SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5396

Sažimanje slika neuronskim mrežama

Alen Štruklec

Zagreb, lipanj 2018.

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

Zahvaljujem prof. dr. sc. Domagoju Jakoboviću na mentorstvu i pomoći pruženoj pri pisanju završnog rada.

Također zahvaljujem dr. sc. Ivanu Vučaku, mag. ing. na pruženoj pomoći oko organizacije završnog rada.

SADRŽAJ

1. Uvod **1**

1.1. Uvod 1

1.2. Neuronske mreže 1

1.3. Konvolucijske neuronske mreže 3

2. Kompresija slike **4**

2.1. Kompresija bez gubitaka 4

2.1.1. PNG - Portable Network Graphics 4

2.1.2. WebP 4

2.2. Kompresija s gubicima 5

2.2.1. JPEG 5

2.3. Kompresija pomoću neuronskih mreža 5

2.3.1. Prime 5

2.3.2. WaveOne 6

3. Metode **7**

3.1. Korišteni alati 7

3.2. Upsampling 7

3.3. SubPixel Upscaling 7

3.4. Arhitektura 8

3.4.1. Autoenkoder 8

3.4.2. Pokušaji realizacije 9

3.4.3. Konačna mreža 10

4. Učenje mreže **11**

4.1. Postupak učenja 11

4.2. Skup podataka 11

4.3. Gradijentni spust 12

4.3.1. Stohastički gradijentni spust 12
4.3.2. ADAM 13
4.4. Funkcija gubitka 13
4.4.1. Srednja kvadratna pogreška 13
4.4.2. Strukturalna sličnost 14
4.4.3. Prilagođena mjera gubitka 14

5. Rezultati **15**

6. Zaključak **19**

Literatura **20**

Popis slika **23**

A. Dio izvornog koda za početnu mrežu **24**

B. Dio izvornog koda za konačnu mrežu **25**

1. Uvod

1.1. Uvod

Napretkom tehnologije i sve većom uporabom iste dolazi do povećanja potrebe za dodatnim prostorom uređaja i "cloud" servisa. Većina društvenih mreža temelji se na dijeljenu fotografija kao sredstvo za opis doživljaja. Fotografije samim time postaju jedan od najčešće dijeljenih medija.

Iz tog razloga ovaj se rad bavi kompresijom slike korištenjem neuronske mreže. Prije same analize modela kompresije, u teorijskom dijelu je obrašnjen zapis kompresirane slike, načini kompresije te način određivanja kvalitete kompresirane slike u odnosu na original.

Za opisivanje slike koristimo razne veličine koje nama daju objektivan podatak o odnosu kompresirane slike i originalne.

Odstupanje od originalne slike ovisi o više faktora, značajan faktor je način spremanja slike. Ako se koristi način kompresije koji gubi dio podataka, tijekom dekompresije doći će do određenih artefakta te će slika odstupati od originalne slike. Kvaliteta nakon dekompresije je relativan pojam, osim samog objektivnog matematičkog odstupanja, te ovisi i o percepciji osobe koja promatra razlike. Slika može djelovati više ili manje kvalitetna ovisno što osoba promatra, promjene boja nužno ne utječu toliko kao što je gubitak detalja.

U nastavku rada bit će opisana arhitektura neuronske mreže korištene u ovom radu te opis nekih njezinih komponenti. Također će biti opisani skupovi podataka za učenje i samo učenje.

1.2. Neuronske mreže

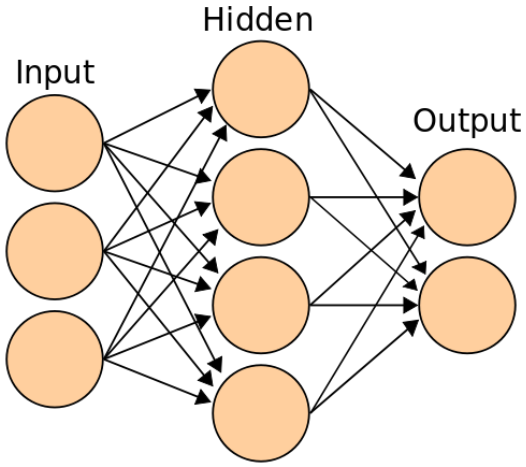
Umjetne neuronske mreže ili skraćeno neuronske mreže su paradigma koja opisuje obradu informacija inspirirano načinom na koji ljudski mozak obrađuje podatke. Ne-

Slika 1.1: Usporedba kvalitete kompresije kod PNG (desno) i JPEG (lijevo) s 10% kvalitete na slici iz skupa podataka Kodak [10]

uronske mreže za obradu koriste računalne modele neurona koji su međusobno povezani i interaktivni kroz operacije obrade signala. Takav model omogućuje visoku razinu paralelizma.

