UNIVERZITET CRNE GORE
ELEKTROTEHNIČKI FAKULTET

Momčilo Mitrić

Primjena tehnika mašinskog učenja u kompresiji digitalnih slika

MASTER RAD

Podgorica, 2023. godine

UNIVERZITET CRNE GORE
ELEKTROTEHNIČKI FAKULTET

Momčilo Mitrić

Primjena tehnika mašinskog učenja u kompresiji digitalnih slika

MASTER RAD

Podgorica, 2023. godine

PODACI I INFORMACIJE O STUDENTU

Ime i prezime:	Momčilo Mitrić
Datum i mjesto rođenja:	20.06.1992. godine, Nikšić, Crna Gora
Naziv završenog osnovnog studijskog programa i godina završetka studija:	Studijski program Primijenjeno računarstvo, Elektrotehnički fakultet, Univerzitet Crne Gore, 180 ECTS kredita, 2021. godine.

INFORMACIJE O MASTER RADU

Naziv master studija:	Master studije primijenjenog računarstva
Naslov rada:	Primjena tehnika mašinskog učenja u kompresiji digitalnih slika
Fakultet na kojem je rad odbranjen:	Elektrotehnički fakultet

UDK, OCJENA I ODBRANA MASTER RADA

Datum prijave master rada:	03.07.2023. godine
Datum sjednice Vijeća na kojoj je prihvaćena tema:	22.09.2023. godine
Komisija za ocjenu/odbranu rada:	Prof. dr Miloš Daković Doc. dr Miloš Brajović Prof. dr Vesna Popović-Bugarin
Mentor:	Doc. dr Miloš Brajović
Datum odbrane:	godine

Ime i prezime autora: Momčilo Mitrić BApp

E T I Č K A I Z J A V A

U skladu sa članom 22 Zakona o akademskom integritetu i članom 18 Pravila studiranja na master studijama, pod krivičnom i materijalnom odgovornošću, izjavljujem da je master rad pod naslovom

"Primjena tehnika mašinskog učenja u kompresiji digitalnih slika"

moje originalno djelo.

Podnosilac izjave,
Momčilo Mitrić, BApp

U Podgorici, dana 21.12.2023. godine

Predgovor

Porodici, Vama meni najmilijima, Milutinu i Slavici ovim putem želim da izrazim zahvalnost. Hvala vam što ste me učinili boljim čovjekom, boljim drugom i vjerovali u mene. Vaša podrška, strpljenje i blaga riječ su osnova svega što ja jesam i zahvaljujući vama sam naučio kako da neustrašivo slijedim svoje snove. Vaša podrška nije samo podsticaj u mom akademskom putu već putokaz ka novim izazovima i pobjedama.

Posebnu zahvalnost dugujem svome mentoru, Doc. dr Milošu Brajoviću. Predanost i znanje omogućili su mi da brzo prepoznam snage i vrijednosti ovog istraživanja kao i dali smjernice u daljem istraživanju i usavršavanju. Vaša posvećenost me je navela na put kritičkog mišljenja kojim se dolazi do intelektualnog rasta.

Na kraju želim da se zahvalim svim dragim ljudima koji su ostavili trag u mom životu. Čovjek je povezan beskonačnim niti sa drugim ljudima. Te niti su klupko koje posjeduje svaki prijatelj, svaki učitelj, svaki poznanik koji nam je nekada pokazao put, otpjevao pjesmu i izmamio osmjeh na lice. Zato hvala Vama što ste mrseći konce meni život uljepšali i od mene napravili boljeg čovjeka.

Momčilo Mitrić, BApp

IZVOD RADA

Rast primjena tehnika mašinskog učenja u različitim oblastima podstakla je istraživače da istraže njihov potencijal i u kompresiji digitalnih slika. Ovaj rad ima za cilj da predstavi kompresione modele dobijene različitim tehnikama mašinskog učenja kao korisnu altrernativu popularnim algoritmima za kompresiju. Fokus je stavljen na algoritme kompresije sa gubicima koji su zasnovani na konvolucionim neuralnim mrežama.

Osnovni koncepti kompresije predstavljeni su kroz analizu JPEG kompresionog algoritma, koji je jedan od najvažnijih algoritama koji se koristi za kompresije digitalnih slika. Detaljno su objašnjene sve prednosti i mane ovog algoritma kroz detaljan opis svih koraka u kompresiji. Ovaj algoritam i njegove posebnosti koriste se kao referenca na pristupe nove generacije zasnovane na mašinskom učenju.

Posebno su prikazani koncepti mašinskog učenja, sam proces učenja i osnovni gradivni elementi arhitektura kompresionih modela. U opisu, neuralne mreže su raščlanjene na najsitnije gradivne elemente. Poseban značaj je dat gradivnim elementima koje koriste kompresioni modeli, kao i specijalizovanim pristupima procesu minimizacije unutar mašinskog učenja.

Predstavljeno je više kompresionih modela koji se koriste za kompresiju digitalnih slika. Performanse razmatranih kompresionih algoritama ispitane su korišćenjem više standardizovanih mjera kvaliteta, uključujući one koje su bazirane na podražavanju čovjekovog čula vida, kao i one koje se uzimaju u obzir.

Eksperimentalni rezultati predstavljeni u radu zasnovani su na implementaciji kompresivnih modela i metoda kontrole kvaliteta na specijalizovanom nepoznatom setu podataka u programskom jeziku Python. Eksperimentalna evaluacija i validacija zasnovane su na više specijalizovanih Python biblioteka koje se standardno koriste u predmetnoj istraživačkoj oblasti i mašinskom učenju.

Rezultati eksperimentalne validacije pokazuju da slike dobijene tehnikama mašinskog učenja u kompresiji digtalnih slika imaju iste ili bolje karakteristike u odnosu na slike dobijene konvencionalnim algoritmima. Karakteristike pojedinih modela mogu poslužiti za njihovo bolje shvatanje, kao i za afirmaciju njihovog korišćenja u budućnosti.

Ključne riječi:

Kompresioni algoritmi, kompresioni modeli, digitalne slike, kontrole kvaliteta, afirmacija kompresivnih modela

ABSTRACT

The widespread application of machine learning techniques in various fields has prompted researchers to explore their potential in digital image compression. This paper aims to present compression models obtained through various machine learning techniques as a useful alternative to popular compression algorithms. The focus is on lossy compression algorithms based on neural convolutional networks.

The basic concepts of compression are presented through an analysis of the JPEG compression algorithm, which is one of the most important algorithms used for digital image compression. All the advantages and disadvantages of this algorithm are detailed through a thorough description of each compression step. This algorithm and its specificities are used as a reference for next-generation approaches based on machine learning.

Machine learning concepts, the learning process itself, and the basic building blocks of compression model architectures are specifically highlighted. In the description, neural networks are broken down into the smallest building blocks. Special emphasis is given to the building elements used by compression models, as well as specialized approaches to the minimization process within machine learning.

Several compression models used for digital image compression are presented. The performance of the considered compression algorithms is examined using various standardized quality measures, including those based on human visual perception and those that do not take it into account.

The experimental results presented in the paper are based on the implementation of compression models and quality control methods on a specialized dataset in the Python programming language. Experimental evaluation and validation are based on several specialized Python libraries commonly used in the relevant research area and machine learning.

The results of experimental validation show that images obtained using machine learning techniques in digital image compression have the same or better characteristics compared to images obtained using conventional algorithms. The characteristics of individual models can serve to better understand them and affirm their use in the future.

Keywords:
Compression algorithms, compression models, digital images, quality control, affirmation of compression models

Sadržaj:

Uvod

Vještačka inteligencija i njena primjena predstavlja jedan od najvećih izazova XXI vijeka. Na prvi pogled njena primjena djeluje poprilično jednostavno i tiče se automatizacije osnovnih čovjekovih djelatnosti, kao što je rad u fabrici, rad na otvorenom pa čak i prevoz materijala, ali danas vještačka inteligencija sve više i više pretenduje da pojedinim industrijama zamjeni čovjeka mašinom. Ubrzanim razvojem vještačke inteligencije otvaraju se mnoga etička pitanja na koja treba posebno obratiti pažnju. Pojavom autonomnih vozila, automatizacijom većine poslova unutar modernih fabričkih postorojenja direktnom primjenom vještačke inteligencije se gubi veliki broj radnih mjesta što može negativno uticati na društvo. Pored negativnog uticaja, posljednjih godina vještačka inteligencija je dobila široku primjenu u medicini korišćenjem specijalizovanih alata za pronalazak veza unutar velikih skupova podataka. Otkriće teških bolesti u ranim stadijuma, automatizovano prepoznavanje bolesti koje u nekim slučajevima prevazilazi ekspertizu medicinskih stručnjaka govori nam o važnosti naučnih radova baziranih na vještačkoj inteligenciji. Vještačka inteligencija ima veoma široku primjenu. Od prikupljanja, obrade i analize podataka, do automatizacije ljudskih aktivnosti i robotike pa čak i do kreiranja umjetnosti, vještačka inteligencija se prožima čovječanstvom.

Sa druge strane, potreba za kompresijom prati pojavu prvih računara. Kompresija je proces kodiranja informacija u drugoj memorijskoj reprezentaciji. Njen cilj je smanjiti memorijske zahtjeve u odnosu na originalnu reprezentaciju informacije. Računarska memorija je ograničen računarski resurs. Potreba da se taj resurs na najbolji način iskoristi je dovela do ubrzanog razvoja algoritama za kompresiju. Na početku, kompesija se bavila samo običnim (numeričkim ili alfanumeričkim) podacima a kasnije i kompleksnim (digitalnim slikama). Cilj je dobiti slike smanjene veličine, sa vizuelnim karakteristikama originala i najboljim sačuvanim kvalitetom. Rezultati kompresije digitalnih slika dobijeni primjenom tehnika mašinskog učenja treba da imaju iste ili manje veličine podataka u odnosu na popularne kompresione algoritme, kao što je JPEG (engl. *Joint Photographic Experts Group*).

Digitalna slika je mreža piksela, najmanjh jedinica slike, koji nose infomacije o boji (u RGB modelu, to su informacije o prisustvu crvene, zelene i plave boje) i informacije o intezitetu (vrijednosti piksela od 0 do 255). Razvoj pametnih telefona na početku XXI vijeka, je pratilo enormno povećanje broja generisanih digitalnih slika. Te slike je potrebno na siguran način sačuvati i prenijeti putem interneta. Kompresioni algoritmi pomažu da se više slika na siguran

način prenese i sačuva. U mašinskom učenju kompresija digitalnih slika predstavlja široku granu, koja pored klasičnih zahtjeva (smanjenje veličine i zadržavanje kvaliteta), je više fokusirana na specifične mjere kvaliteta (kompresioni modeli su optimizovani za jednu mjeru kvaliteta). To dovodi do novih modela, novih algoritama koji su namijenjeni za rješenje specifičnog problema sa boljim rezultatima.

Ovaj istraživački rad za cilj ima da pokaže da algoritmi mašinskog učenja primijenjeni u kompresiji digitalnih slika imaju potencijal da zamijene popularne algoritme kao što je JPEG.

1. Uvodne informacije o radu

Mašinsko učenje je veoma široka oblast koja obuhvata mnoštvo različitih tehnika, različitih pristupa problemu, odnosno različitih načina da se optimizuje put do nekog boljeg rješenja. To znači da u mašinskom učenji postoji mnogo načina da se riješi isti problem. U konkretnom slučaju, već se pojavio veći broj modela za kompresiju digitalnih slika, pa je potrebno napraviti njihovo poređenje i ispitati njihove performanse, korišćenjem posebno odabranih skupova podataka (dataset-ova). U oblasti vještačke inteligencije i mašinskog učenja, ne postoji najbolje rješenje za sve probleme, niti najbolji model za sve probleme, već samo „najbolje“ rješenje za pojedini problem. To „najbolje“ rješenje je samo trenutno najbolje rješenje i postoji mogućnost da daljim razvojem ili kombinacijom postojećih, ili pak razvitkom novih modela i metoda može doći do boljih rješenja sa boljim željenim parametrima.

Tehnike mašinskog učenja nijesu strogo determinističke. Postoji velika zavisnost rezultata od dostupnih skupova podataka. Samo eksperimentalnom validacijom sa realnim podacima moguće utvrditi koliko su zaista efikasne pri rješavanju određenjog problema.

Problem smanjenja veličine datoteke koji sadrži digitalnu sliku decenijama privlači pažnju naučne javnosti. Kompresioni algoritam JPEG je donio dobre rezultate pri kompresiji slika, ali to ne znači da razvoj alternatativnih pristupa više ne privlači pažnju naučne javnosti. Naprotiv, zbog velike dostupnosti istraživačkih materijala na internetu kao i veće povezanosti naučne zajednice, došlo je do veće saradnje među naučnicima te većih pomaka i većeg interesovanja za proces kompresije u mašinskom učenju.

Ključni razlozi istraživanja prezenzovanog u radu su:

1. Efikasnije korišćenje memorijskog prostora. Memorijski prostor je ograničen i smanjenje veličine datoteka zarad bolje iskorišćenosti memorije je jedan od

osnovnih razloga za istraživanja u kompresiji digitalnih slika. Navedeno je relevantno i zbog prenosa digitalnih slika putem mreže.

2. Potreba za pravljenjem što povoljnijeg kompromisa između nivoa kompresije i kvaliteta komprimovanih slika.
3. Pored gore navedenih razloga, moguće je doći do zaključaka o osobinama modela za posebne grupe slika. Ovaj princip će dovesti do boljeg razumijevanja modela za specifične slučajeve.
4. Detaljna komparativna analiza različitih kompresionih modela. Poređenjem više modela moguće je na osnovu performansi izabrati najbolji, pa prezentovani rezultati mogu doprinijeti povećanju upotrebne vrijednosti razmatranih modela.

Motivi istraživanja:

1. Prepoznavanje modela mašinskog učenja kao boljeg rješenja u odnosu na popularne kompresione algoritme. Potrebno je ispitati da li slike dobijene na ovaj način imaju bolje karakteristike u odnosu na JPEG kompresioni algoritam.
2. Bolja prepoznatljivost pojedinih modela. Rezultati dobijeni na velikom skupu podataka, koji iznova potvrđuju njegovu relevantnost, mogu dovesti i do šireg korišćenja modela.
3. Bolji vizuelni kvalitet komprimovanih slika. Jedan od osnovnih motiva je dobiti što bolji vizuelni prikaz, sačuvati bitne informacije pri manjim nivoima kompresije i u tom smislu nadmašiti konvencionalne kompresione pristupe.
4. Primjena pojedinih modela na specijalnim grupama slika. Rezultati predstavljeni u radu se mogu koristiti kao osnova za buduća istraživanja i korišćenje tehnika mašinskog učenja u kompresiji digitalnih slika. Slike korišćene u medicini, i ostalim naukama gdje je važan osvrt na detalje moraju da imaju posebne zadatke u pogledu očuvanja detalja ili preciznosti reprodukcije boja. Rezultati komparativne analize različitih modela mašinskog učenja namijenjenih kompresiji digitalnih slika mogu dati veoma relevantne informacije u ovom kontekstu.

Ciljevi istraživanja:

1. Utvrđivanje da li modeli mašinskog učenja imaju potencijal da zamijene klasične pristupe za kompresiju digitalnih slika.
2. Utvrđivanje kako različiti parametri: dimenzije slike, rezolucija, sadržaj slike, kolorit, itd. utiču na rezultate kompresije.
3. Detaljna komparativna analiza, koja će za veći broj modela, primijenjena na jednom skupu podataka, kvantitativno prezentovati performanse različitih pristupa, u cilju utvrđivanja najboljeg.

1.1. Predmet dosadašnjih istraživanja u oblasti kompresije digitalnih slika u mašinskom učenju

Predmet istraživanja u ovom radu je mašinsko učenje primijenjeno na kompresiju digitalnih slika. Konkretno, rad će se baviti analizom performansi kompresionih modela, dobijenih obučavanjem različitih vrsta konvolucionih neuralnih mreža.

Nakon dobijanja komprimovanih slika, rezultati će biti vrednovani putem objektivnih mjera kvaliteta, u svrhu njihovog poređenja. Metode kontrole kvaliteta, ali i prikaz rezultata i vizuelizacija primjera će biti implementirani u programskom jeziku Python. Python posjeduje mnoštvo biblioteka koje se bave mašinskim učenjem, obradom slike i podataka. Istraživanja će pokazati kako se mogu biblioteke i istrenirani kompresioni modeli iskoristiti za kompresiju, te kakve će performanse pokazati nad konkretno odabranim skupovima podataka (bazama slika, odnosno, dataset-ovima).

Glavne biblioteke korišćene u programskom jeziku Python su:

- **TensorFlow** biblioteka koja se bavi mašinskim učenjem.
- **OpenCV** biblioteka koja se bavi obradom slike.
- **OS** biblioteka koja se bavi sistemskim komandama.
- **NumPy** biblioteka za operacije sa matricama.
- **TensorFlow-**Compression models biblioteka sa istreniranim modelima kompresije.
- **Matplotlib** biblioteka je korišćena za vizuelizaciju rezultata.

Rad će prikazati kakve rezultate kompresioni pristupi zasnovani na mašinskom učenju

daju u poređenju sa algoritmima za kompresiju digitalnih slika. Cilj je dobiti odgovor na pitanje: da li pristupi zasnovani na neuralnim mrežama mogu i treba da zamijene klasične pristupe?

Mnoštvo radova definišu mašinsko učenje pri kompresiji slika i mjere vrednovanja kvaliteta kompresije digitalnih slika. Ti radovi opisuju procese dobijanja najboljih rješenja, specifičnosti pojedinih rješenja kao i ideja na kojima su rješenja nastala. Potrebno je sagledati ta istraživanja. U istraživanjima su opisani elementi, transformacije, kao i različiti pristupi vrednovanja pojedinih modela.

Balle et al. [1] i drugi su uveli parametarsku linearnu transformaciju, radi približavanja normalnoj distribuciji. Sama transformacija se sastoji od nekoliko koraka i nakon toga se vrši normalizacija. Dalja optimizacija je na bazi podataka sa prirodnim slikama gdje je cilj optimizacije smanjenje odstupanja od normalne raspodjele. Kao glavna prednost transformacije je njena diferencijabilnost, što omogućava inverznu transformaciju. To omogućava konstrukciju modela sa statičkim podacima na slikama. Ti podaci će se koristiti da bi se na osnovu predikcije otklanjao šum.

Jedan od načina obrade slika jeste pomoću sistema koji simulira ljudski vizuelni sistem (na mrežnjači oka). Taj sistem se sastoji od lokalne kontrole pojačanja (intezitet piksela se prilagođava ostalim pikselima u blizini da bi se poboljšala percepcija kontrasta) i lokalnog oduzimanja svjetlosti (gdje se od svjetlosti pojedinog piksela oduzima srednja svjetlost okoline). Na ovaj način oko raspoznaje važne informacije o objektima. Metriku kvaliteta slike zasnovanu na ranom vizuelnom sistemu su predstavili Lappara et al. [2], gdje ova metrika ima bolje rezultate u odnosu na standardne mjere kvaliteta.

Balle et al. [3] sa kolegama je opisao kompresiju slika koristeći analizu nelinearne transformacije, uniformnog kvantizera i nelinearne sintezne transformacije. Nelinearne sintezne transformacije se odnose na generisanje novih podataka sa otklonjenim šumom, a nelinearne transformacije analize se odnose na analiziranje određenih ulaznih podataka radi otkrivanja skrivenih veza između oblika ili struktura koje ne mogu lako biti otkrivene linearnom transformacijom. Uniformni kvantizer dijeli opseg kontinualnih vrijednosti na određeni broj intervala, pri čemu svaki interval predstavlja jedan mogući simbol. Vrši se aproksimacija najbližim simbolom i kontinualne vrijednosti postaju diskretne. Metodologija koja se koristi različita je od metodolologije konvolucionih neuralnih mreža (mreže koje su posebno dizajnirane za obradu podataka sa prostornom strukturom, odnosno slika), jer koristi lokalnu kontrolu pojačanja pri modelovanju neurona. Stohastički gradijentni spust (gradijent se procjenjuje na

osnovu uzorka podataka ne na osnovu svih podataka trening seta) poboljšava brzinu konvergencije (računa se mnogo manji broj gradijenata). Takođe, rezultati su bolji u odnosu na savremene kompresione algoritme kao što je JPEG.

Rasprostranjenost mašinskog učenja dovodi do različitih algoritama koji se koriste za optimizacione probleme. Kingma et al. [4] je predstavio algoritam koji ima čestu primjenu kod treninga neuralnih mreža - Adam. Zasnovan je na gradijentu stohastične funkcije cilja (funkcija cilja se stohastički definiše, sa određenim uticajem slučajnosti) i ima veliku adaptivnost. Definiše se preko tzv. momenta i adaptivne stope učenja. Moment se odnosi na ponašanje na osnovu prethodnih gradijenata a stopa učenja se definiše za svaki parametar na osnovu gradijenata, što omogućava efikasniju i stabilniju konvergenciju. Dakle, Adam je algoritam prilagodljiv različitim parametrima. Pogodan je za velike skupove podataka i mašinsko učenje, zbog malih memorijskih zahtjeva, velike brzine konvergencije i jednostavnosti implementacije.

U radu [5] se istražuje evaluacija pojedinih generativnih modela. Generativni modeli su modeli koji imaju zadatak stvaranja novih podataka koji su slični podacima na kojima su trenirani. Pored primjene u kompresiji slika, moguće je generativne modele koristititi za uklanjanje šuma, dodavanje i rekonstrukciju nedostajućih podataka, polu-nadgledano učenje. Zbog širokih primjena nije lako definisati i ocijeniti pojedini model. U fokusu rada je više načina upoređivanja generativnih modela sa naglaskom na generativne modele slika.

Statističke karakteristike i statistička raspodjela piksela koja se javlja unutar prirodnih slika je interesantna za modelovanje. Prirodne slike su slike sa objektima i dešavanjima koja se mogu naći u stvarnom svijetu. Analizira se boja, tekstura, oblik da bi se vršilo modelovanje. Modelovanje je kompleksno, zbog zavisnosti koje se prostiru na više stotina piksela. Theis et al. [6] je opisao rekurentne neuralne mreže (mreže koje uzimaju u obzir prethodne informacije o podacima) sa rekurentnim modelom, koje su uspješne u prepoznavanju zavisnosti u raznim problemima, a tek u skorije vrijeme imaju svoju primjenu u generativnim modelima. Model se može primijeniti na slikama različite veličine i ima svoju primjenu pri sintezi tekstura i popunjavanja oštećenih djelova slike.

Specijalizovana neuralna mreža koja ima za zadatak da kodira i dekodira podatke pri čemu se postiže kompresija naziva se autoenkoder (engl. *Autoencoder*). Najčešće se sastoji se od dva dijela: dekodera kojim se vrši dekodiranje i kodera koji vrši kodiranje. Postoji više vrsta autoenkodera pri čemu varijacioni rekurentni autoenkoder je opisao Gregor et al. [7]. Zbog povezanosti tj. zbog uzimanja prošlih informacija u obzir, pokazalo se da je ovaj način

modelovanja bolji u odnosu na druge metode modelovanja. Autoenkoder ima osobinu da zadržava bitne globalne karakteristike, a manje važni elementi se zanemaruju.

Kompresija može biti sa i bez gubitaka. Kompresioni JPEG algoritam spada u klasu algoritama sa gubicima (gubici su na kvalitetu i informacijama). U radu [8] opisana je kompresija sa gubicima za slike sa velikom rezolucijom, primjenom neuralnih mreža. Koder i dekoder su bazirani na rekurzivnim neuralnim mrežama, binarizatoru (pretvara kontinualne vrijednosti u binarne) i neuralnim mrežama za kodiranje entropije (kodira binarne kodove u kodove entropije). Entropija u kontekstu kompresije slika predstavlja mjeru neizvjesnosti i cilj je što više smanjiti radi bolje kompresije.

U prethodno referenciranim radovima već je pomenuta prirodna distribucija slika. Novi model su predstavili Oord i kolege u radu [9]. Specifičnost ovog modela je da sekvencijalno predviđa piksele duž dvije prostorne dimenzije, primjenom neuralnih mreza. Pod prostornim dimenzijama se smatra broj piksela po kordinatama. Za 1600×1200 prostore piksela dimenzije su: 1600 i 1200. Uzorci dobijeni na ovaj način izgledaju oštro, raznoliko sa određenim smislom u rasporedu objekata.

U radu [10] se govori o svojstvima neuralnih mreža. U radu se vrše eksperimenti na više poznatih skupova podataka zarad boljeg prepoznavanja uticaja povezanosti neurona - aktivacionih jedinica. Govori se o vezama između neurona u dubokom učenju, osobinama neurona kao i procesima koju utiču na postupak mapiranja ulaz-izlaz odnosa neuralne mreže.

1.2. Metodologija istraživanja

Kako bi se testirale hipoteze i istraživačka pitanja u vezi sa primjenom tehnika mašinskog učenja u kompresiji digitalnih slika mogu se koristiti različite metode, uključujući:

1. Odabir skupa podataka. Skup digitalnih slika nad kojima se vrši kompresija mora da ima određenu dosljednost. Zahvaljujući velikom broju dostupnih skupova slika i velikog broja internet stranica posvećenih mašinskom učenju, lako je doći do velikog broja slika sa određenim sadržajem (slika prirode, slika automobila, cvjetova), koje mogu poslužiti za ciljano ispitivanje efikasnosti razmatranih kompresionih algoritama.
2. Eksperimentalna evaluacija modela i algoritama. Pošto se u radu koristi veliki broj modela za kompresiju digitalnih slika, koji već imaju istrenirane parametre za

kompresiju, samo prikupljanje rezultata predstavlja upoređivanje originalne slike sa slikom dobijenom preko određenog modela. Rezultati se odnose na veličinu kompresovane slike, mjere kvaliteta kompresije i veličinu slike dobijene kompresijom primjenom tehnika mašinskog učenja i slike u .jpeg formatu koji je danas najviše rasprostranjen.

3. Obrada eksperimentalnih rezultata. U radu je izabran reprezentativni uzorak od 400 slika iz jednog skupa slika. Kompresijom, od uzorka se generišu 28 800 slike. Od jedne slike se generišu 72 nove slike koje predstavljaju različite stepene kompresije za različite kompresione modele i sa njih se preuzimaju rezultati o kvalitetu kompresije. Oni se dalje obrađuju preko raznih alata i biblioteka programskog jezika Python. Ova obrada predstavlja rad sa velikom količinom podataka, zarad bolje i preglednije, organizacije i analize.
4. Integracija eksperimentalnih rezultata. Dobijeni rezultati će biti semantički povezani sa drugim značajnim podacima korišćenjem alata i biblioteka programskog jezika Python. Biće izvedeni zaključci na osnovu kvaliteta kompresije, ali i izdvojeni prosječni rezultati za određene modele kompresije kao i odnos dobijenih slika i slika u .jpeg formatu.
5. Vizuelizacija rezultata. Rezultati će biti vizuelizovani korišćenjem programskog jezika Python. Pošto su prikupljeni eksperimentalni podaci već obrađeni, lako će biti preuzeti i prikazani preko namjenskih biblioteka za vizuelizaciju. Alati i tehnologije korišćene za vizuelizaciju će dovesti do bolje primjene modela kao i do boljeg uvida u rezultate istraživanja.
6. Diskusija rezultata. Svi prethodno dobijeni rezultati će biti detaljno diskutovani i upoređeni sa teorijskim očekivanjima u vezi ispitivanih pristupa.

1.3. Očekivani rezultati, primjena i doprinosi

Predstavljeni eksperimentali rezultati i prateća diskusija imaju za cilj da pokažu validnost izbora pojedinih modela u odnosu na klasične algoritme kompresije. Predstavljeni doprinosi su:

1. Istraživanje će pokazati da su rezultati dobijeni primjenom kompresionih modela mašinskog učenja, da nude poboljšanja, u odnosu na klasične kompresione algoritme.

2. Perceptualne mjere kvaliteta komprimovanog, odnosno, dekomprimovanog sadržaja, ostaju u okviru zadovoljavajućih i objektivno prihvatljivih nivoa, što će biti eksperimentalno potvrđeno primjenom modela na realne dataset-ove.
3. Dobijeni statistički rezultati će pružiti relevantne informacije o upotrebljivosti analiziranih kompresionih modela u praksi, odnosno, u okviru specijalizovanih aplikacija, servisa i usluga koje iziskuju njihovu upotrebu.
4. Posebno bitan doprinos predstavljaju rezultati dobijeni testiranjem algoritama mimo skupova podataka (dataset-ova) koji su korišćeni za obučavanje modela. U tom smislu, doprinosi teze će ili potvrditi publikovane rezultate, ili ukazati na određene specifičnosti ili nedostatke, koji su relevantni za upotrebljivost algoritama.

Primjena rezultata je moguća za dalja istraživanja vezana za mašinsko učenje u kompresiji digitalnih slika, prvenstveno na radove koji se bave kompresijom te radove u kojima se pominje evaluacija pojednih modela. Naravno, podaci o modelima dobijeni na ovaj način predstavljaju podatke vezane za datasetove.

Istraživanje prezentovano u radu je navelo na zaključak da uprkos postojanju velikog broja modela, kao i velikog broja algoritama za kompresiju, da se svakodnevno razvijaju novi modeli. Većina predstavljenih modela u radu je stara samo nekoliko godina i ne postoji idealan model za sve digitalne slike već samo trenutni najbolji model za pojedinačnu sliku.

1.4. Struktura master rada

U narednim poglavljima će biti detaljinje objašnjeni osnovni principi kompresije (tradicionalni, oni zasnovani na mašinskom učenju), eksperimentalni rezultati i doprinosi rada.

U sljedećem poglavlju biće predstavljen JPEG kompresioni algoritam, koji je već pomenut kao primjer algoritama za kompresiju sa gubicima. Detaljno će biti objašnjen princip, osnovni koraci i DCT transformacija. Takođe će slikovito biti pojašnjeni neki od problema i nedostataka, koji se javljaju korišćenjem konvencionalnog JPEG algoritma.

Treće poglavlje sadrži fundamentalne teorijske koncepte na kojima je zasnovana kompresija u mašinskom učenju, uključujući: neuralne mreže, konvolucione neuralne mreže, odnosno autoenkoder kao i varijacioni autoenkoder. Biće objašnjene osnovne ideje iza ADAM arhitekture, kao i osnove stohastičkog gradijentnog algoritma. Posebna pažnja biće posvećena

detaljnom teorijskom opisu pristupa zasnovanog na optimizaciji parametara neuralnih mreža, za koji se očekuje da unosi određena poboljšanja u odnosu na druge pristupe.

U četvrtom poglavlju će biti prikazana metodologija postizanja rezultata. Podaci biće obrađeni primjenom više specijalizovanih Python biblioteka (modula), kao što su: Tensorflow, OpenCV, OS i Tensorflow-Compression. U poglavlju će biti detaljno objašnjene spomenute biblioteke, kao i dataset-ovi nad kojima će biti sprovedena eksperimentala evaluacija. Biće predstavljene funkcije koje se koriste kao perceptualne mjere kvaliteta PSNR, MSSSIM i SSIM i opis eksperimentalne postavke.

U petom poglavlju će biti prezentovani, sumirani i analizirani rezultati eksperimentalne validacije. Rezultati će biti prezentovani (tabelarno i vizuelno), gdje se upoređuju različite mjere kvaliteta primijenjenih kompresionih algoritama i modela.

U poslednjem poglavlju biće prezentovan nacrt istraživanja u oblasti kompresije digitalnih slika primjenom tehnika mašinskog učenja. Biće diskutovani rezutati eksperimenta i biće predstavljene smjernice za korišćenje kompresionih modela.

2. JPEG algoritam za kompresiju

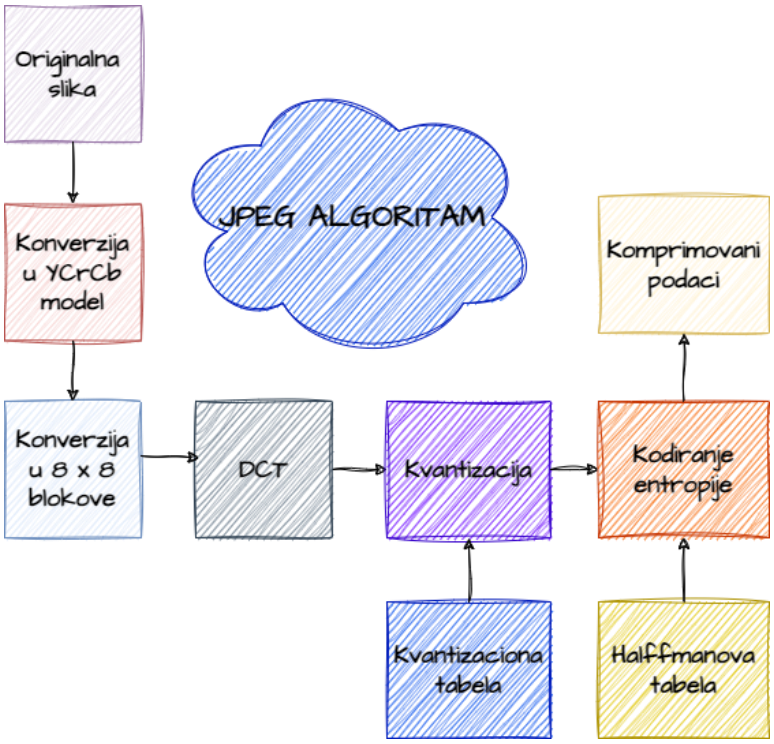
JPEG predstavlja vodeći standard za kompresiju digitalnih slika i najčešću metodu kompresije sa gubicima u radu sa digitalnom fotografijom. Stepen kompresije je podesiv, što znači da je moguć odabir odnosa kvaliteta kompesije i potrebne memorije za čuvanje digitalnih slika. Ovaj standard je uveden 1992. godine i poseban značaj ima u kompresiji digitalnih slika. Njegov značaj je posebno izažen pojavom pametnih telefona, koji je doveo do naglog porasta u broju generisanih digitalnih slika. Zbog lakoće fotografisanja, od 2015. godine broj digitalnih fotografija je povećan na čak nekoliko milijardi dnevno.

Standard koji se bavi obradom digitalnih slika EXIF (Exchangeable image file format) koji specificira formate za digitalne slike koristi JPEG za kompresiju fajlova sa gubicima. Važni podaci o digitalnim slikama, uključujući informacije kao što su postavke kamere, vrijeme snimanja, GPS koordinate i druge relevantne informacije o slici su opisane unutar standarda.

Od osnovnih informacija važno je napomenuti da slike komprimovane JPEG algoritmom sadrže ekstenziju fajla ".jpg" ili ".jpeg" i da .jpeg slika može sadržati najviše 65 535×65 535 piksela.

Osnovni koraci JPEG algoritma su predstavljeni na slici 1. Svi koraci će biti detaljno

objašnjeni u daljem izlaganju.



Slika 1: Osnovni koraci JPEG kompresionog algoritma

2.1. Konverzija slike u drugi kolorni model

U digitalnoj obradi slike je definisano više kolornih modela. Određeni kolorni modeli se koriste za štampanje, drugi za prikaz na internetu, a neki od njih da bi se istakle značajne informacije koje će se koristiti u procesu kompresije. Neki od kolornih modela su:

- **CMYK** (**C**yan, **M**agneta, **Y**ellow, **B**lack) koristi se za štampanje sa 4 glavne boje u imenu (Cijan, Magneta, Žuta, Crna).
- **Pantone** sistem koristi se za štampanje ali za razliku od CMYK boje su definisane posebno. Svaka boja će imati svoj indetifikacioni broj što će omogućavati doslednost boja u različitim medijima i proizvodnim procesima.
- **RGB** (**R**ed, **G**reen, **B**lue) je model boja sastavljen od uređenje trojke (R, G, B), gdje se svaka od boja može predstaviti kao kombinacija crvene, zelene i plave. Drugim riječima, crvena boja je predstavljena uređenom trojkom (255, 0, 0),

plava je predstavljena uređenom trojkom (0, 0, 255), zelena (0, 255, 0) a sve ostale boje su kombinacija crvene, plave i zelene. Kombinacija (0, 0, 0) predstavlja crnu boju, a (255, 255, 255) predstavlja bijelu boju.

Slika 2: Ilustracija RGB modela

Postavlja se pitanje: Zbog čega je uopšte potrebna konverzija u drugi sistem boja? Većina slika koje se nalaze na internetu su opisane u RGB kolornom modelu. Ovaj model boja nije pogodan za JPEG algoritam, jer sadrži tri komponente sa istom važnosti, a to kompresiju čini bespotrebno kompleksnom. Kao što je već napomenuto, boja unutar RGB modela je kombinacija od $256 \times 256 \times 256$ boja (vrijednosti komponenta boja su u intervalu od 0 do 255). Da bi se izbjegao veliki broj kombinacija boja, kao i da bi se stavila komponenta svjetline u prvi plan, JPEG algoritam će koristiti drugi prostor (model) boja. Naime, koristi se YCbCr model, koji je opisan sa tri komponente :

1. Y (eng. *luminance*) predstavlja jačinu ili svjetlinu piksela i ima najveći značaj za JPEG algoritam.
2. Cb (eng. *chrominance blue*) predstavlja plavu komponentu boje i ima manji značaj za JPEG algoritam.
3. Cr (eng. *chrominance red*) predstavlja crvenu komponentu boje i ima manji značaj za JPEG algoritam.

