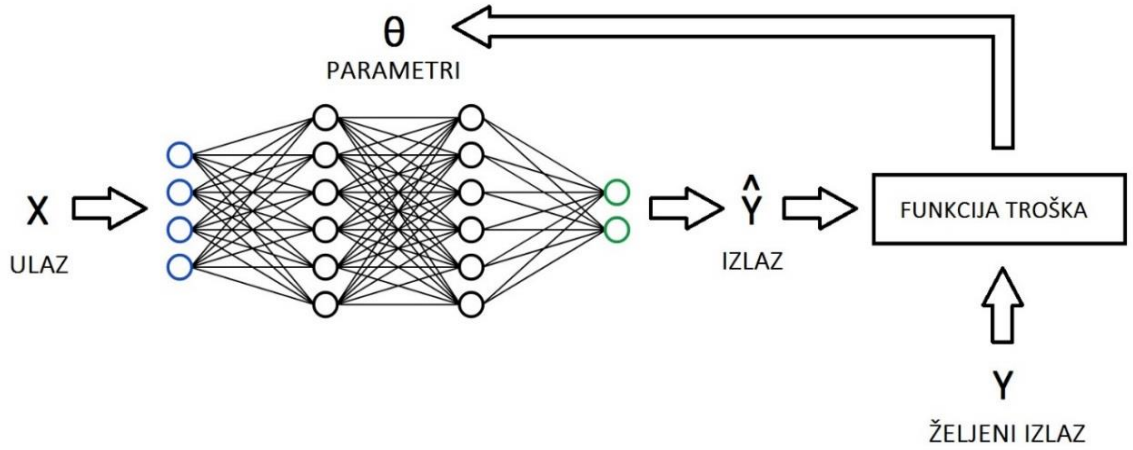
Funkcija generatora često se zapisuje kao P(X | Y). Dakle model vjerojatnosti izlaznih svojstava, odnosno vrijednosti X za danu ulaznu klasu Y. Često se generiraju primjeri iz samo jedne klase pa se u tom slučaju funkcija generatora može zapisati kao P(X).

Učestalost pojave različitih svojstava unutar generiranih primjera zavisit će isključivo o njihovoj učestalosti u skupu originalnih primjerima koji su korišteni za treniranje mreže. Npr. ako se model trenira za generiranje slika pasa, unutar generiranih primjera češće će se nalaziti slike onih vrsta pasa koji su češće bili korišteni tijekom procesa učenja mreže. Vjerojatnost pojave različitih svojstava može se prikazati u trodimenzionalnom prostoru tako da se svojstva koja se češće generiraju prikažu u centru, a na rubovima se prikažu svojstva koja se nalaze samo unutar specifičnih primjera. Takav prikaz može biti vrlo koristan jer se kod složenijih algoritama GEKON modela koriste generatori kojima se nakon procesa učenja može striktno definirati kakva se svojstva traže kod generiranih primjera.

2.3.2. Diskriminator

Diskriminator je tip klasifikatora. Cilj takvog tipa neuronske mreže je naučiti razdijeliti ulazne primjere u ispravne skupine odnosno klase. U najjednostavnijem primjeru klasifikatori razdjeljuju primjere u dvije klase, međutim broj klasa nije limitiran pa tako postoje klasifikatori koji razdjeljuju dane primjere u više mogućih klasa. Klasifikatori također nisu limitirani na tip ulaznih podataka pa se tako za ulazne podatke mogu koristiti: slike, tekst, video, itd.

Proces učenja klasifikatora može se prikazati na sljedeći način:

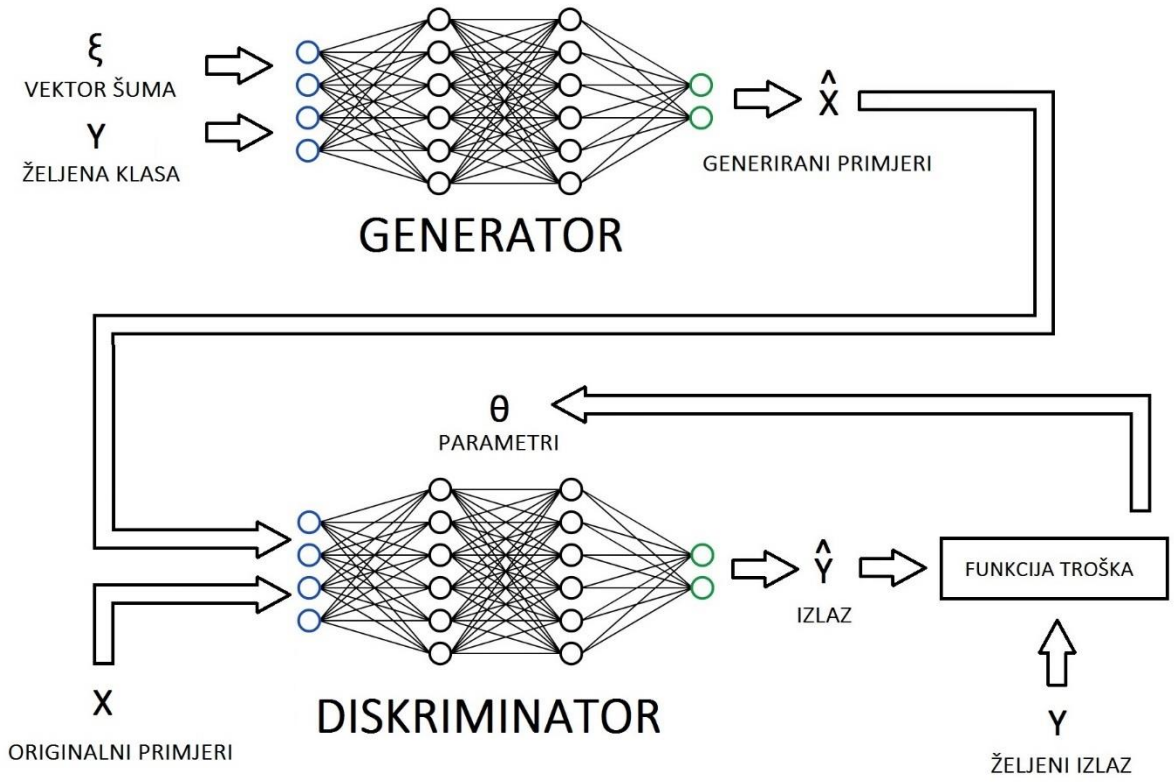


Slika 2.10. Vizualni prikaz procesa učenja klasifikatora

Funkcija klasifikatora često se zapisuje kao P(Y | X). Dakle model vjerojatnosti klase Y za dane ulazne vrijednosti X.

Kod GEKON modela se za diskriminator uglavnom koristi klasifikator samo s jednim neuronom u izlaznom sloju obzirom na to da se traži samo klasifikacija između generiranih i originalnih primjera. Gradijent funkcije troška koja se nalazi na izlazu iz diskriminatora koristi se kako bi se prilagodili parametri diskriminatora ili generatora zavisno o mreži koja se trenira.