Temeljno obilježje svih neuronskih mreža je svojstvo učenja. Inspiracija za svojstvo učenja opet je mozak, u neuronskom biološkom sustavu učenje podrazumijeva podešavanje sinapsi, a slično tome se podešavaju parametri umjetne neuronske mreže.

Sam model neuronske mreže sastoji se od slojeva. Početni ili ulazni sloj sastoji se od jednog ili više ulaza, a završava s jednim ili više izlaza. Između ulaznog i završnog sloja nalazi se jedan ili više skrivenih slojeva.



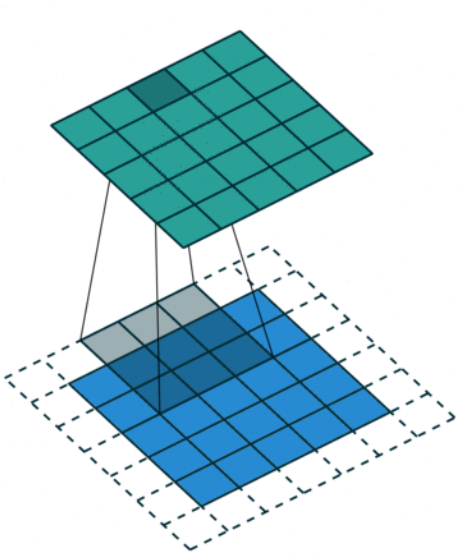
Slika 1.2: Izgled jednostavne umjetne neuronske mreže

1.3. Konvolucijske neuronske mreže

Konvolucijske mreže se danas primjenjuju u većini problema računalnog vida gdje vrlo često postižu vrhunske rezultate. Za razliku od potpuno povezanih mreža, koje za ulaznu sliku od 128x128 pixela i uz skriveni sloj od 100 neurona imaju preko 1 milijun težina već u prvom sloju, konvolucijske mreže višestruko smanjuju broj težina. Za usporedbu jednostavna konvolucijska mreža s 2 skrivena sloja ima otprilike 10 000 težina dok ista takva mreža konstruirana s potpuno povezanim slojevima ima otprilike $7.5e8$ težina

Za područje računalnog vida važno je da se određeni oblici detektiraju neovisno o njihovoj lokaciji na slici. Potpuno povezani sloj ima posebne težine za svaki piksel te samim time nije u mogućnosti detektirati svaki oblik. Konvolucijske mreže koriste filter koji se pomiče po ulaznom vektoru i na svakoj poziciji daje određeni izlaz te se tako dolazi do konvolucije[12], matematičkog pojma definiranog kao učestalost primjenjivane funkcije u podacima.

Na slici 1.3 koristi se filter veličine 3x3 piksela. Kako se pomiče po slici za slične oblike dat će slične vrijednosti. Koristeći više slojeva konvolucije možemo ostvariti mrežu koja će na cijeloj slici prepoznati određene objekte. Takav koncept koristan je kod računalnog vida za detekciju objekata.



Slika 1.3: Primjer 2d konvolucije s 3x3 filterom

2. Kompresija slike

2.1. Kompresija bez gubitaka

Kompresija bez gubitaka iznimno je važna grana kompresije podataka. Vrlo često važno je da zadržimo sve detalje i obilježja slike, tako da kod dekompresije dobijemo sliku istovjetnu orginalu [18]. Jedan od najjednostavniji algoritama bio bi prebrojavanje koliko ima istih boja u nekom području i zapisao sumu uz samo jedan zapis boje. Postoji mnogo različitih algoritama nad kojima su napravljeni modeli za kompresiju. U daljnjem tekstu prikazat ćemo dva najčešće korištena.