Ovaj model boja je mnogo pogodni za JPEG algoritam. Čovjekovo oko najviše primjećuje promjene u Y komponenti ili promjene u jačini svjetlosti piksela. Druge komponente imaju manji uticaj unutar JPEG kompresije i u narednim koracima najviše značaja će imati Y komponenta.

Za digitalni RGB uzorak od 8 bita, gdje svaka RGB komponenta ima vrijednost od 0 do

255, jednačine za transformaciju iz RGB modela boja u YCbCr model su:

$$Y = 16 + \frac{65.738R}{256} + \frac{129.057G}{256} + \frac{25.064B}{256},$$

$$Cb = 128 - \frac{37.945R}{256} - \frac{74.494G}{256} + \frac{112.439B}{256},$$

$$Cr = 128 + \frac{112.439R}{256} - \frac{94.154G}{256} - \frac{18.285B}{256},$$

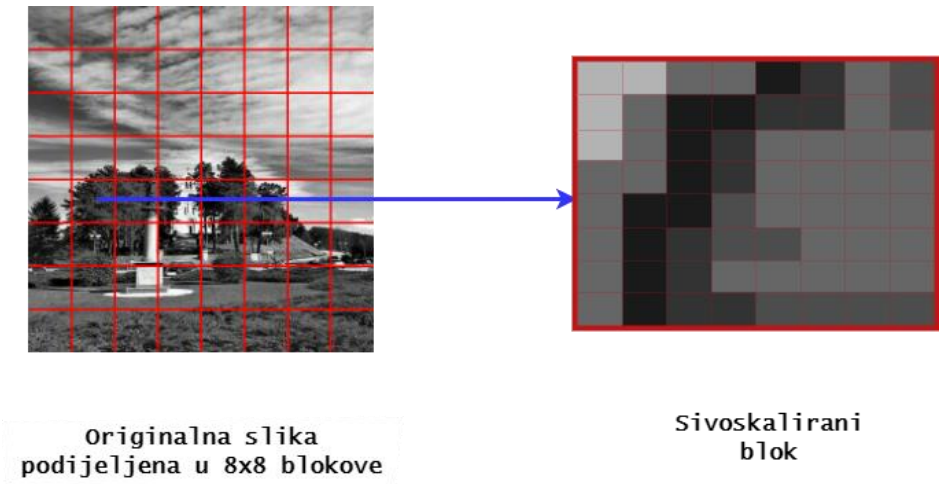
gdje Y, Cb i Cr respektivno predstavljaju vrijednosti osvjetljaja, plave i crvene komponente unutar YCbCr modela, a $R, G,$ i B respektivno predstavljaju vrijednosti crvene, zelene i plave komponente u RGB modelu.

Dalje se vrši zaokruživanje dobijenih vrijednosti ka najbližem cijelom broju, a množenje i dijeljenje se mijenja pomjeračkim (engl. *shift*) registrima.

Y komponenta YCbCr kolornog sistema ima najveći značaj za JPEG algoritam i u daljem izlaganju sve transformacije će se vršiti na Y komponenti slike.

2.2. Konverzija slike u 8×8 blokove

Drugi korak u kompresiji je pretvaranje slike u 8×8 blokove piksela. JPEG standard definiše veličinu bloka na koje će slika biti podijeljena. Digitalna sivoskalirana slika (dobijena kao Y komponenta YCbCr kolornog sistema) se dijeli na 8×8 blokove. Blokovi se dijele od gornjeg lijevog ugla i odvajaju po 8 piksela u desno do kraja slike. Ako veličina slike nije djeljiva sa 8, moguće je dodati pojedine piksele koji su potrebni, ili čak skratiti sliku, da bi bila dobijena efektivna veličina korisna za dalju DCT.



Slika 3: Podjela slike na 8×8 blokove piksela

Svaki 8×8 blok će sadržati informacije o 64 vrijednosti pojedinih piksela. Pošto je u pitanju sivoskalirana slika (samo sadrži Y komponentu), vrijednost na zadatoj koordinati (u posmatranoj vrsti i posmatranoj koloni) ove matrice će biti od 0 (crna boja) do 255 (bijela boja). Početna vrijednost luminance komponente (komponente osvjetljenja) 8×8 bloka ima sljedeće karakteristike:

$$\boldsymbol{Blok}_{8\times8} = \begin{pmatrix} 56 & 59 & 65 & 70 & 73 & 68 & 67 & 72 \\ 61 & 64 & 68 & 95 & 116 & 91 & 72 & 78 \\ 66 & 57 & 74 & 119 & 152 & 107 & 71 & 70 \\ 62 & 64 & 72 & 132 & 160 & 112 & 75 & 71 \\ 68 & 63 & 70 & 109 & 132 & 94 & 73 & 65 \\ 82 & 65 & 55 & 61 & 73 & 64 & 53 & 82 \\ 81 & 70 & 58 & 55 & 53 & 50 & 61 & 80 \\ 74 & 73 & 72 & 65 & 62 & 71 & 74 & 98 \end{pmatrix}.$$

Posljednji korak kojim se završava priprema podataka za DCT je oduzimanje vrijednosti 128 od vrijednosti pojedinih piksela u svakom bloku. Na ovaj način, vrijednost osvjetljaja se postavlja u opseg od -128 do 127. Cilj je da srednje vrijednosti svjetline budu bliske nuli. Ova reprezentacija će dalje pomoći prilikom DCT, budući da je pogodnija za prikaz pozitivnih i negativnih frekvencija. Izgled transformisanog bloka spremnog za DCT je dat u slijedećoj matrici:

$$\boldsymbol{Blok}_{8\times8} = \begin{pmatrix} -72 & -69 & -63 & -58 & -55 & -60 & -61 & -56 \\ -67 & -64 & -60 & -33 & -12 & -37 & -56 & -50 \\ -62 & -71 & -54 & -9 & 24 & -21 & -57 & -58 \\ -66 & -64 & -56 & 4 & 32 & -16 & -53 & -57 \\ -60 & -65 & -58 & -19 & 4 & -34 & -55 & -63 \\ -46 & -63 & -73 & -67 & -55 & -64 & -75 & -46 \\ -47 & -58 & -70 & -73 & -75 & -78 & -67 & -48 \\ -54 & -55 & -56 & -63 & -66 & -57 & -54 & -30 \end{pmatrix}.$$

2.3. Diskretna kosinusna transformacija

Treći korak u JPEG algoritmu je DCT. Pored JPEG algoritma, DCT se takođe koristi u MPEG algoritmu (MPEG algoritam se koristi za kompresiju video sadržaja). DCT-om se prelazi iz prostornog domena, definisanog na već pomenut način preko vrijednosti piksela, u frekvencijski domen. Ova transformacija je posebno korisna u postupcima kompresije podataka, jer omogućava koncentraciju većeg dijela energije signala ili slike u manjem broju koeficijenata,

čime se postiže efikasna kompresija.

Postoji 1D (jednodimenziona) i 2D (dvodimenziona) DCT. 2D DCT se primjenjuje na 2D signale kao što su digitalne slike i u daljem izlaganju kada se pominje DCT misli se na 2D DCT. Dimenzije su broj vrsta i broj kolona, dok su koordinate pojedinačnog piksela redni brojevi vrsta i kolona. Te dimezije su veličina bloka (8×8).

Ova transformacija će nezavisno biti primijenjena na svim 8×8 blokovima na cijeloj digitalnoj slici koja je već podijeljena u blokove i sivoskalirana.

DCT-om se dobija transformaciona 8×8 matrica sa korisnim informacijama o spektralnom sadržaju slike. Te informacije su:

1. U gornjem lijevom uglu matrice se nalazi DCT koeficijent čija vrijednost predstavlja srednju vrijednost promjene svjetline piksela 8×8 bloka sivoskalirane slike. Ime ovog polja je DC.
2. U donjem lijevom uglu se nalazi DCT koeficijent čija vrijednost predstavlja najbržu vertikalnu promjenu frekvencije piksela. Ovaj koeficijent, kao i svi ostali koeficijenti (osim koeficijenta koji se nalazi u gornjem lijevom uglu) su AC koeficijenti.
3. Gornji desni ugao sadrži DCT koeficijent čija vrijednost predstavlja najbržu horizontalnu promjenu frekvencije piksela. Ovaj koeficijent, kao i svi ostali koeficijenti (osim koeficijenta koji se nalazi u gornjem lijevom uglu) su AC koeficijenti.

DCT bloka piksela $a(i,j)$ 8×8 veličine se definiše kao :

$$DCT(k_1, k_2) = \frac{C(k_1)}{2}\frac{C(k_2)}{2}\sum_{i=0}^{7}\sum_{j=0}^{7} a(i,j)\cos\left(\frac{(2i+1)k_1\pi}{16}\right)\cos\left(\frac{(2j+1)k_2\pi}{16}\right),$$

pri čemu važi da je:

$$C(k_1) = \begin{cases} \frac{1}{\sqrt{2}} \text{ za } k_1 = 0 \\ 1 \text{ za } k_1 > 0 \end{cases},$$

$$C(k_2) = \begin{cases} \frac{1}{\sqrt{2}} \text{ za } k_2 = 0 \\ 1 \text{ za } k_2 > 0 \end{cases},$$

gdje $C(k_1)$ i $C(k_2)$ predstavljaju faktor normalizacije, k_1 i k_2 su frekvencijski indeksi, a $a(i,j)$ predstavljaju vrijednost piksela koji se transformiše.

Pokrivene su sve $8 \times 8 = 64$ kombinacije. DCT koeficijient za vrijednost $(k_1, k_2) = (0,0)$ predstavlja vrijednost DC koeficijenta. Ostale vrijednosti $((k_1,\ k_2) = (1,..,7),(1,..,7)\ )$ koeficijenata ***DCT*** matrice su AC koeficijenti, a ***DCT*** predstavlja dobijenu matricu.

DC koeficijenti se kodiraju tako što se DC koeficijent tekućeg bloka oduzme od DC koeficijenta prethodnog bloka, pa se onda kodira njihova razlika. Postupak kodiranja DC koeficijenta će naknadno biti prikazan.

Same kodirane vrijednosti od početnih osmobitnih skaliranih vrijednosti [-128, 127] primjenom DCT-a prelaze u *DCT* koeficijente matrice, koji su kodirani sa 11 bita, pa uzimaju vrijednosti iz opsega [-1024, 1023]. To znači da ovako predstavljeni *DCT* koeficijenti zahtjevaju 3 bita više u odnosu na osmobitne vrijednosti piksela. Međutim, da bi se izvršila kvantizacija i kodiranje, ti koeficijenti se skaliraju tako da se mogu reprezentovati pomoću manjeg broja bita, što čini kodiranje efikasnijim.

Sada će biti prikazana matrica sa vrijednostima pojedinih piksela nakon DCT. Vrijednost bloka ***DCT*** matrice za $(k_1,\ k_2) = (0,0)$ je najveća i, kao što je već pomenuto, predstavlja srednju vrijednost sivoskaliranog piksela u datom bloku - DC koeficijent. Ova vrijednost će imati najveći uticaj na finalni blok. Primijećuje se da vrijednosti koje odgovaraju visokim frekvencijama $((k_1, k_2)\ = (5,6,7),(5,6,7))$ imaju manji uticaj na finalni blok, jer pretežno imaju vrijednosti bliske nuli.

$$\boldsymbol{DCT} = \begin{pmatrix} -406 & -32 & -68 & 24 & 55 & -22 & -3 & 3 \\ 30 & -19 & -69 & 16 & 9 & -3 & -7 & 4 \\ -56 & 6 & 79 & -33 & -30 & 7 & 10 & -3 \\ -52 & 16 & 44 & -9 & -10 & 4 & 2 & -1 \\ 17 & -9 & -20 & -3 & -6 & 4 & -4 & -2 \\ -12 & 6 & 5 & -2 & 1 & 3 & -2 & -1 \\ -4 & 0 & 6 & 0 & 2 & -2 & 9 & 1 \\ -2 & 4 & -7 & 2 & -2 & 6 & -2 & 5 \end{pmatrix}$$

DCT postupak je invertibilan i moguće je rekonstruisati vrijednost originalnog bloka. Kada kažemo da je JPEG kompresija kompresija sa gubitkom, ne govori se o DCT. Gubici će

biti produkt procesa kvantizacije koji se vrši nakon DCT. U procesu dekompresije podataka moguće je izvršiti invertibilnu DCT i ponovo preći iz frekvencijskog domena u prostorni.

2.4. Kvantizacija

Kvantizacija predstavlja ključni korak unutar JPEG algoritma. Tokom ovog koraka vrši se zaokruživanje na najbliži cijeli broj *DCT* koeficijenata skaliranih matricom kvantizacije $\boldsymbol{Q}$, pri čemu se dolazi do gubitka informacija. Ovim se postiže bitno smanjenje prostora potrebnog za čuvanje podataka, što omogućava efikasnu kompresiju slike ili videa.

Posmatrajući $\boldsymbol{DCT}$ matricu koeficijenata primjećuje se da DC koeficijent ima najveću vrijednost, niskopropusni je koeficijent, na poziciji je koja odgovara najmanjoj frekvenciji i ima najveću važnost. Važnost koeficijenata matrice $\boldsymbol{DCT}$ koji odgovaraju nižim frekvencijama će dodatno biti naglašena preko procesa kvantizacije.

Proces kvantizacije je dat formulom:

$$\boldsymbol{DCT}_{\mathrm{Q}}(k_1, k_2) = \mathrm{round}\left\{\frac{\boldsymbol{DCT}(k_1, k_2)}{\boldsymbol{Q}(k_1, k_2)}\right\},$$

pri čemu $\boldsymbol{DCT}_{\mathrm{Q}}(k_1, k_2)$ je nova kvantizovana DCT matrica, $\boldsymbol{Q}(k_1, k_2)$ je matrica kvantizacije (koja će biti predstavljena u nastavku), a $\boldsymbol{DCT}(k_1, k_2)$ je već pomenuta matrica DCT koeficijenata.

Matrice kvantizacije se dobijaju empirijski, na bazi velikog broja eksperimenata. Izbor matrice može se napraviti na osnovu izračunatih grešaka, odnosno, razlike između originalne slike i slike dobijene kompresijom, ili pak na osnovu vizuelnih rezultata. Može se desiti da proizvoljna izabrana matrica kvantizacije produkuje bolji vizuleni rezultat od matrice sa manjom greškom, te efikasan izbor matrice kvantizacije predstavlja kompleksan problem.

Većina matrica kvantizacije za sivoskalirane slike je definisano preko matrice $\boldsymbol{Q}_{50}$:

$$\boldsymbol{Q}_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}.$$

Po rasporedu koeficijenata se primjećuje da su koeficijenti sa većim vrijednostima na

pozicijama koje odgovaraju visokofrekvencijskim DCT koeficijentima. Drugim riječima kada je vrijednost koeficijenta matrice $\boldsymbol{Q}(0,0) = 16$ a $\boldsymbol{Q}(7,7) = 99$ za matricu kvantizacije $\boldsymbol{Q}_{50}$, u procesu kvantizacije $\boldsymbol{DCT}(0,0)$ koeficijent će biti mnogo manje skaliran u odnosu na $\boldsymbol{DCT}(7,7)$ koeficijent.

Pomoću prethodne matrice moguće je izvesti ostale matrice kvantizacije po formuli:

$$\boldsymbol{Q}_{QF} = \text{Round}(\boldsymbol{Q}_{50} q),$$

gdje je matrica $\boldsymbol{Q}_{50}$ definisana za 50% kompresije (eng. *quality factor* - *QF*=50), a nove matrice će biti definisane pomoću parametra *q* koji uzima vrijednosti:

$$q = \begin{cases} 2 - 0.02QF, za\ QF\ \geq 50 \\ \\ \dfrac{50}{QF}, za\ QF < 50 \end{cases}.$$

Dobijeni su $\boldsymbol{DCT}_Q$ koeficijenti matrice:

$$\boldsymbol{DCT}_Q = \begin{pmatrix} -25 & -3 & -7 & 2 & 2 & -1 & 0 & 0 \\ 2 & -2 & -5 & 1 & 0 & 0 & 0 & 0 \\ -4 & 0 & -5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Gubitak informacija je nepovratne prirode jer povratni korak dekvantizacije (prilikom dekompresije kompresovanih podatataka) dovodi do drugačijih koeficijenata u odnosu na prvobitne DCT koeficijente. Drugim riječima, originalni DCT koeficijenti će se znatno razlikovati u odnosu na DCT koeficijente dobijene dekompresijom. Ova osobina je najvažnija osobina JPEG algoritma i kada se pomene da JPEG algoritam predstavlja algoritam sa gubicima, prvenstveno se misli na gubitke u kvantizacijonom procesu. Originalna DCT matrica je:

$$\boldsymbol{DCT\_o} = \begin{pmatrix} -406 & -32 & -68 & 24 & 55 & -22 & -3 & 3 \\ 30 & -19 & -69 & 16 & 9 & -3 & -7 & 4 \\ -56 & 6 & 79 & -33 & -30 & 7 & 10 & -3 \\ -52 & 16 & 44 & -9 & -10 & 4 & 2 & -1 \\ 17 & -9 & -20 & -3 & -6 & 4 & -4 & -2 \\ -12 & 6 & 5 & -2 & 1 & 3 & -2 & -1 \\ -4 & 0 & 6 & 0 & 2 & -2 & 9 & 1 \\ -2 & 4 & -7 & 2 & -2 & 6 & -2 & 5 \end{pmatrix},$$

gdje $\boldsymbol{DCT\_o}$ matrica predstavlja DCT koeficijente prije kvantizacije. Primjećuje se da je u

matrici mali broj koeficijenata sa vrijednošću nula, da matrica zadržava sva svojstva koja su već pomenuta.

Na istoj matrici poslije procesa kvantizacije će biti izvršen proces dekvantizcije:

$$\boldsymbol{DCT\_d}=\begin{pmatrix} -400 & -33 & -70 & 32 & 48 & -40 & 0 & 0 \\ 24 & -24 & -70 & 19 & 0 & 0 & 0 & 0 \\ -56 & 0 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & 0 & 0 & 0 & 0 & 0 \\ 18 & 0 & -37 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

gdje ***DCT_d*** predstavlja matricu poslije dekvantizacije. Primjećuje se veći broj koeficijenata sa vrijednosti nula u odnosu na $\boldsymbol{DCT\_o}$ matricu**.**

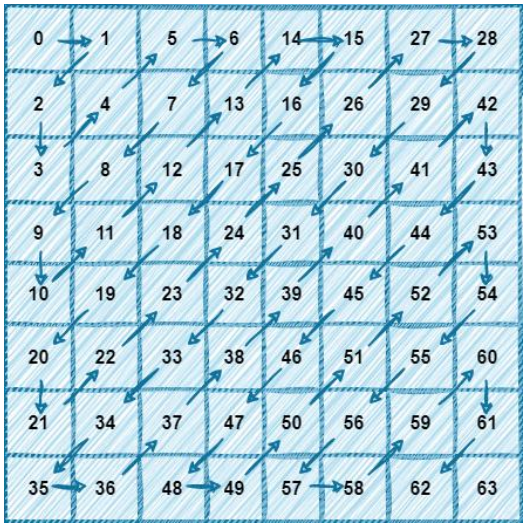
Prilikom procesa dekvantizacije došlo je do promjena u koeficijentima. DC koeficijenti prve i druge matrice su slični (-406 i -400) i primjetno je da sa povećanjem frekvencije gube određene vrijednosti koeficijenata. Već pomenuti niskofrekventni elementi imaju najveći značaj pri formiranju slike, odnosno, oni nose glavni dio informacija o slici i njenom sadržaju. Preostali DCT koeficijenti nose manje relevantne informacije o detaljima. Jedan dio tih informacija se može odbaciti. Primjećuje se da većina preostalih AC koeficijenata dobija nultu vrijednost, čime se gube neznačajne informacije u slici. Ovim postupkom se najčešće gube informacije visokofrekventnih koeficijenata čime se smanjuje veličina slike.

Nakon kvantizacije, kodirane vrijednosti DCT koeficijenata se dalje kompresuju koristeći različite tehnike, kao što je na primjer Hafmanovo kodiranje, što dodatno smanjuje veličinu (memorijskog) prostora potrebnog za čuvanje informacija. Iako se i u ovim fazama može izgubiti dio informacija, kvantizacija se smatra glavnim izvorom gubitaka u JPEG algoritmu i zbog toga ima presudan značaj.

2.5. Kodiranje entropije

Sljedeći i poslednji korak u procesu kompresije podataka je kodiranje entropije. Ovaj korak je veoma važan, jer se njime rješavaju problemi redundancije i olakšava skladištenje. Cilj je predstaviti podatke na način koji će koristiti što manje bitova za skladištenje, što će dovesti do dodatnih ušteda u veličini komprimovane datoteke.

U matrici $\boldsymbol{DCT}_Q$ kvantizovanih koeficijenata najviše se javlja nula, kao vijednost pojedinog koeficijenta. Kada bi svi elementi bili kodirani na isti način, sa istim brojem bita, izgubljena bi bila značajna ušteda memorije na osnovu kodiranja. Elementi se predstavljaju linearnim nizom kvantizovanih koeficijenata.



Slika 4: Cik-cak postupak kodiranja

Prvo treba predstaviti podatke sa matrice koeficijenata kao linearni niz brojeva koji je pogodniji za čuvanje. Pridržavajući se već pomenutih osobina o značaju pojedinih koeficijenata matrice, kodiranje počinje od DC koeficijenta i prati cik-cak obrazac. Ovaj obrazac održava kontinuitet podataka i smjer, jer kod prirodnih slika razlike između susjednih piksela su male. Priroda $\boldsymbol{DCT}_Q$ matrice dovodi do grupisanja nula, tj. formiranja uzastopnih nula u linearnom nizu. Kasnije će se ove nule lako kodirati preko Hafmanovog i RLE kodiranja.

Sljedeći korak je kombinacija RLE (engl. ***R**un **L**ength **E**ncoding*) i Hafmanovog kodiranja na AC koeficijentima. DC koeficijent se kodira na drugi način od ostalih 63 AC koeficijenata (kodira se blok veličine 8×8). Prethodno prikazanim cik-cak postukom za $\boldsymbol{DCT}_Q$ matricu se dobija linearni niz:

-25, -3, 2, -4, -2, -7, 2,-5, 0, -4, 1, 1, 5, 1, 2, -1, 0, -1, 2, 0, 0, 0, 0, -1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Može se uočiti da u posmatranom nizu postoji veliki broj nula na kraju. To je preduslov za dobro kodiranje, kao što će biti objašnjeno u nastavku.

RLE kodiranje je vrsta kodiranja bez gubitaka koja se često koristi u JPEG algoritmu, ali i drugim algoritmima kada se u sekvenci dešava veliki broj ponavljanja pojedinih elemenata. Za proizvoljnu sekvencu AAABBBBACCCCC RLE kodiranje bi dalo rezultat 3A4B1A5C. Broj ponavljanja pojedinih elemenata je (3, 4, 1, 5), a elementi koji se ponavljaju su (A, B, A, C). Broj A kao nezavisni element se pojavljuje dva puta. Postoji više načina RLE kodiranja. JPEG algoritam koristi ovaj način:

$$(RUNLENGTH, SIZE)(AMPLITUDE),$$

gdje za svaki nenulti AC koeficijent važi da je: $RUNLENGTH$ je broj nula prije nenutltog koeficijenta, $SIZE$ je broj bita koji treba da reprezentuju veličinu potrebnu za čuvanje nenultog koeficijenta a $AMPLITUDE$ je reprezentacija nenultog koeficijenta u bitima.

U sledećem tek boksu je prikaz transformacije linearnog cik-cak niza kodiranjem.

(0, 2)(-3);(0, 2)(2);(0, 3)(-4);(0, 2)(-2);(0, 3)(-7);(0, 2)(-2);(0, 2)(-5);(0, 1)(1);(0, 2)(2);(0, 2)(-1);(1, 2)(-1);(0, 2)(2);(4, 2)(-1);(2, 2)(-1);(0, 0);

DC koeficijent se ne kodira na prvom mjestu. Vrijednost 0 iz linearnog niza se neće kodirati, već će se kodirati broj nula između dva nenulta elementa. To znači da u modifikovanoj verziji RLE algoritma se ne broje ponavljanja pojedinih nenultih elemenata, već ponavljanja nula između nenultih elemenata. Linearni niz se završava kodom (0, 0). Ovaj kod predstavlja terminacioni karakter u bloku, odnosno konstataciju da se u bloku ne nalazi ni jedan nenulti koeficijent nakon karaktera (0, 0) što nosi informacije.

Važno konstatovati da je maksimalan broj nenultih elemenata dozvoljenih kodiranjem ograničen sa 4 bita na vrijednost od 15. To znači da ako postoji nenulti AC koeficijent u nastavku a broj nula između dva koeficijenta je veći od 15 koristi se Hafmanova kodna oznaka (15, 0)(0).

Dalje će biti primijenjeno Hafmanovo kodiranje koje će elementima sa više ponavljanja dati kraće kodove zbog učestalosti. U prethodnom primjeru (0, 1) ili (0, 2) kombinacija bi se kodirala sa najmanje bita jer je načešća unutar koda, dok kombinacije koje se ređe pojavljuju bi bile kodirane sa više bitova.

Prikaz kodiranja DC koeficijenta je bio do sada izostavljen. Najvažniji koeficijent se kodira preko DPCM postupka (engl. ***D**ifferential **P**ulse **C**ode **M**odulation*). Umjesto da se kodiraju sve informacije vezane za DC koeficijent, kodira se samo razlika između blokova

pojedinih DC koeficijenata. Za prikaz DPCM kodiranja uzete su slučajne vrijednosti DC koeficijenta. Postupak DPCM kodiranja jednog 5×5 bloka koji sadrži DC koeficijente prikazan je matricama:

$$\begin{pmatrix} 78 & 72 & 81 & 84 & 88 \\ 89 & 88 & 83 & 72 & 71 \\ 75 & 78 & 76 & 81 & 83 \\ 65 & 70 & 69 & 73 & 77 \\ 43 & 40 & 42 & 43 & 49 \end{pmatrix} \xrightarrow{DPCM\ kodiranje} \begin{pmatrix} 78 & -6 & 9 & 3 & 4 \\ 1 & -1 & -5 & -11 & -1 \\ 4 & 3 & -2 & 5 & 2 \\ -18 & 5 & -1 & 4 & 3 \\ -34 & -3 & 2 & 1 & 6 \end{pmatrix}.$$

Pošto razlika između DC blokova najčešće nije velika, jer svaka slika će da sadrži određeni prirodni ton, kontrast, svjetlinu, na ovaj način se čuva vrijednost DC koeficijenta sa mnogo manje potrebne memorije. Kodira se razlika između DC koeficijenata počevši od gornjeg lijevog DC koeficijenta.

Nakon DPCM postupka, AC koeficijenti su predstavljeni Hafmanovim kodiranjem a DC koeficijenti čine jedan niz koji sadrži informacije o slici. Ovaj bitni niz, zajedno sa informacijama o veličini slike, o drugim komponentama modela boja (koje takođe prolaze kroz sve kompresione korake ali imaju manji značaj unutar JPEG algoritma pa nije govoreno o njima), kvantizacionim tabelama i drugim neophodnim metapodacima čini JPEG kompresovani fajl. Taj kompresovani faj je moguće sačuvati na disku ili poslati ga preko interneta, a na drugom mjestu vršiti dekompresiju. Dekompresija će uključivati obrnute korake:

1. Dekodiranje kodova,
2. Dekvantizaciju,
3. Inverznu DCT,
4. Rekonstrukciju orignalne slike.

Već je pomenuto da proces kvantizacije je glavni gubitni proces u JPEG algoritmu.

2.6. Zaključak JPEG

Rekonstruisana slika dobijena JPEG algoritmom će se, zbog kvantizacije, u određenoj mjeri razlikovati u odnosu na originalnu sliku. U nastavku su prikazane dvije matrice sa vrijednostima piksela 8×8 bloka. U lijevoj matrici se nalazi početna sivoskalirana vrijednost bloka koja je predstavljena na samom početku prolaska kroz JPEG algoritam. U desnoj matrici je vrijednost istog bloka poslije oduzimanja vrijednosti 128, DCT, kvantizacije, dekvantizacije,

inverzne DCT i dodavanja vrijednosti 128.

$$\begin{pmatrix} 56 & 59 & 65 & 70 & 73 & 68 & 67 & 72 \\ 61 & 64 & 68 & 95 & 116 & 91 & 72 & 78 \\ 66 & 57 & 74 & 119 & 152 & 107 & 71 & 70 \\ 62 & 64 & 72 & 132 & 160 & 112 & 75 & 71 \\ 68 & 63 & 70 & 109 & 132 & 94 & 73 & 65 \\ 82 & 65 & 55 & 61 & 73 & 64 & 53 & 82 \\ 81 & 70 & 58 & 55 & 53 & 50 & 61 & 80 \\ 74 & 73 & 72 & 65 & 62 & 71 & 74 & 98 \end{pmatrix} \begin{pmatrix} 57 & 66 & 77 & 71 & 70 & 71 & 77 & 66 \\ 68 & 65 & 102 & 92 & 82 & 74 & 104 & 77 \\ 73 & 88 & 129 & 68 & 77 & 93 & 114 & 56 \\ 77 & 74 & 74 & 46 & 84 & 76 & 63 & 55 \\ 69 & 78 & 78 & 74 & 88 & 74 & 78 & 78 \\ 77 & 55 & 63 & 76 & 84 & 46 & 74 & 74 \\ 73 & 56 & 114 & 93 & 77 & 68 & 129 & 88 \\ 68 & 77 & 104 & 74 & 82 & 92 & 102 & 65 \end{pmatrix}$$

Poređenjem dvije matrice, može se uočiti razlika između vrijednosti piksela. Razlika je mala. Čovjek je često ne može uočiti, ali ipak ona je prisutna.

Postoje mnogi problemi i ograničenja, vezani za JPEG kompresiju ali, zbog njene brzine, dobre efikasnosti i dobrih rezultata, ti problemi se često ignorišu. Neki od problema su:

1. Pojava artifakata koji su vidljivi na rekonstruisanoj slici najčešće kao blokovi. Direktna su posledica kvantizacije i vidljivi su na oštrim prelazima i ivicama.
2. Gubitak detalja je neželjeni proizvod kvantizacije. Najviše je vidljiv na teksturama.
3. Aliasing je pojava pruga kao neželjeni efekat kvantizacije. Ovo se posebno može primijetiti u oblastima sa finim linijama ili teksturama.
4. Blooming, koji predstavlja pojavu kada se svijetle oblasti prelivaju u tamne zbog procesa kvantizacije.
5. Šum u JPEG kompresiji može biti pojačan, posebno u oblastima sa malim brojem boja i detalja.

Slika 5: Pojava blokova kao proizvod kompresije

Uprkos svim navedenim problemima i ograničenjima, JPEG kompresija ostaje kao

najrasprostranjeni standard za kompresije sa gubitkom. Ipak, važno je pravljenje kompromisa između kvaliteta i veličine fajla.

JPEG je jedan od najefikasnih pristupa za kompresiju sa gubicima. Koriste ga svi moderni pretraživači uključujući Chrome, Firefox, Safari, i Edge. Probleme sa JPEG algoritmom imaju starije verzije Internet Explorer-a. Jedan od najčešće korišćenih formata za slike je .jpeg. U odnosu na njega prednjači samo .PNG format, ali ovaj format ima veće memorijske zahtjeve za čuvanje.

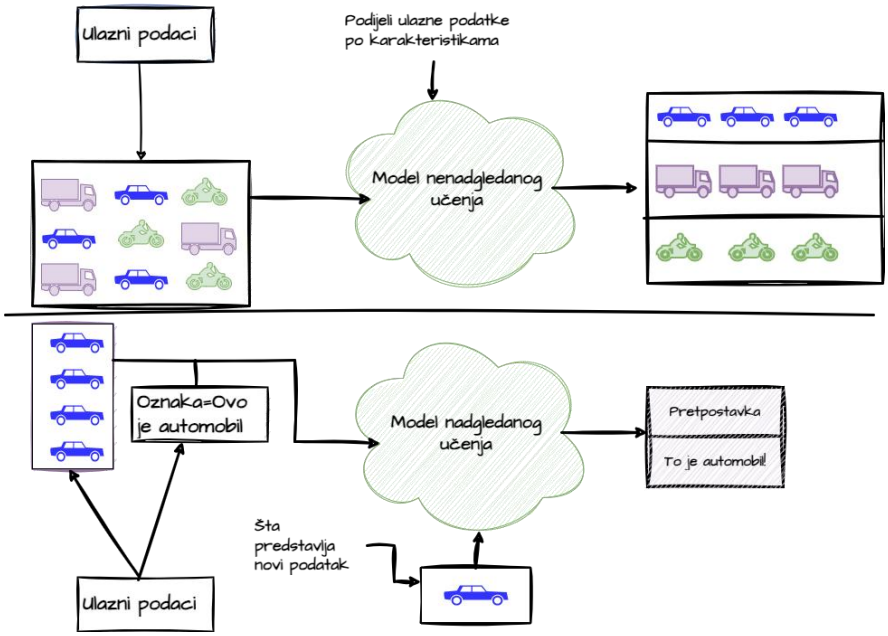
3. Kompresija zasnovana na mašinskom učenju

Razvitkom modernih računara, vještačka inteligencija polako prelazi iz svijeta fikcije u stvarnost. Vještačka inteligencija koja je bila proizvod mašte na kraju 19 vijeka, sredinom 20 vijeka počinje da poprima prve obrise nauke od velikog značaja. Unutar široke oblasti vještačke inteligencije razvija se i podoblast mašinskog učenja. Računar će da "uči" na sličan način kao što to radi čovjek – ponavljanjem i naglašavanjem osobina. Kada djeca uče o vrstama životinja, da bi dijete asociralo određeni izgled sa životinjom, roditelj pokaže djetetu više slika pojedinačnih životinja. Djetetu će se više puta naglasiti koja slika predstavlja koju životinju. Biće prikazane određene osobine životinja (krava ima rogove, mačka ima brkove, pas njušku). Na osnovu prošlih slika i predočenih osobina dijete će, na nekoj novoj slici, prepoznati životinju. Na isti način će algoritmi mašinskog učenja da daju bolje rezutate što su više izloženi podacima. Mašinsko učenje ima veoma veliku primjenu u brojnim ljudskim aktivnostima, u procesima gdje čovjek ne može jasno da nađe povezanosti među podacima. Pored mogućnosti obrade govora, predviđanja ponašanja na berzi, prepoznavanja pisanog jezika, mašinsko učenje se koristi za ranu detekciju bolesti.

Cilj je sprovesti mašinsko učenje primjenom odgovarajućih algoritama nad velikim skupom podataka. Algoritam treba da uoči određene pravilnosti, svojstva, šablone podataka na osnovu kojih je moguće povezati više podataka u cjelinu. Moguće je naznačiti da određeni podaci pripadaju jednoj grupi, moguće je algoritmu proslijediti podatke da on sam uoči šablone u grupi. Na primjer, u medicini na osnovu velikog broja podataka o raznim pacijentima, je moguće da se analizom podataka utvrde obrazci koji upućuju na određene bolesti i stanja. Na bazi velike količine podataka označenih od strane eksperata, algoritam će "naučiti" da prepozna bolesti kod

čovjeka. Iako je sama procedura učenja relativno jednostavna, numerički je veoma kompleksna i najčešće zahtijeva velike računarske resurse. Danas na početku 21. vijeka većina ograničenja po pitanju resursa znatno su smanjena razvitkom računara visokih performansi.

Razvijena su dva principa mašinskog učenja: nadgledano i neneadgledano.



Slika 6: Ilustracija nadgledanog i nenadgledanog učenja

Nadgledano učenje se bazira na podacima za treniranje, koji su već na neki način označeni. Ovi podaci imaju za cilj da usmjere algoritam ka uspješnoj klasifikaciji i na rješavanje nekog kompleksnijeg problema. Na primjer, prilikom klasifikacije cvijeća, kod nadgledanog učenja cvijeće će već biti podijeljeno u grupe (ruže, narcis, karanfili). Set za treniranje treba da sadrži veliki broj pojedinačnih vrsta cvjetova, da bi nadgledano učenje imalo dobre rezultate. Kada algoritmu bude prikazana nova, nepoznata slika ruže, on bi na osnovu njenih karakteristika trebalo da sliku prirdruži grupi ruža, da je kvalifikuje kao ružu. Prateći ulazne i izlazne podatke, moguće je pratiti kako se mijenja uspješnost algoritma tokom vremena. U postupku rudarenja podataka postoje dva načina pristupa problemima:

[1]. Klasifikacionim algoritmima će biti odvojeni u specifične kategorije. Na primjer korišćnjem algoritma klasifikacije se može odvojiti neželjena od korisne pošte.

[2]. Regresija je drugi tip nadgledanog učenja gdje se traže veze između zavisnih i nezavisnih karakteristika. Primjeri su kretanja tržišta na berzi, prognoza rezultata u sportskom klađenju, u poslovima gdje se predviđa brojna vrijednost.

Dalje je predstavljen princip rada nenadgledanog učenja kod algoritama mašinskog učenja. Ovaj tip mašinskog učenja će da ima za cilj da poveže podatke koji nijesu još grupisani. Drugim rijećima, u ovom vidu učenja podaci nijesu označeni, već na "slijepo", bez čovjekove intervencije, se pristupa procesu mašinskog učenja (klasifikaciji, prepoznavanju šablona i skrivenog znanja i grupisanju). Načini pristupa problemima kod ovog tipa mašinskog učenja su:

[1]. Grupisanje (engl. *clustering*) je povezivanje ili grupisanje podataka na osnovu njihovih sličnosti ili razlika. Ovaj princip će stvoriti prirodne grupe ili cluster-e bez potrebe za umiješanoću čovjeka.