Proces treniranja diskriminatora modela može se prikazati na sljedeći način:



Slika 2.11. Proces učenja diskriminatora

2.3.3. Funkcija troška GEKON modela

Kod generativnih kontradikcijskih mreža se za funkciju troška najčešće koristi prethodno definiran gubitak unakrsne entropije (1.8). Takav oblik funkcije troška odgovara GEKON modelu jer je gubitak unakrsne entropije definiran baš za klasifikacijske zadatke kada je rezultat

mreže potrebno klasificirati u dvije kategorije. U slučaju GEKON modela se radi o kategoriji generiranih primjera i kategoriji originalnih primjera.

Funkcija gubitka unakrsne entropije može se zapisati u sljedećem obliku:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log h(x_i,\theta) + (1-y_i)\log(1-h(x_i,\theta))\right]. \qquad (2.1)$$

Funkcija uzima u obzir sve primjere koji su ušli u diskriminator s obzirom na to da se koristi metoda grupnog učenja pa je potrebno sumirati sve primjere i pronaći srednju vrijednost troška. Predikcija modela koju smo ranije označavali s $\hat{y}_i$, ovdje je zapisana u obliku $h(x_i, \theta)$ kako bi se naznačio utjecaj parametara na ishod funkcije troška.

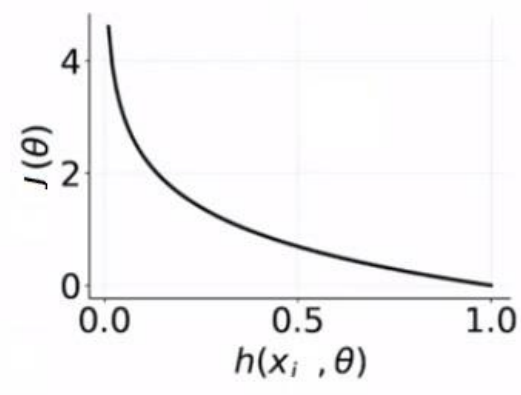
Izraz unutar sume može se podijeliti u dvije skupine. Ako se za prvi dio izraza unutar sume uzmu različite kombinacije željenog izlaza i predikcije koju daje diskriminator, dobit će se sljedeće:

Tablica 1. Utjecaj predikcije modela na prvi dio funkcije troška

$\boldsymbol{y_i}$	$\boldsymbol{h(x_i, \theta)}$	$\boldsymbol{y_i \log h(x_i, \theta)}$
0	bilo koja vrijednost	0
1	~1	~0
1	~0	-∞

Dakle, prvi dio izraza imat će utjecaj na funkciju troška samo kada željena izlazna vrijednost diskriminatora bude jednaka 1, odnosno kada se radi o originalnom primjeru za kojeg želimo da bude tako i označen od strane diskriminatora.

Ovisnost funkcije troška o predikciji za slučaj kada je željena vrijednost diskriminatora jednaka 1 može se i grafički prikazati na sljedeći način:



Slika 2.12. Ovisnost funkcije troška o predikciji kada je željena vrijednost diskriminatora jednaka 1

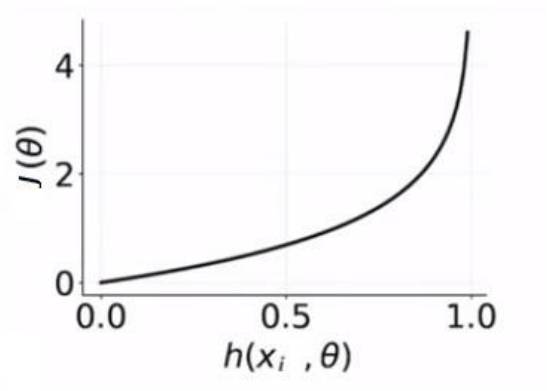
Ako se za drugi dio izraza unutar sume uzmu različite kombinacije željenog izlaza i predikcije koju daje diskriminator, dobit će se sljedeće:

Tablica 2. Utjecaj predikcije modela na drugi dio funkcije troška

$\boldsymbol{y_i}$	$\boldsymbol{h(x_i, \theta)}$	$\boldsymbol{(1 - y_i) \log(1 - h(x_i, \theta))}$
1	bilo koja vrijednost	0
0	~0	~0
0	~1	-∞

Drugi dio izraza imat će utjecaj na funkciju troška samo kada željena izlazna vrijednost diskriminatora bude jednaka 0. Dakle, kada se radi o generiranom primjeru za kojeg želimo da bude tako i označen od strane diskriminatora.

Ovisnost funkcije troška o predikciji za slučaj kada je željena vrijednost diskriminatora jednaka 0, može se i grafički prikazati na sljedeći način:



Slika 2.13. Ovisnost funkcije troška o predikciji kada je željena vrijednost diskriminatora jednaka 0

Ako se pogledaju najnepovoljniji primjeri iz tablica, a to su primjeri kod kojih diskriminator daje predikciju suprotnu od željene vrijednosti izlaza, može se vidjeti da će izlazna vrijednost iz sume težiti u -∞. S obzirom na to da se kod funkcije troška uglavnom koriste pozitivne vrijednosti koje se tada pokušavaju minimizirati, čitavu je sumu potrebno pomnožiti s -1 kao što se može vidjeti u izrazu (2.1).

2.3.4. Treniranje cijelog modela

Prethodno je pokazano kako se svaka od mreža unutar GEKON modela trenira zasebno. Međutim, kod treniranja modela je također potrebno alternirati između procesa učenja svake od mreža zasebno. Razlog tome su problemi koji mogu nastati kada jedna od mreža nadmaši drugu tijekom procesa učenja. U slučaju kada se zbog neuravnoteženog učenja dobije superioran diskriminator, izlaz iz diskriminatora će uvijek upućivati na to da su generirani primjeri gotovo stopostotno generirani. Takav rezultat ne može generatoru ukazati na specifične pogreške pa stoga otežava ili onemogućuje proces učenja generatora. U suprotnom, ako se zbog

neuravnoteženog učenja dobije superioran generator, izlaz iz diskriminatora će uvijek upućivati na to da su generirani primjeri gotovo stopostotno originalni što opet stvara problem kod treniranja generatora koji ne dobiva povratnu informaciju o tome kako prilagoditi svoje parametre za daljnji napredak. Iz navedenog razloga je tijekom procesa učenja potrebno održavati generator i diskriminator na što bližem nivou.

Diskriminator obavlja lakši zadatak od generatora. Zadatak diskriminatora je samo klasificirati ulazne vrijednosti u dvije skupine dok generator mora u potpunosti modelirati određenu klasu kako bi mogao generirati nove primjere. Zato je kod mnogih GEKON modela potrebno provesti više vremena trenirajući generator u odnosu na diskriminator. Međutim, iz ranije spomenutih razloga je potrebno pronaći ravnotežu i ne pretjerati ni s treniranjem generatora.

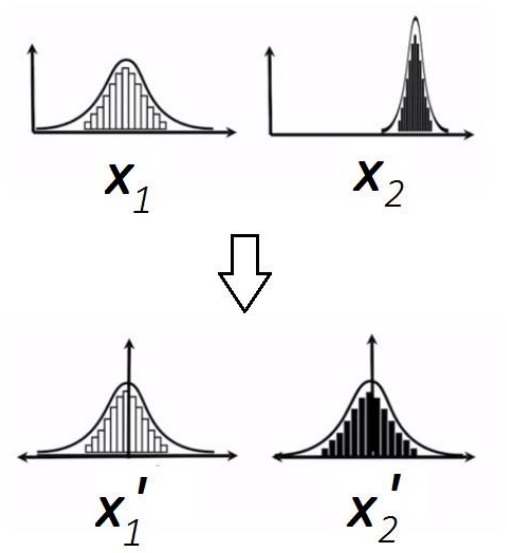
2.4. Optimizacija GEKON modela

U ovom će se poglavlju prikazati metode koje se koriste kako bi se optimizirao proces treniranja GEKON modela.