2.1.1. PNG - Portable Network Graphics

PNG format je često korišten format na internetu. Sadrži mnogo prednost nad formatom JPEG (obrađen kasnije u tekstu) kao što su izbjegavanje digitalnih artefakta zbog kompresije bez gubitaka. Vrlo važno svojstvo koje PNG nudi je transparentnost čime omogućava korištenje raznih efekata. PNG datoteke zauzimaju više prostora od JPEG datoteka, zbog toga se ne koriste u određenim područjima gdje je važnija veličina datoteke od kvalitete slike.[11]

2.1.2. WebP

WebP je format izrađen od strane Google-a te zamišljen kao format za moderno doba interneta. Cilj WebP formata je ubrzati učitavanje i otvaranje stranica uz zadržavanje kvalitete slika. Podržava sve moderne značajke i pokušava zamijeniti trenutne formate na internetu.

Također podržava kompresiju s i bez gubitaka. Tako za usporedbu s PNG formatom veličina datoteka je 26% manja, dok za JPEG format može biti i za 34% manja uz istu razinu kvalitete.[6]

2.2. Kompresija s gubicima

Povećanjem korištenja digitalnih slika došlo je do potrebe smanjiti veličinu potrebnu za pohranu. Ideja je prihvatljivost gubitka pojedinih značajki slike kako bi višestruko smanjili potreban prostor. Modeli kompresije s gubicima dizajnirani su na percepciji vida ljudskog oka. Ljudsko oko je više osjetljivo na male promjene u luminiscenciji scene nego na male promjene u boji.[18]

2.2.1. JPEG

Joint Photographic Experts Group ili kraće JPEG je jedan od najkorištenijih formata slike u moderno doba. Koristi se za fotoaparate, mobilne uređaje, internet, itd. Sadrži veliki raspon boja, podržava slike visoke rezolucija i kompresirane datoteke zahtijevaju malo prostora za pohranu.[13]

Format koristi kompresiju s gubicima. Algoritam odbacuje nepotrebne informacije koje ljudsko oko nije u stanju raspoznati, a može se i podesiti kvaliteta kompresije. Važno je napomenuti ako se kvaliteta previše smanji slika izgubi većinu detalja te artefakti postanu iznimno uočljivi. Primjer kompresije sa samo 10% kvalitete prikazano je na slici 1.1

2.3. Kompresija pomoću neuronskih mreža

Porastom popularnosti neuronskih mreža sve više modela kompresije slika se pojavljuje. Veliki uzrok tome je bliska veza između strojnog učenja i kompresije podataka. U posljednjih par godina ova je grana doživjela velike uspjehe i do sada nedostižne razine kompresije.