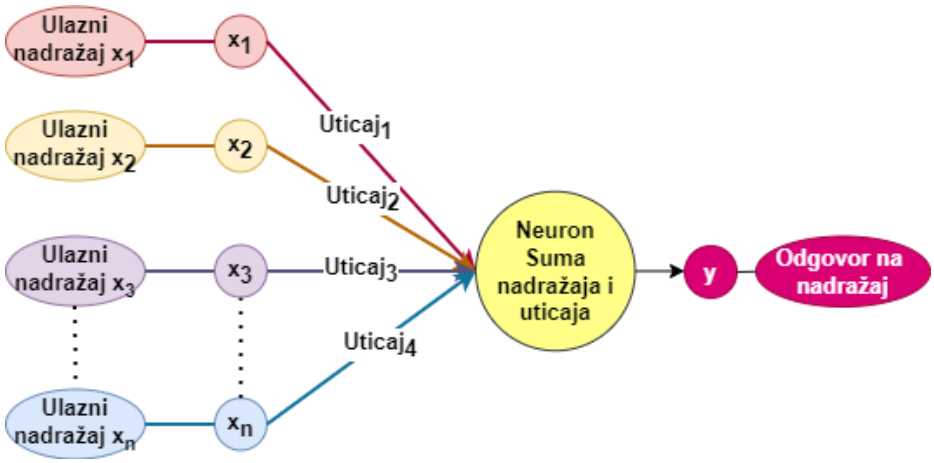
[2]. Redukcija dimenzionalnosti (engl. dimensionality reduction) je proces smanjivanja (redukcije) odbacivanjem nebitnih informacija. U algoritmima koji koriste ovaj pristup cilj je očuvati bitne informacije o podacima, samo u manjem broju dimenzija. Smanjenjem dimezija će se naglo smanjiti i potreba za računarskim resursima.

[3]. Generativno modeliranje će biti vršeno na osnovu raspodjele podataka. Treba napraviti model koji će da kreira nove podatke na osnovu podataka iz trening seta. Generativno modeliranje ima velike uspjehe u primjenama mašinskog učenja na digitalnu fotografiju. Kreiranje slika u boji na osnovu sivoskaliranih slika, kreiranje slika na osnovu upisanih riječi (opisa), kreiranje novih digitalnih slika ljudi su samo neka od mogućih primjena generativnog modeliranja. Ovaj princip ima dva veoma popularna generativna modela: VAE (engl. *Variational Autoencoders*) i GAN (engl. *Generative Adversarial Networks*). U ovom radu veliki značaj će biti dat VAE generativnim modelima, jer oni opisuju modele kompesije digitalnih slika.

Treća kategorija pristupa mašinskog učenja je polunadgledano učenje. U ovom postupku se koriste skupovi podataka koji su označeni, kao i skupovi koji nijesu označeni. Mali je broj označenih podataka u odnosu na neoznačene. U ovom pristupu podaci koji nijesu označeni će služiti za bolje modeliranje cijelog skupa podataka i da se bolje razumije sama struktura podataka. Sa označenih podataka će da se izvuku korisne informacije i da se prošire na neoznačene podatke. Ovaj postupak ima svoju primjenu u medicini prilikom dijagnostike. Mali broj označenih slika će se koristiti sa velikim brojem neoznačenih slika, kako bi se stvorio model koji zna da prepoznaje strukture. Korišćenjem, na primjer generativnih modela, moguće je stvoriti nove slike koje će da izgledaju slično kao podaci korišćeni za treniranje.

3.1. Neuralne mreže

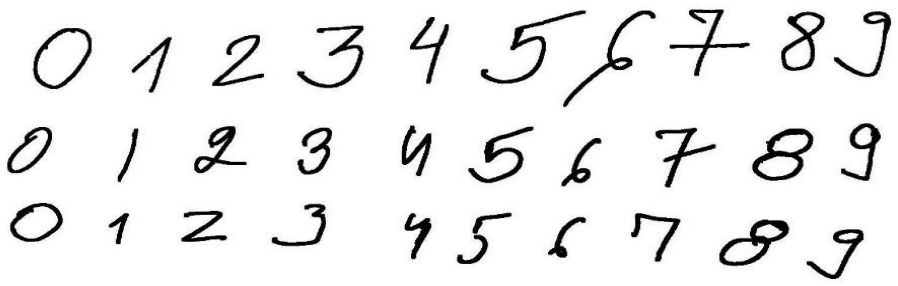
Neuralna (neuronska) mreža predstavlja jedan od osnovnih koncepata u mašinskom učenju. Bazirana je na modelovanju rada čovjekovog mozga. Klučni elementi neuralnih mreža su neuroni, koji imaju moć da razmjenjuju informacije između sebe da bi riješili zadate probleme. Naučnici modeluju veze između neurona, da bi riješili različite probleme. Da li će se pojedini neuron aktivirati zavisi od ulaznog nadražaja i uticaja.



Slika 7: Ilustracija neurona kod (vještačkih) neuralnih mreža

3.1.1. Neuralne mreže kroz prepoznavanje napisanih brojeva

Neuralne mreže poslednjih godina imaju veliku upotrebu u raznim oblastima mašinskog učenja, od prepoznavanja govora, slika i oblika do primjena koncepata mašinskog učenja u autonomnim vozilima. Pogodan primjer za ilustraciju osnovnih koncepata jeste neuralna mreža koja prepoznaje pisane brojeve.



Slika 8: Ilustracija skupa podataka za problem prepoznavanja napisanih brojeva

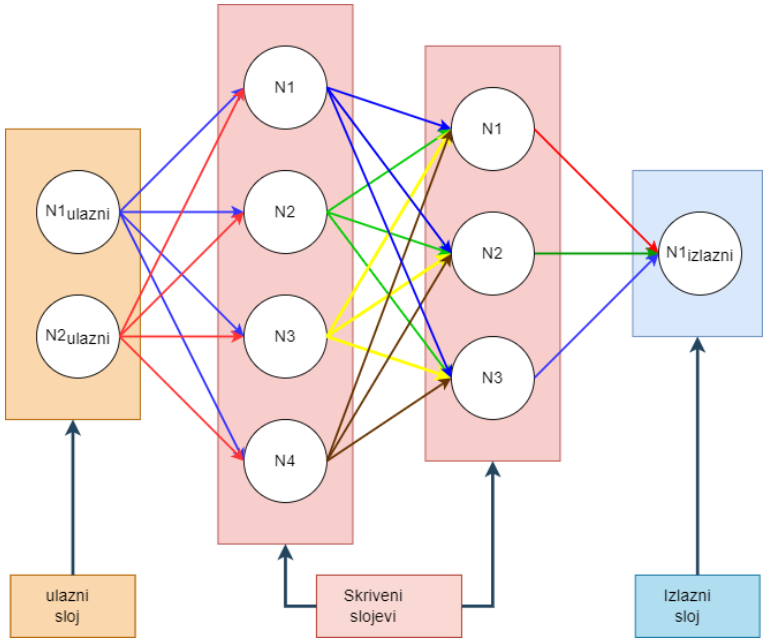
Može se primijetiti da su brojevi na slici 8 napisani na različite načine. Čovjekov mozak

kao proizvod procesa evolucije, lako će pojedine brojeve asocirati, čak iako su različito napisani. Čovjekov mozak sadrži milione neurona sa velikim brojem veza između njih. Naučnicima je cilj da simuliraju proces učenja biološkog nervnog sistema korišćenjem vještačkih neurona. Ova vrsta neurona ima mnogo manje veza nego što je slučaj sa biološkim neuronima.

Čovjek prepoznaje oblike kao cjelinu sastavljenu od mnoštva specifičnih detalja. Na primjer, kada prepoznaje broj 8, čovjekov mozak prepoznaje 8 kao cjelinu koja se sastoji od dva kružića, jedan iznad drugog. Detaljnije kružiće je moguće asocirati sa više oblika: sa dvije horizontalne linije, dvije vertilkalne linije, četiri polukružne linije. Iako se intuitivno krug može prestaviti sa dvije polukružne linije kombinacijom već navedenih oblika, čovjekov mozak ima percepciju kružića i za ostale slične oblike koji sadrže prave horizontalne, prave vertikalne i (ili) četiri polukružne linije. Ove komponente procesa razmišljanja simuliraju i neuralne mreže.

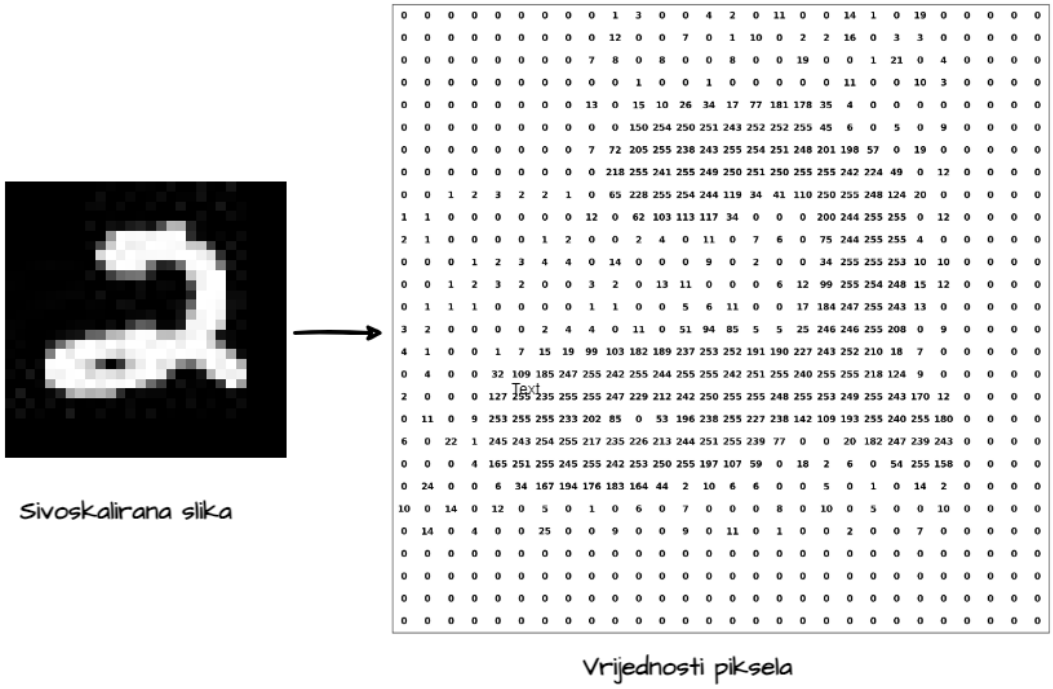
Naime, kada se aktiviraju biološki neuroni, oni će preko veza determinisati aktivaciju drugih neurona, što će aktivirati treće neurone. Navedena svojstva inspirisala su neke od arhitektura neuralnih mreža, pri čemu je broj neurona i veza između neurona mnogo manji nego kod bioloških neuralnih mreža.

Na slici 9 je prikazana arhitektura jedne višeslojne neuralne mreže. Ona se sastoji od ulaznog sloja, izlaznog sloja i dva skrivena sloja. Neuralna mreža se može sastojati od mnogo više skrivenih slojeva.



Slika 9: Arhitektura višeslojne neuralne mreže

Neuralna mreža, koja se koristi za posmatrani primjer prepoznavanja napisanih brojeva, za ulazni sloj ima sliku od 28×28 piksela. Slika je sivoskalirana (ima samo jednu komponentu boje – sivu), da bi primjer bio što jasniji. Izlaz produkuje 10 vrijednosti koje odgovaraju ciframa i predstavljaju vjerovatnoću da je predložena slika baš ta cifra. Neuralna mreža koja se posmatra sastavljena je od ulaznog sloja, dva skrivena sloja, i jednog izlaznog sloja.



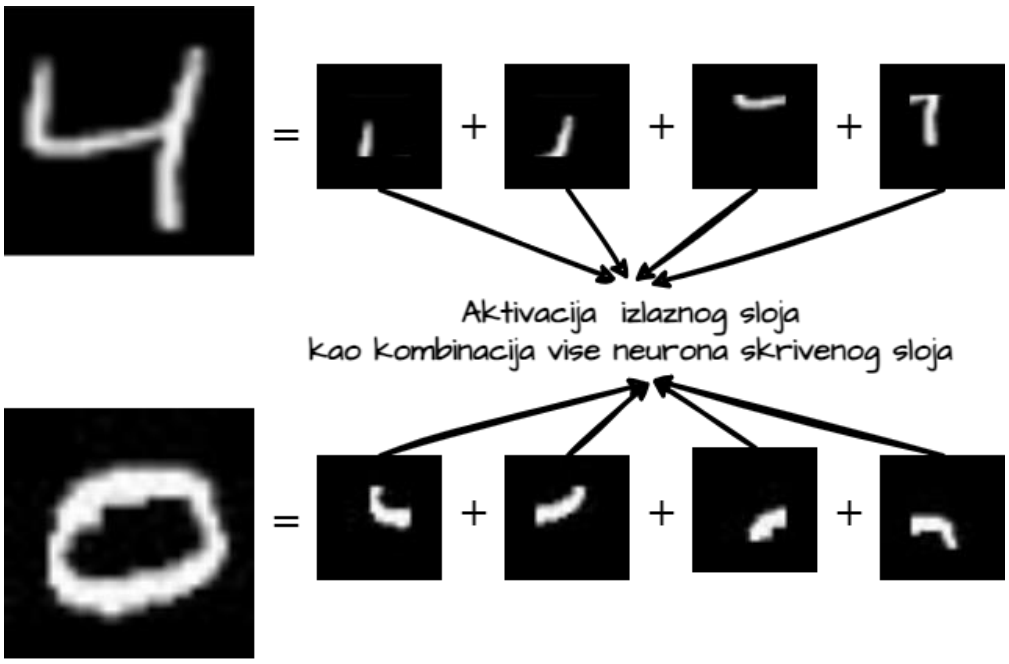
Slika 10: Sivoskalirana slika broja koja predstavlja broj 2 (lijevo)
i njegova reprezentacija matricom piksela (desno)

Na slici 10 su predstavljene vrijednosti pojednačnih piksela. Te vrijednosti će biti skalirane na interval $(0, 1)$ i predstavljaće prvi značajan podatak neuralne mreže – vrijednost aktivacije ulaznog sloja neurona.

Kao izlazna informacija javljaju se vjerovatnoće da pojedini broj predstavlja određenu cifru. Te vjerovatnoće su vrijednosti aktivacije izlaznog sloja neurona. Od izlaznih aktivacionih vrijednosti uzima se najveća vrijednost i algoritam na osnovu vjerovatnoće zaključuje koji broj je proslijeđen ulaznom sloju. Drugim riječima, za tri vrijednosti aktivacije neurona izlaznog sloja: $a_8 = 0.69$ (vjerovatnoća da je na novoj slici broj 8 je 69%), $a_6 = 0.20$ i $a_3 = 0.11$, neuralna mreža će zaključiti da je ulaznom sloju proslijeđen broj 8.

Skriveni slojevi sadrže glavne komponente procesa učenja u neuralnoj mreži. U prikazanom primjeru, su izabrana dva skrivena sloja (prvi sa 16 neurona drugi sa 14). Vrše se

transformacije iz 784 (dimenzije slike su 28×28) ulaznih neurona u 16 neurona prvog skrivenog sloja, pa iz 16 neurona prvog skrivenog sloja u 14 neurona drugog skirvenog sloja i na kraju 14 neurona iz drugog skrivenog u 10 izlaznih neurona. Po ugledu na biološko rješenje, neka pojedini neuroni skrivenog sloja predstavljaju određene oblike. Ako ulazna aktivacija predstavlja vrijednost piksela, prvi skriveni sloj će predstavljati određenu asocijaciju piksela u neki osnovni oblik, a drugi će možda biti neki kompleksni oblik. Ideja je da će vrijednost aktivacije pojedinačnih neurona, koji bi trebalo da predstavljaju određene oblike, biti približno jednaka jedinici ako taj oblik na slici postoji, a biće jednaka približno nuli ako taj oblik ne postoji na slici.



Slika 11: Simulacija formiranja brojeva 4 i 0 kroz poslednji skriveni sloj

Specijalizovana neuralna mreža CNN(engl. *convolutional neural network* – konvoluciona neuralne mreža) čija arhitektura funkcioniše baš na gore prikazan način, gdje konvolucioni slojevi odgovaraju prisustvu određenih oblika. Slika 11 ilustruje prethodno opisane koncepte, koji se odnose na konvolucione neuralne mreže (CNN).

Neuroni su usko povezani i vrijednost aktivacije sa ulaznog sloja utiče na sve vrijednosti aktivacije neurona u slijedećim slojevima. Svi neuroni između slojeva su povezani, odnosno mreža pripada potpuno povezanim neuralnim mrežama. Neće svaka aktivacija neurona da ima isti značaj, odnosno, uvode se težinski koeficijenti koji predstavljaju vrijednost koliko je pojedini neuron aktivan u kreiranju neurona novog sloja. Vrijednost težinskih faktora može biti pozitivna

i negativna a sugerisaće kako pojedini neuron učestvuje u stvaranju neurona novog sloja. Pored ovog keoficijenta dodaje se bias (sistematska greška). Ovaj broj omogućava pomjeranje po X osi i pomaže da pored linearnih podataka efikasno modeluje i nelinearne veze. Uz pomoć biasa, moguće je bolje prilagođavanje podacima. Za vrijednost aktivacije jednog neurona slijedećeg sloja važi:

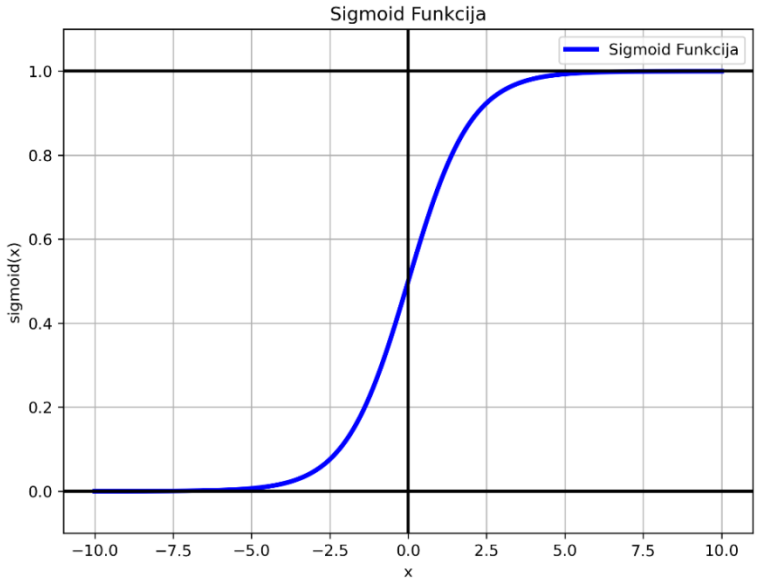
$$S_2 = w_1 a_1{}^{(0)} + w_2 a_2{}^{(0)} + w_3 a_3{}^{(0)} + \cdots + w_n a_n{}^{(0)} + b\,,$$

gdje $a_1{}^{(0)}, a_2{}^{(0)}, \ldots a_n{}^{(0)}$ predstavljaju aktivaciju neurona koji odgovaraju ulaznom sloju, $w_1, w_2, \ldots, w_n$ predstavljaju težinska stanja svakog od neurona, dok b predstavlja bias.

Suma S_2 može da ima veliki raspon vrijednosti. Da bi vrijednosti bile pogodne za klasifikaciju kao i za probabilističku interpretaciju, zgodno je pretvoriti vrijednost sume u interval od 0 do 1. Vrši se regulacija vrijednosti sume na željeni interval sigmoid funkcijom:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Koncept će biti ilustrovan u nastavku. Sigmoid funkcija će za veoma velike vrijednosti promjenljive x poprimati vrijednost broja 1, a za veoma male vrijednosti broja x će poprimati vrijednost 0.



Slika 12: Sigmoid funkcija ili logistička kriva

Na kraju, formula kojom se dobija vrijednost aktivacije jednog neurona prvog skrivenog

sloja poprima slijedeći oblik:

$$a_0^{(1)} = \sigma\left(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + w_{0,2}a_2^{(0)} + \cdots + w_{0,n}a_n^{(0)} + b_o\right),$$

gdje vrijednost aktivacije $a_0^{(1)}$predstavlja vrijednost aktivacije jednog (slučajnog) neurona prvog skrivenog sloja, $w_{0,0}a_0^{(0)}$predstavlja uređeni par težinskih koeficijenata (faktora) i vrijednosti aktivacije ulaznog sloja (u prikazanom primjeru neuralne mreže napisanih brojeva postoje 784 ulazne vrijednosti za jedan neuron prvog sloja $a_0^{(0)} \ldots a_{784}^{(0)}$ i 784 vrijednosti $w_{0,0} \ldots w_{0,784}$ težinska faktora), a b_o predstavlja bias za jedan neuron. Pošto su u pitanju uređeni parovi, za jedan neuron prvog skrivenog sloja postoji 784 uređena para za primjer napisanih brojeva.

Prethodna formula predstavlja vrijednost aktivacije za jedan neuron. Moguće je matrično predstaviti vrijednost aktivacije svih neurona jednog sloja. Prikazana je ispod:

$$\boldsymbol{a}^{(1)} = \sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix}\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_o \\ b_1 \\ \vdots \\ b_k \end{bmatrix}\right),$$

ili:

$$\boldsymbol{a}^{(1)} = \sigma(\boldsymbol{W}\boldsymbol{a}^{(0)} + \boldsymbol{b}),$$

gdje $\boldsymbol{W}$ predstavlja težinske faktore za prvi skriveni sloj (matrica od $k = 16$ vrsta i $n = 784$ kolone za posmatrani primjer napisanih brojeva), $\boldsymbol{a}^{(0)}$(matrica od $n = 784$ vrste i jednu kolonu za prikazani primjer) predstavlja vrijednost aktivacije ulaznog sloja, b (matrica biasa od od $k = 16$ vrsta i jedne kolone), $\boldsymbol{a}^{(1)}$ matrica vrijednosti aktivacije prvog skrivenog sloja (dimenzije ove matrice su 16 vrsta (prvi skriveni sloj sadrži 16 neurona) i jedna kolona) a $\sigma(.)$ predstavlja aktivacionu funkciju.

Ovaj matrični oblik ima veoma veliku primjenu i značaj. Razvijene su efikasne biblioteke koje pojednostavljuju rad sa matričnim podacima. U programskom jeziku Python, jedna takva biblioteka je Numpy, koja se često koristi sa drugim bibliotekama kao što su Pytorch, Skit learn i TensorFlow. Biblioteke kao što su TensorFlow i Pytorch rade sa tenzorima. Većina izvedenih modela neuralnih mreža koriste ove specijalne nizove za mnogo kompleksnije mreže koje sadrže veliki broj skrivenih slojeva.

U neuralnoj mreži jedine poznate vrijednosti su: vrijednost ulazne aktivacije i željena vrijednost izlazne aktivacije. Sve ostale vrijednosti (težinske koeficijente i bias vrijednosti)

mreža mora da na neki način nauči. Za prije posmatranu neuralnu mrežu prepoznavanja pisanih brojeva koja se sastoji od jednog ulaznog, dva skrivena sloja i jednog izlaznog sloja potrebno je:

1. Naučiti težinske vrijednosti:

$$784 \times 16(\ za\ računanje\ aktivacije\ prvog\ skrivenog\ sloja)$$
$$+16 \times 14(za\ računanje\ aktivacije\ drugog\ skrivenog\ sloja)$$
$$+14 \times 10\ (za\ računanje\ aktivacije\ izlaznog\ sloja) = 12908,$$

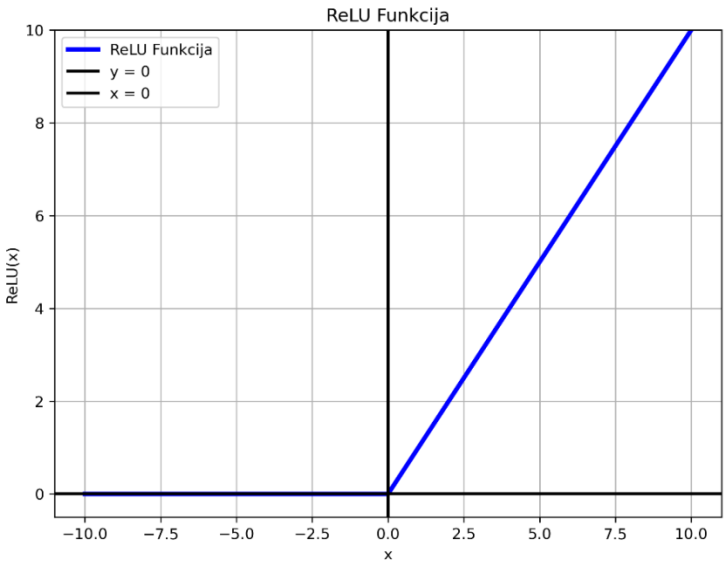
2. Naučiti bias vrijednosti:

$$16(\ za\ računanje\ aktivacije\ prvog\ skrivenog\ sloja)$$
$$+14(za\ računanje\ aktivacije\ drugog\ skrivenog\ sloja)$$
$$+10(za\ računanje\ aktivacije\ izlaznog\ sloja) = 40,$$

što dovodi do konačne cifre od $12908 + 40 = 12948$ vrijednosti koje opisuju samo jedan trening podatak.

Na trenutak treba pomenuti ReLu funkciju koja se često koristi za obradu slike. U određenim uslovima ReLu funkcija ima bolje rezultate nego sigmoid funkcija i ima primjenu u nekim modelima koji će kasnije biti detaljno objašnjeni.

Problem sigmoid funkcije jesu vrijednosti u blizini minimuma i maksimuma. U ovim slučajevima u okolini minimuma i maksimuma funkcija će imati gotovo konstantnu vrijednost bez promjene što onemogućava efektivno korišćenje težinskih faktora. Ovaj problem se rješava ReLu funkcijom, koja za vrijednosti ulaznog argumenta manje od 0 poprima vrijednost 0, a za vrijednosti ulaznog argumenta veće ili jednake nuli zadržava vrijednost ulaznog argumenta.



Slika 13: ReLU funkcija

Ova funkcija ima veći značaj za više skrivenih slojeva i mnogo je brža za implementaciju. Takođe, lakše se izračunava od sigmoidne funkcije. Odbacivanjem negativnih vrijednosti ulaznog argumenta ReLu funkcijom vrši se aktivacija samo pozitivnih djelova izlaza. Ona ima poseban značaj u procesu dubokog učenja. Problem minimizacije (minimizacija obavlja gradijentnim spustom) lakše se prevazilazi korišćenjem ReLU funkcije. Pojava da se usporava promjena sigmoidne funcije kod minimuma i maksimuma kroz više slojeva u neuralnoj mreži može dovesti do „ispada„ ili zaglavljivanja gradijenta funkcije u nekom izračunavanju u višim skrivenim slojevima. Pošto ReLU funkcija nema ovaj problem, ona se češće koristi kod dubokog učenja. Pod dubokim učenjem često se mogu smatrati arhitkture neuralne mreže sa mnogo skrivenih slojeva, sa specijalno definisanom arhitekturom.

3.1.2. Učenje u neuralnim mrežama

U ovoj sekciji će biti razmatran algoritam obučavanja neuralnih mreža. Neuralnoj mreži će biti proslijeđen veliki broj slika koje su označene. Inicijalno, sve vrijednosti biasa i težinskih faktora će biti postavljene na proizvoljne vrijednosti. U praksi je moguća inicijalizacija u blizini vrijednosti broja 0, ili po nekim heurističkim pravilima koja model zahtjeva. Cilj je naravno poboljšati rezultate rada algoritma.

U neuralnoj mreži na osnovu postavljenih težinskih koeficijenata i biasa, se izračunava izlazna aktivacija neurona. Pored ove vrijednosti, postoji vrijednost oznake koja reflektuje stvarno stanje (vrijednost 1 na izlazu koji označava broj koji se traži na ostalim izlazima nula). Ako postoje izlazne aktivacije: $a_8 = 0.39$, $a_6 = 0.30$, $a_0 = 0.21$, $a_5 = 0.10$, ostale vrijednosti izlaznih aktivacija su jednake nuli, a oznaka je broj 5, vidi se da neuralna mreža slučajno postavljenih koeficijenata nije dobro odradila svoj zadatak. Sada je potrebno ispraviti pojedine koeficijente i biase, da mreža ima adekvatnu izlaznu aktivaciju. Za potrebe treniranja i učenja težinskih faktora i biasa, je potrebna velika baza podataka za obučavanje. Za potrebe klasfikacije pisanih brojeva, može se koristi MNIST dataset. On u sebi sadrži 60 000 slika sa oznakama i pogodan je za treniranje neuralne mreže.

Ako vrijednost aktivacije izlaznih neurona bude predstavljena sa a_i, vrijednost oznake tj. stvarna vrijednost bude predstavljena sa y_i a broj izlaznih podataka sa N, može se definisati slijedeća formula za funkciju cijene C (engl. *Cost function*):

$$C = \frac{1}{N}\sum_{i=1}^{N}(a_i - y_i)^2.$$

Funkcija kojom se kontroliše efikasnost neuralne mreže predstavlja funkciju cijene i u ovom slučaju ona se definiše kao srednja kvadratna greška (engl. *Mean Square Error* –MSE). U opštem slučaju, ona predstavlja razliku između očekivane vrijednosti i vrijednosti koja je dobijena kao izlazna vjerovatnoća predstavljena realnim brojem. Ta izlazna vrijednost predstavlja vjerovatnoću da podaci koji su proslijeđeni ulaznom sloju pripadaju određenom skupu podataka. U slučaju neuralne mreže za prepoznavanje pisanih cifara, izlazna vjerovatnoća predstavlja vjerovatnoću da sivoskalirana ulazna slika broja (koja je proslijeđenja ulaznom sloju) predstavlja jednu od traženih 10 cifara (0-9). Za prije predstavljene izlazne aktivacije ($a_8 = 0.39$, $a_6 = 0.30$, $a_0 = 0.21$, $a_5 = 0.10$) C_1 funkcija ima vrijednost:

$$C_1 = \frac{1}{10}((a_0 - y_0)^2 + (a_1 - y_1)^2 + \cdots + (a_8 - y_8)^2 + (a_9 - y_9)^2)$$
$$C_1 = \frac{1}{10}((a_0 - y_0)^2 + (a_5 - y_5)^2 + (a_6 - y_6)^2 + (a_8 - y_8)^2)$$
$$C_1 = \frac{1}{10}((0.21 - 0)^2 + (0.10 - 1)^2 + (0.30 - 0)^2 + (0.39 - 0)^2)$$
$$C_1 = \frac{1}{10}(0.0441 + 0.81 + 0.09 + 0.1521) = \frac{1.0962}{10} = 0.10962$$

Neka se razmotri slučaj za dobro istrenirani model. Ako su vrijednosti izlaznih aktivacija $a_5 = 0.97$ i a_6=0.03, a ostale vrijednosti izlaznih aktivacija jednake nuli, vrijednost C_2 funkcije je:

$$C_2 = \frac{1}{10}((0.97 - 1)^2 + (0.03 - 1)^2) = \frac{0.0018}{10} = 0.00018.$$

Vrijednost C_1 *i* C_2 su različite. Pošto je prva funkcija C_1 bazirana na netreniranim vrijednostima, a druga C_2 bazirana na vrijednostima koje su proizvod treniranja, može se izvesti zaključak da je cilj treninga smanjenje vrijednosti C funkcije, odnosno traženje njene minimalne vrijednosti.

Pored MSE funkcije, postoje i alternativne definicije *C* funkcije. Neke od njih su:

1. Među-entropija (engl. *Cross Entropy*) je funkcija koja se najčešće koristi za klasifikacione probleme. Slična je MSE funkciji, u smislu da se koriste oznake i vjerovatnoće za izlaznu vrijednost. Takođe se koristi za klasifikacione probleme s tim da za loše predikcije model se „kažnjava“. Data je formulom:

$$C = -\sum_{i=1}^{N} y_i \log(a_i),$$

gdje je vrijednost aktivacije izlaznih neurona predstavljena sa a_i, vrijednost oznake tj. stvarna vrijednost predstavljena sa y_i a broj izlaznih podataka sa N.

2. Binarna među-entropija (engl. *Binary Cross Entropy*) je funkcija koja pomaže u binarnoj klasifikaciji. Ima čestu primjenu u dubokom učenju. Binarna klasifikacija ima dvije izlazne klase 0 ili 1 (tačno ili netačno) te u formuli *C* funkcije važi da je:

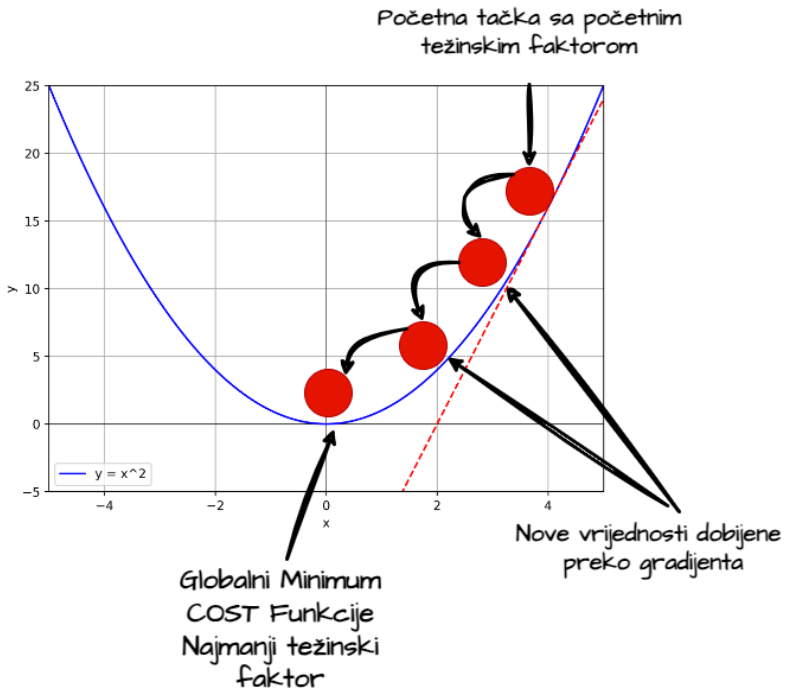
$$C = -\sum_{i=1}^{N} y_i \log(a_i) - (1 - y_i)\log(1 - a_i),$$

gdje je vrijednost aktivacije izlaznih neurona predstavljena sa a_i, vrijednost oznake tj. stvarna vrijednost predstavljena sa y_i (ovdje ima samo vrijednosti 0 i 1) a broj izlaznih podataka sa N. Ova formula je konstruisana tako da kažnjava model kada vrijednost oznake ne odgovara predviđenoj. Za vrijednost $y = 1$ i $y = 0$ aktiviraće se prvi, ili drugi dio formule, respektivno.

3.1.3. Gradijentni spust

Postoji mnogo načina da se nađe minimum određene funkcije. Pitanjem efektivnog pronalaska minimuma se bavio, među prvim naučnicima, Njutn gdje se preko prvog i drugog izvoda tražio minimum. Uslov za primjenu ovakvog pristupa je diferencijabilnost posmatrane funkcije.

Ipak zbog problema diferencijabilnosti funkcija, težine pronalaska drugog izvoda funkcije u određenim tačkama, zatim velikog broja promjenljivih (u primjeru neuralne mreže pisanih brojeva sa četiri sloja imali smo 12948 promjenljive) ustalio se gradientni spust koji je zasnovan na gradijentu funkcije, odnosno, vektoru sastavljenom od parcijalnih izvoda funkcije po svim promjenljivim.