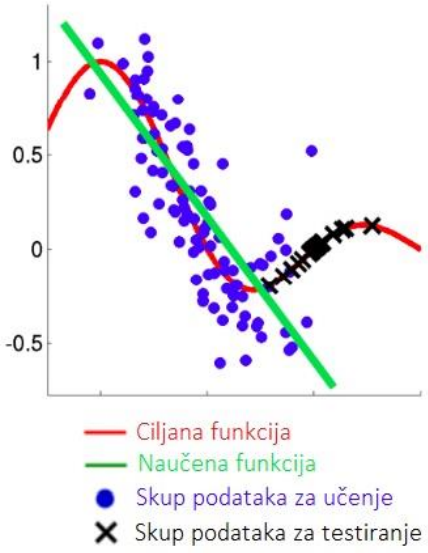
2.4.1. Normalizacija serije podataka

Proces učenja složenijih zadataka kod GEKON modela može potrajati dugo vremena. Također postoji i spomenuti problem izjednačavanja brzine učenja generatora i diskriminatora pa se iz tih razloga koriste svi mogući načini optimizacije procesa učenja. Jedna od korištenih metoda koja ubrzava proces učenja se naziva normalizacija serije podataka (engl. batch normalization).

Kod treniranja se radi statistika ulaznih podataka unutar svake iteracije te se potom distribucija ulaznih podataka transformira u oblik u kojem svi ulazni podaci imaju istu aritmetičku sredinu i standardnu devijaciju (slika 2.11). Npr. uzima se da je aritmetička sredina jednaka 0 i standardna devijacija jednaka 1. Nakon treniranja se za skup podataka za testiranje koristi statistika svih ulaznih podataka iz faze treniranja kako bi se njihova distribucija približila centriranoj distribuciji podataka za treniranje. Taj se postupak provodi kako bi se izbjegao kovarijantni pomak (engl. covariate shift). Kovarijantni pomak znači promjenu u distribuciji ulaznih podataka za testiranje modela u odnosu na ulazne podatke korištene pri treniranju (slika 2.12). Ovom se metodom postiže ravnoteža utjecaja ulaznih podataka unutar funkcije troška što ubrzava i olakšava cjelokupni proces učenja.



Slika 2.14. Normalizacija serije podataka [23]



Slika 2.15. Kovarijantni pomak [24]

U nastavku će biti opisan postupak provođenja normalizacije serije podataka za vrijeme treniranja i testiranja modela.

Normalizacija serije podataka za vrijeme treniranja odvija se u samim čvorovima mreže. Postupak se provodi na izračunatoj vrijednosti $z_i^{(L)}$ prije nego ona ulazi u aktivacijsku funkciju za svaki zasebni primjer koji se koristi tijekom grupnog učenja. Tako se uzimaju sve izračunate vrijednosti promatranog čvora $z_i^{(L)}$ iz cijelog seta primjera za gledanu iteraciju grupnog učenja

i na svakoj od njih se izvršava sljedeća prilagodba:

$$\hat{z}_i^{(L)} = \frac{z_i^{(L)} - \mu_{z_i^{(L)}}}{\sqrt{\sigma^2_{z_i^{(L)}} + \varepsilon}}. \tag{2.2}$$

Gdje su:

$\mu_{z_i^{(L)}} -$ aritmetička sredina svih $z_i^{(L)}$ unutar promatrane iteracije grupnog učenja

$\sigma^2_{z_i^{(L)}} -$ varijanca $z_i^{(L)}$ unutar promatrane iteracije grupnog učenja

$\varepsilon -$ konstanta standardne devijacije

Dakle, kako bi se ostvarila distribucija vrijednosti $z_i^{(L)}$ s aritmetičkom sredinom jednakom 0 i standardnom devijacijom s iznosom 1, svakoj se pojedinačnoj vrijednosti $z_i^{(L)}$ oduzima vrijednost prvotne aritmetičke sredine promatrane iteracije grupnog učenja, nakon čega se dobivena vrijednost dijeli sa standardnom devijacijom koja ovdje predstavlja korijen varijance kojoj se dodaje konstanta standardne devijacije ε kako bi se osiguralo da će vrijednost unutar korijena biti veća od 0.

Nakon što se dobiju normalizirane vrijednosti $z_i^{(L)}$ koje se označavaju sa $\hat{z}_i^{(L)}$, dolazi se do parametara sloja normalizacije serije podataka koji se označavaju s β i γ.

$$y_i^{(L)} = \gamma \hat{z}_i^{(L)} + \beta. \tag{2.3}$$

Gdje su:

β – faktor pomaka

γ – faktor skaliranja

Faktori β i γ također se prilagođavaju tijekom postupka učenja kako bi se osiguralo da će distribucija u koju se transformiraju vrijednosti $z_i^{(L)}$ biti optimalne za zadatak mreže. Dakle, nakon što se podaci normaliziraju u vrijednosti $\hat{z}_i^{(L)}$, ponovno se transformiraju u novu distribuciju korištenjem spomenutih faktora nakon čega vrijednosti $y_i^{(L)}$ ulaze u aktivacijsku funkciju. Upravo ta nova prilagodba distribucije čini razliku između klasične normalizacije i

normalizacije serije podataka.

Kod testiranja modela provodi se drugačiji postupak jer se zahtijeva da aritmetička sredina i standardna devijacija budu jednake za sve iteracije provođenja grupnog učenja umjesto samo za promatranu iteraciju kao što je bio slučaj kod postupka treniranja.

$$\hat{z}_i^{(L)} = \frac{z_i^{(L)} - E(z_i^{(L)})}{\sqrt{Var(z_i^{(L)}) + \varepsilon}}. \tag{2.4}$$

Gdje su:

$E(z_i^{(L)})$ – pretpostavljena srednja vrijednost od $z_i^{(L)}$

$Var\left(z_i^{(L)}\right)$ – varijanca vrijednosti $z_i^{(L)}$

Pretpostavljena srednja vrijednost i varijanca su ovdje fiksne vrijednosti koje proizlaze iz svih primjera korištenih tijekom procesa treniranja.

Nakon provedene prve transformacije distribucije vrijednosti $z_i^{(L)}$, nastavlja se isti postupak kao i kod treniranja mreže gdje dobivene vrijednosti $\hat{z}_i^{(L)}$ulaze u funkciju (2.3) koja ponovno transformira distribuciju podataka prije ulaza u aktivacijsku funkciju.

2.4.2. Konvolucijski GEKON model

Konvolucija je jedan od temeljnih procesa na kojem se bazira obrađivanje slika pa tako poprima bitnu ulogu i kod mnogih vrsta GEKON modela. Korištenjem procesa konvolucije moguće je detektirati različite značajke na specifičnim dijelovima slike korištenjem prethodno definiranih filtera gdje svaki filter predstavlja matricu čiji su članovi realni brojevi naučeni tijekom procesa treniranja mreže.

U nastavku će biti prikazan proces konvolucije na crno-bijeloj slici veličine 5x5 piksela i naučenim filterom za detekciju vertikalnih linija veličine 3x3 piksela.