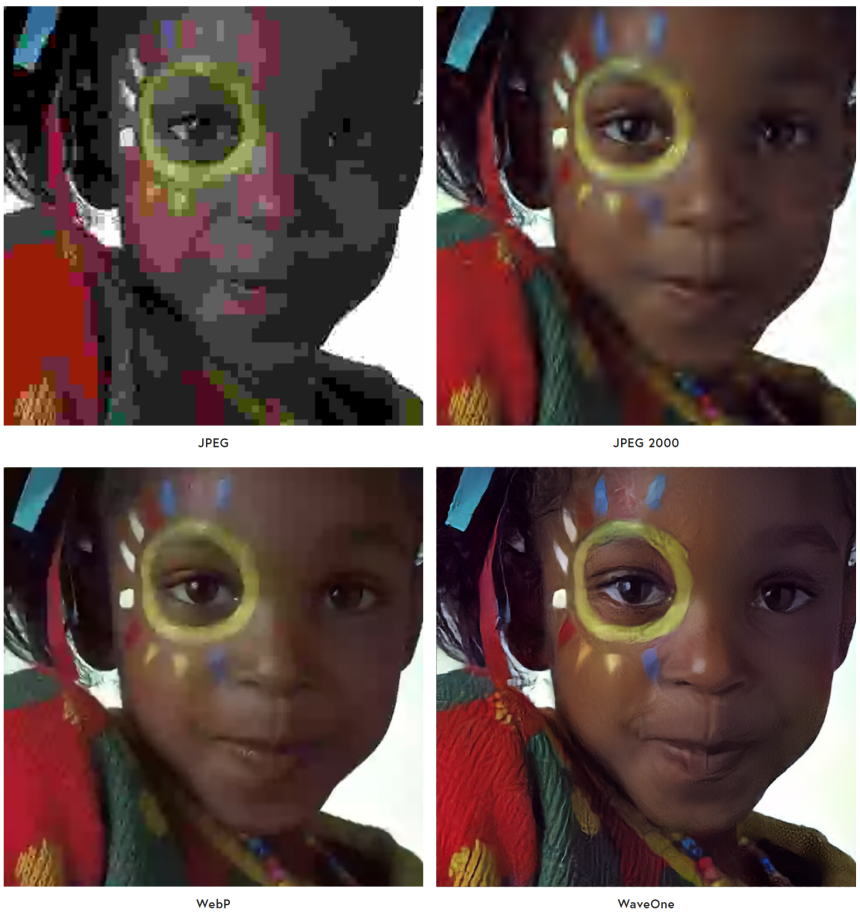
2.3.1. Prime

Prime [8] je model kompresije uz gubitak podataka temeljen na neuronskom mrežama. Nadmašuje većinu modernih formata kao što su WebP, BPG, itd. Prime koristi strukturu autoenkodera(opisano u nastavku) kojeg proširuje s rekurentnom neuronskom mrežom [19]. Značajnost ove mreže je dokaz da modeli temeljeni na neuronskim mrežama mogu dati bolje rezultate od postojećih standardnih modela. Ujedno ovo je prvi model koji pokazuje bolje rezultate od WebP standarda.

2.3.2. WaveOne

WaveOne je model temeljen na strojnom učenju koji nadmašuje sve komercijalne i istraživačke pristupe kompresije slika koji su u vrijeme izrade WaveOne kompresije bili dostupni. Model je kompresija s gubitkom podataka, te uz pristup grafičkoj procesnoj jedinici (GPU) WaveOne može kompresirati ili dekompresirati do 100 slika u sekundi.[14]

Na slici 2.1 prikazana je usporedba 4 različita modela kompresije podataka nad slikom 480x480 piksela uz ograničenje na veličinu slike od 2.3kB. Iz same slike vidi se kakav je trenutni pomak iz standardnih modela na modele temeljene na strojnom učenju.



Slika 2.1: Usporedba 4 modela kompresije podataka[14]

3. Metode

3.1. Korišteni alati

Mreža je izgrađena pomoću programskog okvira Keras[3] koji u pozadini koristi okvir TensorFlow[1]. TensorFlow je danas jedan od najpopularnijih okvira za strojno učenje te je zbog tog razloga odabran, dok je Keras samo omotač oko TensorFlow-a kako bi olakšao pisanje i brzo mijenjanje koda. Kao programski jezik korišten je python u razvojnoj okolini otvorenog koda Jupyter Notebook.

3.2. Upsampling

Konvolucijski slojevi često služe kako bi smanjili dimenzionalnost podataka kao što je prikazano na slici 1.3. No ako želimo povećati dimenzionalnost trebamo drugačije slojeve. Jedan takav sloj je Upsampling sloj čija se implementacija nalazi u okviru Keras.

Upsampling sloj povećava dimenzionalnost podataka kopiranjem postojećih podataka te slaganjem tih podataka na određena mjesta. Uzmemo li za primjer sliku koja dimenzije 64x64 piksela prolaskom kroz Upsampling sloj slika postaje dimenzije 128x128 piksela, ali na slici nema novih detalja. Postoje različite implementacije, neke od kojih pružaju interpolaciju sadržaja, ali se u posljednje vrijeme zamjenjuju drugim slojevima.

3.3. SubPixel Upscaling

Vrlo važan sloj u mreži je SubPixel Upscaling sloj. Zadaća ovog sloja je da iz slike manje rezolucije izradi sliku veće rezolucije uz predviđanje i dodavanje detalja. Postoji nekolicina slojeva koje mogu raditi takav postupak, no ovaj sloj pokazuje vrhunske rezultate uz relativno malo korištenje računalnih resursa.