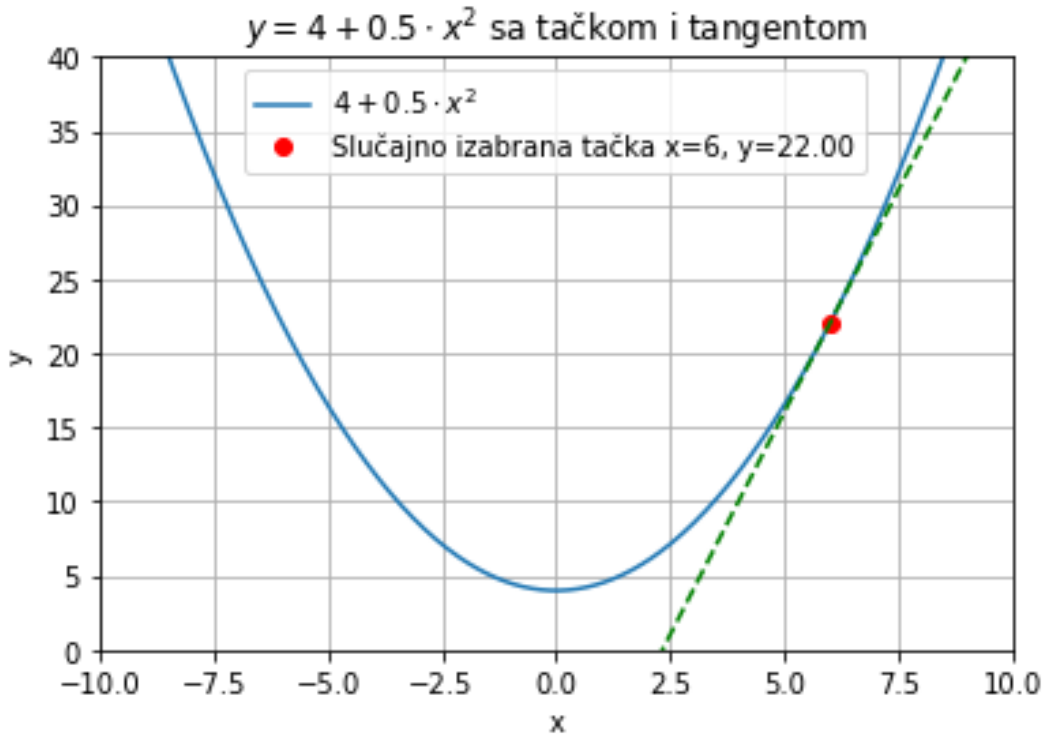
Slika 14: Simulacija dobijanja minimuma putem gradijenta funkcije

Na slici 14 je prikazan princip traženja minimuma funkcije spuštanjem u pravcu negativog gradijenta. Ideja je da se sa neke početne tačke $\boldsymbol{x}_0$ lopta (na slici 14) premjesti u neku novu tačku $\boldsymbol{x}_1$ koja će biti bliža minimumu. Kasnije, u sljedećem koraku, lopta će se ponovo približiti minimumu i premjestiti u neku drugu tačku $\boldsymbol{x}_2$. Ova ažuriranja pozicija se vrše preko gradijenta. Gradijent kao promjena se svodi na pronalazak prvog izvoda funkcije po različitim promjenljivim od kojih ta funkcija zavisi. Za potrebe neuralnih mreža je potrebno računati prve parcijalne izvode za veliki broj promjenljivih. Preko gradijenta je moguće vršiti minimizaciju funkcije. U dvije uzastopne iteracije imamo ažuriranje oblika:

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \alpha \nabla y(\boldsymbol{x}_0),$$

gdje $\nabla y(\boldsymbol{x}_0)$ predstavlja gradijent (vrijednost prvog parcijalnog izvoda po svim promjenljivim) u tački $\boldsymbol{x}_0$, α je korak učenja a $\boldsymbol{x}_1, \boldsymbol{x}_0$ su tačke koje se posmatraju pri čemu za $\boldsymbol{x}_1$ funkcija *y* ima manju vrijednost nego za tačku $\boldsymbol{x}_0$.

Gradijent u tački $\boldsymbol{x}_0$ predstavlja pravac najbržeg rasta funkcije u toj tački. Negativni gradijent predstavlja pravac najbržeg pada funkcije što može poslužiti za traženje minimuma. Zarad boljeg objašnjenja posmatra se primjer dat na slici 15.



Slika 15: Tangenta funkcije u tački $\boldsymbol{x_0} = 6$.

Koeficijent pravca tangente je prvi izvod funkcije u toj tački

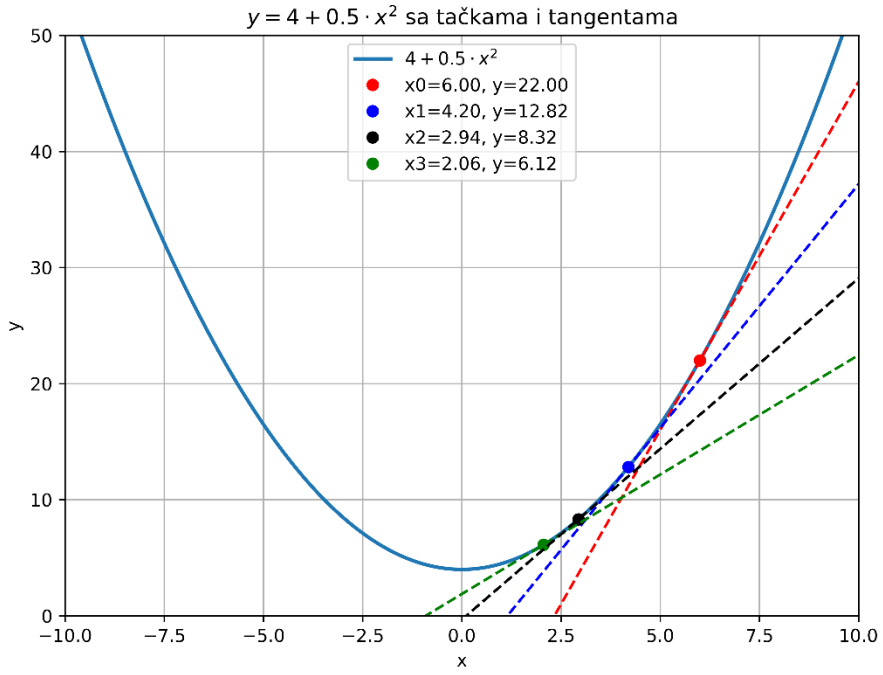
Na slici 15 je prikazana konveksna diferencijabilna funkcija. Nagib tangente u početnoj tački $\boldsymbol{x_0} = 6$ je jednak prvom izvodu funkcije y u toj tački. Funkcija će da raste ako je i vrijednost prvog izvoda u toj tački pozitivna. Prvi izvod mjeri brzinu promjene funkcije $y = 4 + 0.5x^2$ u odnosu na tačku $\boldsymbol{x_0}$.

Zbog čega je značajna ova informacija? Ako se uzme proizvoljna tačka (ovdje u primjeru je slučajno izabrana vrijednost $\boldsymbol{x_0} = 6$), moguće je na osnovu gradijenta (prvog parcijalnog izvoda) utvrditi kako se funkcija y smanjuje ili povećava. Drugim riječima, ako se uzme pozitivan pravac gradijenta vrijednost funkcije y će se povećavati, a za negativan pravac gradijenta vrijednost funkcije y će se smanjivati. Tj. ako se od početne tačke oduzme vrijednost gradijenta pomnožena sa veoma malim brojem, moguće je za konveksnu diferencijalnu funkciju tvrditi: vrijednost funkcije y u novoj tački $\boldsymbol{x_1}$ će imati manju vrijednost nego u početnoj tački $\boldsymbol{x_0}$. Važi da je:

$$\boldsymbol{x_1} = \boldsymbol{x_0} - \alpha(\mathrm{y}\,(\boldsymbol{x_0}))',$$

gdje je $\boldsymbol{x_0}$ je početna tačka, $\boldsymbol{x_1}$ je nova tačka za koju funkcija y ima manju vrijednost, α

(korak učenja) predstavlja malu pozitivnu konstantu koja definiše brzinu konvergencije gradijentnog algoritma, a $(y(\boldsymbol{x}_0))'$ predstavlja vrijednost prvog izvoda funkcije y u tački $\boldsymbol{x}_0$ odnosno gradijent u tački $\boldsymbol{x_0}$. U prikazanom primjeru postavljen je korak učenja $\alpha = 0.3$ i tačka ima samo jednu dimenziju pa se traži prvi izvod funkcije.



Slika 16: Postupak pronalaženja minimuma preko prvog izvoda

Osobine gradijenta će važiti i za funkcije cijene sa velikim brojem varijabli na isti način kao što važe za funkcije sa jednom promjenljivom. Sada će se tražiti parcijalni izvodi.

Posmatraće se funkcija:

$$y = x_1^2 + 3x_2^2.$$

Funkcija y je definisana preko dvije promjenljive x_1 i x_2. Postupkom gradijentnog spusta će se tražiti minimum funkcije na određenom intervalu. Slučajno je izabrana početna tača $\boldsymbol{x}_0(x_1, x_2) = (5, 5)$. Vrijednost funkcije y u toj tački je $y = 100$. Proizvoljno je izbran mali konstantan korak učenja $\alpha = 0.01$ i ažuriranje vrijednosti tačke $\boldsymbol{x}_0$ će se vršiti po formuli:

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \alpha \nabla y(\boldsymbol{x}_0).$$

Pošto je funkcija y funkcija dvije promjenljive traži se njen parcijalani izvod po x_1 i x_2. Za funkciju y prvi parcijalni izvodi po x_1 i x_2 su:

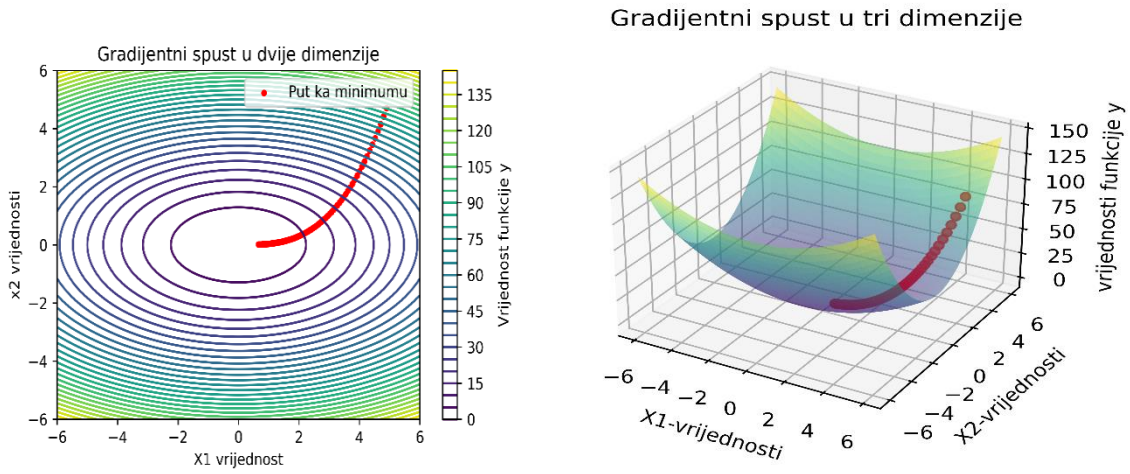
$$y'_{(x_1)} = \frac{\partial y(x_1, x_2)}{\partial x_1} = \frac{\partial}{\partial x_1}[x_1^2 + 3x_2^2] = \frac{\partial(x_1^2)}{\partial x_1} + \frac{\partial(3x_2^2)}{\partial x_1} = 2x,$$

$$y'_{(x_2)} = \frac{\partial y(x_1, x_2)}{\partial x_2} = \frac{\partial}{\partial x_2}[x_1^2 + 3x_2^2] = \frac{\partial(x_1^2)}{\partial x_2} + \frac{\partial(3x_2^2)}{\partial x_2} = 6x.$$

Tačka $\boldsymbol{x_1}$u kojoj *y* ima manju vrijednost je:

$$\boldsymbol{x_1} = \boldsymbol{x_0} - \alpha \nabla y(\boldsymbol{x_0}) = (5,5) - 0.01\,(2 \cdot 5, 6 \cdot 5) = (5,5) - (0.1, 0{,}3) = (4.9, 4.7).$$

Vrijednost funkcije *y* u tački $\boldsymbol{x_1}$ je $y = 90.28$ i manja je od početne vrijednosti za tačku $\boldsymbol{x_0}$. Postupak će se ponavljati u više iteracija da bi se našao minimum funkcije *y*. Na slikama 17 i 18 je grafički prikaz traženja minimuma funkcije *y* za više iteracija. Predstavljeni princip važi za mnogo više promjenljivih.



Slike 17 i 18: Prikaz gradijentnog spusta funkcije *y* u dvije dimenzije (lijevo) u tri dimenzije (desno)

Već je data formula za promjenu aktivacije neurona u neuralnoj mreži. Ona ima sljedeći oblik:

$$\boldsymbol{a}^{(1)} = \sigma\,(\boldsymbol{W}\boldsymbol{a}^{(0)} + \boldsymbol{b}),$$

gdje $\boldsymbol{a}^{(1)}$ i $\boldsymbol{a}^{(0)}$ predstavljaju matrice aktivacije neurona u prvom skrivenom i ulaznom sloju respektivno, $\boldsymbol{b}$ predstavlja matricu bias vrijednosti za prvi skriveni sloj, $\boldsymbol{W}$ predstavlja matricu težinskih faktora (stanja) prvog skrivenog sloja, a σ(.) predstavlja aktivacionu funkciju.

Ova formula je samo za aktivaciju jednog neurona jednog sloja. U prikazanom primjeru neuralne mreže za prepoznavanje ručno pisanih brojeva korišćeno je 784 neurona sloja ulaznih podataka (dimenzije sivoskalirane slike su 26×26), dva skrivena sloja (16 neurona i 14 neurona) i 10 neurona izlaznog sloja. Posmatrana mreža je sastavljena od četiri sloja:

1. Ulaznog sloja čija će vrijednost aktivacije biti obilježena sa matricom $\boldsymbol{a}^{(0)}$.

2. Prvog skrivenog sloja čija će vrijednost aktivacije biti obilježena sa matricom $\boldsymbol{a}^{(1)}$.
3. Drugog skrivenog sloja čija će vrijednost aktivacije biti obilježena sa matricom $\boldsymbol{a}^{(2)}$.
4. Izlaznog sloja čija će vrijednost aktivacije biti obilježena sa matricom $\boldsymbol{a}^{(3)}$ a vrijednost aktivacije jednog slučajno izabranog neurona biti prikaza sa ${a_0}^{(3)}$.

Za slučajno izabranu vrijednost aktivacije četvrtog izlaznog sloja ${a_0}^{(3)}$ važi:

$$ {a_0}^{(3)} = \sigma\left(\boldsymbol{W}_3\boldsymbol{a}^{(2)} + b_3\right) = \sigma\left(\boldsymbol{W}_3\sigma\left(\boldsymbol{W}_2\boldsymbol{a}^{(1)} + \boldsymbol{b}_2\right) + b_3\right) = $$
$$ \sigma\left(\boldsymbol{W}_3\sigma\left(\boldsymbol{W}_2\sigma\left(\boldsymbol{W}_1 a^{(0)} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + b_3\right), $$

gdje $\boldsymbol{a}^{(0)}, \boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}$ i $\boldsymbol{a}^{(3)}$ su matrice aktivacije pojedinih slojeva, $\boldsymbol{W}_3, \boldsymbol{W}_2, \boldsymbol{W}_1$ su matrice težinskih faktora, $\boldsymbol{b}_1$ i $\boldsymbol{b}_2$ su matrice biasa, b_3 je jedna vrijednost biasa koja odgovara slučajno izabranom neuronu, a $\sigma(\,.\,)$ je aktivaciona funkcija.

Ovo je prikaz za izlaznu vrijednost samo jednog neurona neuralne mreže koja vrši detekciju napisanih brojeva. Uzimajuću u obzir računanje C funkcije preko srednje kvadratne greške, formula na kraju ima sledeći oblik:

$$ C = \frac{1}{10}\Big(\left({a_0}^{(3)} - y_0\right)^2 + \left({a_1}^{(3)} - y_1\right)^2 + \left({a_2}^{(3)} - y_2\right)^2 + \left({a_3}^{(3)} - y_3\right)^2 + \left({a_4}^{(3)} - y_4\right)^2 $$
$$ + \left({a_5}^{(3)} - y_5\right)^2 + \left({a_6}^{(3)} - y_6\right)^2 + \left({a_7}^{(3)} - y_7\right)^2 + +\left({a_9}^{(3)} - y_9\right)^2\Big), $$

gdje funkcija cijene C računa vrijednost aktivacije četvrtog izlaznog sloja za jednu sliku koja je proslijeđena kao ulazni sloj, ${a_0}^{(3)} \ldots {a_9}^{(3)}$ predstavljaju izlazne vrijednosti aktivacije pojedinačnih neurona a $y_0 \ldots y_9$ predstavlja stvarnu vrijednost. Za ažuriranje parametara u neuralnoj mreži će se koristiti drugačije oznake:

$$ \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla \mathrm{C}(\boldsymbol{\theta}_t), $$

gdje $\boldsymbol{\theta}_{t+1}$ predstavlja vrijednost parmetara u sledećoj iteraciji, $\boldsymbol{\theta}_t$ predstavlja vrijednost parametara u trenutnoj iteraciji, α je korak učenja a $\nabla \mathrm{C}(\boldsymbol{\theta}_t)$ predstavlja vrijednost gradijenta.

Sada se od funkcije C traži minimalna vrijednost gradijentnim spustom. Postupak je veoma kompleksan i zahtjeva računare visokih performansi da bi se brzo došlo do rješenja. Ipak princip pronalska minimuma preko gradijenta se vrši na isti način kao što se vršila za funkciju sa jednom ili dvije promjenljive.

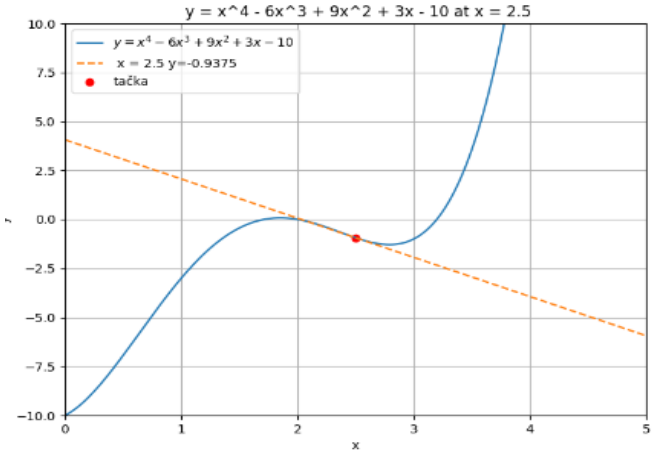
Važno je naglasiti da opisani proces ažuriranja odgovara samo jednom trening podatku i ne odvija se na isti način kao što je prikazano u linearnom modelu. Za više trening podataka, odnosno za potrebe predstavljene neuralne mreže za više slika sa oznakama, vrši se izračunavanje gradijenta za svaki trening podatatak. Na početku biće izračunata 12948

parcijalna izvoda C funkcije za jedan trening podatak. Postupak se ponavlja i gradijent se izračuna za ostale trening podatake. Vrijednosti svih gradijenanta se sabiraju i dijele sa brojem trening podataka. Usrednjena vrijednost se oduzima od početnih vrijednosti težina i biasa, nakon čega se vrši ažuriranje. Ovaj proces koji obuhvata jedno računanje gradijenta za svaki trening podatak se naziva epohom. U sledećoj epohi se računa vrijednost gradijenta sa ažuriranim podacima (težinom i biasom). Ponovo se vrši računanje za više epoha. MINST dataset sadrži 60 000 slika za treniranje. Odnosno normalnim gradijentnim spustom epoha predstavlja $12948 \times 60\,000 = 776\,880\,000$ računanja pracijalnih izvoda. Ovaj princip gradijentnog spusta se naziva gradijentni spust sa kompletnim skupom podataka (engl. *full batch gradient descent*). Zbog kompleksnosti i broja računica je razvijeno više različitih gradijentnih algoritama koje se ne bave kompletnim skupom podataka.

3.1.4. Korak učenja

Korak učenja (engl. *learning rate*) je važan parametar gradijetnog algoritma koji utiče na brzinu konvergencije algoritma. Postoji više pristupa određivanju koraka učenja.

Posmatra se strogo konveksna funkcija bez lokalnih minimuma. U slučaju da postoji više lokalnih minimuma, potrebno je da se pronađe globalni minimum funkcije. Ovdje se javlja jedan od prvih problema pri izboru koraka učenja: ako izaberemo veoma malu stopu učenja, pored sporije konvergencije ka minimumu, moguće je da se algoritam zaustavi na prevojnim tačkama. Posmatra se slika 19.



Slika 19: Inlustracija funkcije sa prevojnim tačkama

Kada bi se za vrijednost koraka učenja uzela veoma mala proizvoljna vrijednost,

algoritam bi zakočio u lokalnom minimumu. Za veliki korak učenja algoritam bi možda preskočio globalni minimum. U prikazanoj situaciji na slici 19 sa velikim korakom učenja algoritam je zaglavio u dvije tačke.

Ovi problemi se rješavaju različitim pristupima koraku učenja. Neki od tih pristupa su:

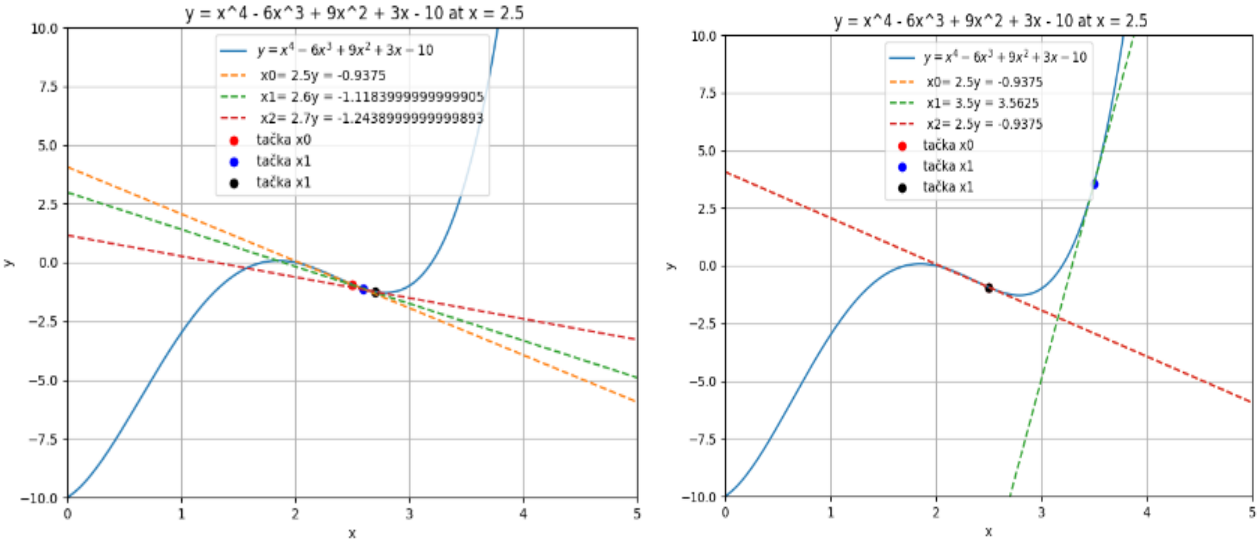
1). Fiksni korak je metod gdje se korak učenja ne mijenja tokom konvergencije. Koristan je ako su dobro poznate specifičnosti sistema koji treba minimizirati.

2). Smanjenje koraka učenja je metod po kojem se korak učenja postepeno smanjuje tokom iteracija. Ovim postupkom bi bilo izbjegnuto kočenje u dvije tačke na slici 19 ali bi algoritam najverovatnije kovergirao ka lokalnom minimumu a ne ka globalnom minimumu.

3). Adaptivne metode su metode modernih algoritama gradijentnog spusta gdje se korak učenja mijenja zavisno od prošlih gradijenata tokom iteracija. Time je moguća brža kovergencija (pronalazak minimuma) funkcije cijene.

4). Rano zasustavljanje je metoda koja zaustavlja treniranje kada se postigne neki željeni stepen greške, odnosno u predstavljenom primjeru kada se *C* funkcija prestane smanjivati. Veoma je pogodna za sprečavanje procesa preprilagođavanja.

U narednoj sekciji će biti kratko predstavljeni osnovi tipovi gradijentnih algoritama i Adam (gradijentni algoritam korišćen za minimizaciju funkcije cijene više modela mašinskog učenja digitalne kompresije slika koji će biti predstavljeni).



Slike 20 i 21: Minimizacija funkcije sa malim fiksnim korakom učenja (slika 20) i velikim fiksnim korakom učenja (slika 21)

Na slikama 20 i 21 su prikazani mali i veliki fiksni koraci učenja pri traženju minimuma funkcije y na prikazanom intervalu. U oba slučaja nije pronađen minimum funkcije. Na slici 20 je pronađen lokalni minimum, a nije globalni. Na slici 21 algoritam je velikim konstantnim korakom učenja zaglavljen u dvije tačke i nije pronađen ni lokalni minimum.

3.1.5. Varijante gradijentnog spusta i Adam algoritam

U dosadašnjem izlaganju razmatran je primjer neuralne mreže za prepoznavanje pisanih brojeva jednostavne strukture. Ta mreža se samo sastoji od 4 sloja (ulazni, izlazni i dva skrivena sloja). U stvarnosti broj slojeva u neuralnoj mreži može biti veći. Neuralna mreža najčešće se sastoji od 3 sloja (ulaznog, izlaznog i jednog skrivenog). U problemima dubokog učenja broj slojeva može biti veći, čime se znatno povećava broj promjenljivih i vrijeme za koje je potrebno naći težinske koeficijente i biase. GTP-3 neuralna mreža, kojom se generiše tekst, u svojoj arhitekturi ima 175 milijardi parametara. Neki parametri ove mreže su fiksirani, a neki trenirani. U postupku treniranja ili minimizacije C funkcije, osnovni princip gradijentog spusta će zahtjevati ogromno vrijeme i imaće slabu efikasnost. Zbog toga, su naučnici odlučili da razviju više modifikacija klasičnog gradijentnog algoritma. Neke od modifikacija su:

1. Stohastički gradijentni spust
2. Minibatch gradijentni spust
3. Momentum gradijentni spust
4. Adagard
5. Rms prop
6. ADAM

1). Stohastički gradijenti spust (SGD – engl. *Stochastic Gradient Descent*). Ideja ove vrste gradijentnog algoritma je da se zamijeni računanje svih gradijenata sa trening seta računanjem samo jednog. Odnosno biće postavljene vrijednosti svih parametara za sve trening podatke na početku prve epohe. U drugom koraku biće slučajno izabran jedan trening podatak i za njega će biti izračunat gradijent. Svi podaci će biti ažurirani preko vrijednosti tog gradijenta. To znači da nova vrijednost za sve trening podatke će samo biti ažurirana preko gradijenta jednog slučajno izabranog trening podatka. Time se završava prva epoha. U narednim epohama će se vrši ponovno računanje gradijenta za nove slučajno izabrane trening podatke. Veoma je važno

naglastiti da u ovom slučaju rezutati funkcije cijene neće imati konstantan pad, već je moguća pojava oscilacija. Jedan slučajno izabran gradijent jednog trening podatka možda neće imati izračunatu vrijednost koja na adekvatan način reflektuje stanje neuralne mreže. Da bi se umanjio uticaj gradijenta loše izabranog slučajnog trening podatka korak učenja mora biti mnogo manji nego u osnovnom gradijentnom spustu. U praksi, korak učenja se može smanjivati kada se funkcija približava minimumu. Za velike dataset-ove SGD algoritam je mnogo brži od osnovnog gradijentnog spusta jer veliki dataset-ovi zahtjevaju veliki broj računanja parcijalnih izvoda.

2). Minibatch gradijentni spust ne koristi cijeli trening set već grupu podataka. Ovim postupkom se praktično preuzima brzina iz SGD algoritma a smanjuje oscilacija funkcije cijene *C* koja nastaje kao posledica loše izračunatog gradijenta. Umjesto računanja gradijenta jednog trening podatka računaće se gradijent na skupu slučajno izabranih trening podataka i usrednjiti. Dalji postupak ažuriranja je isti kao i u SGD algoritmu.

3). Momentum gradijent descent (MGD) predstavlja prvu modifikaciju algortima gradijentnog spusta. Minibatch i SGD su treniranje definisali preko slučajno odabranih trening setova a MGD algoritam će imati kratkoročnu memoriju prošlih gradijenata. Neželjeno zaustavljanje na prevojnim tačkama, kao posledica usporenja gradijentnog spusta se na ovaj način izbjegava. Ako se funkcija naglo nađe u lokalnom minimumu, gradijent funkcije može da zaustavi ažuriranje podataka, jer je približno jednak nuli. Kada se uvede moment u algoritam, odnosno ako vrijednosti prethodnog gradijenta imaju uticaj na buduće vrijednosti preko akumuliranog momenta, može se preskočiti stacionarna tačka, ili lokalni minimum. Problem traženja globalnog minimuma neke funkcije se može ilustrovati preko spusta niz planinu. Osnovni gradijentni spust, kada traži podnožje planine, posmatra tačke u kojima dolazi do najvećeg pada visine i pomjera se ka njima. Ovaj postupak odgovara prvom izvodu, odnosno, brzini promjene funkcije po promjenljivima. U trenutku kada dodje na visoravan (koja nije traženi minimum, već stacionarna tačka) gradijentni algoritam bi zaključio da je ta visoravan minimum jer promjena je veoma mala odnosno, gradijent je približno jednak nuli. MGD posmatra isti spust na drugačiji način. Neka se lopta spušta niz istu planinu. U ovom slučaju početni moment lopte će biti nula i neće imati uticaj na prvo računanje gradijenta. Lopta nastavlja da akumulira brzinu na osnovu gradijenta i dolazi u poziciju iste visoravni. Akumulirana brzina pomaže loptici da iskoči iz rupe i traži globalni minimum. Moment se definiše formulom:

$$\boldsymbol{v}_t = \beta \boldsymbol{v}_{t-1} + (1-\beta)\,\nabla C(\boldsymbol{\theta}_t),$$

gdje t predstavlja iteraciju, $\boldsymbol{v}_t, \boldsymbol{v}_{t-1}$ predstavljaju vrijednosti momenta u novoj i prošloj iteraciji, β je koeficijent koji odlučuje koliko će uticati vrijednost prošlih momenata na novi, a $\nabla C(\boldsymbol{\theta}_t)$ predstavlja gradijent funkcije za iteraciju *t*. Vrijednost parametara se ažurira po formuli:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{v}_t,$$

gdje $\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t$ predstavljaju vrijednosti promjenljive u novoj (*t)* i prošloj iteraciji (*t*-1*)*, α je korak učenja a $\boldsymbol{v}_t$ predstavlja moment.

U sljedećoj epohi se koriste informacije iz prošle epohe (vrijednost gradijenta) da bi se ažurirali težinski koeficijenti. Više važnosti se daje novim koeficijentima. Važno je naglasititi da u MGD algoritmu korak učenja generalno ostaje isti, odnosno, moment predstavlja vrijednost na koju utiče istorija gradijenata.Većinu principa gradijentnih algoritama je moguće koristiti sa drugim gradijentnim algoritmima. Postoji na primjer, varijanta SGD algoritma sa momentom.

4). Adagard četvrti algoritam gradijentog spusta koji ponovo uzima u obzir istoriju promjena vrijednosti gradijenata. Istorija promjena gradijentnih vrijednosti sada utiče samo na korak učenja. Parametri koji se brže mijenjaju osjetljivi su na promjene funkcije, pa korak učenja za njih ima malu vrijednost. Parametri čiji je gradijent približno jednak nuli mogu imati veći korak pri ažuriranju. Data je formula za sumu prošlih gradijenata:

$$s^2{}_{t,b} = \mathrm{s}^2{}_{t-1,b} + (\nabla C_b(\boldsymbol{\theta}_t))^2,$$

gdje $s^2{}_{t,b}$ predstavlja sumu svih trenutnih gradijenata sa indeksom *b*, $(\nabla C_b(\boldsymbol{\theta}_t))^2$ trenutnu vrijednost gradijenta za parametar *b* (za indeks promjenljive), $s^2{}_{t-1,b}$ predstavlja sumu svih prošlih gradijenata za indeks a *t* predstavlja iteraciju. Ažuriranje vrijednosti se vrši na sljedeći način:

$$\theta_{t+1,b} = \theta_{t,b} - \frac{\alpha}{\sqrt{0.000001 + s^2{}_{t,b}}} \nabla C_b(\boldsymbol{\theta}_t),$$

gdje α predstavlja korak učenja, $s^2{}_{t,b}$ sumu svih trenutnih gradijenata za parametar sa indeksom *b*, broj 0.000001 je dodat da bi se izbjeglo moguće dijeljenje sa nulom, *b* predstavlja indeks parametra koji se ažurira, $\theta_{t,b}$ predstavlja vrijednost parametra sa indeksom b u trenutnoj iteraciji, $\theta_{t+1,b}$ vrijednost parametra sa indeksom *b* u sljedećoj iteraciji a $\nabla C_b(\boldsymbol{\theta}_t)$ predstavlja gradijent za parametar sa indeksom *b*.

Dakle, za sve parametre, sa različitim indeksima *b*, će se posebno ažurirati vrijednost koraka

učenja. Drugim riječima, $s^2{}_{t,1}$ će biti suma svih prošlih gradijenata parametra sa indeksom 1. Indeks predstavlja jednu vrijednost težine ili biasa, odnosno, jedan parametar *b* odgovara samo jednom parametru u modelu. U MGD algoritmu β konstanta je imala istu vrijednost za sve promjenljive. U Adagard algoritmu, korak učenja će se posebno prilagođavati za svaki indeks.

Adagard ima dobre rezultate kod primjena u kojima je veliki broj gradijenata približno jednak nuli. Primjer su problemi računarskog vida i procesuiranja prirodnog jezika.

5). RmsProp algoritam se zasniva na sličnim principima kao i i Adagard algoritam. Osnovna razlika je u načinu na koji se ažuriraju vrijednosti u svakoj iteraciji u odnosu na indeks b. Postupak ažuriranja unutar RmsProp algoritma se vrši eksponencijalnim pokretnim prosjekom kvadrata gradijenata. Ovim postupkom, veću važnost će imati trenutni gradijenti (gradijenti u trenutnoj iteraciji t). Izmijenjena formula za akumuliranu sumu glasi:

$$v^2{}_{t,b} = \beta v^2{}_{t-1,b} + (1-\beta)\left(\nabla C_b(\boldsymbol{\theta}_t)\right)^2,$$

gdje $v^2{}_{t,b}$ predstavlja sumu svih trenutnih gradijenata sa indeksom *b*, $v^2{}_{t-1,b}$ predstavlja sumu svih prošlih gradijenata sa indeksom b, β predstavlja konstantu koja govori koliko (eksponencijalno) utiče trenutna vrijednost gradijenta na sumu $v^2{}_{t,b}$ a $\nabla C_b(\boldsymbol{\theta}_t)$ predstavlja vrijednost gradijenta parametra sa indeksom b u iteraciji *t*.

Formula za ažuriranje parametra $\theta_{t+1,b}$ glasi:

$$\theta_{t+1,b} = \theta_{t,b} - \frac{\alpha}{\sqrt{0.000001 + v^2{}_{t,b}}} \nabla C_b(\theta_t),$$

gdje α predstavlja korak učenja, $v^2{}_{t,b}$ sumu svih trenutnih gradijenata za parametar sa indeksom *b*, broj 0.000001 je dodat da bi se izbjeglo moguće dijeljenje sa nulom, *b* predstavlja indeks parametra koji se ažurira, $\theta_{t,b}$, predstavlja vrijednost parametra sa indeksom *b* u trenutnoj iteraciji, $\theta_{t+1,b}$ vrijednost parametra sa indeksom *b* u sledećoj iteraciji a $\nabla C_b(\boldsymbol{\theta}_t)$ predstavlja gradijent za parametar sa indeksom *b*.

Dakle, jedina razlika između Adagard algoritma i RmsProp algoritma je što se konstantom β daje veći značaj nedavnim vrijednostima gradijenata. Uticaj prošlih gradijenta će ekponencijalno da opada u svakoj iteraciji *t*. Ponovo će da se računa vrijednost posebno za svaki parametar *b*, što znači da se ažuriranje vrši nezavisno za svaki parametar.

RmsProp ima dobre rezultate sa problemima koji imaju greške, ili nesigurnost u podacima - šum (engl. *Noise* - noisy problems). Takođe, za probleme mašinskog učenja gdje podaci nijesu ograničeni na početni skup već pristižu (suprotno od batch gradient pristupa) RmsProp je

pokazao dobre rezultate.

Sve u svemu, RmsProp i Adagard dobro rješavaju različite probleme u mašinskom učenju.

6). Adam (engl. *Adaptive Estimation Movement*) će kombinovati osobine RmsPropa i Adagarda. Do sada je predstavljeno više pristupa gradijentnom spustu. Svi ti pristupi su ciljano pokušavali da nadoknade neke mane osnovnog gradijentnog spusta. ADAM algoritam pokušava da preuzme pozitivne osobine RmsPropa i Adagard algoritma. Koristiće se dvije vrijednosti momenta. Prva vrijednost momenta predstavlja vrijednost prošlih gradijenata sada modifikovanu za hiperparametar β_1 a druga vrijednost predstavlja kvadratnu vrijednost gradijenta ponovo modifikovanu preko parametra β_2. Adam koristi prvi i drugi moment da bi ažurirao parametre. Prvi moment se računa preko osnovnog gradijenta, a drugi moment je varijansa odnosno kvadratna vrijednost gradijenta. Za prvi moment važi:

$$m_{t,b} = \beta_1 m_{t-1,b} + (1-\beta_1)\nabla C_b(\boldsymbol{\theta}_t),$$

gdje $m_{t,b}$ predstavlja trenutnu vrijednost momenta za parametar sa indeksom b, β_1 je hiperparametar koji govori o uticaju prošlih gradijenata na isti način na koji je konstanta β uticala kod RmsProp algoritma, $m_{t-1,b}$ je prošla vrijednost momenta za parametar sa indeksom b a $\nabla C_b(\boldsymbol{\theta}_t)$ je gradijent *C* funkcije za parametar sa indeksom *b*.

Drugi moment je dat formulom:

$$v^2{}_{t,b} = \beta_2 v^2{}_{t-1,b} + (1-\beta_2)\left(\nabla C_b(\boldsymbol{\theta}_t)\right)^2,$$

gdje predstavlja $v^2{}_{t,b}$ trenutnu vrijednost momenta za parametar sa indeksom b, β_2 je hiperparametar koji govori o uticaju prošlih gradijenata na isti način na koji je konstanta β uticala kod RmsProp algoritma, $v^2{}_{t-1,b}$ je prošla vrijednost momenta za paramater sa indeksom b a $(\nabla C_b(\boldsymbol{\theta}_t))^2$ je kvadratna vrijednost gradijenta *C* funkcije za parametar sa indeksom *b*.