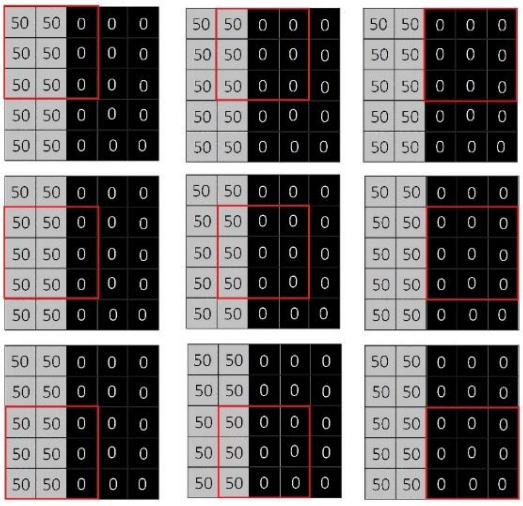
Crno-bijela slika bit će definirana matricom **A** dimenzije 5x5 gdje svaki član matrice preuzima vrijednost između 0 (crno) i 255 (bijelo).

$$\boldsymbol{A} = \begin{bmatrix} 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \end{bmatrix}. \tag{2.5}$$

Analogno tome, filter će biti definiran matricom **B** dimenzije 3x3:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}. \tag{2.6}$$

Postupak konvolucije izvodi se tako da se matrica filtera B stavlja u gornji lijevi kut matrice slike A, nakon čega se sumiraju umnošci članova matrice filtera B s preklapajućim članovima matrice slike A čime se dobiva prvi član rezultirajuće matrice. Nakon toga se matrica filtera pomiče za jedno mjesto udesno te se proces ponavlja sve dok matrica filtera ne prođe čitavu matricu slike (slika 2.13).



Slika 2.16. Primjer izvođenja konvolucije

Postupkom konvolucije za dani primjer dobit će se sljedeće:

$$\begin{bmatrix} 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \\ 50 & 50 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 150 & 150 & 0 \\ 150 & 150 & 0 \\ 150 & 150 & 0 \end{bmatrix}. \quad (2.7)$$

Visoke vrijednosti unutar rezultirajuće matrice ukazuju da se na slici nalazi vertikalna linija. Kako bi se na slici pronašle značajke koje nije jednostavno opisati s jednom matricom filtera, u praksi se često koriste slojevi koji se sastoje od nekoliko filtera.

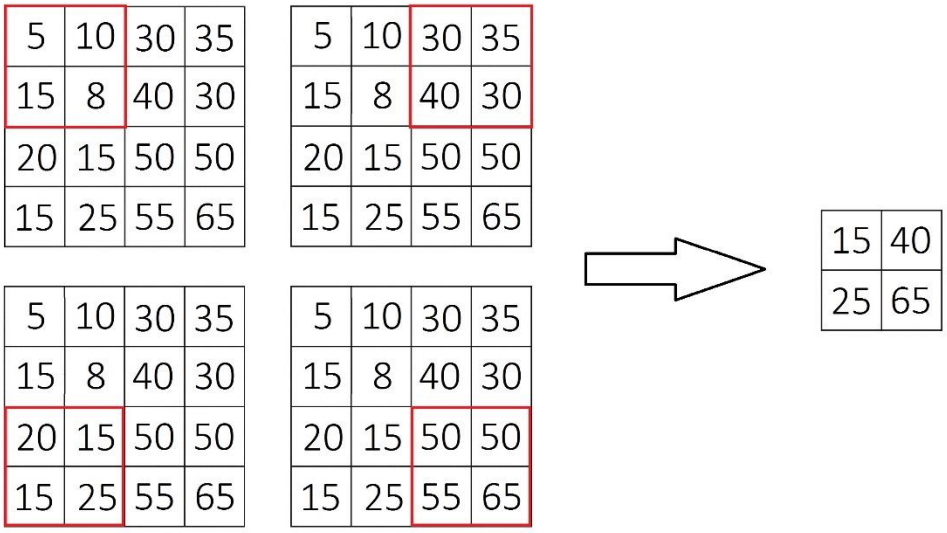
Kada se radi o velikim slikama, proces konvolucije može dugo potrajati pa se ponekad uzima veći korak od 1. To znači da se matrica filtera može pomaknuti za više od jednog mjesta udesno ili dolje nakon svakog novog izračuna sume umnoška članova. Rezultirajuća matrica takvog postupka davat će manje informacija, međutim, ponekad je takav postupak isplativ radi uštede vremena.

Također se na slike koje prolaze postupak konvolucije često dodaje okvir. To znači da se matrici slike povećavaju dimenzije dodavanjem stupaca i redova na samom početku i kraju matrice gdje svi članovi dodanih redova i stupaca poprimaju istu brojčanu vrijednost. Taj se postupak provodi kako bi se osiguralo da će matrica filtera jednak broj puta prijeći preko svakog člana matrice slike.

U konvolucijskim GEKON modelima koriste se slojevi sažimanja (engl. pooling layer) i slojevi naduzorkovanja (engl. upsampling) kako bi se smanjile ili povećale dimenzije slika.

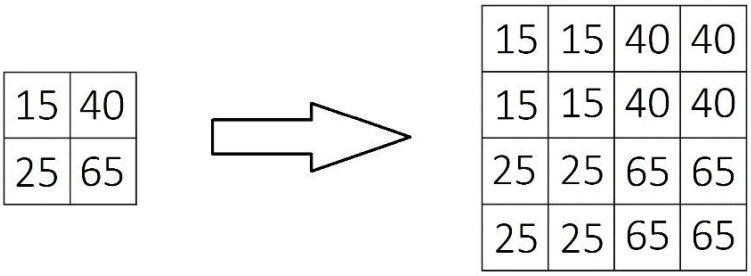
Kao što je već spomenuto, odvijanje konvolucije može biti vremenski i računski zahtjevno pa se iz tog razloga provodi sažimanje slika. Postupak sažimanja provodi se korištenjem funkcija sažimanja tako da se ulazna matrica podijeli na nekoliko manjih dijelova od kojih se s obzirom na korištenu funkciju uzimaju: maksimalna, minimalna ili srednja vrijednost, koje zamjenjuju promatrani dio ulazne matrice unutar nove rezultirajuće matrice manjih dimenzija. Postupak se provodi slično konvoluciji gdje se dijelovi originalne ulazne matrice dijele korištenjem jezgre (engl. kernel). Za razliku od filtera, jezgra prolazi kroz ulaznu matricu s korakom jednakim svojim dimenzijama, čime je osigurava da će svaki član ulazne matrice samo jednom biti

promatran od strane jezgre (slika 2.14).