Model izgrađen na temelju SubPixel Upscaling slojeva prvi je model koji omogućava korištenje super rezolucije[5] na HD videu u realnom vremenu uz korištenje samo jedne GPU.[15]

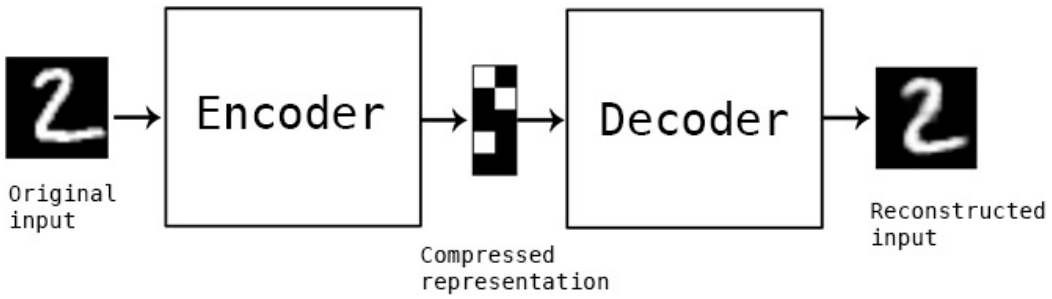
3.4. Arhitektura

3.4.1. Autoenkoder

Autoenkoder je vrsta umjetne neuronske mreže koja nastoji naučiti reprezentaciju za neki skup podataka s ciljem smanjenja dimenzionalnosti [17]. Osnovna ideja autoenkodera je neuronska mreža u kojoj su ulazni i izlazni sloj jednaki po broju neurona, a u sredini imamo sloj koji sadrži manje neurona. Srednji sloj tada poprima ulogu "uskog grla" koji može efikasno naučiti reprezentaciju ulaznog podatka.

Arhitektura najjednostavnijeg autoenkodera je, kao što je gore navedeno, ulazni, skriveni i izlazni sloj. Cilj je rekonstrukcija podataka na izlazu. Autoenkoder se sastoji od dva dijela, enkoder i dekoder. Enkoder služi kako bi kompresirao podatke u neku reprezentaciju, dok dekoder služi da se ta reprezentacija rekonstruira u podatak što više sličan ulaznom.

Autoenkoder daje kompresiju s gubitkom podataka, no pošto se koristi strojno učenje "algoritam" za kompresiju se generira umjesto da ga ljudi pišu. Nažalost zbog toga je model moguće koristiti samo sa sličnim podacima od onih nad kojima se model učio. Na primjer ako je autoenkoder bio učen samo na nekim slikama voća radit će dobro za ostalo voće, ali neće za primjerice vozila.



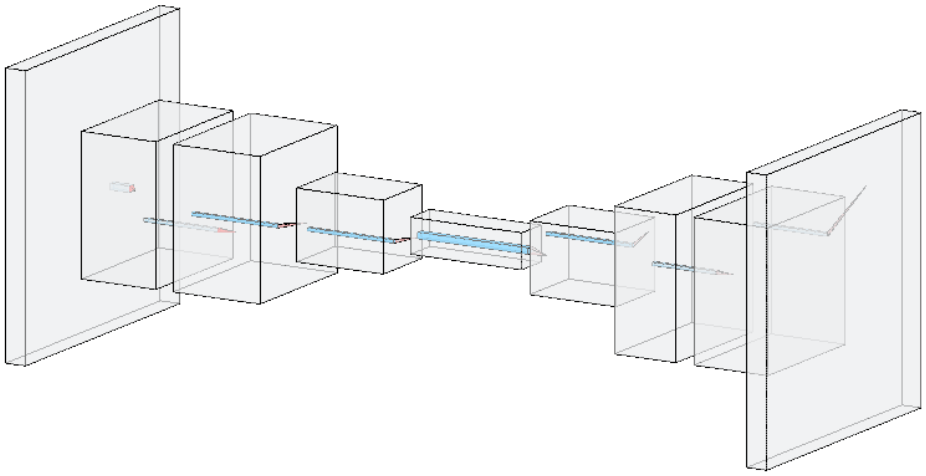
Slika 3.1: Primjer autoenkodera koji koristi slike [16]

3.4.2. Pokušaji realizacije

Konvolucijske neuronske mreže koriste se za računalni vid i dobre su u tome. Autoenkoderi služe za kompresiju podataka te su također dobri u tome. Cilj ovog rada bio je napraviti model za kompresiju slike, takav model ostvaren je korištenjem konvolucijskih neuronskih mreža kao autoenkodera.