Dakle, u Adam algoritmu će se vršiti dva nezavisna računanja (prvi i drugi moment) za svaki parametar sa indeksom *b*. Računanje će se vršiti posebno za svaki parametar.

Autori ovog algoritma su primijetili da prilikom inicijalizacije parametara prvog i drugog momenta, njihove vrijednosti imaju određeni bias prema 0. Za hiperparmetre β_1 i β_2 koji su bliski vrijednosti 1, a vrijednost momenata se inicijalizuje na 0 u početnim koracima, konvergencija bi bila veoma spora. To proizilazi iz jednačina za moment.

$$(1-\beta_2) \approx 0 \rightarrow (1-\beta_2)\left(\nabla C_b(\boldsymbol{\theta}_t)\right)^2 \approx 0,$$

Da bi se smanjio uticaj početnog biasa, uvode se korigovane vrijednosti prvog i drugog momenta. Važe jednačine:

$$\widehat{m_{t,b}} = \frac{m_{t,b}}{1-\beta_1},$$

$$\widehat{v^2{}_{t,b}} = \frac{v^2{}_{t,b}}{1-\beta_2},$$

gdje $m_{t,b}, v^2{}_{t,b}$ predstavljaju vrijednost prvog i drugog momenta za parametar *b* respektivno, $\widehat{m_{t,b}}$ i $\widehat{v^2{}_{t,b}}$ predstavljaju vrijednosti prvog i drugog momenta za parametar *b* korigovane za bias.

Na kraju finalna formula ažuriranja unutar Adam algoritma glasi:

$$\theta_{t+1,b} = \theta_{t,b} - \frac{\alpha}{0.00000001 + \sqrt{\widehat{v^2{}_{t,b}}}}\widehat{m_{t,b}},$$

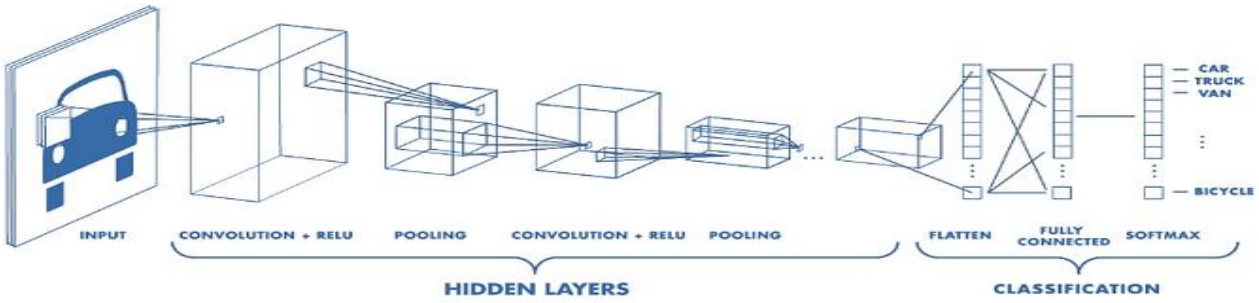
Gdje $\theta_{t,b}, \theta_{t+1,b}$ presdtavljaju staru i ažuriranu vrijednost za parametar sa indeksom *b*, $\widehat{m_{t,b}}$ i $\widehat{v^2{}_{t,b}}$ predstavljaju vrijednosti prvog i drugog momenta korigovane za bias za parametar sa indeksom b, $\alpha = 0.001$ korak učenja, hiperparametar $\beta_1 = 0.9$, hiperparametar β_2=0.999. Vrijednosti α, β_1 i β_2 su dobijene eksperimentalnim putem.

Adam algoritam ima bolje performanse u odnosu na prije prikazane algoritme. Često brže kovergira, ima bolju kontrolu nad ispadajućim gradijentima. Ideje svih algoritama koji koriste gradijentni spust je da se do rješenja dođe na brži i bolji način. Neki od algoritama koji su nastali od Adam algoritma su: SAdam, FusedAdam i A2grad.

3.2. Konvolucione neuralne mreže (CNN)

Ove mreže predstavljaju posebnu podgrupu neuralnih mreža koja je pogodna za rad sa vizuelnim podacima (slikama). Kada je posmatrana prosta neuralna mreža u prethodnim poglavljima, pomenuto je da skriveni slojevi mogu imati na poseban način definisanu arhitekturu. Jedan od mogućih načina definisanja slojeva je preko konvolucionih neuralnih slojeva. U matematici konvolucija predstavlja proces kombinovanja dvije funkcije u svrhu dobijanja mjere sličnosti između njih. Slike se posmatraju kao matrice piksela a konvolucione mreže pomažu da se pikseli povežu u manje cjeline, ili grupe, čime se simulira čovjekov vid na računaru. Uvodi se više filtera po kojima se posmatra slika da bi se pikseli grupisali u cjeline. Intuitivno, proces posmatranja se može opisati kao proces gledanja kroz durbin. Da bi čovjek imao uvid u stvarno stanje nekog dalekog broda neće polako da posmatra cijelu sliku broda. Staklo na durbinu neće biti savršeno glatko, pa pri posmatranju neki djelovi broda u jednom

trenutku će možda imati malo drugačije karakteristike u odnosu na stvarnost. Cijela slika broda predstavljaće polako posmatranje broda po visini i širini i konstrukciju te slike u čovjekovoj mašti.



Slika 22: Arhitektura konvolucione neuralne mreže[1]

Uvode se posebni filter slojevi (kerneli). Taj kernel je durbin kojim će se posmatrati slika. On će imati manje dimenzije u odnosu na sliku (opisivaće lokalne grupe neurona ili djelove slike) i sadržaće više parametara koje treba mreža da nauči kroz minimizaciju. Posebni kernel slojevi će imati posebnu namjenu. Jedna mreža često ima veliki broj kernel filtera čije parametre mreža treba da uči.

Posmatraju se RGB slike sa tri kanala: crvenim, zelenim i plavim. Ako je originalna slika predstavljena RGB modelom boja, to znači da će ulazne informacije biti $\text{širina} \times \text{visina} \times 3 (\text{informacije o boji})$. Kernel sloj ima istu formu, samo manje dimenzije. Početni kernel podaci će tokom treniranja slučajno biti inicijalizovani i imaće vrijednosti za sve kanale boja. Kod konvolucionih neuralnih mreža uče se parametri filtera.

Vrši se matrično množenje između matrice parametara filtera i aktivacionih vrijednosti neurona koji odgovaraju gornjem lijevom uglu i dimenzijama filtera. Vršiće se samo množenje između filtera dimenzija $8 \times 8 \times 3\,(\text{visina}, \text{širina}, \text{dubina})$ i prvog isječka slike dimenzija $8 \times 8 \times 3$. Filter će da se pomjera po širini slike i da vrši matrična množenja za svaki isječak. Klizanje ili korak ne mora imati vrijednost 1. Dimenzije vektora dobijenog množenjem su:

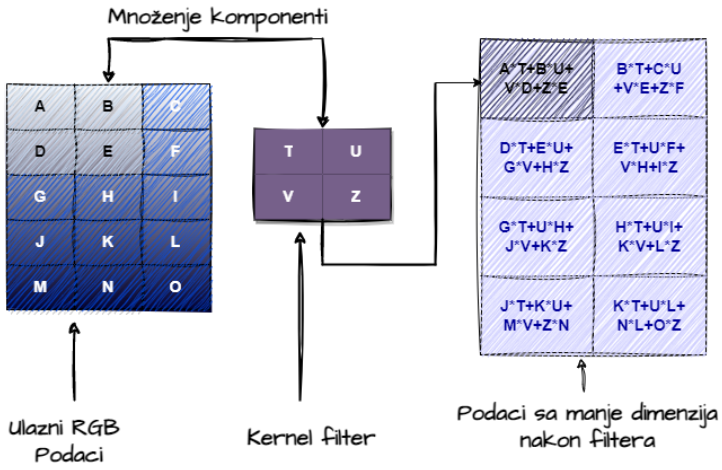
$$vektor = ((W - F + 2P) / S) + 1,$$

gdje je F dimenzija filtera (za filter je moguće da ima i različite dimenzije na primjer 3 x 4), W je dimenzija slike, P je pading (moguće dodavanje nula oko slike za potrebe filtera), a S predstavlja korak (u prije navedenom primjeru durbina korak ima vrijednost 1).

[1] https://miro.medium.com/v2/resize:fit:720/format:webp/1*kkyW7BR5FZJq4_oBTx3OPQ.png

Moguće je posebno računati širnu i dužinu preko gore navedene formule. To je za slučaj za filtere koji nijesu kvadratnih dimezija.

Ako se posmatra slika dimenzije $16 \times 16 \times 3$ sa filterom dimenzije $8 \times 8 \times 3$ i korakom 1. Izlazni vekor će imati dimenzije $vektor = (16 - 8 + 2 \times 0)/1 + 1 = 9$. Izlazni vektor će imati vrijednost $9 \times 9 \times 3$. Detaljan proces dobijanja izlaznog vektora je prikazan na sledećoj slici.



Slika 23: Proces konvolucije sa 2×2 kernelom

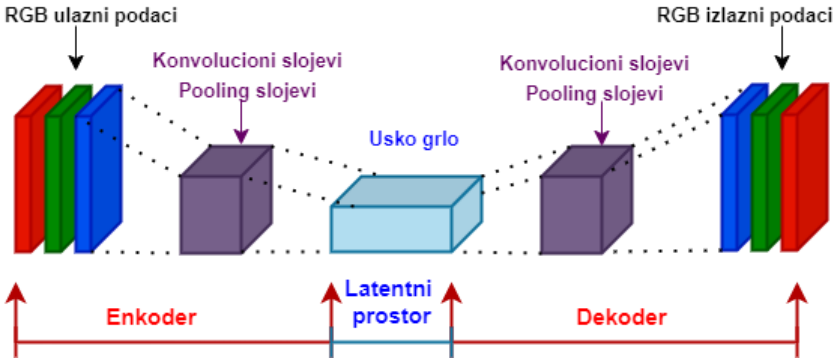
Vrijednosti dobijene matričnim množenjem se nazivaju aktivacionom mapom. Ona je u suštini odgovor na to kako filter reaguje na sliku. Važno je istaći da se ovim matričnim množenjem smanjuje broj potrebnih parametara za reprezentaciju slike. Kao što je već pomenuto, cijela slika se posmatra kroz kernel. To znači da za jednu aktivacionu mapu jednog kernela vrijednosti težina će samo jednom biti incijalizovane za cijelu sliku. U cijeloj neuralnoj mreži postoji više kernel slojeva koji će imati različite namjene (detekcija ivice, teksture). Svaki kernel će imati zadatak da detektuje jednu osobinu slike.

Nakon dobijanja aktivacione mape, često se primjenjuje ReLu funkcija, da bi se odbacili negativni težinski koeficijenti, ubrzao proces učenja i izbjegli problemi sa računanjem gradijenta.

Konvolucione neuralne mreže zbog kernela koji pomaže detekciji oblika, imaju veliku primjenu u mašinskom učenju u digitalnim slikama. Mnogi kompresioni modeli koji su ispitani sadrže konvolucione slojeve kojima se detektuje određeni oblik. U prethodnom tekstu je samo opisan jedan konvolucioni sloj, koji je dio arhitekture konvolucione neuralne mreže sa više različitih slojeva. Moguće je da izlazne vrijednosti konvolucionog sloja budu ulaz nekog drugog sloja, koji za cilj ima da pronađe kompleksnije oblike i veze.

3.3. Autoenkoderi (AE)

Konvolucione neuralne mreže su samo jedna od arhitektura koju je moguće koristiti za modele neuralnih mreža koje vrše proces kompresije slika. Konvolucioni slojevi su česti unutar arhitektura, zbog mogućnosti povezivanja djelova slike u cjelinu. Drugi sličan pristup, na kojem su bazirana mnoga rješenja u oblasti kompresije digitalnih slika zasnovanom na mašinskom učenju, je koncept autoenkodera. Neuralna mreža će biti podijeljenja na tri dijela: enkoder, dekoder i dio koji predstavlja kodiranu reprezentaciju podataka (latentni prostor). Ova arhitektura je veoma pogodna za digitalnu kompresiju sa gubicima jer je moguće odlučiti na koji način će podaci biti kodirani i dekodirani. Taj proces kodiranja značajnih podataka se obavlja u enkoderu i moguća je manipulacija podacima u centralnom sloju (latentnom prostoru) preko modifikacije parametara. Na ovaj način se mogu vještački isticati određene osobine kao što su glatkoća i oštrina određenih detalja. Važno je naglasiti da autoenkoderi sami po sebi ne moraju nužno predstavljati cijelu neuralnu mrežu koja ima zadatak da kodira i dekodira podatke. Većina arhitektura mreže sadrži mnogo posebnih elemenata kao što su konvolucioni slojevi, više autoenkodera specijalizovane namjene, softmax funkcije, funkcije aktivacije i drugo.



Slika 24: Arhitektura autoenkoder mreže za kompresiju slika

Enkoder preuzima ulazne podatke (slike u pikselima). Od ulaznih podataka do kodnog dijela se nalazi više skrivenih slojeva koji imaju za cilj da uspješno i kompaktno predstave sliku sa manjim brojem dimenzija. Ti slojevi mogu da budu konvolucioni, mogu da vrše agregaciju podataka pooling filterom, mogu biti potpuno povezani slojevi (slojevi koji su prikazani u prvom primjeru neuralne mreže gdje svaki neuron je povezan sa drugim preko težine i biasa), ili kombinacija više različitih slojeva u cilju smanjenja dimenzionalnosti podataka i efektivnog prikaza podataka u latentnom prostoru. Enkoder modeluje značajne podatke i moguće je naglasiti

koji podaci imaju veću važnost preko skrivenih slojeva. Između slojeva enkodera se koriste različite funkcije kao što je ReLu funkcija ili sigmoid funkcija, koje imaju za cilj lakše računanje gradijenata i izbjegavanje preprilagođavanja. Procesi normalizacije (smanjenje uticaja ispada gradijenta na sistem koji uči), regulacije (pokušavanje izbjegavanja pre prilagođavanja mreže) mogu biti nakon svih slojeva, ili pojedinačno nakon svakog sloja. Ova osobina, kao i mogućnost odabira broja slojeva čini ovu arhitekturu primamljivom za rad i istraživanje.

Latentni prostor predstavlja sloj sa najmanjom dimenzionalnošću unutar mreže autoenkodera. On predstavlja usko grlo unutar spejalizovane neuralne mreže. Cilj je napraviti dobru predstavu o podacima, zadržati informacije o ulaznom podatku na podacima sa mnogo manjim dimenzijama. U latentnim podacima broj dimenzija zna stvarno biti mali. Za sliku od 28 piksela, kada se posmatra slika kao ulazni podatak u RGB sistemu boja, će biti potrebno $3 \times 28 \times 28 = 2352$ aktivacije neurona (pomatra se ulazna aktivacijja za RGB sliku). Latentni prostor za jednu autoenkoder mrežu može imati na primjer 2 dimenzije. To znači da informacija o vrijednosti aktivacije 2352 neurona se sažima unutar latentnog prostora na vektor dvije dimenzije. Naravno, unutar mnogih arhitektura latentni prostor može imati veću dimenzionalnost. Međutim, dimenzije slika su veće od 28×28 piksela u stvarnosti što znači da nivo ulaznih informacija je često mnogo veći. Nivo sačuvanih detalja često zavisi od dimenzija latentnog prostora jer smanjenjem dimezionalnosti gube se važne informacije o slici. Moguće je produženim treniranjem mreže postići slične rezultate sa manjim dimenzijama ali to znatno usporava proces učenja u autoenkoder neuralnoj mreži.

Treći nezavisan dio strukture je predstavljen kao dekoder. U ovom dijelu se vrši obrnuti smjer operacija koji je opisan u polju enkodera. Polako se od latentnog prostora gradi slika koja će biti izlaz autoenkodera. Sama struktura autoenkodera dovodi do zaključka da su enkoder i dekoder simetrični djelovi. To znači da se obrnutim operacijama rekonstruiše slika koja neće imati iste karakteristike ali će podsjećati na original.

Mjera različitosti ulazne i izlazne slike će predstavljati funkciju cijene koja se minimizuje. Proces stvaranja izlazne slike je generativne prirode. Kao dobar primjer autoenkodera je autoenkoder koji uklanja šum. To znači da se iz latentnih podataka pri kodiranju uklanjaju tačkice koje predstavljaju šum i pravi se nova slika. Moguće je trenirati autoenkodere da stare sivoskalirane slike predstave slikama u boji. Sve ovo su generativne operacije. U kompresiji slika moguće je odlučiti koje informacije će slika zadržati. Moguće je na specifične načine kodirati oblike, ivice, artifekte i brinuti o nivou detalja. Za izlaznu sliku je moguće

podesiti veličinu, nivo sačuvanih detalja kao i obraćati pažnju na specifične detalje. Naravno proces kodiranja i dekodiranja je proces sa gubitkom jer se koristi mnogo funkcija za normalizaciju, regulaciju i funkcija za aktivaciju.

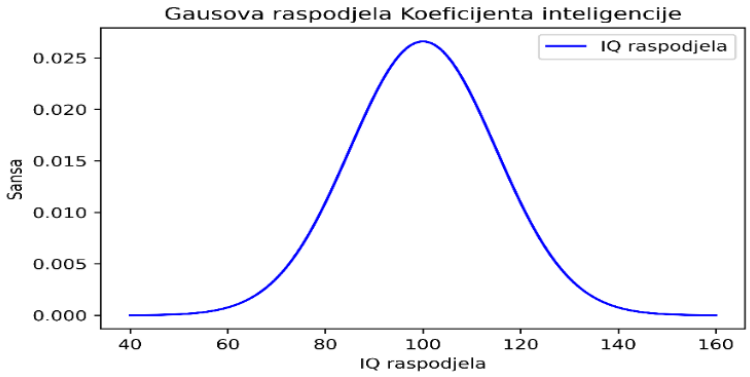
Proces učenja je već objašnjeni proces kojim se uče težinski koeficijenti enkodera i dekodera. Princip učenja je zasnovan na minimizaciji razlike između ulaza i rekonstruisanog izlaza i primjeni funkcije cijene koja će biti minimizovana preko neke varijante gradijentnog spusta. Kao i kod konvolucionih mreža ReLu funkcija je najčešće zastupljena aktivaciona funkcija pored sigmoidne funkcije. U praksi je definisano više vrsta autoenkodera koji odgovaraju različitim namjenama i problemima. Pregled najzančajnih varijanti je dat u nastavku izlaganja.

3.4. Varijacioni autoenkoderi (VAE)

Varijacioni autoenkoderi predstavljaju modernu vrstu autoenkodera sa većom slobodom u latentnom prostoru. Arhitektura neuralne mreže varijacionog autoenkodera ima istu strukturu kao arhitektura običnog autoenkodera. Osnovna razlika je u reprezentaciji latentnog prostora. Kod autoenkodera latentni prostor je predstavljao višedomenzioni vektor koji nije imao specijalne strukturne karakteristike. Pojavom struktura u latentnom prostoru kod varijacionog autoenkodera omogućava bolje organizovanje podataka.

Osnovne generativne osobine autoenkodera su više prisutne u varijacionom autoenkoderu. Razlika je što se vektor latentnog prostora na drugačiji načim kodira. VAE kodira parametre latentnog prostora kao normalnu Gausovu distribuciju. Ta distribucija ima srednju vrijednost i parametre devijacije. Ideja iza ovog načina kodiranje jeste da je moguće uzeti uzorak iz predstavljene kodirane distribucije koji će imati slične ali ne potpuno iste osobine kao original. Drugim riječima, uzorak iz latentnog prostora će imati slične osobine kao original, ali ne potpuno iste. Zbog ove osobine moguće je reći da VAE pripadaju grupi autoenkodera sa potpuno generativnim modelima. Slike dobijene na ovaj način će podsjećati na original, ali neće imati potpuno iste osobine.

Te generativne sposobnosti se mogu iskoristiti za kreiranje novih informacija o slici. GAN modeli koji se koriste na primjer za rekonstukciju starih sivoskliranih slika koriste generativna svojstva da bi slikama dodijelili boje.

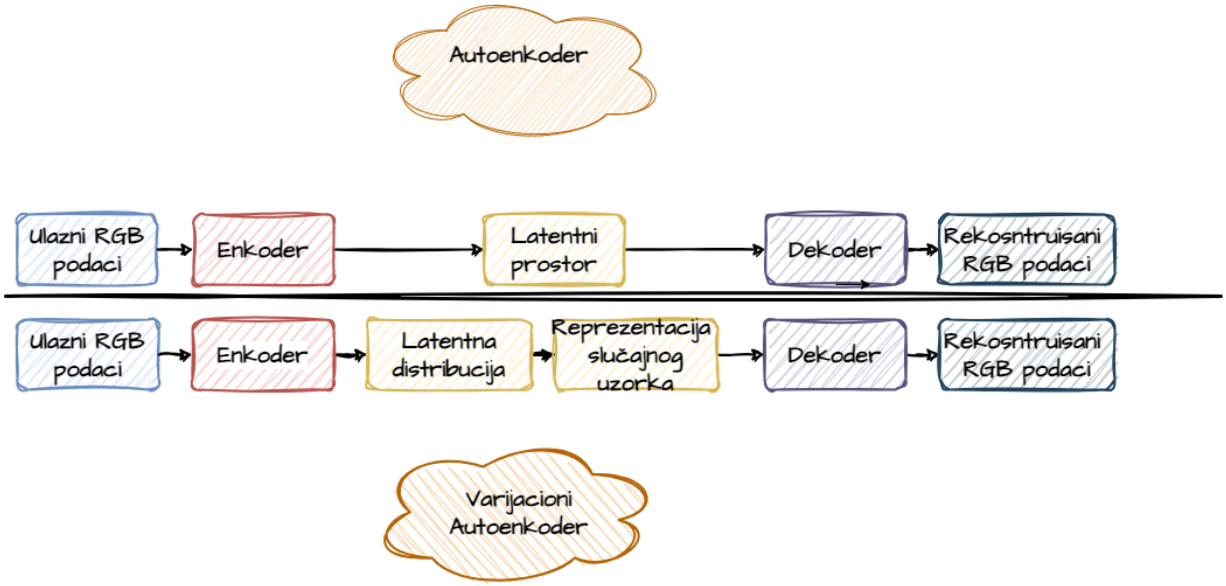


Slika 25: Gausova normalna distribucija koeficijenta inteligencije

Kodiranje latentog prostora na ovaj način omogućava uvođenje principa vjerovatnoće i korišćenje slučajnog uzorka sa Gausove krive za reprezentaciju rezultata. Ideja je da se umjesto Latentnog prostora modeluju parametri Gausove distribucije koje će funkcija učiti za svaki podatak. Pored učenja svih težinskih parametara enkodera i dekodera uče se i parametri Gausove krive. Naknadno sa latnente repreznetacije uzima se slučajni uzorak od koga će biti vršena rekonstrukcija podataka. Ostali koraci u dekoder dijelu arhitekture su potpuno isti kao i kod regularnih AE.

Primjena VAE unutar digitalne kompresije primjenom tehnika mašinskog učenja je veoma velika. Većina modela koji će naknadno biti predstavljeni sadrži modifikovane VAE za određeni problem. Generativna svojstva unutar kompresije dolaze do velikog značaja. Pri primjeni JPEG algoritma svi podaci unutar slike su posmatrani na isti način. Kod mašinskog učenja, posredstvom generativnih mogućnosti VAE, moguće je mijenjati strukturu slike, pri čemu gubitak informacija neće predstavljati problem za posmatrača.

Na primjer šara na tigru, JPEG algoritam će da sve šare na isti način predstavi bez obzira na njihovu važnost, a algoritmi koji u sebi sadrže VAE možda će imati mogućnost da istu sliku postave bez nekih sitnih detalja koji ne utiču mnogo na kvalitet slike. Na slici 26 su prikazane razlike između autoenkodera i varijacionog autoenkodera.



Slika 26: Blok dijagrami autoenkodera i varijacionog autoenkodera

4. Metode validacije rezultata i Python biblioteke

Ovo poglavlje ima za cilj da bliže pojasni načine i metode koji su korišćeni za dobijanje eksperimentalnih rezultata predstavljenih u radu. Konceptualno je podijeljena u dvije cjeline: Korišćene biblioteke programskog jezika Python i metode evaluacije rezultata.

Eksperimentalna postavka se generalno odnosi na sve parametre eksperimenta. U radu su predstavljene tri hipoteze:

1. Pristupi za kompresiju digitalnih slika zasnovani na mašinskom učenju (neuralnim mrežama) nude poboljšanja u kvalitetu kompresije, u odnosu na klasični JPEG kompresioni algoritam. Navedeno važi čak i u slučaju slika koje su mimo skupa koji je korišćen za obučavanje neuralne mreže.
2. Promjena parametara i svojstava slika koje se kompresuju, uključujući: veličinu slike, rezoluciju, nivo osvjetljaja, itd., ali i promjena samog sadržaja slike, nemaju presudan uticaj na efikasnost kompresionog algoritma zasnovanog na tehnikama mašinskog učenja (modelu dobijenom obučavanjem neuralne mreže).

3. Određeni modeli koji će se ispitivati u radu dovešće do boljeg vizuelnog kvaliteta komprimovanih slika, za isti nivo kompresije, u odnosu na konvencionalne pristupe JPEG algoritma.

Da bi trvđenja iz hipoteza bila tačna odabran je specijalizovani set podataka. Izabran je DIV2K [2] sa web stranice Kaggle[3] koja sadrži veliki broj dataset-ova pogodnih za mašinsko učenje. Moguće je izabrati specijalizovane dataset-ove automobila, životinja kao i setove koji sadrže podatke za medicinsku dijagnostiku.

Set podataka DIV2K sadrži preko 800 slika koji nemaju uniformno definisane dimenzije, uniformno definisan opseg veličina i uniformno prisustvo istih oblika u sadržaju slike. Slike variraju po sadržaju i strukturama. Dataset sadrži slike veoma visokog kvaliteta koje su pogodne za vizuelno kvalifikovanje slika dobijenih kompresijom.

Većina biblioteka je dostupna za Python i većinu drugih programskih jezika. Python je izabran za implementaciju jer ima veoma kompaktan i lako razumljiv kod uz veoma dobru podršku. Značajne biblioteke TensorFlow-Compression i TensorFlow su široko korišćene u mašinskom učenju. TensorFlow-Compression je biblioteka koja, između ostalog, sadrži alate i modele za kompresiju sa gubicima i ima svoju implementaciju u Python okruženju operativnog sistema Linux.

4.1. Pregled korišćenih Python biblioteka

U programskom jeziku Python je razvijeno mnoštvo modula i biblioteka za široku naučnu upotrebu. Jezik je jednostavan i čitljiv, sa veoma dobrom onlline podrškom. Multiplatformski je, sa veoma širokom primjenom i sa usko specijalizovanim bibliotekama (biblioteke za obradu slike, web programiranje, mašinsko učenje).

Python programski jezik je jedan od najviše korišćnih programskih jezika današnjice. Razvitkom novih verzija vodi se računa o pristupačnosti i povećava se brzina izvršavanja. Za skok sa verzije 3.10 na 3.11 brzina izvršavanja programa se prosječno povećala za 25%. Python je živ programski jezik i njegovi kreatori se trude da pronađu bolja rješenja i da ga unaprede.

[2] https://www.kaggle.com/datasets/joe1995/div2k-dataset?resource=download
[3] https://www.kaggle.com/

Jedan od interesantnih projekata je biblioteka koji ima za cilj da zamijeni Java Script programski jezik. To je Pyscript biblioteka. Rješenja dobijena preko ove biblioteke programskog jezika Python još su mnogo sporija, ali obećavaju.

U narednim cjelinama će biti predstavljene najznačajnije biblioteke koje su korišćene u ovom Master radu.

4.1.1. NumPy bibloteka

Za rad sa listama, često se koristi blioteka NumPy koja je 50 puta brža u odnosu na direktni rad sa listama u nativnom Python okruženju. Biblioteka je veoma pogodna za rad sa nizovima i operacije nad članovima nizova (bilo da su to jednodimenzioni, dvodimenzioni i višedimenzioni). Sam naziv NumPy-Numerical Python nam govori o cilju same biblioteke: brz i efikasan rad sa numeričkim vrijednostima.

Biblioteka sadrži generatore slučajnih vrijednosti, univerzalne funkcije koje još više upotpunjavaju rad sa brojevima, nizovima, funkcijama. Kombinacija NumPy i Matplotlib nam omogućava vizuelizaciju funkcija koja podsjeća na softver MATLAB. Funkcije kojim se liste porede, pronalazak minimuma i maksimuma unutar višedimenzionalnih nizova, su samo neke od ugrađenih funkcija.

Važno je naglasiti da je NumPy biblioteka implementirana u programskom jeziku C što nudi veliku brzinu rada sa matricama kao dvodimenzionalnim nizovima. Nudi se čitav niz alata za rad sa višedimenzionim podacima, rad sa inverznim matricama, matričnim množenjem, transponovanim matricama. U biblioteci postoji gradijent funkcija numpy.gradient() koja vrši izračunavanje gradijenta, što ovu biblioteku čini pogodnom za mašnsko učenje.

Slijedi prikaz nekoliko primjera iz ove biblioteke vezanih za osnovne operacije sa matricama

```
import numpy as np
niz1=np.array([1,2,3,4])
niz2=np.array([5,6,7,8])
niz3=np.array([[1,2,3],[5,6,7],[9,10,11]])
niz4=np.array([[3],[4],[5]])

sabiranje=np.add(niz1,niz2)               #sabiranje=[6,8,10,12]
oduzimanje=np.subtract(niz1,niz2)         #oduzimanje=[-4,-4,-4,-4]
mnozenje=np.matmul(niz1,niz2)             #mnozenje=70
stepenovanje=np.power(niz1,niz2)          #stepenovanje=[1,64,2187,65536]
mod=np.mod(niz2,niz1)                     #mod=[0,0,1,0]
reminder=np.remainder(niz2,niz1)          #remainder=[-4,-4-4-4]
max_jednodim=np.amax(niz1,0)               #max_jednodim=4
max_visedimen=np.amax(np.amax(niz3,0))     #max_visedim=11

trasnponovana=np.transpose(niz3)           #transponovana matrica
Jednodimenziona=np.squeeze(niz3)           #jednodimenzioni niz
mnozenje_matricno=np.matmul(niz3,niz4)     #mnozenje matricno=[26,
                                           #                   64,
                                           #                   172]
inverzna=np.linalg.inv(niz3)               #inverzna=[-1.930,    3.860, -1.930
                                           #           3.860,   -7.720,  3.860
                                           #          -1.930,    3.860, -1.930]
```

4.1.2. Matplotlib biblioteka

Matplotlib biblioteka služi za vizuelizaciju podataka unutar Python programskog jezika. Moguće je vizuelizovati podatke u vidu neprekidne linije (najčešća vizualizacija), tačke (pogodna tehnika vizuelizacije za velike skupove podataka), stubića (pogodno za vrijednosti gdje se naglašava rast ili pad na primjer u ekonomiji), pie grafikona i histograma. Na kanvas je moguće „nalijepiti“ mnoštvo elemenata kao objekte. Ti nezavisni objekti mogu biti linija, osa, tačka, legenda pa čak i napomene vezane za grafik kroz komentar.

Sve slike, grafike, na kojima su izvršene određene promjene i prikazane preko Matplotlib biblioteke moguće je sačuvati u .PNG, .jpeg i u mnoštvu drugih formata za čuvanje digitalnih slika.

U kodu koji slijedi je prikazan isječak vizuelizacije rezultata.

```
  plt.figure(figsize=(20, 12), dpi=600) #visok kvalitet rezultata garantovan sa dpi=600
  plt.legend(prop={'size': 20})
 plt.scatter(x, a1, label='bmshj2018_hyperprior_msssim-5', marker='o', s=20)#tip velicina
                                          markera
                                      plt.plot(x, a1)
  plt.scatter(x, a2, label='bmshj2018_hyperprior_msssim-6', marker='x', s=20) #legenda
                    plt.scatter(x, a3, label='jpeg 50', marker='s', s=20)
                for i in range(100):#povezivanje pojedinih tačaka na grafiku
        plt.plot([x[i], x[i]], [a1[i], a2[i]], color='blue', linestyle='--', linewidth=1)
        plt.plot([x[i], x[i]], [a1[i], a3[i]], color='red', linestyle='--', linewidth=1)

   plt.xlabel('X-Slike sortirane po  bmshj2018_hyperprior_msssim-5 MSSIM
vrijednosti',size='20')
   plt.ylabel('Y-Modeli',size='20')
   plt.legend()
   plt.tight_layout()
```

4.1.3. Os biblioteka

Os biblioteka se koristi za rad sa sistemskim komandama. Vrši se pristup direktorijumima, brisanje datoteka, njihovo otvaranje, kopiranje i manipulacija putanjama. Pri implementaciji numeričkih eksperimenta prezentovanih u ovom radu, Os biblioteka je korišćena za izvršenje sistemskih komandi unutar programskog jezika Python. U sljedećem textbox-u će biti prikazano aktiviranje Cuda alata unutar Windows podsistema. Ovaj proces će se ponavljati za svaku sliku iz skupa slika.

```
subprocess.run(['wsl', 'bash', '-c', 'CUDNN_PATH=$(dirname $(python -c "import
nvidia.cudnn;print(nvidia.cudnn.__file__)")) && echo $CUDNN_PATH'], capture_output=True,
text=True)

subprocess.run(['wsl', 'bash', '-c', 'export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/:$CUDNN_PATH/lib && echo $LD_LIBRARY_PATH'],
capture_output=True, text=True)
```

Dakle, Python program koji vrši evaluaciju modela mora da sadrži biblioteku koja će da radi sa sistemskim komandama za Windows i Linux okruzenje. Iz određenog foldera na Windows okruženju kopira se slika na Linux okruženje. Na toj slici se vrši kompresija i

dekompresija. Slika dobijena dekompresijom se šalje natrag u Windows okruženje gdje će se vršiti dalja evaluacija. Posebno će se kao stringovi čuvati različite putanje da bi se izvršio proces kopiranja, brisanja i preimenovanja.

Nakon kopiranja moguća je evaluacija rezultata na dobijenim slikama. Ovdje se cijeli proces kompresije i dekompresije vrši za sve modele iz biblioteke TensorFlow-Compression. Krajni rezultat ovih naredbi (ne obazirući se na stringove koji su nazivi kompresionih modela i strogo definisani u Compression biblioteci) od 1 slike se dobija 64 kompresionih slika i posebnim putem još 8 slika JPEG algoritmom. U narednom kodu je prikazan proces kreiranja, kompresovanja, kopiranja i dekompresovanja korišćenjem OS biblioteke.

```
subprocess.run(['wsl', '/home/legolasnk/anaconda3/envs/1/bin/python',
'~/compression/models/tfci.py', 'compress', mod_za_kompresiju, ime_za_kompresiju],
capture_output=True, text=True)
ime_za_dekompresiju='~/kompresije/'+filename+'.tfci'
novo_ime='~/kompresije/'+filename+mod_za_kompresiju+'.tfci'
subprocess.run(['wsl', 'mv', ime_za_dekompresiju,novo_ime], capture_output=True,
text=True)
subprocess.run(['wsl', '/home/legolasnk/anaconda3/envs/1/bin/python',
'~/compression/models/tfci.py', 'decompress', novo_ime], capture_output=True, text=True)
za_cp='~/kompresije/'+filename+mod_za_kompresiju+'.tfci'+'.png'
subprocess.run(['wsl', 'cp', za_cp,
'/mnt/c/Users/Korisnik/Desktop/compression/static/uploads/slike'], capture_output=True,
text=True)
```

U sledećem tekst polju će biti prikazan ekvivalent kompresije unutar Linux operativnog sistema.

```
python tfci.py compress neki_od_modela /path/to/image.png
#(relativna kod nas u Linuxu od homa)

python tfci.py compress b2018-leaky_relu-128-1  putanja_do slike #naredba je compress
b2018-leaky_relu-128-1 je model a putanja
phyton tfci.py decompress  /kompresije/original.png.tfci
```

4.1.4. OpenCV biblioteka

OpenCV biblioteka nudi alate za digitalnu obradu slike. Neki od korisnih alata su: formatiranje i segmentaciju slike, prepoznavanje objekata, stvaranje grafičkih interfejsa za

potrebe aplikacija. Slika u ovoj biblioteci predstavlja poseban objekat sa mnogim interesantnim osobinama koje se mogu iskoristiti.

Ova biblioteka se široko koristi u kompjuterskoj viziji (engl. *computer vision*). Detekcija lica, praćenje objekata, pa čak i algoritmi koji vrše prepoznavanja osoba nalazi na osnovu slika često koriste OpenCV biblioteku. Njena primjena u mašinskom učenju svodi se na rad sa slikama i video sadržajem. U problemima klasifikacije ima posebnu primjenu zbog mogućnosti detekcije objekata i segmentacije slike. Moguć je pristup slici, pristup pojedinačnom pikselu, podacima iz slike, metapodacima, izvršiti transformaciju slike u sivoskaliranu, rotirati sliku i još mnogo toga..