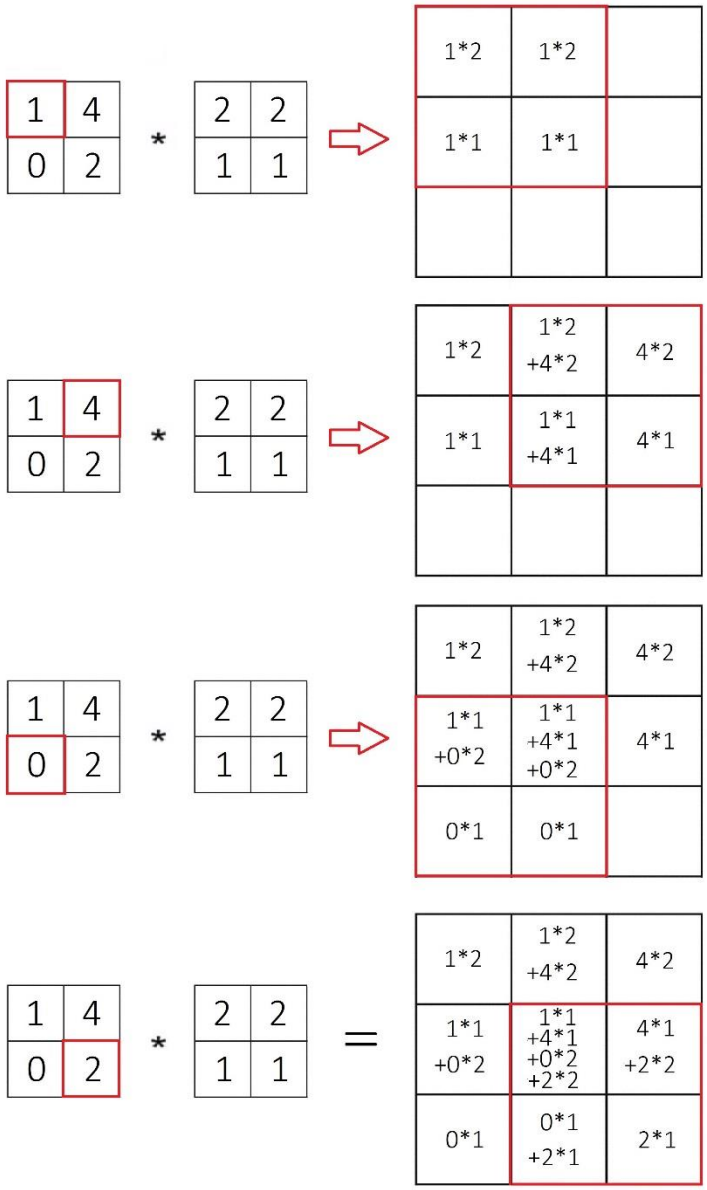
Slika 2.17. Primjer sažimanja korištenjem max funkcije

Suprotno od sažimanja, naduzorkovanje se koristi kada je potrebno povećati dimenzije ulazne matrice. Kod naduzorkovanja se može koristiti postupak ponavljanja istih vrijednosti članova ulazne matrice kako bi se popunila nova matrica većih dimenzija ili se provode funkcije kao što su linearna ili bilinearna interpolacija.



Slika 2.18. Primjer naduzorkovanja ponavljanjem

Još jedan od postupka povećavanja dimenzija ulaznih podataka koji se koristi u GEKON modelu naziva se transponirana konvolucija (engl. transposed convolutions). Kod ove se metode koristi naučeni filter koji popunjava rezultirajuću matricu s korakom jednakim 1. Matrica filtera množi svaki član ulazne matrice te dobivene vrijednosti preslikava direktno u rezultirajuću matricu nakon čega prelazi na sljedeći član ulazne matrice i ponavlja istu operaciju (slika 2.16).



Slika 2.19. Primjer transponirane konvolucije

Jedan od problema koji se javlja pri korištenju transponirane konvolucije je ponavljanje uzoraka u obliku šahovskog polja (engl. checkerboard problem). Do navedenog problema dolazi jer se pikseli koji se nalaze u središtu rezultirajuće matrice više puta provlače kroz filter od onih koji

se nalaze na rubovima. Usprkos navedenom problemu, transponirana konvolucija još uvijek se često koristi u GEKON modelima, a u slučaju kada se u potpunosti želi izbjeći ponavljanje uzoraka, umjesto transponirane konvolucije koristi se kombinacija naduzorkovanja i konvolucije.

2.4.3. Wasserstein GEKON model

Funkcija unakrsne entropije koja je prethodno definirana izrazom (1.8), najčešće se koristi za funkciju troška GEKON modela. Međutim, funkcija unakrsne entropije također može dovesti model u nepovoljno stanje tijekom učenja. Postoje dva glavna problema koja se javljaju pri korištenju unakrsne entropije kao funkcije troška, a to su:

1. Kolaps modova (engl. mode collapse) koji se događa kada generator pronađe lokalni minimum za specifični mod unutar kojeg je sposoban generirati primjere koje diskriminator klasificira kao originalne, međutim, generirani primjeri postaju sve sličniji jedni drugima. Npr. kada generator čiji je zadatak naučiti generirati slike jednoznamenkastih brojeva krene generirati samo slike broja 1 jer one najčešće prolaze kroz diskriminator kao originalne.

2. Problem nestanka gradijenta (engl. vanishing gradient problem) koji nastaje kada se učenje diskriminatora odvije brže od učenja generatora, odnosno kada se distribucija generiranih primjera udalji od distribucije originalnih. U tom će se slučaju značajno smanjiti nagib funkcije koju aproksimira funkcija troška što znači da će nestati gradijent s kojim se prilagođavaju parametri mreže tijekom učenja.

Kako bi se izbjegli navedeni problemi, neke od GEKON mreža koriste Wasserstein model troška.

Funkcija unakrsne entropije može se jednostavnije zapisati u sljedećem obliku korištenjem oznake očekivane vrijednosti *E(x)*:

$$J(\theta) = -\left[E\left(\log\left(d(x)\right)\right) + E\left(1 - \log\left(d(\hat{x})\right)\right)\right]. \tag{2.8}$$

Gdje prvi dio funkcije daje informaciju o tome koliko dobro diskriminator klasificira originalne primjere, a drugi dio funkcije koliko dobro klasificira generirane primjere.
Wasserstein trošak poprima sličan izraz:

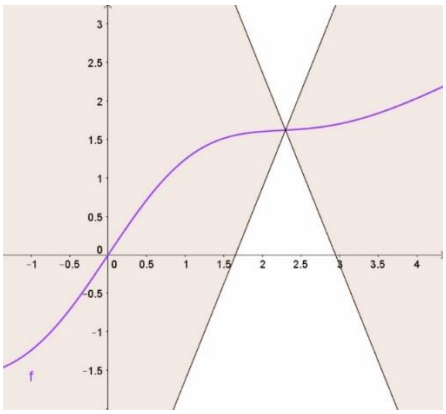
$$J(\theta) = E(c(x)) - E\left(c(\hat{x})\right). \tag{2.9}$$

Bitna razlika je što se ovdje predikcije diskriminatora *d(x)* i $d(\hat{x})$ zamjenjuju s predikcijama kritičara (engl. critic) *c(x)* i $c(\hat{x})$.

Za razliku od diskriminatora sa sigmoidalnom aktivacijskom funkcijom u izlaznom sloju koji se koristi u klasičnom GEKON modelu, izlazna vrijednost kritičara nije ograničena na vrijednosti između 0 i 1 pa se iz tog razloga u funkciji troška ne upotrebljavaju logaritamske funkcije. S obzirom na to da se u Wasserstein GEKON modelu ne zahtijeva uporaba sigmoidalne aktivacijske funkcije, koristi se linearni sloj na izlazu iz mreže koji omogućava izlaz u obliku bilo kojeg realnog broja. Kritičar tada može prilagođavati vlastite parametre bez utjecaja na kvalitetu povratne informacije koju generator prima tijekom svojeg učenja. Na taj se način rješava problem nestanka gradijenta, ali i omogućava izlaz iz kolapsa modova s obzirom na to da će generator uvijek primati korisnu povratnu informaciju od kritičara.