Zbog korištenja konvolucijskih neuronskih mreža ograničeni smo na fiksne veličine ulaznih podataka (dalje obrađeno u nastavku), ali to omogućuje da odredimo kakvo "usko grlo" želimo u autoenkoderu. Razinu kompresije moguće je podesiti omjerom broja neurona na ulazu i broja neurona u "uskom grlu" mreže.

Početni model sastavljen je kao simetrična mreža kao što je prikazano na slici 3.2 te je služio kao provjera koncepta mreže te učenje izgradnje autoenkodera. Model se sastoji od ulaznog sloja veličine 64x64 piksela u 3 kanala (RGB), koristi se još i oznaka 64x64x3. Nakon toga slijede 2 konvolucijska sloja veličine 32x32x64, jedan sloj 16x16x128 te jedan sloj 8x8x128 koji predstavlja kodiranu reprezentaciju ulaza. Nakon toga slijedi dekoder koji se sastoji od Upsampling slojeva te konvolucijskih slojeva simetričnih veličina kao i enkoder. Odsječak koda za početni model mreže nalazi se u dodatku A.



Slika 3.2: Izgled početne verzije mreže

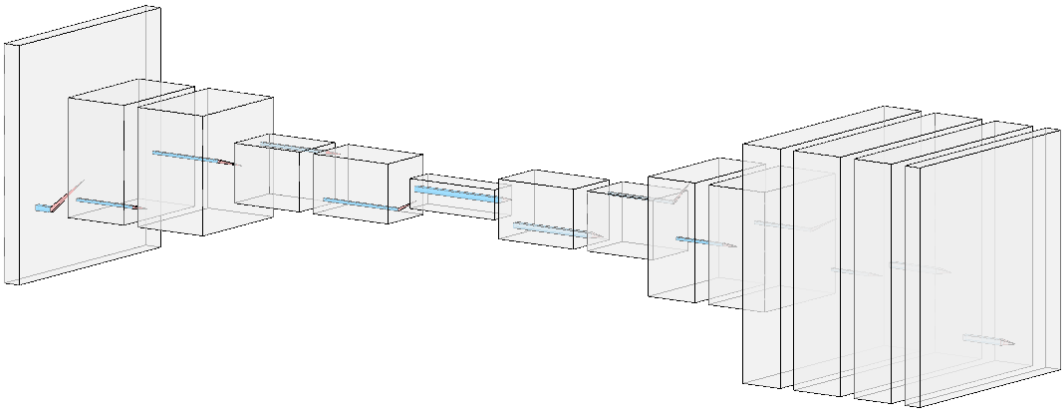
3.4.3. Konačna mreža

Kako je početna mreža samo provjera koncepta potrebno je konstruirati kvalitetniju mrežu. Prve iteracije mijenjaju samo parametre slojeva, kasnije se dodaje više slojeva. Samim povećanjem mreže kvaliteta kompresije se nije poboljšala, ali zato se drastično povećalo vrijeme treniranja mreže.

Konačna verzija mreže (slika 3.3) uzima prijašnju ideju i dodaje SubPixel Upscaling slojeve kako bi dekodiranje mreže bilo učinkovitije. Također mreža postaje asimetrična tako da enkoder ima više slojeva za bolju kompresiju slike i dekoder više slojeva kako bi dekodirao kvalitetniju reprezentaciju enkodirane slike.

Veličina ulazne slike ograničena je na 64x64 piksela te se zbog toga početna slika mora rastaviti na skupove veličine 64x64 piksela ili manje. Ako se slika rastavi na manje dijelove tada će se prije ulaska u mrežu slika skalirati na pravilnu veličinu te proći kroz mrežu. Iz tog razloga postoji mogućnost da mreža stvori sliku veće rezolucije nego početna uz veći gubitak detalja. Poželjno je koristiti slike koje su djeljive sa 64 po visini i širini.