U predmetnom radu je korišćena OpenCV biblioteka za čitanje korisnih informacija o slici. Slika će biti pročitana kao NumPy niz. Sadržaće informacije o visini, širini i kanalima boje. Te informacije dalje mogu biti korišćene za metode kontrole kvaliteta (kao što je PSNR metoda). U sledećem tekst boksu su prikazane neke od mogućnosti rada sa bibliotekom OpenCV a rezultati napisanog koda su predstavljeni na slici 27.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from PIL import Image
image_path = "C:\\Users\\Korisnik\\Desktop\\compression_novi\\5.jpg"   # Otvaranje slike
color_slika = Image.open(image_path)
siva_slika = color_slika.convert('L')
velicina = (300, 300)
smanjena_siva_slika = siva_slika.resize(velicina, Image.BILINEAR)  # Interpolacija i
smanjenje dimezija
smanjena_siva_slika_array = np.array(smanjena_siva_slika)

gaussian_sum = np.random.normal(0, 25, smanjena_siva_slika_array.shape) #dodavanje gausovog
suma
slika_sa_sumom = np.clip(smanjena_siva_slika_array + gaussian_sum, 0, 255).astype(np.uint8)
```

U narednom kodu je prikazano otvaranje i čitanje slika za potrebe numeričkih eksperimenata prezentovanih u radu. Proces čitanja se ponavlja za sve modele kako bi se izračunale sve vrijednosti PSNR funkcije. Čitanje i upisivanje unutar CV funkcije takođe pomaže prilikom obrade slika. U programskom jeziku Python, ova biblioteka zahtjeva NumPy biblioteku.

```
cv_original=cv2.imread(original_ime)
cv_kompresovana=cv2.imread(ime_kompresovano)
```

Slika 27: Prikaz transformacije slika posredstvom OpenCv biblioteke

4.1.5. TensorFlow biblioteka

TensorFlow biblioteka predstavlja jednu od najvažnih biblioteka za mašinsko učenje. Tenzori su matematičke strukture kojima se generalizuju vektori i matrice. U TensorFlow biblioteci, broj predstavlja tenzor nultog reda (nema dimenzije), vektor predstavlja tenzor prvog reda a matrica višedimenzioni tenzor (može imati dvije ili više dimenzija). Ovakva predstava omogućava brz rad sa velikim brojem promjenljivih što većina problema mašinskog učenja zahtjeva.

Na ovaj način preko multidimenzionalnih nizova je moguće predstaviti slike, video sadržaj, CSV, Exel datoteke. Slike predstavljaju tenzor trećeg reda, gdje su dimenzije širina, visina i boja. Video zapisi su tenzori četvrtog reda koji pored tri dimenzije slika sadrže i četvrtu – vrijeme. CSV datoteke kao tenzori su tenzori drugog reda sa vrstama i kolonama. Veoma je važno naglasiti da ovaj način predstavljanja preko tenzora je mnogo brži za obradu velike količine informacija. U TensorFlow biblioteci optimizovan je rad sa slikama na gotovim dataset-ovima, kao što je već pomenuti MNIST dataset. Python program kojim se efektivno trenira model prepoznavanja ručno napisanih slika sa velikim stepenom prepoznavana (tačnost prepoznavanja

preko 97%) se izvršava preko podmodula Keras u nekoliko linija koda.

U sljedećem tekstualnom polju je prikazan jednostavan kod neuralne mreže. Može se vidjeti da je veoma lako podesiti parametre i kreirati veoma kompleksne strukture. Kompajliranje može biti izvršeno na više načina. Moguće je postaviti još više slojeva, predstaviti različite tipove neuralnih mreža kao što su Konvolucione neuralne mreže, Generativne neuralne mreže, Auto enkodreri i Varijabilni Auto enkoderi. Kao izlazno rješenje treniranja dobija se velika tačnost predviđanja. Cijeli proces treniranja za 15 epoha je 30 sekundi. Autoenkodri i Varijacioni autoenkdoeri sadrže dva dijela koji odgovaraju Enkoderu i Dekoderu.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist                          # MNIST dataset u Keras modulu
(x_train, y_train),(x_test, y_test) = mnist.load_data()# MNIST set i trening set slika oznakama
x_train, x_test = x_train / 255.0, x_test / 255.0      #vrijednosti piksela u  opseg
vrijednosti od 0-1
model = tf.keras.models.Sequential([           #slojevi se izvršavaju redom (sekvencijalno)
  tf.keras.layers.Flatten(input_shape=(28, 28)),# slike dvodimenzioni vektor od 28x28 piksela
                                               # pretvara u jednodimenzioni vektor 784 piksela
  tf.keras.layers.Dense(128, activation='relu'), # skriveni sloj neurona 128  koji se aktivira
                                                # ReLu  funkcijom
  tf.keras.layers.Dropout(0.2),                 #izbjegavanje  preprilagođavanja (overfitting)
  tf.keras.layers.Dense(10, activation='softmax') # izlazni sloj sa 10 vrijednosti cifara,
                                                  #softmax funckija uzima najveće vrijednosti
])
model.compile(optimizer='adam',                   #  koristi se Adam za minimizaciju funkcije
              loss='sparse_categorical_crossentropy',# odgovarajuća cost funkcija
              metrics=['accuracy'])                  # Praćenje tačnosti tokom treniranja

model.fit(x_train, y_train, epochs=15) # treniranje kroz 15 epoha (15 prolazaka kroz dataset)
model.evaluate(x_test, y_test)         #evaluacija modela
```

```
Epoch 15/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0292 - accuracy: 0.9900
313/313 [==============================] - 0s 951us/step - loss: 0.0681 - accuracy: 0.9828
```

Slika 28: Povratne informacije nakon izvršenog treniranja neuralne mreže

Sada je moguće proslijediti modelu sliku broja i trenirani model će sa tačnosti od preko 99% utvrditi kojoj od 10 klasa taj broj pripada. Manjim ispravkama koda i procesom segmetacije slike moguće je utvrditi napisane vrijednosti brojeva sa više cifara.

Pri sprovođenju numeričkih eksperimenata koji su osnov predmetnog rada, korišćene su ugrađene funkcije TensorFlow biblioteke kao što je MSSSIM i SSIM. U Compression biblioteci logika i arhitektura svih modela je postavljena preko TensorFlow biblioteke.

TensorFlow je razvijen u kompaniji Google 2015. godine. Navedena biblioteka radi dobro sa linearnom algebrom, statistikom i vjerovatnoćom. Primjena pri klasifikaciji slika je posebno značajna u oblasti medicinske dijagnostike. Pri varijantama koje odgovaraju potrebama personalnih računara uz NVIDIA CUDA alat može vršiti procese paralelizacije i direktno ubrzati kreiranje modela.

4.1.6. TensorFlow-Compression models biblioteka

TensorFlow-Compression nudi alate za kompresiju slika uz posredstvo TensorFlow biblioteke. U sebi sadrži istrenirane kompresione modele za digitalne slike. Riječ je o kompresiji sa gubicima. Biblioteka nije optimizovana za rad u operativnom sistemu Windows. Ona isključivo radi u operativnom sistemu Linux.

Važan alat, koji koriste TensroFlow i TensorFlow-Compression biblioteke, je CUDA alat. CUDA zahtjeva grafičku karticu visokih performansi i nema svoju verziju za sve grafičke kartice. Pored svih ovih manjkavosti, CUDA alat predstavlja jedan od najkorišćenih alata unutar TensorFlow biblioteke. U brzini treniranja modela CUDA može da ima veliku ulogu. CUDA alatom moguće je paralelizovati procese što dodatno smanjuje vrijeme treniranja modela neuralnih mreža. Svaka nova kompresija slika u Compression biblioteci zahtjeva ponovno uključenje CUDA alata preko Python programskog jezika.

Biblioteka TensorFlow-Compression nudi alate za kompresiju korišćenjem tehnika mašinskog učenja. Moguće je trenirati svoj model, napraviti složenu neuralnu mrežu koja sadrži konvolucione slojeve, enkoder-dekoder arhitekturu, varijacioni autoenkoder i drugo. Cijeli proces treniranja zna biti veoma dug i zahtjeva računare visokih performansi.

U ovom radu su prikazani neki od već treniranih modela kompresije slika koje ova biblioteka nudi:

(1) "Joint Autoregressive and Hierarchical Priors for Learned Image Compression":
   (a) `mbt2018-mean-mse-[1-8]`
   (b) `mbt2018-mean-msssim-[1-8]`

(2) "Variational Image Compression with a Scale Hyperprior":
   (a) `bmshj2018-factorized-mse-[1-8]`
   (b) `bmshj2018-factorized-msssim-[1-8]`
   (c) `bmshj2018-hyperprior-mse-[1-8]`

(d) `bmshj2018-hyperprior-msssim-[1-8]`

(3) "Efficient Nonlinear Transforms for Lossy Image Compression":

(a) `b2018-leaky_relu-128-[1-4]`

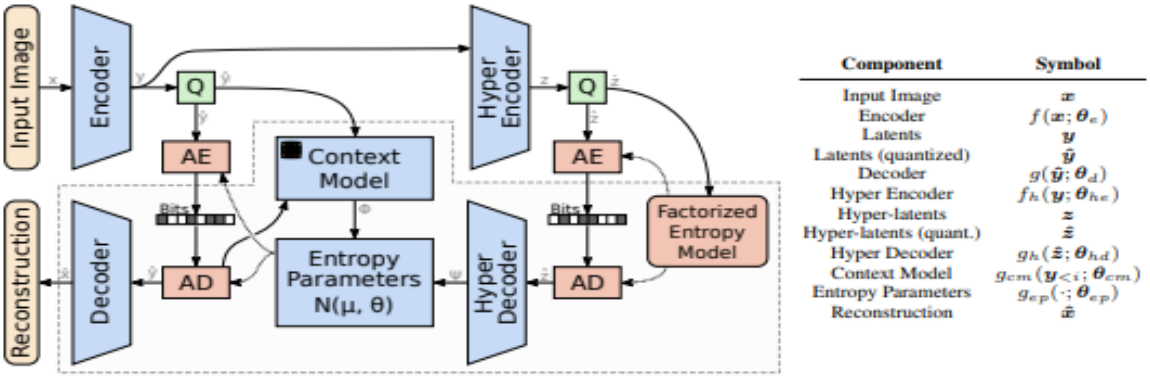
(b) `b2018-leaky_relu-192-[1-4]`

(c) `b2018-gdn-128-[1-4]`

(d) `b2018-gdn-192-[1-4]`

Razmatrani modeli su podijeljeni u tri grupe sa tri različita principa kompresije.

Prva grupa modela koristi enkoder dekoder arhitekturu, autoregresivne modele i hijerarhijske priore prilikom kompresije slika sa gubicima. Autoregresivni modeli se koriste za modelovanje odnosa piksela (vjerovatnoća budućeg piskela u odnosu na prethodne), a hijerarhiski priori modeluju strukturu slike na više nivoa detalja. Cijela arhitektura je veoma komplesna i kombinuje više principa i tehnika iz oblasti mašinskog učenja, posebno neuralnih mreža.

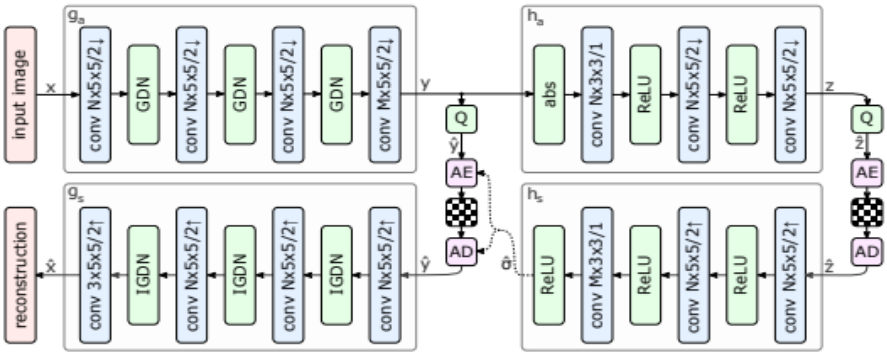


Slika 29: "Joint Autoregressive and Hierarchical Priors for Learned Image Compression arhitektura[4]

Model "Variational Image Compression with a Scale Hyperprior" koristi varijacione autoenkodere uz kombinaciju konvulucionih slojeva i hyperpriora za detekciju struktura. U arhitekturi, pored slojeva neuralne mreže se koriste i funkcije za normalizaciju kao što je ReLu funkcija ili funkcija apsolutne vrijednosti. Autoenkoderi prvenstveno imaju ulogu pri kompresiji slike, dok hyperpriori imaju ulogu u definisanju struktura i oblika. Na slici arhitekture se mogu vidjeti dvije odvojene cjeline. Lijeva strana predstavlja kodiranje slike putem varijacionog autoenkodera, dok desna predstavlja implementaciju hyperpriora preko autoenkodera.

[4] https://arxiv.org/pdf/1809.02736.pdf figure 1

Hyperpriori predstavljaju specijalan način modelovanja latentnog prostora. Ovim modelovanjem se mogu postaviti dodatne prestavke o distribuciji podataka i on može sadržati dodatne slojeve koji služe za isticanje željenih karakteristika slike.



Slika 30: "Variational Image Compression with a Scale Hyperprior" arhitektura[5]

Posledna grupa modela koristi dvije tehnike za različite pristupe kompresiji slika sa gubicima. Koristi se SADAM, specijalizovana verzija ADAM algoritma za potrebe rada sa slikama i kompresijama, i GDN (Generalized Divisive Normalization) koja se takođe koristilla u arhitektruri drugog modela. Vrši se nelinearna transformacija preko neuralnih mreža te optimizacija funkcija cijene preko navedenih metoda (SADAM i GDN). Po složenosti arhitekture predstavlja jednostavnije rješenje u odnosu na prethodne algoritme. Koriste se konvolucione neuralne mreže sa velikim brojem filtera.

Argumenti prilikom biranja modela imaju formu `bmshj2018-factorized-msssim-[1-8]`. Ovdje `bmhj2018` predstavlja broj modela, `hyperprior` sugeriše korištenje hyperpriora u strukturi, `msssim` govori da je model optimizovan za msssim metriku a `[1-8]` govori o stepenu kompresije. Drugi nazivi će bliže govoriti o osobinama kompresionih modela. `ReLu` i `GDN` respektivno upućuju na korišćenje Rectified Linear unit funkcije i Generalized Divisive Normalization funkcije. `Mse` govori da su pojedini modeli optimizovani za parametar srednje kvadratne greške, `msssim` sugeriše da su modeli optimizovani za MSSSIM indeks dok broj `128` i `192` predstavlja broj filtera u konvolucionoj neuralnoj mreži.

Na kraju sintaksa procesa kompresije i dekompresije slike može imati sledeći oblik u Linux operativnom sistemu:

[5] https://arxiv.org/pdf/1802.01436.pdf figure 4

```
python tfci.py compress neki_od_modela /path/to/image.png(relativna kod nas u Linuxu od homa)
python tfci.py compress b2018-leaky_relu-128-1  putanja_do slike #naredba je compress b2018-leaky_relu-128-1 je model a putanja
phyton tfci.py decompress  /kompresije/original.png.tfci
```

4.1.7. Ključni programi i prikaz rezultata

U nastavku će biti objašnjeno kako je u numeričkim eksperimentima implementiran proces prikupljanja informcija.U TensorFlow-Compression biblioteci je prikazano 10 modela kompresije sa više različitih kvaliteta. U kodu koji slijedi je prikazano prikupljanje informacija sa jednog modela.

```
        for i in range(1,5):# Racunanje Indeksa za slike prvog modela
                mod_za_kompresiju=mod_za_kompresiju+str(i)
                subprocess.run(['wsl', '/home/legolasnk/anaconda3/envs/1/bin/python', '~/compression/models/tfci.py', 'compress', mod_za_kompresiju, ime_za_kompresiju], capture_output=True, text=True)
                ime_za_dekompresiju='~/kompresije/'+filename+'.tfci'
                novo_ime='~/kompresije/'+filename+mod_za_kompresiju+'.tfci'
                subprocess.run(['wsl', 'mv', ime_za_dekompresiju,novo_ime], capture_output=True, text=True)
                subprocess.run(['wsl', '/home/legolasnk/anaconda3/envs/1/bin/python', '~/compression/models/tfci.py', 'decompress', novo_ime], capture_output=True, text=True)
                za_cp='~/kompresije/'+filename+mod_za_kompresiju+'.tfci'+'.png'
                subprocess.run(['wsl', 'cp', za_cp, '/mnt/c/Users/Korisnik/Desktop/compression/static/uploads/slike'], capture_output=True, text=True)
                ime_kompresovano="C:\\Users\\Korisnik\\Desktop\\compression\\static\\uploads\\slike\\" + filename+mod_za_kompresiju+'.tfci'+'.png'
                cv_original=cv2.imread(original_ime)
                cv_kompresovana=cv2.imread(ime_kompresovano)
                b2018_leaky_128_psnr[i-1]=PSNR(cv_original,cv_kompresovana)
                za_cp2='~/kompresije/'+filename+mod_za_kompresiju+'.tfci'
                subprocess.run(['wsl', 'cp', za_cp2, '/mnt/c/Users/Korisnik/Desktop/compression/static/uploads/slike'], capture_output=True, text=True)
                ime_kompresovano2=  "C:\\Users\\Korisnik\\Desktop\\compression\\static\\uploads\\slike\\"+filename+mod_za_kompresiju+'.tfci'
                b2018_leaky_128_velicina[i-1]=os.path.getsize(ime_kompresovano2)/1024
                b2018_leaky_128_msssim[i-1]=msssim(original_ime,ime_kompresovano)
                b2018_leaky_128_ssim[i-1]=ssim(original_ime,ime_kompresovano)
                mod_za_kompresiju=mod_za_kompresiju[:-1]
```

Sa 64 slike se preuzimaju podaci o veličini, MSSSIM, PSNR, SSIM za svaku sliku. Prikupljanje informacija za jedan model prikazano je u kodu koji slijedi. Sekvencijalno se vrši kompresovanje da bi se dobile slike koje odgovaraju svim nivoima svih modela. U suštini prethodni isječak kompresuje slike koje odgovaraju jednom modelu i preuzima sve informacije o njima (MSSSIM, PSNR, SSIM i veličina).

Da bi se sačuvale vrijednosti svih mjera kvaliteta za posebne slike kreira se klasa slika koja u sebi sadrži nizove vrijednosti koje odgovaraju pojedinim mjerama kvaliteta. Odnosno ako model sadrži 8 nivoa kvaliteta, nizovi koji odgovaraju tom modelu za posebne vrijednosti SSIM, PSNR, MSSSIM i veličine će imati 8 vrijednosti za jednu sliku (biće kompresijom dobijeno 8 slika sa kojih će biti preuzete vrijednosti indeksa).

```
class slika:                                                    # Modeli
    def __init__(self,   original_ime      ,original_velicina,
          b2018_leaky_128_psnr             ,b2018_leaky_128_velicina
         ,b2018_leaky_128_msssim           ,b2018_leaky_128_ssim             #prvi
         ,mbt2018_mean_mse_psnr            ,mbt2018_mean_mse_velicina           ,
         mbt2018_mean_mse_msssim           ,mbt2018_mean_mse_ssim            #drugi
         ,mbt2018_mean_msssim_psnr         ,mbt2018_mean_msssim_velicina        ,
         mbt2018_mean_msssim_msssim        ,mbt2018_mean_msssim_ssim         #treci
         ,bmshj2018_factorized_mse_psnr    ,bmshj2018_factorized_mse_velicina   ,
         bmshj2018_factorized_mse_msssim   ,bmshj2018_factorized_mse_ssim  #cetvrti
         ,bmshj2018_factorized_msssim_psnr ,bmshj2018_factorized_msssim_velicina ,
         bmshj2018_factorized_msssim_msssim,bmshj2018_factorized_msssim_ssim #peti
         ,bmshj2018_hyperprior_mse_psnr    ,bmshj2018_hyperprior_mse_velicina   ,
         bmshj2018_hyperprior_mse_msssim   ,bmshj2018_hyperprior_mse_ssim    #sesti
         ,bmshj2018_hyperprior_msssim_psnr ,bmshj2018_hyperprior_msssim_velicina ,
         bmshj2018_hyperprior_msssim_msssim,bmshj2018_hyperprior_msssim_ssim #sedmi
         ,b2018_leaky_relu_192_psnr        ,b2018_leaky_relu_192_velicina       ,
         b2018_leaky_relu_192_msssim       ,b2018_leaky_relu_192_ssim        #osmi
         ,b2018_gdn_128_psnr               ,b2018_gdn_128_velicina              ,
         b2018_gdn_128_msssim              ,b2018_gdn_128_ssim              #deveti
         ,b2018_gdn_196_psnr               ,b2018_gdn_196_velicina              ,
         b2018_gdn_196_msssim              ,b2018_gdn_196_ssim              #deseti ):
```

Sve vrijednosti nizova klase na početku se incijalizuju na nizove nula koje će služiti za smještanje vrijednosti. U narednom tekst polju je prikazan jedan prazni niz za jedan model (u radu se opisuju 10 kompresionih modela). Pošto je predstavljeni model sastavljen od četiri nivoa kvaliteta, vrijednosti ovog niza predstavlaju vrijednosti za 4 kompresijom dobijene slike.

```
b2018_leaky_128_psnr=[] #Prazan niz u kojem se čuvaju vrijednosti psnr za prvi model
b2018_leaky_128_psnr=[0] * 4
b2018_leaky_128_ssim=[] #prazan niz u kojem se čuvaju vrijednosti SSIM za prvi model
b2018_leaky_128_ssim=[0] *4
```

vrše mjerenja i za slike kompresovane JPG algoritmom na sličan način.

```
with open('poslednji_eks1.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow([  'original_ime '                  ,'original_velicina'
                    , # Modeli prvi
                     'b2018_leaky_128_psnr'             ,'b2018_leaky_128_velicina'          ,
                     'b2018_leaky_128_msssim'           ,'b2018_leaky_128_ssim '             ,
                      # drugi
                     'mbt2018_mean_mse_psnr'            ,'mbt2018_mean_mse_velicina'         ,
                     'mbt2018_mean_mse_msssim '         ,'mbt2018_mean_mse_ssim'             ,
                      # treci
                     'mbt2018_mean_msssim_psnr'         ,'mbt2018_mean_msssim_velicina'      ,
                     'mbt2018_mean_msssim_msssim'       ,'mbt2018_mean_msssim_ssim'          ,
                     # cetvrti
                     'bmshj2018_factorized_mse_psnr'    ,'bmshj2018_factorized_mse_velicina' ,
                     'bmshj2018_factorized_mse_msssim'  ,'bmshj2018_factorized_mse_ssim'     ,
                      # peti
                     'bmshj2018_factorized_msssim_psnr' ,'bmshj2018_factorized_msssim_velicina,
                     'bmshj2018_factorized_msssim_msssim','bmshj2018_factorized_msssim_ssim'  ,
                      # sesti
                     'bmshj2018_hyperprior_mse_psnr  '  ,'bmshj2018_hyperprior_mse_velicina' ,
                     'bmshj2018_hyperprior_mse_msssim'  ,'bmshj2018_hyperprior_mse_ssim'     ,
                      # sedmi
                     'bmshj2018_hyperprior_msssim_psnr' ,'bmshj2018_hyperprior_msssim_velicina,
                     'bmshj2018_hyperprior_msssim_msssim','bmshj2018_hyperprior_msssim_ssim'  ,
                      # osmi
                     'b2018_leaky_relu_192_psnr'        ,'b2018_leaky_relu_192_velicina'     ,
                     'b2018_leaky_relu_192_msssim'      ,'b2018_leaky_relu_192_ssim '        ,
                      # deveti
                     'b2018_gdn_128_psnr '              ,'b2018_gdn_128_velicina'            ,
                     'b2018_gdn_128_msssim'             ,'b2018_gdn_128_ssim'                ,
                     # deseti
                     'b2018_gdn_196_psnr'               ,'b2018_gdn_196_velicina'            ,
                     'b2018_gdn_196_msssim'             ,'b2018_gdn_196_ssim'                ,
                        ])
    for i in slike:
        writer.writerow(i.prevod na cscv())
```

U prethodnom tekst polju je opisan proces upisivanja vrijednosti u CSV datoteku. Podaci iz pojedinačnih objekata klase (pojedinačne slike) se upisuju u CSV datoteku. Slika kao istanca

klase će zauzimati jedan red datoteke. Iz CSV datoteke pogodno je vršiti proračune, upoređivati rezultate i vizuelno ih prikazivati. Prvo se sve informacije čuvaju kao objekat slika, pa se taj objekat upiše u jednu kolonu (red). Ovaj postupak se sprovodi funkcijom prevod_na_csv().

4.2. Metode validacije

Metode validacije se koriste da na adekvatan način opišu rezultate eksperimenta. Izabrano je više mjera kvaliteta koje će rezultate opisati na više načina. MSSSIM i SSIM mjere kvaliteta simuliraju čovjekov vid i veću važnost daju struktrama. PSNR i odnos veličine posmatra razlike između originala i kompresovanog rezultata analitički kao mašina. Cilj je uporediti rezultate dobijene na uređenim parovima:

1. Originalna slika - slika dobijena kompresijom preko JPEG algoritma.
2. Originalna slika - slika dobijena primjenom modela iz TensorFlow-Compression biblioteke

Upoređivanjem ovih rezultata moguće je doći do validnih zaključaka o odnosu kvaliteta JPEG algoritma i algoritama koji su prikazani u TensorFlow Compression biblioteci. Važno je napomenuti da set slika na kojima se vrši mjerenje nije korišćen prilikom treniranja modela, nema uskodefinsanu veličinu i rezoluciju i nema uniformnost oblika i struktura.

Validacija rezultata je izvršena preko 4 mjere kvaliteta (metode):

1. PSNR(engl. *Peak Signal to Noise Ratio*)
2. MSSSIM (engl. *Multiscale Structural similarity*)
3. SSIM (engl. *Structural Simularity*)
4. Odnos veličina originala i kompresovanog fajla (za pojedine slike).

U nastavku slijedi detaljan opis pojednih mjera kvaliteta.

4.2.1. PSNR

PSNR (engl. *Peak Signal to Noise Ratio*) predstavlja mjeru koja opisuje kvalitet kompresije slike ili video sadržaja. Pogodna je za upoređivanje originalne slike sa kompresovanim materijalom. Šum se pojavljuje kao posledica kompresije originalne slike. U ovoj funkciji njegovo računanje je predstavljeno preko MSE (engl. Mean Square Error) srednje kvadratne greške.

Već je u glavi o neuralnim mrežama pomenuta srednja kvadratna greška koja se računala

za razliku stvarne i predviđene vrijednosti aktivacije pojedinih izlaznih neurona. Ovdje ona predstavlja srednju razliku između piksela originalne slike i slike dobijene kompresijom na kvadrat. Data je formulom:

$$MSE = \frac{1}{mn}\sum_{\mathrm{i}-0}^{\mathrm{m}-1}\sum_{\mathrm{j}-0}^{\mathrm{n}-1}\|f(i,j) - g(i,j)\|^2 \ ,$$

gdje m i n predstavljaju dimenzije slike u pikselima, $f(i,j)$ originalna vrijednost pojedinačnog piksela a $g(i,j)$ vrijednost pojedinačnog piksela slike dobijene kompresijom koji su na istom položaju u slici.

Pošto signali često imaju veoma širok dinamički opseg (velika je razlika između najtamnih i najsvjetlih tačaka na slici), može doći do stvaranja velikog spektra nivoa inteziteta što čini MSE neprikladnom metodom kontrole kvaliteta.

Ovaj problem se rješava logaritamskom skalom. Izražava se odnos maksimalne vrijednosti piksela sa vrijednošću šuma (srednje kvadratne greške). Visoke razlike između maksimalne vrijednosti piksela i šuma je moguće predstaviti na kompaktniji, bolji način. Posebno je vidno poboljšanje na slikama čiji je raspon između ovih vrijednosti veliki. Formula za računanje PSNR indeksa je:

$$PSNR = 20\log_{10}\left(\frac{Max_f}{MSE}\right),$$

gdje MSE predstavlja srednju kvadratnu grešku, Max_f maksimalnu vrijednost piksela na originalnoj slici.

Na osnovu formule srednje kvadratne greške se može zaključiti da PSNR nema vrijednost kada se upoređuju iste slike. Ova pojava je posledica dijeljenja sa nulom. U drugim slučajevima, PSNR ima relativno mali raspon vrijednosti. Što je PSNR indeks veći to znači da je kvalitet slike dobijene kompresijom veći. Rezultati PSNR metode kontrole kvaliteta za različite kompresione algoritme variraju i za primjere na kojima je rađeno testiranje se drže u rapsonu od 30 dB do 50 dB.

U nastavku je prikazana imlentacija PSNR metode u programskom jeziku Python. U prikazanoj implementaciji funkcija ima dva ulazna argumenta (slike koje se porede).

```
def PSNR(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if(mse == 0):  # MSE je jednaka nuli kada se upoređuju iste slike.
        return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr
```

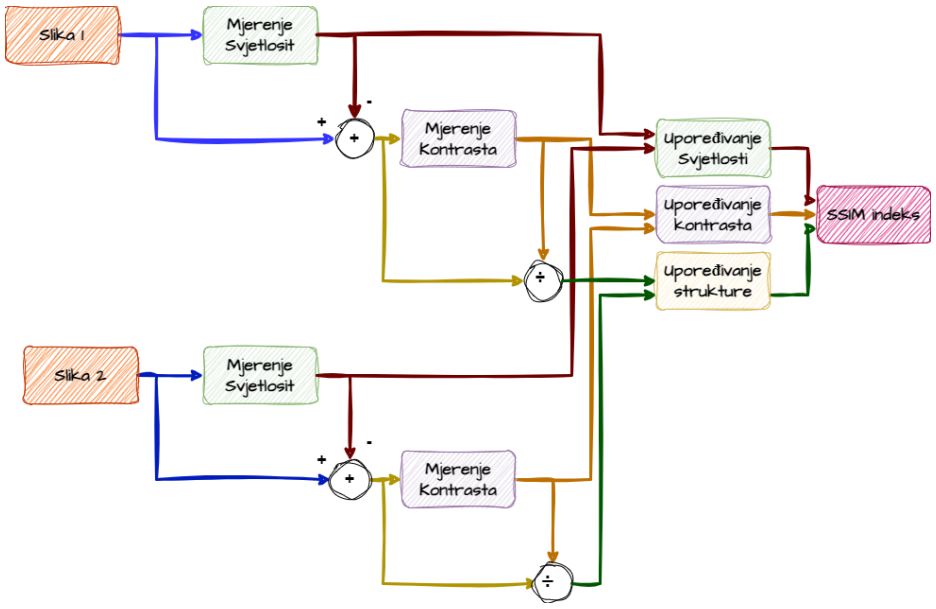
Ova metoda ne oponaša čovjekovu vizuelnu percepciju već računa razliku u signalima. Različiti ljudi na različite načine percipiraju kvalitet putem čula vida, a MSE navedenu distinkciju ne uzima u obzir. To znači da se može desiti da za slike sa lošim vizuelnim kvalitetom, PSNR vrijednost je bolja nego kod slike sa boljim vizuelnim kvalitetom. Metode koje više oponašaju čovjekova čula su strukturne metode MSSSIM i SSSIM.

4.2.2. SSIM

SSIM ili strukturna sličnost predstavlja mjeru sličnosti između dvije slike koja se često koristi za ocjenjivanje kvaliteta određenih algoritama. Predstavljena je 2004. godine od strane IEEE istraživača Wang-a [13] i njegovih kolega. Bazirana je na strukturnoj sličnosti. Pokušava da oponaša čovjekova čula. PSNR kao metrika posmatra i kvantifikuje greške da bi vidjelo koliko je degradirana sama slika. SSIM posmatra promjene ne kao greške prilikom kvantizacije (ili nekog drugog kompresionog procesa) već promjene u strukturnim informacijama tj. promjeni same strukture slike.

Strukturna sličnost koristi tri ključne informacije da bi izačunala razlike u strukturi. To su:

1. Svjetlost (engl. *luminance*)
2. Kontrast (engl. *constrast*)
3. Struktura (engl. *structure*)



Slika 31: SSIM arhitektura

Luminance (svjetlina) predstavlja ukupnu svjetlost i tamu na slici. Računa se kao srednja vrijednost svjetline piksela koji odgovaraju odgovarajućoj komponenti signala (slike). Kao što je već pomenuto, u JPEG kompresiji svjetlina može da ima najveći uticaj na percepciju slike. Ova pojava se bazira na percepciji slika u čovjekovom oku. Nezavisna je od preostale dvije komponente. Računa se kao:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_{\mathrm{i}},$$

gdje μ_{x} predstavlja svjetlinu prve slike, N je broj piksela a x_{i} vrijednost svjetline pojedinačnog piksela. Računa se za obje slike.

Kontrast mjeri standardnu devijaciju između svih vrijednosti piksela na slici. Prije upoređivanja sa drugim signalom (slikom) važno je saznati koliko dobro slika zadržava nivo detalja ili oštrinu. Veći kontrast sugeriše jasnije detalje, bolji kvaliet slike, bolje vidljivu strukrturu slike. Standardna devijacija se računa kao kvadratni korijen varijanse. Standardna devijacija predstavlja mjeru raspršenosti (mjeru kontrasta) između srednje vrijednosti svjetline piksela i trenutne vrijednosti piksela. Zavisna je od komponente svjetline. Data je formulom :

$$\sigma_{\mathrm{x}} = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_{\mathrm{i}} - \mu_{\mathrm{x}})^2\right)^{\frac{1}{2}},$$

gdje σ_{x} predstavlja kontrast prve slike, N broj piksela, μ_{x} već izračunatu srednju vrijednost svjetline piksela za sliku a $\mathrm{x_i}$ vrijednost svjetline pojedinačnog piksela.

Struktura se računa preko standardne devijacije. Vrši se normalizacija signala čime otklanja prekomjeran uticaj razlike u amplitudi dva signala. Za dvije slike za koje se mjeri strukturna sličnost, mogu biti izražene veoma velike razlike komponente svjetlosti u pojednim pikselima što ih čini nezahvalnim za strukturno poređenje. Normalizacija rješava ovaj problem:

$$Normalizacija = (x - \mu_{\mathrm{x}})/\sigma_{\mathrm{x}}\,,$$

gdje x predstavlja ulzani signal (sliku), μ_{x} već izračunatu srednju vrijednost svjetline piksela za sliku a σ_{x} predstavlja kontrast prve slike.

Nakon posebnog predstavljanja tri važne komponente signala (pojedinačne slike), potrebno je kombinovati signale da bi se izvršilo poređenje. SSIM predstavlja:

$$S(\mathbf{x},\mathbf{y}) = f(l(\mathbf{x},\mathbf{y}), c(\mathbf{x},\mathbf{y}), s(\mathbf{x},\mathbf{y})),$$

gdje $\mathrm{S}(\mathbf{x},\mathbf{y})$ predstavlja SSIM, strukturnu sličnost između dva signala (dvije slike **x** i **y**), $l(\mathbf{x},\mathbf{y})$ je svjetlost (luminanca) između dvije slike, $c(\mathbf{x},\mathbf{y})$ kontrast i $s(\mathbf{x},\mathbf{y})$(napisano malim slovom) predstavlja strukturni odnos između dvije slike.

Sada je potrebno pojedinačno definisati sve komponente preko ulaznih signala **x** i **y**. Komponenta upoređivanja svjetlosti $l(\mathbf{x},\mathbf{y})$ je data formulom:

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$

gdje μ_x, μ_y predstavljaju srednju vrijednost svjetline obje slike, C_1 konstantu koja je data formulom:

$$C_1 = (K_1 L)^2,$$

gdje K_1 predstavlja slućajno odabranu konstantu (jedan od argumenata u implementaciji SSIM u TensorFlow biblioteci) i L maksimalni raspon za piksele (max_val argument u TensorFlow implementaciji za osmobitne slike 255).

Komponenta koja odgovara kontrastu dvije slike **x** i **y** $c(\mathbf{x},\mathbf{y})$, je data formulom:

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_{\mathrm{x}}\sigma_{\mathrm{y}} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$

gdje $\sigma_{\mathrm{x}}, \sigma_{\mathrm{y}}$ predstavljaju standardnu devijaciju obje slike **x** i **y**, C_2 konstantu koja je data formulom:

$$C_2 = (K_2 L)^2,$$

gdje K_2 predstavlja slućajno odabranu konstantu (jedan od argumenata u implementaciji SSIM u TensorFlow biblioteci) i L maksimalni raspon za piksele (max_val argument u TensorFlow implementaciji za osmobitne slike 255). Konstante C_1 i C_2 služe da bi se izbjeglo dijeljenje

nulom.

Treća komponenta, strukuturna vrijednost između dva ulazna signala **x** i **y** je opisana formulom:

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_y \sigma_x + \mathrm{C}_3},$$

gdje σ_x, σ_y predstavljaju standardnu devijaciju slika **x** i **y** a slike σ_{xy} se računa kao:

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)\big(y_i - \mu_y\big),$$

gdje x_i, y_i predstavljaju vrijednost pojedinačnog piksela slika **x** i **y**, N broj piksela a μ_x, μ_y vrijednosti svjetline.