Kako bi se osiguralo da će funkcija troška kod Wasserstein GEKON modela biti diferencijabilna i stabilna tijekom učenja, na funkciju se postavlja uvjet 1-Lipschitz kontinuiteta. To znači da norma gradijenta u svakoj točki funkcije mora biti manja ili jednaka 1 kao što je prikazano u izrazu (2.10).

$$\|\nabla f(x)\|_2 \leq 1. \tag{2.10}$$



Slika 2.20. Vizualni prikaz Lipschitz kontinuiteta [25]

Postoje dvije metode koje se koriste kako bi se zadovoljio uvjet 1-Lipschitz kontinuiteta:

1. Ograničavanje težinskih faktora (*engl. weight clipping*) na minimalne i maksimalne vrijednosti unutar neuronske mreže kritičara. Dakle, svi parametri koji bi se nakon iteracije učenja trebali prilagoditi na vrijednosti iznad maksimalne ili ispod minimalne, ostat će na definiranim graničnim vrijednostima. Nedostatak ove metode je što otežava ili ponekad onemogućava proces učenja kritičara.

2. Gradijentna kazna (*engl. gradient penalty*) kod koje se u funkciju troška dodaje regularizacijska komponenta koja kažnjava kritičara kada norma gradijenta funkcije postane veća od 1. Kod ove metode se gradijent provjerava tako da se interpoliraju vrijednosti između generiranih i originalnih primjera tako da se nasumično uzme originalni primjer korišten u procesu učenja, pomnoži s faktorom ε' te mu se pritom zbroji nasumično uzeti generirani primjer pomnožen s (1- ε'), gdje se za faktor ε' također uzima nasumična vrijednost između 0 i 1. Tako se dobiva novi primjer kojeg označavamo s $\hat{x}'$ čije se vrijednosti nalaze između seta originalnih i seta generiranih primjera.

$$\hat{x}' = \varepsilon' x + (1 - \varepsilon')\hat{x}. \tag{2.11}$$

Norma gradijenta novog primjera $\hat{x}'$ se tada ubacuje u funkciju troška u sljedećem obliku:

$$J(\theta) = E(c(x)) - E\big(c(\hat{x})\big) + \lambda E(\|\nabla c(\hat{x}')\|_2 - 1)^2. \tag{2.12}$$

S obzirom na to da kritičar želi minimizirati funkciju troška, u interesu mu je da očekivana vrijednost norme gradijenta primjera $\hat{x}'$ bude što bliža 1. Utjecaj regularizacije na čitavu funkciju troška još se iskazuje korištenjem regularizacijskog faktora λ koji množi regularizacijski izraz. Dakle, kod ove metode se ne zadaje strogo uvjet 1-Lipschitz kontinuiteta, već se dodaje regularizacijski izraz koji kažnjava kritičara kada gradijent nije jednak 1. Prednost ove metode je što ne ograničava mrežu u učenju za razliku od metode ograničavanja težinskih faktora.

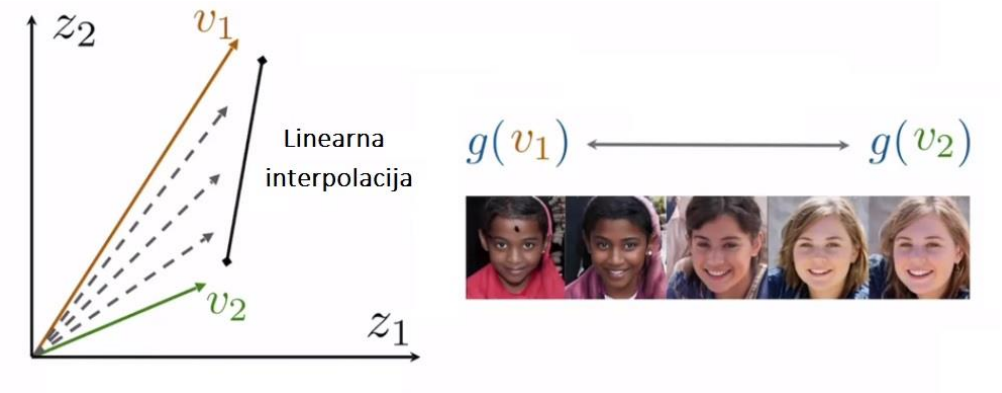
2.4.4. Uvjetna i kontrolirana generacija

Kao što je već spomenuto, osim vektora šuma, u generator se može ubaciti i tražena klasa kada je generator treniran tako da može generirati različite vrste primjera. Generiranje novih vrijednosti s dodatnim ulazom u obliku vektora klase naziva se uvjetna generacija (engl. conditional generation). Na taj se način generatoru može odrediti tražena klasa izlaza. Međutim, kod takve generacije nije moguće u potpunosti kontrolirati sva svojstva dobivenog izlaza. Za tu se svrhu provodi tzv. kontrolirana generacija (engl. controllable generation) kod koje se umjesto korištenja vektora klase provodi postupak mapiranja različitih svojstava u prostoru nasumično generiranih ulaza. Tablica 2.3 prikazuje razlike između uvjetne i kontrolirane generacije.

Tablica 3. Razlike između uvjetne i kontrolirane generacije

Kontrolirana generacija	**Uvjetna generacija**
Definiranje specifičnih traženih svojstava	Definiranje tražene klase
Podaci na kojima se provodi učenje ne moraju biti označeni	Podaci na kojima se provodi učenje moraju biti označeni
Mijenjanje ulaznog vektora šuma	Korištenje dodatnog vektora klase

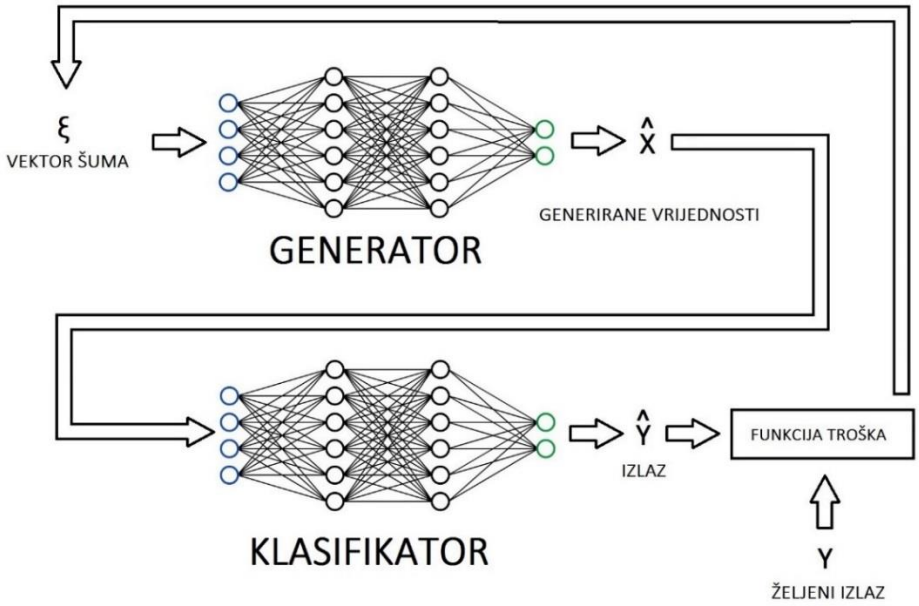
Korištenje kontrolirane generacije također omogućava prikaz tranzicije između dva generirana primjera kao što je prikazano na slici 2.18.