Krajnja formula za izračunavanje SSIM indeksa glasi:

$$S(\mathbf{x},\mathbf{y}) = \big([l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma}\big).$$

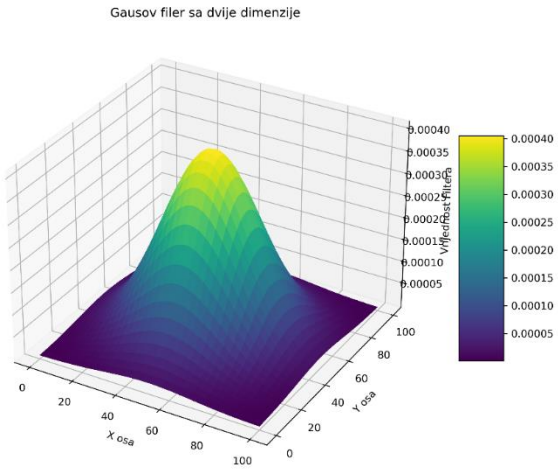
Vrijednosti α, β, γ se odnose na stepen važnosti pojedinih komponenti. Ako se postavi vrijednost važnosti sva tri stepena važnosti na 1, a računa $C_3 = C_2/2$ dobija se uprošćena formula:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{\big(2\mu_x\mu_y + C_1\big)\big(2\sigma_{xy} + C_2\big)}{\big({\mu_x}^2 + {\mu_y}^2 + C_1\big)\big({\sigma_x}^2 + {\sigma_y}^2 + C_2\big)},$$

gdje C_1, C_2, C_3 predstavljaju konstante, $\sigma_x, \sigma_y, \sigma_{xy}$ predstavljaju standardnu devijaciju, a μ_x, μ_y predstavlja svjetlinu dva signala **x** i **y**.

Prethodna formula podrazumijeva globalnu implementaciju odnosno implementaciju na cijeloj slici. Primijećeno je da distorzije u strukturama slika su najčešće lokalne prirode i da se mogu nalaziti na različitim mjestima u prostoru, da prilikom posmatranja slike kao cjeline čovjek se fokusira ne na cijelu sliku već na lokalne karakteristike. Ljudski vid nikada neće davati isti značaj svakom pikselu na slici. Cilj je uhvatiti lokalne aspekte i preciznije ocijeniti kvalitet slike.

Navedeno je postignuto uvođenjem Gausovog filtera dimenzija (11×11) sa standardnom devijaciom Gausove fukcije od 1.5. Tim filtrom je moguće bolje akcentovati lokalne vrijednosti SSIM indeksa, izračunati ih za svaki piksel, sabrati i podijeliti sa brojem piksela. Ovim postupkom dobija se MSSIM odnosno srednja vrijednost strukturne sličnosti. Ona u odnosu na globalnu vjerodostojnije i bolje računa sličnost između dvije slike.



Slika 32: Gausov filter

Ažurirane formule SSIM (**x**,**y**) indeksa signala **x** i **y** su date ispod:

$$\mu_x = \sum_{i-1}^{N} w_i x_i \,,$$

$$\sigma_x = \left(\sum_{i=1}^{N} w_i (x_i - \mu_x)^2 \right)^{\frac{1}{2}},$$

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (x_i - \mu_x)\big(y_i - \mu_y\big),$$

gdje w_i predstavlja težinski faktor iz Gausovog filtera a ostale vrijednosti su već dobro pojašnjene.

Formula za MSSIM (srednju SSIM vrijednost):

$$\mathrm{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{SSIM}(\mathbf{x}_j \,, \boldsymbol{y}_j),$$

gdje MSSIM i SSIM predstavljaju srednju strukturnu vrijednost i strukturnu vrijednost, *M* broj lokalnh prozora, **X**,**Y**, predstavljaju ulazne signale (slike) a $\mathbf{x}_j, \mathbf{y}_j$ odgovarajuće lokalne prozore. Računa se suma strukturnih vrijednosti za svaki prostor i nađe joj se srednja vrijednost.

Važno je naglasiti da SSIM kao metrika radi isključivo sa sivoskaliranim slikama. JPEG kompresija ima istu komponentu svjetlosti ili osvjetljaja (engl. *luminance*). To čini SSIM veoma korisnim alatom za upoređivanje slika dobijenih kompresijom putem JPEG algoritma.

Većina vrijednosti koje će biti proslijeđene SSSIM funkciji su konstantne. K1 i K2 imaju vrijednosti 0.01 i 0.03 respektivno i produkt su eksperimentalnih rezultata. Gausov filter pored vrijednosti 11 može se postaviti na vrijednost 15, dok parametar sigma koji predstavlja

standardnu Gausovu deviajciju može biti postavljen na vrijednost 1 ili 1.5. Da bi rezultati bili validni, slika mora biti pretvorena u sivoskaliranu sliku. Implementacija SSIM indeksa prikazana je u kodu ispod.

```
def ssim(original,compressed):
     # citanje i dekodiranje slika
    slika1 = tf.io.read_file(original)
    slika1 = tf.image.decode_png(slika1, channels=3)
    slika2 = tf.io.read_file(compressed)
    slika2 =tf.image.decode_png(slika2, channels=3)
    # Konvertovanje u crno bijelu sliku
    y_slika1 = tf.image.rgb_to_grayscale(slika1)
    y_slika2 = tf.image.rgb_to_grayscale(slika2)
    ssim=tf.image.ssim(
            y_slika1,
            y_slika2,
            max_val=255,
            filter_size=11,
            filter_sigma=1.5,
            k1=0.01,
            k2=0.03,
            return_index_map=False
        )
    return ssim.numpy()
```

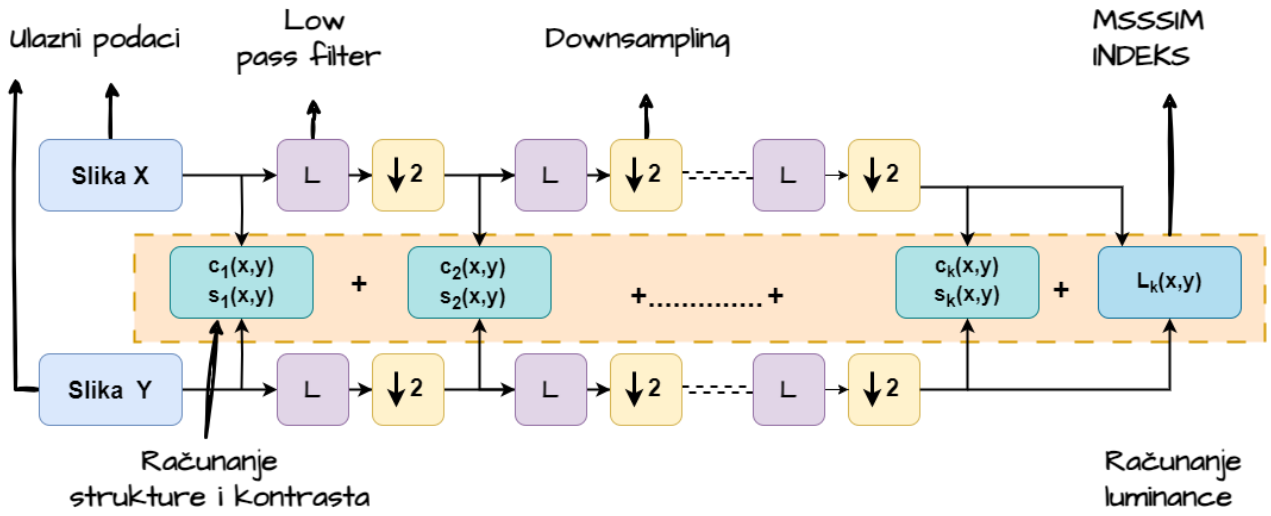
4.2.3. MSSSIM

Još jedna mjera kvaliteta korišćena za ocjenjivanje razlika između originalnih i slika dobijenih putem različitih kompresionih algoritama. Predstavljena je od iste grupe naučnika kao i SSIM metoda. Ideja iza MS-SSIM funkcije je da se koriste više skaliranih verzija slike da bi se još bolje predstavio ljudski vid i percepcija. Kao što je već pomenuto SSIM uz pomoć Gausovog filtera pravi „prozor“ na slici na kojem se računa SSIM vrijednost. Sumirajem vrijednosti SSIM indeksa svakog prozora i dijeljenjem sa brojem prozora izračunava se vrijednost srednjeg strukturnog faktora. Kod MSSSIM indeksa slika će biti podijeljena na više nivoaili skala. Ovaj proces je potreban da bi se uzeo u obzir različit nivo detalja prilikom računanja MSSSIM indeksa. Korišćenjem na primjer različitih rezolucija slike dobija se efekat različite udaljenosti koji više liči na percepciju čovjeka. Neće svaki čovjek na isti način posmatrati sliku, što se želi simulirati ovim pristupom. Sitni detalji će u svakom stepenu skale biti manje izraženi. Ovim postupkom simulira se promjena rastojanja čovjeka i dvije slike kompresovane i originalne.

Arhitektura MSSSIM mnozenja signalima ima dva veoma važna koraka:

[1]. Smanjenje rezolucije (engl. *downsampling*). Slika se skalira na nižu rezoluciju. Ovim procesom različite strukture i oblici će biti bolje izaženi. MSSSIM na slikama manje rezolucije se može bolje baviti strukturnim i kontrasnim detaljima na slici. U arhitekturi MSSSIM koja je predstavljena od strane Wang-a vrši se smanjenje rezolucije za faktor 2 u svakom koraku. To znači da će rezolucija u sledećem koraku imati duplo manju x i y dimenzije (smanjenje će biti izvršeno u oba pravca).

[2]. Niskopropusno filtriranje (engl. *low pass filter*) će takođe biti primijenjeno na obje slike, originalnu i sliku dobijenu kompresijom. Ideja da se smanji kontrast ili oštrina prelaza na određenim slikama prije nego što se izvrši smanjenje rezolucije. Ovim procesom se otklanjaju viskofrekventne promjene u signalu (slici). Odnosno na slici se postiže efekat gladjenja (engl. *smoothing effect*). Primjenom ovog filtera se mogu i ukloniti nepravlnosti na slikama (na primjer licu osobe) pa je u eri pametnih telefona i društvenih mreža veoma popularan.



Slika 33: Arhitektura MSSSIM mjere kvaliteta

Arhitektura prikazana na slici 33 je data ispod:

$$\mathrm{MSSSIM}(\mathbf{x},\mathbf{y}) = [l_m(\mathbf{x},\mathbf{y})]^{\alpha_m} \times \prod_{k=1}^{m} [c_k(\mathbf{x},\mathbf{y})]^{\beta_k} \times [s_k(\mathbf{x},\mathbf{y})]^{\gamma_k},$$

gdje $\mathbf{x}$ i $\mathbf{y}$ predstavljaju ulazne signale (slike), $[l_m(\mathbf{x},\mathbf{y})]^{\alpha_m}$ predstavlja svjetlost i računa se samo za poslednji korak ili poslednju skalu, $[c_k(\mathbf{x},\mathbf{y})]^{\beta_k}$ predstavlja kontrast koji se računa na svakom

koraku, $[s_k(\mathbf{x},\mathbf{y})]^{\gamma_k}$ strukturu u svakom koraku ili skali, $\gamma_k, \beta_k, \alpha_m$ predstavlja koeficijent koji odeđuje važnost komponenti a *m* predstavlja broj koraka ili koliko će puta biti skalirana slika.

Luminance se samo koristi za poslednju skalu ili korak i to je vidno na slici 33 arhitekture. Prilikom izračunavanja SSIM vrijednosti, koeficijenti važnosti su imali vrijednost broja 1. U MSSSIM metodi, koeficijenti važnosti zarad lakše računice imaju iste vrjednosti u svakom koraku ili skali. Važi da je $\gamma_k = \beta_k = \alpha_k$. Eksperimentalnim računanjem za broj skala 5 na velikom broju slika dobijaju se sledeće vrijednosti za pojedine koeficijente:

$$\beta_1 = \gamma_1 = 0.044,\ \beta_2 = \gamma_2 = 0.2856, \beta_2 = \gamma_2 = 0.3001, \beta_4 = \gamma_4 = 0.2363, \alpha_5 = \beta_5 = \gamma_5 = 0.1333$$

Ove vrijednosti, zajedno sa vrijednostima koje su korišćenee kao argumenti SSIM vrijednosti se šalju kao argument funkciji iz biblioteke TensorFlow. U nastavku je data implementacija msssim funkcije unutar TensorFlow biblioteke.

```
def msssim(original,compressed):
    slika1 = tf.io.read_file(original)
    slika1 = tf.image.decode_png(slika1, channels=3)
    slika2 = tf.io.read_file(compressed)
    slika2 = tf.image.decode_png(slika2, channels=3)
    # Konvertovanje u crno bijelu sliku
    y_slika1 = tf.image.rgb_to_grayscale(slika1)
    y_slika2 = tf.image.rgb_to_grayscale(slika2)
    ms=tf.image.ssim_multiscale(
        y_slika1,
        y_slika2,
        max_val=255,
        power_factors=[0.0448, 0.2856, 0.3001, 0.2363, 0.1333],
        filter_size=11,
        filter_sigma=1.5,
        k1=0.01,
        k2=0.03)
    return ms.numpy()
```

5. Eksperimentalni rezultati

U ovom poglavlju će biti predstavljeni rezultati eksperimentalne analize. U metodama validacije je naglašeno da će biti prikazano poređenje sa slikama u .jpeg formatu. Slike su na početku transformisane u slike različitog kvaliteta. Pored slika dobijenih posredstvom TensorFlow Compression biblioteke izrađene su sledeće slike:

1) Slike .jpeg formata kvaliteta 10
2) Slike .jpeg formata kvaliteta 20
3) Slike .jpeg formata kvaliteta 30
4) Slike .jpeg formata kvaliteta 40
5) Slike .jpeg formata kvaliteta 50
6) Slike .jpeg formata kvaliteta 60
7) Slike .jpeg formata kvaliteta 70
8) Slike .jpeg formata kvaliteta 80

Pored slika dobijenih metodama mašinskog učenja za upoređivanje postoje još 3200 slika .jpeg formata. Više nivoa kvaliteta slika .jpeg formata je upoređivano sa slikama dobijenim kompresijom. U radu nije predstavljen samo jedan nivo kvaliteta zbog osobenosti da određeni TensorFlow Compression modeli posjeduju više nivoa kvaliteta. Za određene slike dobijene TensorFlow Compression bibliotekom važi:

1) "Joint Autoregressive and Hierarchical Priors for Learned Image Compression"
    a. `mbt2018-mean-mse-[1-8]` – osam nivoa kvaliteta
    b. `mbt2018-mean-msssim-[1-8]` – osam nivoa kvaliteta
2) "Variational Image Compression with a Scale Hyperprior" model
    a. `bmshj2018-factorized-mse-[1-8]` – osam nivoa kvaliteta
    b. `bmshj2018-factorized-msssim-[1-8]` – osam nivoa kvaliteta
    c. `bmshj2018-hyperprior-mse-[1-8]` – osam nivoa kvaliteta
    d. `bmshj2018-hyperprior-msssim-[1-8]` – osam nivoa kvaliteta
3) "Efficient Nonlinear Transforms for Lossy Image Compression" model
    a. `b2018-leaky_relu-128-[1-4]` – četiri nivoa kvaliteta
    b. `b2018-leaky_relu-192-[1-4]` – četiri nivoa kvaliteta
    c. `b2018-gdn-128-[1-4]` – četiri nivoa kvaliteta
    d. `b2018-gdn-192-[1-4]` – četiri nivoa kvaliteta

Uzorak na kojem je vršen eksperiment se sastoji od 400 slika iz dataset-a DIV2K. Radi validnosti samih rezultata, originalne slike su .png formatu. Ovim se izbjegava prva kompresija (kompresija iz .jpeg formata u .png format koji zahtjeva TensorFlow Compression biblioteka). Od 400 slika se dobijaju 25600 slika koje predstavljaju različite modele i, kao što je već pomenuto, 3200 slika .jpeg formata.

Slučajno je odabrna slika patke kao primjer uspješne kompresije jednog modela i jednog .jpeg kvaliteta. Na slici 34 je predstavljena slika prije kompresije, na slici 35 je prikazana kompresija korišćenjem `b2018-leaky_relu-128-1` modela a na slici 36 je slika koja odgovara `.jpeg 10` kvalitetu.

Slika 34: Primjer slike iz razmatranog dataset-a

Slika 35: Slika koja odgovara `b2018-leaky_relu-128-1` dekompresiji slike

Slika 35 sadrži glatke ivice i nema naglih promjena u boji. Vizuelno ona veoma dobro

predstavlja original ali neki detalji (šare ili slamke u pozadini) su manje naglašeni. Posmatrač mora da se potrudi da uoči veće razlike između slika 34 i 35.

Slika 36: Slika koja odgovara `.jpeg 10` kvalitetu

Na slici 36 je vidna pojava blokova, kao neželjeni efekat JPEG algoritma. Efektom glađenja slika 35 zadržava dovoljno veliki nivo detalja i izbjegava pojavu blokova. Može se zaključiti da slika 35 je bolje vizuelno rješenje u odnosu na sliku 36 što se kompesije tiče.

Slika 35 predstavlja samo jednu od 10 slika (nivoa 1) koje su dobijene korišćenjem TensorFlow-Compression biblioteke. U tabelama 1 i 2 su prikazane vrijednosti mjera kvaliteta svih slika koje su dobijene kompresijom slike 34 za vrijednost argumenta 1 (odgovara prvom nivou).

Ime modela	Jpeg 10	mbt2018 -mean- mse-1	mbt2018- mean- msssim-1	bmshj20- factorized -mse-1	bmshj20- factorized- msssim-1	bmshj201- hyperprior- mse-1
PSNR(dB)	32.740	35.839	35.602	35.307	31.971	34.03
SSIM	0.907	0.944	0.955	0.943	0.949	0.907
MSSSIM	0.936	0.969	0.979	0.970	0.975	0.959
Veličina(kB)	60.61	12.185	25.59	22.63	19.90	15.83

Tabela 1: Mjere kvaliteta za slike dobijene kompresijom slike 34 i argumentom 1 (nivo kvaliteta 1)

Ime modela	Jpeg 10	bmshj2018- hyperprior- msssim-1	b2018- leaky_relu- 128-1	b2018- leaky_relu- 192-1	b2018- gdn-128- 1	b2018- gdn-192- 1
PSNR(dB)	32.740	32.99	34.950	35.060	35.070	35.151

SSIM	0.907	0.949	0.938	0.939	0.939	0.941
MSSSIM	0.936	0.975	0.968	0.969	0.967	0.968
Veličina(kB)	60.61	16.912	23.185	22.811	23.88	23.55

Tabela 2: Mjere kvaliteta za slike dobijene kompresijom slike 34 i argumentom 1 (nivo kvaliteta 1)

Već na osnovu predstavljenih tabela 1 i 2 je moguće primijetiti određene zavisnosti. Modeli koji su optimizovani za kvadratnu grešku (sadrže argument MSE u svom nazivu) imaju veće vrijednosti PSNR indeksa u odnosu na modele koji su optimizovani za MSSSIM metriku. Za rezultate prikazane u tabeli 1, se vidi da je vrijednost `mbt2018-mean-mse-1-PSNR` indeksa veća od vrijednosti `mbt2018-mean-msssim-1-PSNR` indeksa (`mbt2018-mean-mse-1` i `mbt2018-mean-msssim-1` predstavljaju slike dobijene kompresijom slike 34). Obrnuta tvrđenja takođe važe. U tabeli 1, `mbt2018-mean-mse-1-MSSSIM` indeks ima veću vrijednost nego `mbt2018-mean-mse-1-MSSSIM`. Generalno za sve slike će važiti tvrđenja:

`Mbt2018-mean-mse-1-PSNR` veće od `mbt2018-mean-mssim-1-PSNR`,

`Mbt2018-mean-mse-1-MSSSIM` manje od `mbt2018-mean-mssim-1-MSSSIM`,

`bmshj2018-factorized-mse-1-PSNR` veće od `bmshj2018-factorized-msssim-1-PSNR`,

`bmshj2018-factorized-mse-1-MSSSIM` manje od `bmshj2018-factorized-msssim-1-MSSSIM`,

`bmshj2018-hyperprior-mse-1-PSNR` veće od `bmshj2018--hyperprior-msssim-1-PSNR`,

`bmshj2018-hyperprior-mse-1-MSSSIM` manje od `bmshj2018-hyperprior-msssim-1-MSSSIM`.

Ova tvrđenja će dalje pomoći pri odabiru modela prilikom kompresije. Ako je za kompresiju određene slike potrebno izabrati model koji vodi računa o vizuelnoj percepciji biće izabran model sa argumentom msssim. Obratno modeli koji sadrže argument mse će biti izabrani za slučaj kada se više daje značaj razlici između pojedinih piksela slika (razlici između signala). Navedena tvrđenja će odgovarati i za veće nivoe kompresije. U tabelama 3 i 4 će biti prikazane vrijednosti nivoa kvaliteta 4 za sliku 34.

Ime modela	Jpeg 40	mbt2018-mean-mse-4	mbt2018-mean-msssim-4	bmshj201-factorized-mse-4	bmshj201-factorized-msssim-4	bmshj201-hyperprior-mse-4
PSNR(dB)	39.149	39.870	40.153	39.251	38.899	39.251
SSIM	0.970	0.971	0.978	0.967	0.989	0.965

MSSSIM	0.989	0.989	0.994	0.989	0.993	0.988
veličina(kB)	106.9	33.257	52.191	67.04	74.803	38.076

Tabela 3: Mjere kvaliteta za slike dobijene kompresijom slike 34 i argumentom 4 (nivo kvaliteta 4)

Ime modela	Jpeg 40	bmshj2018-hyperprior-msssim-4	b2018-leaky_relu-128-4	b2018-leaky_relu-192-4	b2018-gdn-128-4	b2018-gdn-192-4
PSNR(dB)	39.149	40.104	42.173	42.31	43.43	43.63
SSIM	0.970	0.979	0.980	0.981	0.985	0.985
MSSSIM	0.989	0.994	0.995	0.995	0.996	0.997
veličina(kB)	106.9	51.420	183.353	195.505	181.832	196.996

Tabela 4: Mjere kvaliteta za slike dobijene kompresijom slike 34. i argumentom 4 (nivo kvaliteta 4)

Ovdje su ponegdje vidna i odstupanja od navedenih trvrđenja. U tabeli 4 `mbt2018-mean-mse-4-PSNR` vrijednost indeksa je manja u odnosu na `mbt2018-mean-msssim-4-PSNR` vrijednost indeksa. Ipak u navedenom primjeru se radi samo o jednoj slici gdje struktura ili način minimizacije pojedinih funkcija u posebnim modelima mogu uticati na krajnje rezultate.

Zbog toga će biti na cijelom setu slika vršena usrednjavanja svih vrijednosti. Odnosno biće tabelarno prikazane srednje vrijednosti pojedinih indeksa na svih 400 slika na kojima je rađen eksperiment. Ovaj proces treba da pokaže stvarnu vrijednost rezultata. Zavisno od modela biće prikazano 8 tabela vrijednosti za modele i jedna tabela sa svim vrijednostima za slike .jpeg formata. Modeli koji u sebi sadrže više slojeva i imaju po četiri nivoa kvaliteta su prikazani zajdeno.

	b2018-leaky_relu-128				b2018-leaky_relu-192			
avg	1	2	3	4	1	2	3	4
psnr	31.65371	32.99993	34.59738	35.89752	31.68925	33.13391	34.76205	36.96606
ssim	0.774874	0.858816	0.926296	0.951785	0.780216	0.869638	0.930424	0.964909
msssim	0.931927	0.967818	0.986825	0.993197	0.934207	0.970555	0.987146	0.994531

Tabela 5: Srednje vrijednosti kontrole kvaliteta za `b2018-leaky_relu-128-[1-4]` i `b2018-leaky_relu-192-[1-4]` modele

	b2018-gdn-128				b2018-gdn-192			
avg	1	2	3	4	1	2	3	4
psnr	31.7126	33.2171	34.9722	36.3950	31.7538	33.1870	35.0952	37.4179
ssim	0.78113	0.87020	0.93341	0.95812	0.78372	0.87035	0.93623	0.96903
msssim	0.93337	0.97042	0.98831	0.99452	0.93430	0.97031	0.98836	0.99547

Tabela 6: Srednje vrijednosti kontrole kvaliteta za `b2018-gdn-128-[1-4]` i `b2018-gdn-192-[1-4]` modele

bmshj2018-hyperprior-mse								
avg	1	2	3	4	5	6	7	8
psnr	31.61517	32.6514	33.5577	34.37859	35.61343	37.14352	38.82138	40.41263
ssim	0.793051	0.853254	0.891442	0.917213	0.942514	0.959555	0.971284	0.980729
msssim	0.940159	0.961466	0.974897	0.982404	0.988998	0.99297	0.995363	0.997028

Tabela 7: Srednje vrijednosti kontrole kvaliteta za `bmshj2018-hyperprior-mse-[1-8]` model

bmshj2018-hyperprior-msssim								
avg	1	2	3	4	5	6	7	8
psnr	30.92954	32.26441	33.39054	34.3198	35.52372	36.72616	38.34787	40.07458
ssim	0.795975	0.851898	0.895565	0.918115	0.950766	0.968836	0.980475	0.988005
msssim	0.9474	0.970129	0.982899	0.989045	0.993668	0.996321	0.997822	0.998683

Tabela 8: Srednje vrijednosti kontrole kvaliteta za `bmshj2018-hyperprior-msssim-[1-8]` model

bmshj2018_factorized_mse								
avg	1	2	3	4	5	6	7	8
psnr	31.84046	32.56421	33.32284	34.04044	35.1641	36.36012	37.96855	39.79587
ssim	0.789147	0.834304	0.876289	0.906597	0.935473	0.954816	0.970114	0.979822
msssim	0.936952	0.957296	0.972715	0.981424	0.988333	0.992125	0.995282	0.997019

Tabela 9: Srednje vrijednosti kvaliteta za bmshj2018_factorized_mse-`[1-8]` model

bmshj2018-factorized-msssim								
avg	1	2	3	4	5	6	7	8
psnr	30.43468	31.85617	32.87091	33.91257	34.99292	36.06383	37.41736	38.90055
ssim	0.778316	0.831134	0.875932	0.910691	0.937085	0.958938	0.973745	0.985166
msssim	0.938593	0.962544	0.978067	0.987026	0.99232	0.9954	0.997236	0.99837

Tabela 9: Srednje vrijednosti kontrole kvaliteta za `bmshj2018-factorized-msssim-[1-8]` model

Sada je lako pratiti prije navedene zakonitosti. Pošto se u radu dataset sastoji od mnogo slika različite veličine, odnos srednjih veličina nema značaj, pa je u tabelama izostavljen. Preostale su još dvije tabele sa vrijednostima pojedinih modela i jedna tabela sa vrijednostima JPEG modela.

mbt2018-mean-mse								
avg	1	2	3	4	5	6	7	8
psnr	32.26693	32.97018	33.79051	34.72249	35.92059	37.45174	39.06229	40.76868
ssim	0.793569	0.853075	0.895479	0.925057	0.945893	0.962161	0.973607	0.981958
msssim	0.939853	0.962857	0.976049	0.984538	0.989717	0.993561	0.995787	0.997288

Tabela 10: Srednje vrijednosti kontrole kvaliteta za `mbt2018-mean-mse-[1-8]` model

mbt2018-mean-msssim								
avg	1	2	3	4	5	6	7	8
psnr	32.1680	32.841	33.7525	34.3673	35.5403	36.8048	38.3485	39.9424
ssim	0.80336	0.8362	0.90093	0.91530	0.95285	0.96893	0.98063	0.98745
msssim	0.95065	0.96580	0.9838	0.98892	0.99395	0.99632	0.99778	0.99860

Tabela 11: Srednje vrijednosti kontrole kvaliteta za `mbt2018-mean-msssim-[1-8]` model

Za predstavljanje rezultata ostale su samo vrijednosti vezane za slike dobijene preko JPEG algoritma.

JPEG								
avg	10	20	30	40	50	60	70	80
psnr	31.0254	32.569	33.3652	33.9066	34.3451	34.7419	35.3247	39.9424
ssim	0.8189	0.882	0.9094	0.9241	0.9344	0.9431	0.9531	0.9874
msssim	0.9433	0.974	0.9838	0.9881	0.9906	0.9924	0.9943	0.9986

Tabela 12: Srednje vrijednosti kontrole kvaliteta koje odgovaraju JPEG kompresiji (10-80 kvaliteta)

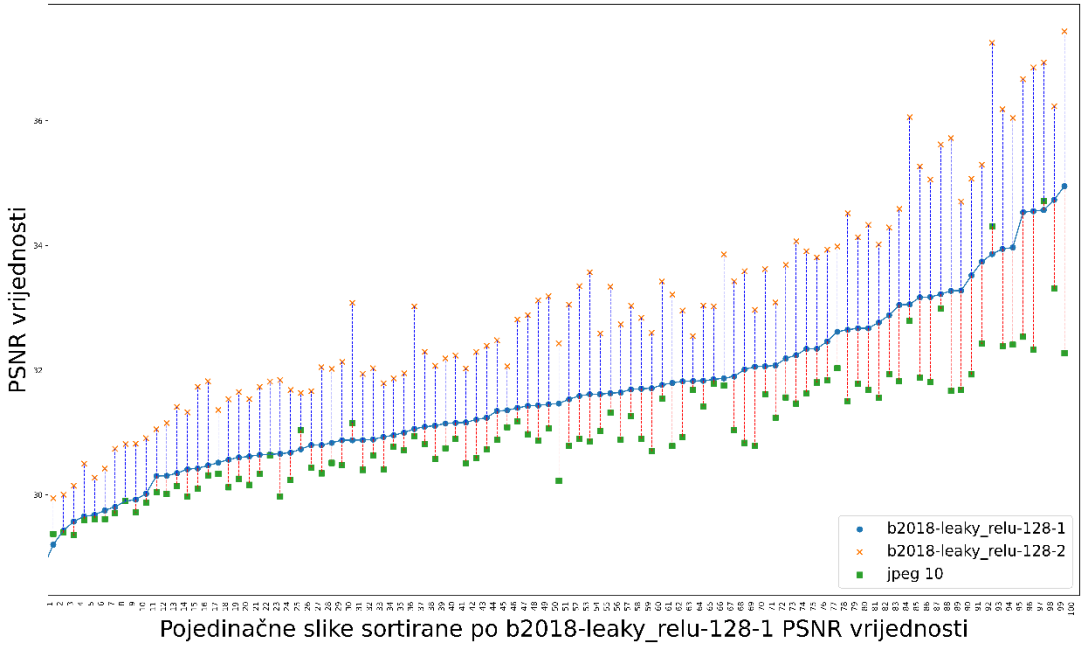
Sada se na velikom uzorku mogu primijetiti neke interesantne osobine modela. Kada je model optimizovan za srednju kvadratnu grešku on će generalno imati veće PSNR vrijednosti nego JPEG nivo sa kojim se upoređuje. Odnosno za slike `.jpeg 30-PSNR` srednja vrijednost će biti manja u odnosu na slike `bmshj2018-hyperprior-mse-3-PSNR` i `mbt2018-mean-mse-PSNR` vrijednost. Isto važi i za `.jpeg 50-MSSSIM` vrijednosti. Modeli `mbt2018-mean-msssim-5`, `bmshj2018-factorized-msssim-5` i `bmshj2018-hyperprior-msssim-5` imaju srednju vrijednost indeksa MSSSIM veću nego slike `.jpeg 50`. U slučajevima kada dolazi do manjih odstupanja od ovih pravila, odstupanja su veoma mala. Za slike `.jpeg 40- MSSSIM` vrijednost i `bmshj2018-factorized-msssim-4-MSSSIM` vrijednost, razlika je 0.001.

Važno je naglasiti da modeli mogu biti na više načina primijenjeni. Ništa korisnika ne sprječava da slike dobijene preko JPEG algoritma sa bilo kojim kvalitetom zamijeni slikom dobijenom nekim većim indeksom.

Za prošli primjer patke prikazane na slikama 34, 35 i 36, vidno je da su rezultati za bilo koji od modela kvalifikativno bolji u odnosu na osnovne `.jpeg 10` rezultate. Može se primijetiti da memorija potrebna za skladištenje dekompresovanih fajlova je manja od polovine memorije potrebne za čuvanje .jpeg fajla. Dakle, moguće je na efektivan način zamijeniti sliku dobijenu preko JPEG kompresije slikom dobijenom preko drugog modela `(mbt2018-mean-mse-2)`. To dalje dovodi do zaključka da se svi modeli mogu koristiti u paru sa različitim

nivoima JPEG kompresije za određeni slučaj.

U nastavku će biti vizuelizovani odabrani rezultati. Zarad preglednije vizuelizacije je slučajno odabrano prvih 100 slika. Moguće je bilo odabrati i više slika, ali pojedine slike u tom slučaju nebi došle do izražaja i grafik nebi bio pregledan. U radu je predstavljeno 10 različitih modela i JPEG algoritam. Dakle, za svaku mjeru kvaliteta (PSNR, msssim, ssim, veličina) je moguće direktno uporediti slike dobijene preko kompresionih modela i slike dobijene preko JPEG algoritma. Ako se još uvedu i nivoi kvaliteta, moguće je predstaviti mnogo grafika koji prikazuju osobine modela.

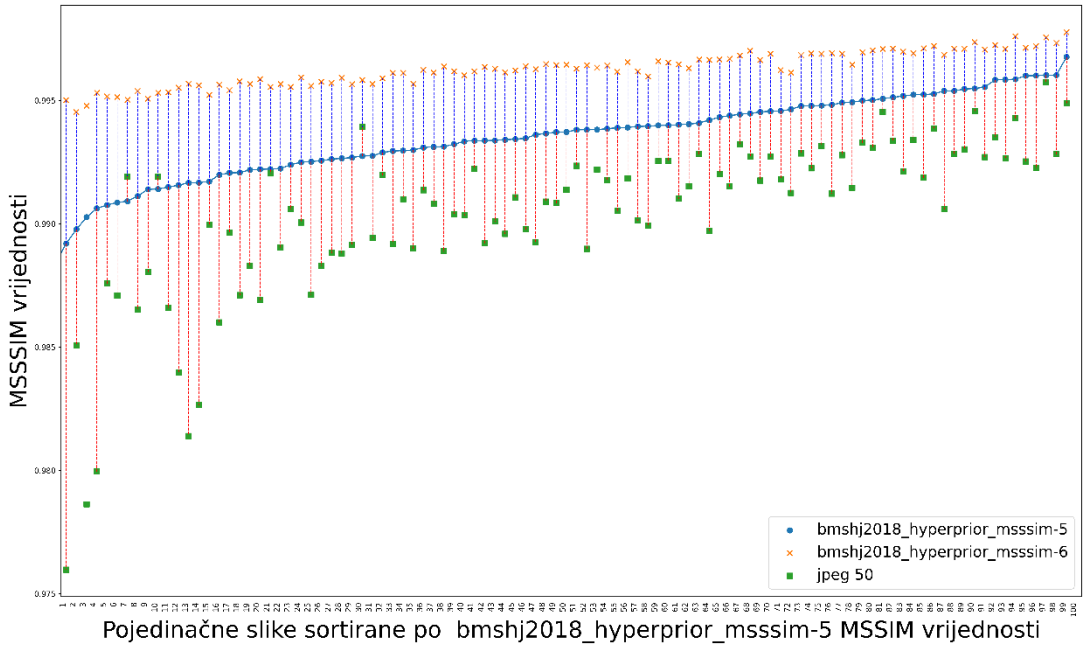


Slika 37: PSNR za `.jpeg 10` slike, `b2018-leaky_relu-128-1` i `b2018-leaky_relu-128-2`

Na slici 37 su prikazane zavisnosti slika pojednih modela i slika `.jpeg 10` kvaliteta. `b2018-leaky_relu-128-1` model je predstavljen kružićom plave boje, `.jpeg 10` je predstavljen kvadratićom zelene boje a `b2018-leaky_relu-128-2` model je predstavljen krstićom žute boje. Za veliki broj slika vrijednosti PSNR `b2018-leaky_relu-128-1` modela ima manju vrijednost u odnosu na vrijednosti PSNR za `.jpeg 10` kvalitet. Moguća su manja odstupanja, ali su ona veoma rijetka. U predstavljenom primjeru 95 `.jpeg 10` slika (od 100) je imalo manju PSNR vrijednost u odnosu na `b2018-leaky_relu-128-1` slike.

Na slici 38 model `bmshj2018_hyperprior_msssim-5` (predstavljen plavim kružićima) ima bolje vrijednosti u odnosu na `.jpeg 50` kvalitet (predstavljen zelenim

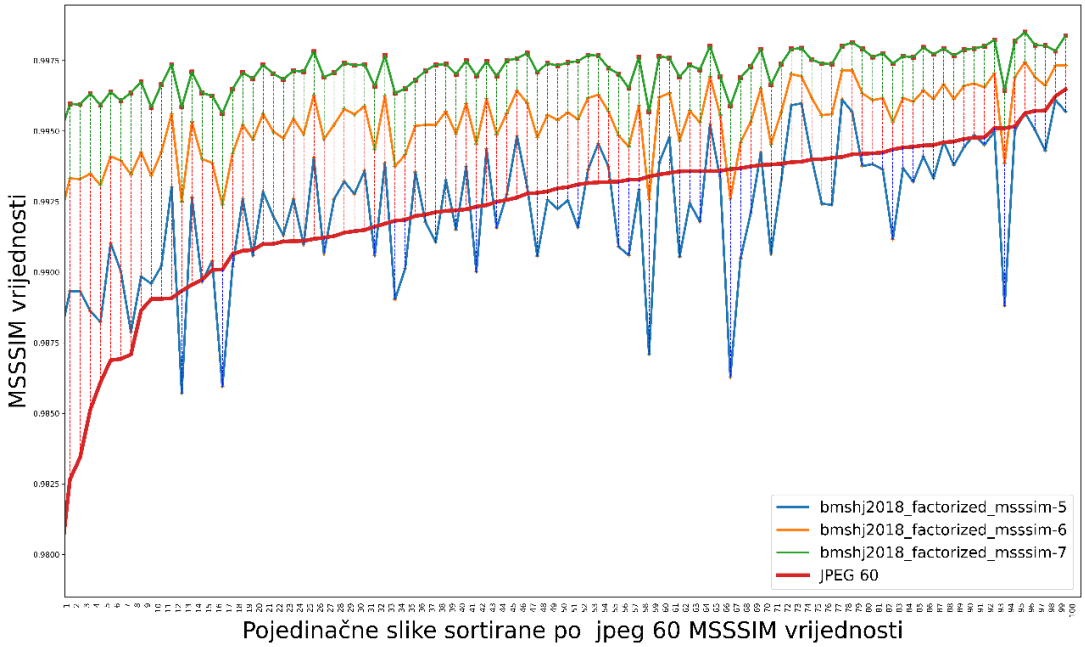
kvadratićima). Radi preglednosti grafika u prikazu je postavljeno ograničenje od 100 slika. Sada u 97 slučajeva (od 100) `.jpeg 50` slike imaju manje vrijednosti MSSSIM indeksa u odnosu na slike `bmshj2018_hyperprior-msssim-5` modela.



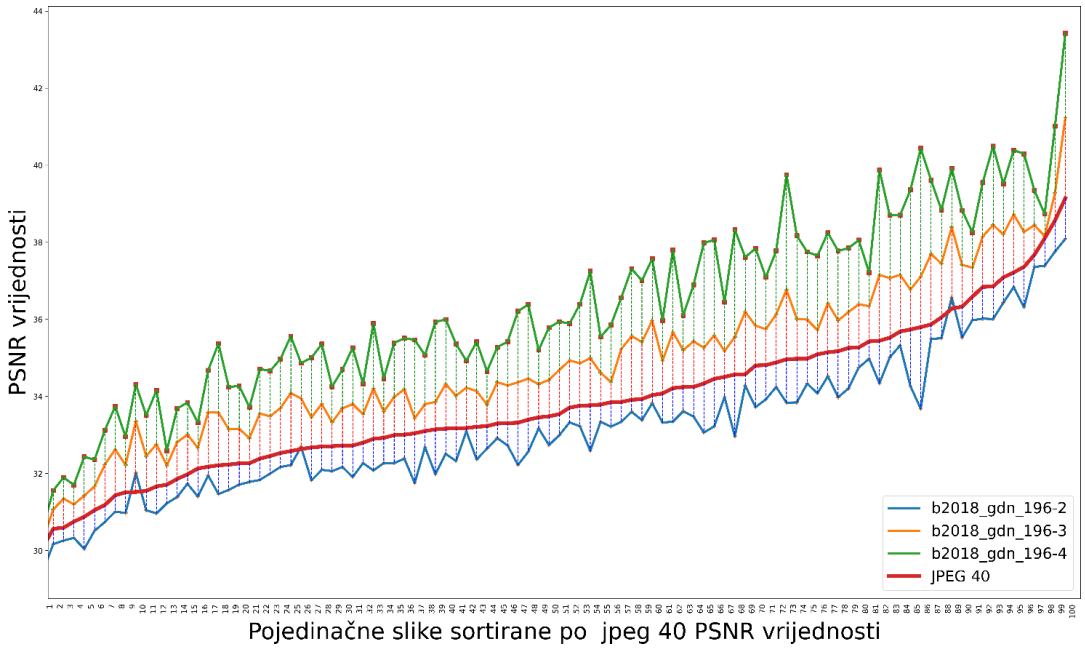
Slika 38: MSSSIM za `bmshj2018_hyperprior-msssim-5, 6` i MSSSIM `.jpeg 50` slika

Na slici 39 će biti prikazana mogućnost korišćenja više modela kao zamjenu za JPEG kompresiju kvaliteta nivoa 60. Sada je sortiranje vršeno po vrijednostima `.jpeg 60` slika. U prikazanom primjeru se potvrđuje ispravnost tvrđenja da je moguće koristiti više različitih modela kao zamjenu jednog nivoa JPEG kompresije. Više slika dobijenih preko različitih modela je uspješno zamijenilo slike `.jpeg 60` kvaliteta. Na slici prikazani model `bmshj2018_factorized_msssim-5` (prikazan plavom bojom) u određenim slikama je imao veću vrijednost MSSSIM indeksa u odnosu na konvencionalni JPEG algoritam sa kvalitetom 60 (prikazan crvenom bojom). Ostali modeli sa većim kvalitetom imaju bolje tražene karakteristike. `Bmshj2018_factorized_msssim-6` model (prikazan narandžastom bojom) ima bolje performanse u odnosu na `.jpeg 60` (prikazan crvenom bojom) nivo kvaliteta a `bmshj2018_factorized_msssim-7` (prikazan zelenom bojom) ima mnogo bolje performanse u odnosu na `.jpeg 60` kvalitet. Navedeni rezultati govore o potencijalnoj mogućnosti za upotrebu različith modela da zamijene određeni nivo JPEG kvaliteta. Dakle, u primjeru na slici 39, vidi se da je moguće zamijeniti `.jpeg 60` slike slikama koje odogvaraju

modelu sa manjim argumentom `bmshj2018_factorized_msssim-5.`



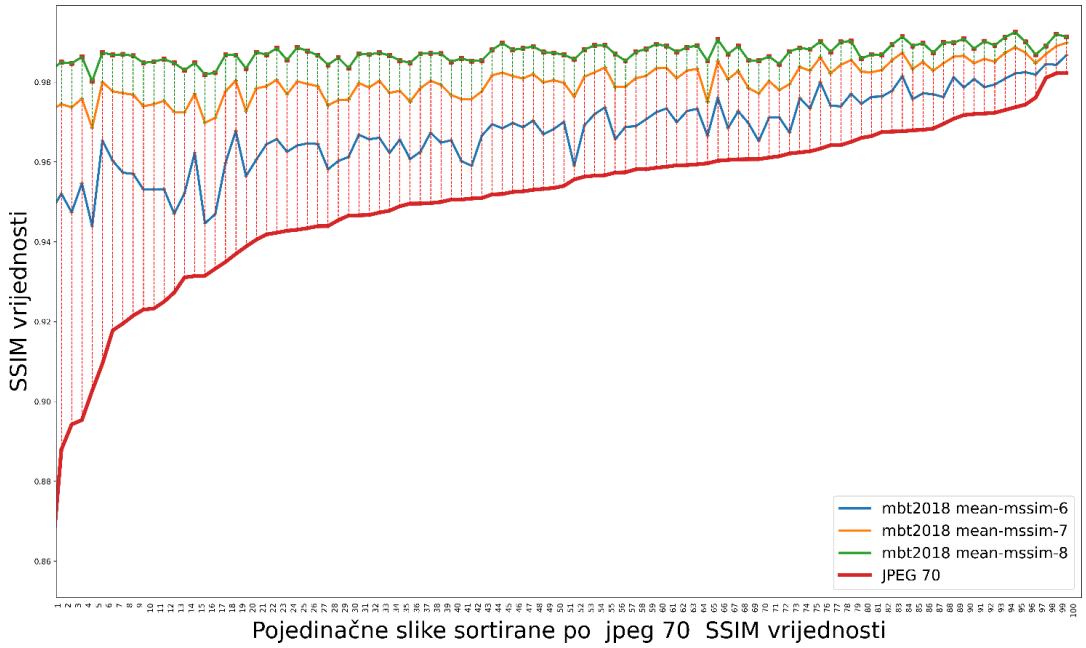
Slika 39: MSSSIM vrijednosti `bmshj2018_factorized_msssim-5, 6, 7` modela i MSSSIM `.jpeg 60` slika



Slika 40 : PSNR `b2018_gdn_196-2, 3, 4` i PSNR `.jpeg 40` slika

Slika 40 odgovara uspoređivaju PSNR vrijednosti `b2018-gdn-196-2, 3, 4` modela i PSNR vrijednosti koja odgovara `.jpeg 40` kvalitetu. Na slici su plavom, narandžastom i zelenom bojom prikazani kompresioni modeli `b2018-gdn-196-2, 3, 4` a crvenom bojom

je prikazana PSNR vrijednost koja odgovara .JPEG 40 kvalitetu. Modeli b2018-gdn-196-3, 4 daju bolje PSNR rezultate u odnosu na .jpeg 40 kvalitet.

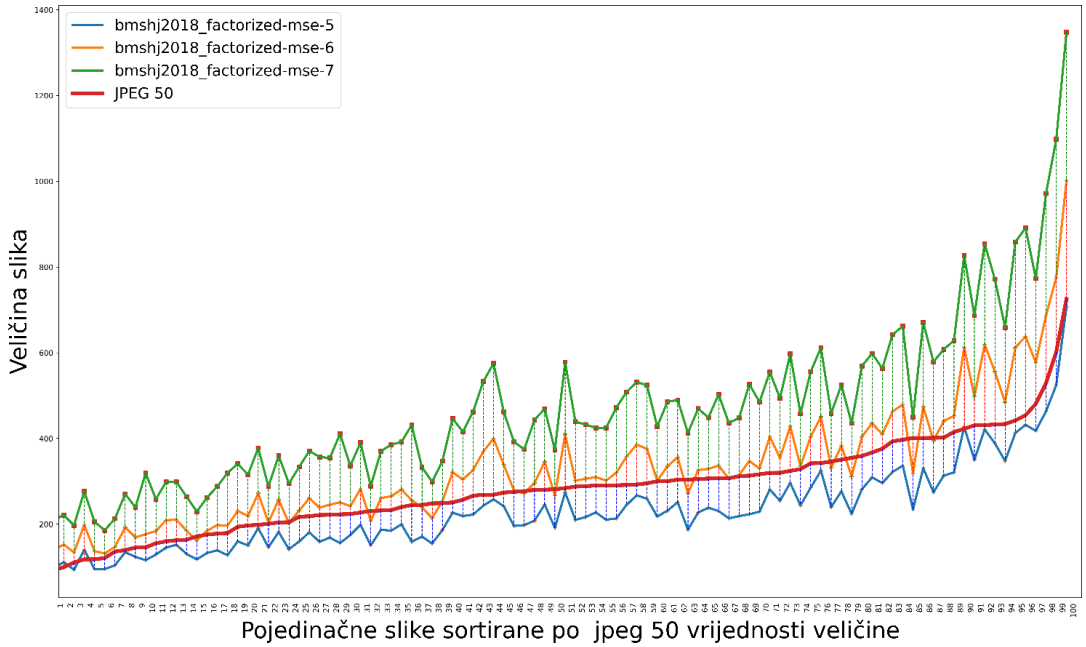


Slike 41: MBT2018 mean-mssim- 5, 6, 7 SSIM vrijednosti i SSIM .jpeg 70 slika

Na slici 41 su prikazane SSIM vrijednosti za slike koje odgovaraju modelima mbt2018-mean-mssim- 6, 7, 8 (prikazane plavom, narandžastom i zelenom bojom respektivno) i slike .jpeg 70 kvaliteta (prikazano crvenom bojom). Svi modeli imaju značajno veće vrijednosti SSIM indeksa u odnosu na JPEG 70 algoritam. Na slici 41 za predstavljene modele mbt2018-mean-mssim- 6, 7, 8 je prikazana sličnost od 0.98 (98%) a najmanja sličnost je veća od 0.94 (94%). Razlika između originala i slike dobijene kompresijom ili JPEG algoritmom je veoma mala kada se predstavljaju SSIM indeksom. Vizuelno razlike posmatraču neće biti lako uočljive. Za potrebe skladištenja, kada je presudna memorija potrebna za čuvanje velikog broja slika, moguće je izabrati model sa manjim argumentom a vizuelna razlika između dobijenih slika će biti veoma mala.

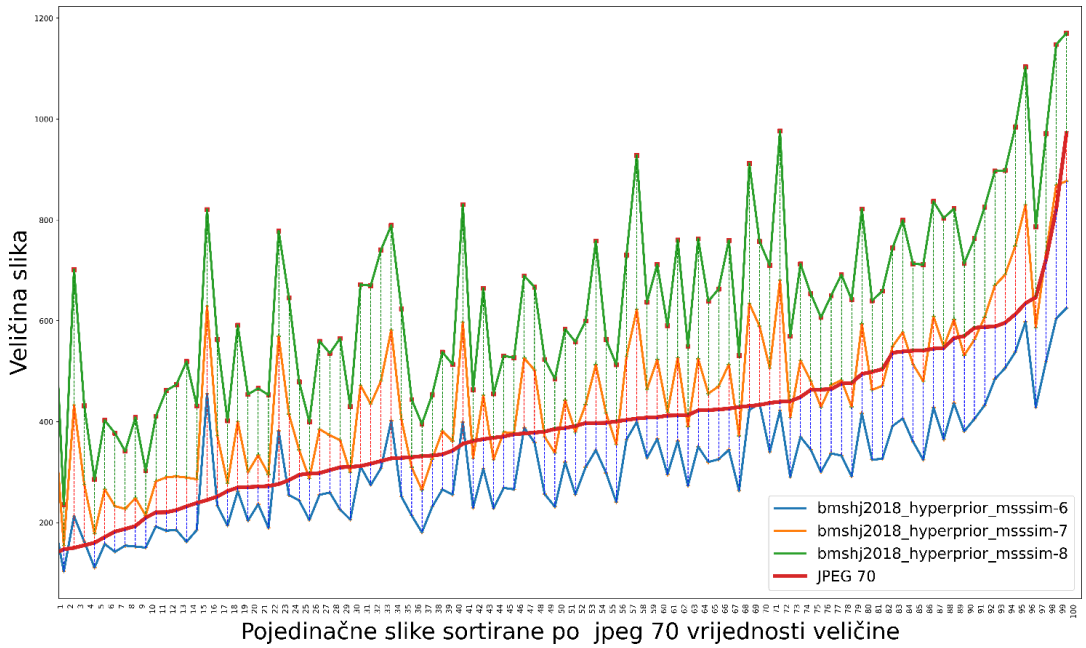
Prikaz na slici 41 predstavlja samo jedan od mogućih (slučajno izabranih) prikaza. Moguće je upariti sve PSNR vrijednosti koje odgovaraju svim modelima. Rezultati modela koji sadrže MSE unutar svog imena će generalno imati bolje vrijednosti u odnosu na rezultate koji sadrže MSSSIM. Drugim riječima, mbt2018-mean-mssim-6-PSNR je generalno manje mbt2018-mean-mse-6-PSNR. Dok za MSSSIM i SSIM indekse važi obratno: mbt2018-mean-mssim-6-msssim je veće od mbt2018-mean-mse-6-msssim. Prilikom odabira

modela koji se koristi za upoređivanje sa JPEG, za različite metode kvaliteta, treba voditi računa za šta je model optimizovan (za mse ili msssim metriku).



Slika 42: Odnos veličina `bmshj2018_factorized-5, 6, 7` i veličine `.jpeg 50` slika

Na slici 42 su prikazani odnosi veličina slika koje su dobijene korišćenjem `bmshj2018_factorized-5, 6, 7` modela kompresije i slika `.jpeg 50` (slika koje odgovaraju nivou kvaliteta 50). Veličina slika `bmshj2018_factorized-5` modela je prikazana plavom bojom, `bmshj2018_factorized-6` narandžastom bojom, `bmshj2018_factorized-7` zelenom bojom, a crvenom bojom je prikazana veličina `.jpeg 50` slika. Može se zaključiti da slike dobijene preko modela `bmshj2018_factorized-5` imaju manju veličinu nego slike dobijene `.jpeg 50` kompresijom. Na predstavljenom grafiku od 100 slika samo jedna slika `bmshj2018_factorized-5` će imati vrijednost veličine kompresovanog fajla veću od `.jpeg 50` odgovarajuće slike. Primjetno je da i slike koje predstavljaju model `bmshj2018_factorized-6` takođe imaju približne veličine kao i slike dobijene `.jpeg 50` pa se u određenim slučajevima mogu koristiti kao zamjena. Moguće je efektivno zamijeniti slike dobijene `.jpeg 50` kompresijom slikama dobijenim korišćenjem `bmshj2018_factorized-5` modela i pri tome iskoristiti manje memorije za skladištenje fajlova.

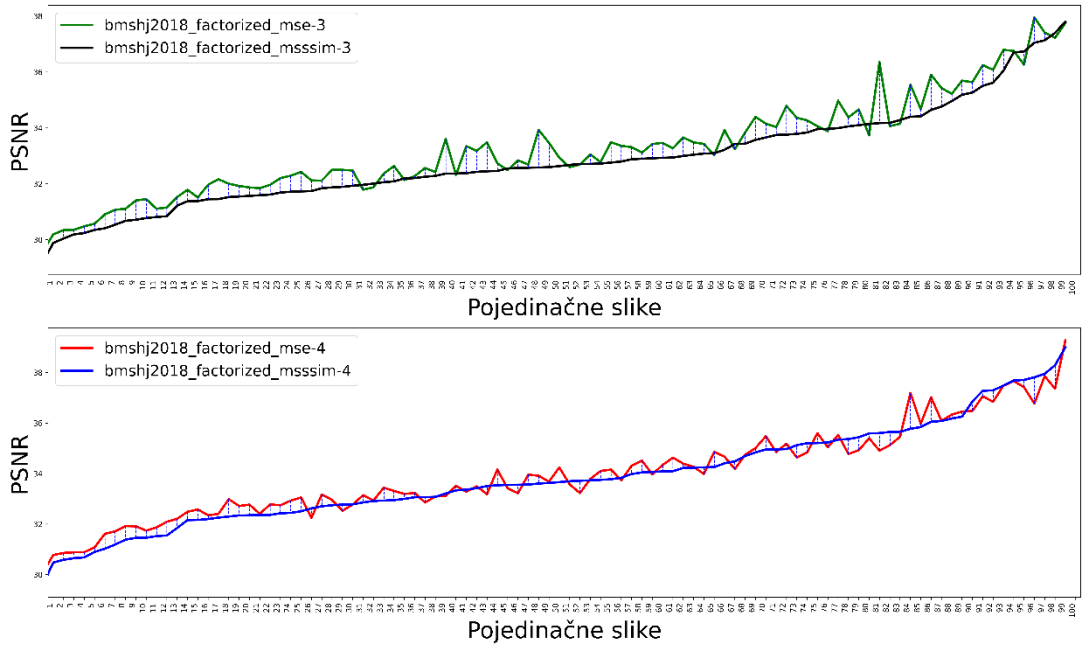


Slika 43: Odnos veličina `bmshj2018_hyperprior_msssim-6, 7, 8` i veličine slika `.jpeg 70` kvaliteta

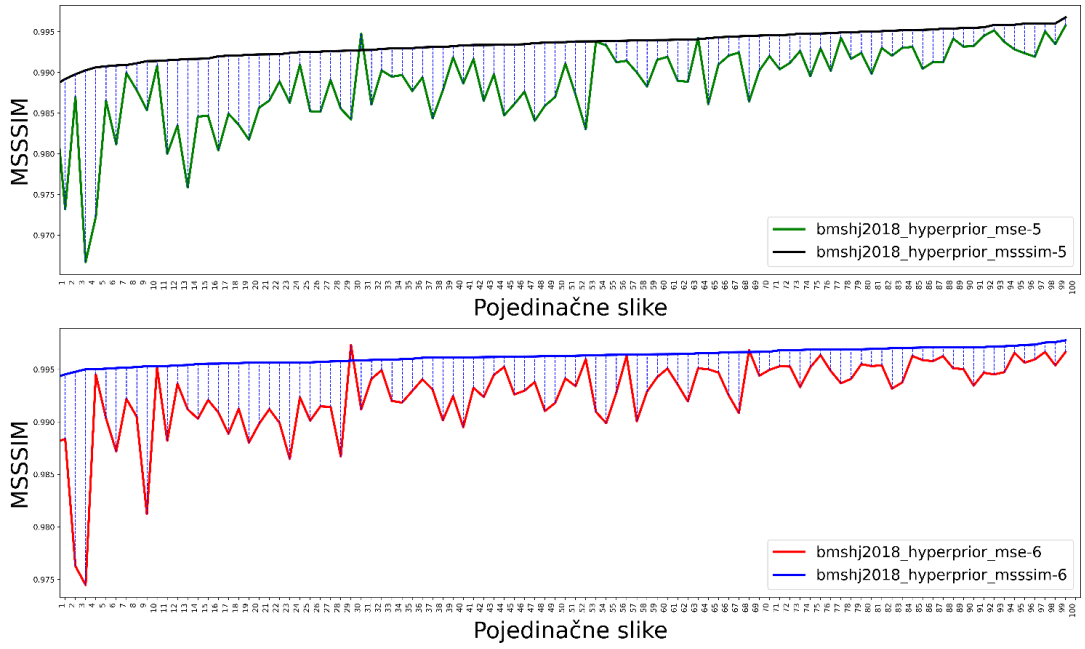
Što se veličina tiče, modeli mašinskog učenja imaju osobinu da naglo povećaju potrebnu memoriju za skladištenje. Na slici 43 plavom bojom je prikazana veličina slika koja odgovara `bmshj2018_hyperprior_msssim-6` modelu kompresije, naradžastom je prikazana veličina slika koja odgovara `bmshj2018_hyperprior_msssim-7` modelu kompresije, zelenom je prikazana veličina slika koja odgovara `bmshj2018_hyperprior_msssim-8` modelu kompresije, a crvenom je prikazana veličina slika `.jpeg 70` (JPEG kompresija nivoa 70). Iako su rezultati veličine potrebne memorije loši u odnosu na modele sa manjim argumentima za sliku 43 važi da slike dobijene modelom `bmshj2018_hyperprior_msssim-6` mogu efektivno zamijeniti slike `.jpeg 70`, što se memorije potrebne za skladištenje tiče.

Za trenutak će biti data pažnja tvrđenjima o optimizaciji modela. Biće pojedinačno prikazana po dva para modela otimizovanih za MSE i MSSSIM metriku. Na slikama 44 i 45 će detaljno biti prikazane specifičnosti pojedinih modela. Modeli koji su optimizovani za MSE (srednju kvadratnu grešku) će imati bolje PSNR rezultate u odnosu na modele koji su optimizovani za MSSSIM vrijednosti. Na slici 44 je prikazan odnos MSSSIM indeksa za sve slike modela `bmshj2018_factorized_mse-3` (prikazano zelenom bojom na slici 44), modela `bmshj2018_factorized_mssim-3` (prikazano crnom bojom na slici 44) i

kompresionog modela `bmshj2018_factorized_mse-4` (prikazano crvenom bojom) `bmshj2018_factorized_mssim-4` (prikazano plavom bojom).



Slika 44: Odnos factorized modela za PSNR vrijednost



Slika 45: Odnos Hyperior modela u odnosu na MSSSIM vrijednost

Na slici 45 su predstavljeni i prikazani rezultati MSSSIM vrijednosti za dvije grupe modela. Modeli koji su optimizovani za MSSSIM vrijednost će generalno imati veću vrijednost MSSIM indeksa za slike dobijene kompresijom u odnosu na modele koji su optimizovani za

MSE vrijednosti. Za sliku 45 model `Bmshj2018-hyperprior-msssim-5` (prikazano zelenom bojom na slici 45) će generalno imati veće vrijednosti MSSSIM indeksa za kompresovane slike u odnosu na `Bmshj2018-hyperprior-mse-5` (prikazano crnom bojom) i `Bmshj2018-hyperprior-msssim-6` (prikazano crvenom bojom) će generalno imati veće vrijednosti MSSSIM indeksa za kompresovane slike na `Bmshj2018-hyperprior-mse-6`(prikazano plavom bojom).

Na ovaj način se još jednom naglašava da je moguće na osnovu argumenta (mse ili msssim) odlučiti koji model je više pogodan za određene probleme.

Pošto je predočeno da vizuelni kvalitet na većim kompresionim nivoima ima izrazito mali raspon razlike u slikama dobijenim preko modela i JPEG kompresije će praktično biti neprimjetne. To dovodi do zaključka da za modele koji odgovaraju malim nivoima kompresije vizuelni efekat će ostaviti najveći uticaj.

Sledeća grupa slika će se truditi da prikaže male vizuelne razlike originala, slika dobijenih JPEG kompresijom i slika dobijenih pojedinačnim modelima. Sa cijelog seta slika slučajno je izabrana slika koja odgovara nivou kvaliteta 3.

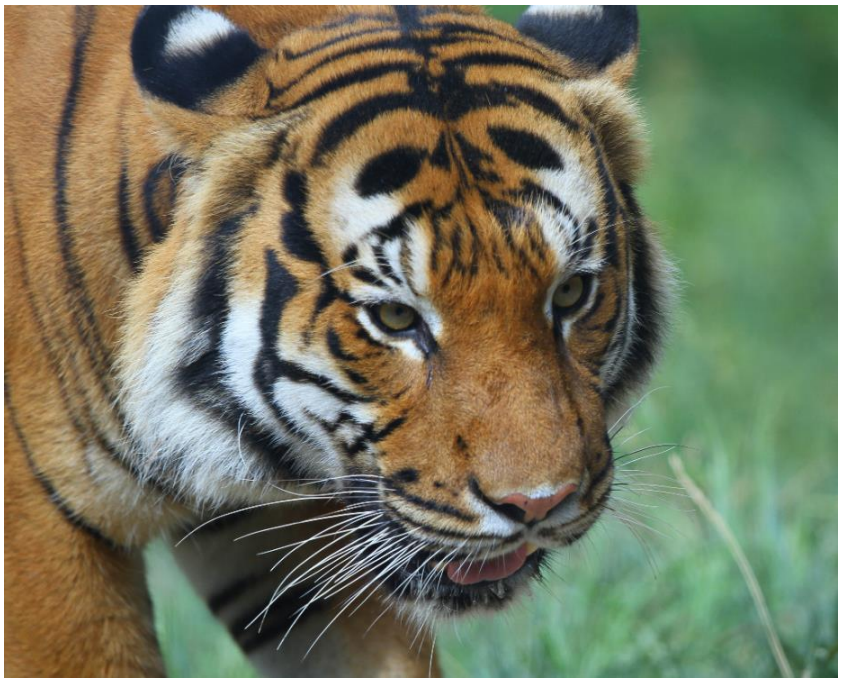
Slika 46: Originalna slika

Slika 47: `Mbt2018-mean-msssim-3` dekompresovana slika

Slika 48: `JPEG 30` slika

Na slikama 46, 47 i 48, su vidne razlike u teksturi. Slika 47 (dobijena korišćenjem `Mbt2018-mean-msssim-3` modela) predstavlja ljepši vizuleni prikaz u odnosu na sliku 48 (`.jpeg 30` slika). Pojavljivanje blokova ne postoji na slici 47 dok na slici 48 su vidljivi blokovi, kao nuspojave JPEG kompresije.

Slika 47 dobro prikazuje efekat glađenja koji je karakterističan za ove generativne modele. Dok se na slici 48, nijanse sive prikazuju kao koncentrični krugovi, na slici 47 je prikazani bolji prelazi koji veoma dobro posjećaju na original. Osnovna vidna razlika u odnosu na original (slika 46) je oštrina slike.



Slika 49: Originala slika

Slika 50: `Bmshj2018-hyperprior-mse-1` dekompresovana slika

Slika 51: `.jpeg 10` slika

Ponovo su vizuelne informacije na ljepsi način prikazane na slici 50 (dobijenoj preko modela) nego na slici 51 (odgovara JPEG kompresiji kvaliteta 10). Ova slika je slučajno odabrana od skupa svih slika dobijenih sa modela koji posjeduju argument 1.

U svim priloženim primjerima slika je odabran različiti model i generalno za sve modele važi da imaju bolji prikaz za niže nivoe kvaliteta nego JPEG algoritam. Posebno je vidljiv efekat glađenja koji ne utiče negativno na kvalitet slika dobijenih preko modela. To znači da .jpeg slike generalno posjeduje veću oštrinu u odnosu na slike modela ali baš oštrine te se pojavljuju kao blokovi i loši rezultati.

Modeli kompresija u mašinskom učenju vizuelno veoma dobro predstavljaju sve podatke zbog korišćenja više različitih tehnika za otklanjanje grešaka prilikom kompresije. Generativni procesi u Varijacionim autoenkoderima stvaraju nove slike u kojima se odbacuju veoma sitni detalji. To na prikazanoj slici 51 može biti dio šare, detalj trave u pozadini ili bilo koji struktruni aspekt za koji model ustanovi da nema prevelik značaj za vizuelni aspekt u slikama. To čini dobijene slike boljim, sa ljepše definisanim strukturama i ljepšim vizuelnim prikazom.

6. Zaključak i nacrt daljih istraživanja

Mašinsko učenje kao grana vještačke inteligencije je dostupno svima tek u poslednjih 20 godina. Za dokazivanje matematičkih teorema i računanje pojedinačnih gradijenata bile su potrebne mašine visokih performansi gdje je korisnik doslovno morao da sam, od početka kreira logiku iza arhitekture modela mašinskog učenja. Danas je situacija mnogo drugačija.

Specijalizovane Python biblioteke TensorFlow, TensorFlow Compression, TensorFlow Keras sadrže kompleksne matematičke alate, kojima veoma lako predstavljaju kompleksne matematičke probleme koji su nekada bili dostupni samo pojedincima. Za primjer treniranja mreže za prepoznavanje brojeva bilo je potrebno 30 sekundi da se mreža trenira na cijelom MNIST dataset-u. Na početku 21. vijeka za isti proces su bili potrebni dani, ako ne i nedjelje. Danas, laka definicija tenzora, mogućnost povezivanja tenzora u kompleksne strukture, kao što su Autoenkoder modeli, Varijacioni autoenkoderi, modeli Generativnih suparničkih mreža čini ovaj svijet primamljivim za sve programere. Svi koraci su veoma dobro definisani unutar kompleksnih struktura. Moguće je koristiti različite aktivacione funkcije, različite normalizacione funkcije, različite kovolucione i pooling filtere – sve zamjenom jednog argumenta u deklaraciji funkcije.

Druga veoma važna stavka je evolucioni proces gradijentnog spusta. ADAM algoritam je predstavljen prije manje od 10 godina a već postoji više verzija Adam algoritma koje se bave specifičnim problemima. To dovodi do zaključka da se i u matematičkom pogledu poslednjih godina radi veoma mnogo, kada je mašinsko učenje u pitanju. Mogućnost testiranja ekperimentalnih pretpostavki na većem broju računara dodatno ohrabruje istraživače da traže nova rješenja za minimizaciju funkcije cijene. Sam program Adam predstavlja kombinaciju pristupa korišćenih unutar dva prošla algoritma AdaGard i RmsProp. Budući algoritmi će imati za cilj da još bolje i brže konvergiraju ka minimumu na nove neistražene načine.

Što se striktno tiče mašinskog učenja u kompresiji digitalnih slika, poslednjih godina primijetan je veliki broj publikacija u oblasti koja se bavi digitalnom obradom slike, generisanjem slika i kompresijom. Većina predloženih modela TensorFlow Compression biblioteke nije starija od 6 godina. Prilikom implementacije pojedinih modela naučnici su koristili već dostupne djelove definisane preko TensorFlow biblioteke. Na osnovu toga moguće je zaključiti da za samostalno kreiranje modela potrebno je osnovno znanje o neuralnim mrežama i personalni računar. Kombinacije slojeva, funkcija, Varijacionih autoenkodera ne može biti

totalno slučajna ali se logika može preuzeti sa velike količine materijala na internetu. Poznavanjem određenih zakonitosti i logike pojedinih djelova unutar arhitekture može uz malo truda dovesti do jednostavnih modela, koji možda neće imati veoma dobre rezultate, ali će biti putokaz za usavršavanje sopstvenih modela. Sama TensorFlow Compression biblioteka nudi alat za kreiranje sopstvenih modela i generisanje novih, boljih slika u odnosu na slike dobijene preko JPEG algoritma.

Mnogi modeli kompresije digitalnih slika zasnovanih na mašinskom učenju uspješno će zamijeniti slike dobijene JPEG algoritmom. Posebno je vidan boljitak na nivoima kompresije koji odgovaraju manjim argumentima. U predstavljenim primjerima, uz efekat glađenja, slike dobijene mašinskim učenjem vizuelno su bolje rješenje u odnosu na slike koje odgovaraju JPEG kompresiji. Rješenja nemaju nuspojave koje su karakteristične za JPEG algoritam, posmatrač mora da se potrudi da nađe velike razlike između originala i slike dobijene preko kompresionih modela a kod slika dobijenih preko JPEG algoritma razlike su vidne. Što se tiče memorije potrebne za skladištenje, kompresioni fajlovi kreirani preko modela imaju manju veličinu u odnosu na .jpeg fajlove čime je moguća ušteda u prostoru koji je potreban za skladištenje. Predstavljeni modeli su optimizovani za više problema. Modeli optimizovani za kvadratnu grešku imaju veće vrijednosti PSNR indeksa u odnosu na modele optimizovane za MSSSIM indeks. Na ovaj način moguće je izabrati model za specifičan problem. Za probleme koji zahtjevaju veću vizuelnu vjerodostojnost kompresovanih digitalnih slika će se koristiti modeli sa argumentom MSSSIM dok za probleme gdje se traži apsolutna razlika između dvije slike će se koristiti modeli sa argumentom MSE. Prilagodljivost, mogućnost šire upotrebe slika dobijenih preko kompresionih modela čine kompresione modele adekvatnom zamjenom za JPEG algoritam.

U rezultatima su prikazane bolje osobine slika dobijenih tehnikama mašinskog učenja u odnosu na JPEG algoritam. Pored mnogo boljih vizuelnih rezultata slike dobijene mašinskim učenjem imaju mnogo bolji kvalitet i za druge metode kojima se mjerio kvalitet slika dobijenih kompresijom. Sve ovo čini kompresione pristupe zasnovane na mašinskom učenju boljim izborom u odnosu na tradicionalne kompresione tehnike. Na kratko je pomenuto da se unutar Python jezika razvija biblioteka PyScript koja će možda zamijeniti JavaScript. To može da bude most koji će da omogući korišćenje svih Python biblioteka na internetu. Ovim postupkom će možda biti moguće koristiti i TensorFlow Compression biblioteku za prikaz slika na WEB stranicama. Time bi algoritmi mašinskog učenja uspješno zamijenili JPEG algoritam.

Mašinsko učenje i kompresija slika u mašinskom učenju predstavlja veoma atraktivne naučno istraživačke oblasti. Nova istraživanja, novi eksperimenti, novi modeli i novi pristupi minimizaciji funkcije cijene za posebne probleme sugerišu da tek prestoje veliki iskoraci ka boljim sveobuhvatnim rješenjima.

Literatura

[1]Balle, Johannes, Valero Laparra, and Eero P. Simoncelli (2015). “Density Modeling of Images Using a Generalized Normalization Transformation,” in: arXiv e-prints, *Presented at the 4th Int. Conf. for Learning Representations*, 2016. arXiv: 1511.06281.

[2]Laparra, Valero et al. (2016). “Perceptual image quality assessment using a normalized Laplacian pyramid”. In: Proceedings of SPIE, Human Vision and Electronic Imaging XXI

[3]J. Balle, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. arXiv preprint arXiv:1611.01704, 2016

[4]Kingma, Diederik P. and Jimmy Lei Ba (2014). “Adam: A Method for Stochastic Optimization”. In: arXiv e-prints. Presented at the 3rd Int. Conf. for Learning Representations, 2015. arXiv:1412.6980.

[5]Theis, Lucas, van den Oord, Aäron, and Bethge, Matthias. A note on the evaluation of generative models. arXiv e-prints, 2015. Under review as a conference paper at the 4th International Conference for Learning Representations, San Juan, 2016.

[6]Theis, Lucas, and Matthias Bethge. "Generative image modeling using spatial lstms," Advances in neural information processing systems 28 (2015). arXiv:1506.03478

[7]Gregor, Karol, Frederic Besse, et al. (2016). “Towards Conceptual Compression”. In: arXiv e-prints. arXiv: 1604.08772.

[8]Toderici, George et al. (2016). “Full Resolution Image Compression with Recurrent Neural Networks”. In: arXiv e-prints. arXiv: 1608.05148

[9]Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). “Pixel Recurrent Neural Networks”. In: arXiv e-prints. arXiv: 1601.06759.

[10]Szegedy, Christian et al. (2013). “Intriguing properties of neural networks”. In: arXiv e-prints. arXiv: 1312.6199

[11]Diederik P Kingma, Max Welling, „Auto-Encoding Variational Bayes“. In: arXiv e-prints arXiv:1312.6114v11

[12]Sebastian Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747v2. [cs.LG] 15 Jun 2017

[13]Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[14]Z. Wang, E. P. Simoncelli and A. C. Bovik, "Multiscale structural similarity for image quality assessment," The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers,2003, Pacific Grove, CA, USA, 2003, pp. 1398-1402 Vol.2, doi:

10.1109/ACSSC.2003.1292216.

[15]Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, Nick Johnston „Variational image compression with a scale hyperprior“ In: arXiv e-prints arXiv:1802.01436v2

[16]David Minnen, Johannes Ballé, George Toderici „Joint Autoregressive and Hierarchical Priors for Learned Image Compression“ In: arXiv e-prints arXiv:1809.02736v1

a. Johannes Ballé „Efficient Nonlinear Transforms for Lossy Image Compression“ In: arXiv e-prints arXiv:1802.00847v2

[17]Boukaye Boubacar Traoré, Bernard Kamsu-Foguem, Fana Tangara. „Deep convolution neural network for image recognition.“ Ecological Informatics, 2018, 48, pp.257-268. ff10.1016/j.ecoinf.2018.10.002ff.ffhal-02053205f