Slika 2.21. Primjer tranzicije između dva generirana primjera [23]

Na slici 2.18 vektori v_1 i v_2 predstavljaju vektore šuma koji su radi preglednosti prikazani samo s dvije ulazne vrijednosti z_1 i z_2. Kod realnih primjera bi se takvi vektori morali prikazati u višedimenzionalnim prostorima s obzirom na to da vektori šuma obično poprimaju velik broj nasumično generiranih ulaznih vrijednosti.

Razne metode se koriste za određivanje smjera u kojem je potrebno promijeniti vektor šuma kako bi rezultat poprimio tražena svojstva. Najčešće se koriste prethodno trenirani klasifikatori za traženo svojstvo čiji gradijent funkcije troška ukazuje na potrebnu promjenu ulazne vrijednosti vektora šuma kao što je prikazano na slici 2.19.



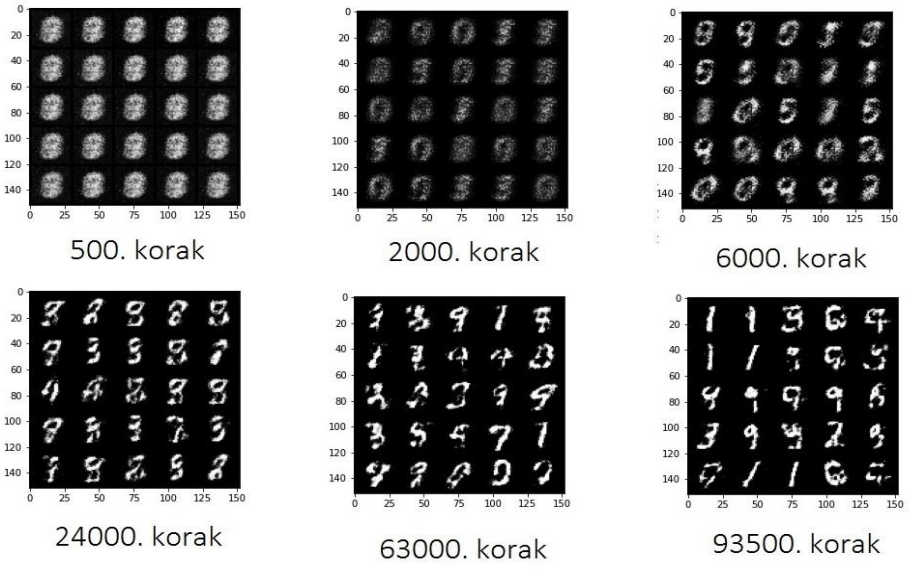
Slika 2.22. Korištenje klasifikatora za generiranje traženih svojstava

2.5. Primjeri korištenja GEKON modela

U ovom će se poglavlju prikazati rezultati korištenja različitih GEKON modela za generiranje „rukom pisanih“ znamenki. Baza podataka na kojoj je obavljen proces učenja modela pod nazivom *MNIST* sadrži 60 000 primjera podataka za treniranje (slika 2.20). Primjeri su spremljeni u obliku crno-bijelih slika rezolucije 28x28 piksela kako bi proces učenja trajao što kraće. Za definiranje modela korišteni su kodovi koji se mogu pronaći na linku [23].

Slika 2.23. Primjer podataka za treniranje [26]

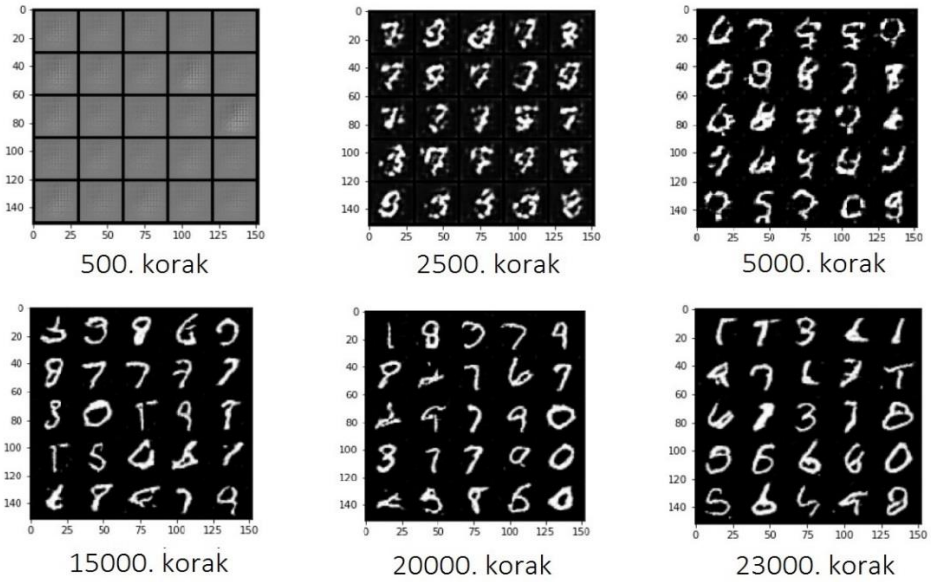
Pri korištenju klasičnog GEKON modela bez konvolucije s gubitkom unakrsne entropije kao funkcijom troška dobiveni su sljedeći rezultati:



Slika 2.24. Rezultati klasičnog gekon modela

Proces učenja na grafičkoj kartici *Intel HD Graphics 520* trajao je oko 15 sekundi za svakih 500 izvršenih koraka.

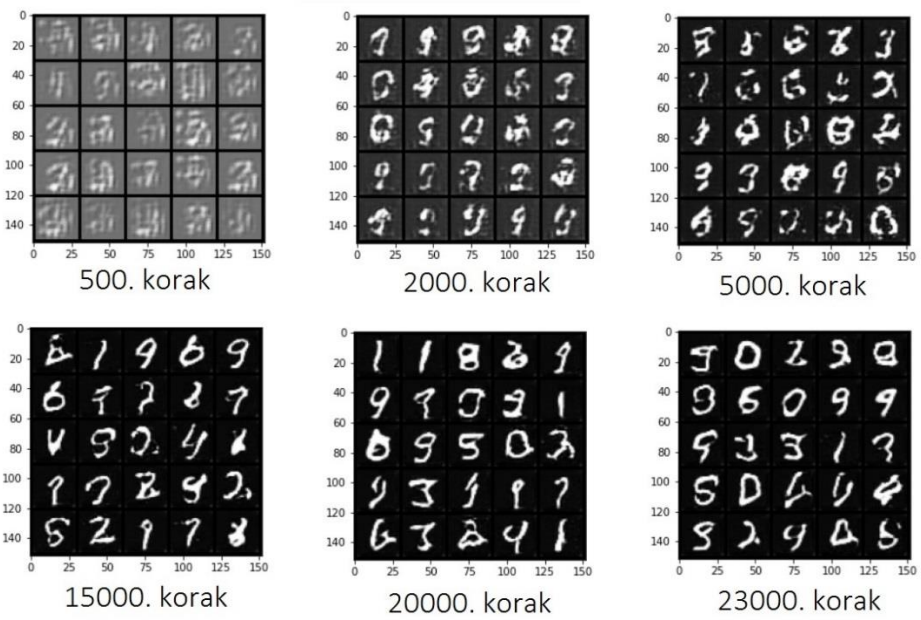
Korištenjem dubokog konvolucijskog GEKON modela dobiveni su sljedeći rezultati:



Slika 2.25. Rezultati dubokog konvolucijskog GEKON modela

Proces učenja na grafičkoj kartici *Intel HD Graphics 520* trajao je otprilike jednako kao i kod klasičnog modela, dakle oko 15 sekundi za svakih 500 izvršenih koraka. Rezultati dubokog konvolucijskog modela vidljivo su bolji od rezultata klasičnog modela čak i kada se uspoređuju rezultati konvolucijskog modela nakon samo 15000. koraka s rezultatima klasičnog modela nakon 93500. koraka. Dakle, s ovom se vrstom modela se za predstavljeni zadatak može uštedjeti na vremenu, ali i dobiti kvalitetnije generirane rezultate.

Korištenjem Wasserstein GEKON modela dobiveni su sljedeći rezultati:



Slika 2.26. Rezultati Wasserstein GEKON modela

Proces učenja na grafičkoj kartici *Intel HD Graphics 520* trajao je oko jedne minute za svakih 500 izvršenih koraka što je otprilike 4 puta duže od klasičnog i dubokog konvolucijskog modela. Učenje Wasserstein modela zahtijeva više vremena zato što je unutar funkcije troška potrebno izračunati više gradijenata za svaki korak. Rezultati ovog model pokazali su se jednake kvalitete kao i kod dubokog konvolucijskog modela. Međutim, prednost ovakvog modela je što je stabilniji od klasičnog i dubokog konvolucijskog modela.

3. ZAKLJUČAK

Generativne kontradikcijske mreže samo su jedan od generativnog oblika strojnog učenja. Međutim, njihov drugačiji pristup učenja koji se temelji na natjecanju dviju neuronskih mreža pokazao se kao postupak koji u mnogim tipovima zadataka nadmašuje ostale generativne modele jer se ne temelji na korištenju funkcije gustoće vjerojatnosti već je naglasak stavljen na generiranju što realističnijih primjera. GEKON model također se pokazao kao vrlo fleksibilna metoda koja pronalazi svoju uporabu u mnogim znanstvenim, tehničkim i umjetničkim sferama. Posebno se ističe njihova uporaba u robotici za: izradu simulacija mogućih budućih ishoda za pojačano učenje, generiranje originalnih putanja mobilnih robota ili gesta socijalnih robota te za rješavanje zadataka kod kojih je učenje potrebno provesti na bazi neoznačenih podataka kao što su prepoznavanje tvrdoće materijala korištenjem taktilnih senzora ili pronalaženje inverzne kinematike i dinamike robotskih manipulatora. Zbog svoje fleksibilnosti i realističnih rezultata, uporaba GEKON modela također dovodi u pitanje i mnoge etičke probleme zbog čega će se uporaba takvih modela morati strogo regulirati u budućnosti.

Rezultati klasičnog, dubokog konvolucijskog i Wasserstein GEKON modela pokazali su važnost poznavanja prednosti i nedostataka svakog od zasebnih vrsta modela za dobivanje željenog rezultata u što kraćem vremenu.

LITERATURA

[1] McCulloch, W.S., Pitts, W.: *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics 5, pp. 115–133, 1943.

[2] Wikipedia, https://hr.wikipedia.org/wiki/Datoteka:Neuron_Hand-tuned.svg [datum pristupa: 6.12.2020.]

[3] Wikipedia, https://en.wikipedia.org/wiki/File:Dirac_distribution_CDF.svg [datum pristupa: 6.12.2020.]

[4] Wikipedia, https://en.wikipedia.org/wiki/File:Logistic-curve.svg [datum pristupa: 6.12.2020.]

[5] A Simple Neural Network- Transfer Functions, MLNotebook, https://mlnotebook.github.io/post/transfer-functions/ [datum posjete: 10.12.2020.]

[6] Ujević Andrijić, Ž.: *Umjetne neuronske mreže*, https://hrcak.srce.hr/file/322233 [datum posjete: 12.12.2020.]

[7] Goodfellow, I.: *Generative Adversarial Networks*, NIPS 2016. Tutorial, https://arxiv.org/pdf/1701.00160.pdf [datum posjete: 15.12.2020.]

[8] Goodfellow I. i dr.: *Generative Adversarial Networks*, 2014., https://arxiv.org/abs/1406.2661v1 [datum posjete: 15.12.2020.]

[9] Goodfellow I., Twitter, https://twitter.com/goodfellow_ian/status/1084973596236144640/photo/1 [datum posjete: 15.12.2020.]

[10] Radhakrishnan P.: *Photoshop 2.0*, https://towardsdatascience.com/photoshop-2-0-a49990e483 [datum posjete: 16.12.2020.]

[11] Jiajun, W. i dr:. *Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling*, Advances in neural information processing systems, 2016.

[12] Yi X., Walia E., Babyn P.: *Generative Adversarial Network in Medical Imaging: A Review*, 2019.

[13] Zakharov E. i dr.: *Few-shot adversarial learning of realistic neural talking head models*, IEEE International Conference on Computer Vision, 2019.

[14] Finn C., Goodfellow I., Levine S.: *Unsupervised Learning for Physical Interaction through Video Prediction*, 2016.

[15] Quian, X. i dr.: *Hardness Recognition of Robotic Forearm Based on Semi-Supervised Generative Adversarial Networks*, 2019.

[16] Ren, H., Ben-Tzvi P.: *Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks*, Robotics and Autonomous Systems, Volume 124, 2020.

[17] Suguitan, M., Bretan, M., Hoffman, G.: *Affective Robot Movement Generation Using CycleGANs*, IEEE, 2019.

[18] Flyr, T., Parsons, S.: *Towards Autonomous Robotic Systems*, Towards Adversarial Training for Mobile Robots, ppt. 197-208, 2019.

[19] Ćurković, P., Jerbić, B., Stipančić, T.: *Hybridization of adaptive genetic algorithm and ART 1 neural architecture for efficient path planning of a mobile robot*, Transactions of FAMENA, Volume 32 Issue 2, p11-20. 10p. 2, 2008.

[20] ThisPersonDoesNotExist, https://thispersondoesnotexist.com/ [datum posjete: 18.12.2020.]

[21] NoArtist, https://www.noartist.io/ [datum posjete: 18.12.2020.]

[22] DeepFaceLab, https://github.com/iperov/DeepFaceLab [datum posjete: 18.12.2020.]

[23] Zhou, S., Zhou, E., Zelikman, E.: *Build Basic Generative Adversarial Networks (GANs)*, Coursera, https://www.coursera.org/learn/build-basic-generative-adversarial-networks-gans [12. mj. 2020.- 1. mj. 2021.]

[24] Shubham, J.: *Covariate Shift- Unearthing hidden problems in Real World Data Science*, Analytics Vidhya, 2017.

[25] Wikipedia, https://en.wikipedia.org/wiki/File:Lipschitz_Visualisierung.gif

[26] LeCun, Y., Cortes, C., Burges, C.J.C.: *MNIST Database*, http://yann.lecun.com/exdb/mnist/ [datum posjete: 20.12.2020.]