Konačna verzija mreže koristi ADAM metodu za optimizaciju (obrađeno u nastavku) i prilagođenu mjeru gubitka (također obrađeno u nastavku). Mreža se sastoji od ulaznog sloja dimenzije 64x64x3, dalje slijede konvolucijski slojevi u parovima dimenzija 32x32x128 i 16x16x256. Sloj koji reprezentira kodirane podatke je dimenzije 8x8x256. Dekoder sadrži jednaka 4 sloja kao i enkoder te dodaje još 3 konvolucijska sloja dimenzija 64x64x64, 64x64x32 i 64x64x16. Važno primijetiti da na slici 3.3 nema vidljivih SubPixel Upscaling slojeva no nalaze se kod svakog povećanja dimenzija. Odsječak koda modela mreže nalazi se u dodatku B.



Slika 3.3: Izgled konačne verzije mreže

4. Učenje mreže

4.1. Postupak učenja

Učenje neuronske mreže podrazumijeva postupak pronalaska optimalnih parametara/težina između slojeva. Optimalni parametri su oni koji danim modelom modeliraju skup podataka za učenje D. D je skup parova (x_i, y_i) gdje je x_i ulazna slika, a y_i izlazna slika. Model treba uz skup podataka za učenje i skup za validaciju, ako nemamo skup za validaciju model se može prilagoditi samo za ulazne podatke i neće ispravno raditi za nove podatke. Općenito je dobra praksa podijeliti 70% skupa podataka na skup za učenje, a preostalih 30% na skup za validaciju.

Kako bi našli optimalne parametre potrebna nama je ciljna funkcija, odnosno funkcija čiji rezultat nam govori koliko su trenutni parametri dobri (koliko se slike razlikuju).

4.2. Skup podataka

Kako bi mogli učiti naš model mreže potreban nam je skup podataka. Početno je uzet skup DIV2K[2]. Skup sadrži 1000 slika dimenzija 2040x1536 piksela što je preveliko za gore navedenu mrežu. Zbog toga sve su slike "razrezane" na segmente od 64x64 piksela čime dobivamo 768000 slika. Većina tih slika je jednobojnih (nebo, mrak, itd.) pa su takve slike uklonjene. Skup podatak smanjen je na otprilike 250000 slika pa je zbog toga dodan još CVRP 2018[4]. CVRP 2018 sastoji se od otprilike 3000 slika različitih dimenzija koje su također obrađene na isti način kao i DIV2K. Nakon obrade skup podataka je fiksiran na 800000 slika. Od toga 500000 slika je za učenje, a preostalih 300000 za validaciju.

4.3. Gradijentni spust

Gradijentni spust je jedan od najpopularnijih optimizacijskih algoritama pronalaska minimuma derivabilne konveksne funkcije. Traženje minimuma postignuto je tako da se u trenutnoj točki nađe smjer gradijenta funkcije te se napravi pomak u smjeru negativnom od gradijenta. Veličina pomaka je vrlo važna. Ako je pomak dovoljno mali nova vrijednost funkcije bit će manja od prethodne. Ako je prevelik možemo preskočiti minimum i dobiti još veće vrijednosti funkcije. Važno je za primijetiti ako imamo premalene pomake, pronalaska minimuma će se znatno povećati.

Neka je $f(x)$ funkcija čiji minimum tražimo, T trenutna točka (odnosno naše rješenje) u kojoj se nalazimo, t je trenutni korak, $t-1$ prethodni korak, a η je pomak algoritma. Jedna iteracija algoritma izgleda :

$$T(t) = T(t-1) - \eta f^{'}[T(t-1)] \tag{4.1}$$

Ako imamo funkciju F v više varijabli :

$$T_i(t) = T_i(t-1) - \eta \frac{\partial F}{\partial T_i}[T_i(t-1)] \tag{4.2}$$

gdje indeks i predstavlja i-tu komponentu vektora funkcije F.

Ako je η dovoljno malen vrijedit će $f(T(t)) < f(T(t-1))$ i pomak će biti u smjeru minimuma funkcije.

4.3.1. Stohastički gradijentni spust

Pretpostavimo da imamo funkciju $F(x)$ koja se sastoji od sume drugih funkcija :

$$F(x) = \sum_{r=1}^{R} F_r(x) \tag{4.3}$$

gdje R može biti u redu veličine tisuća. Tada izračun gradijenta G_F, koji je suma svih gradijenata G_{F_r} funkcija F_R, možemo zamijeniti sumom podskupa tih "parcijalnih" gradijenata. Stohastički gradijent je onda $\hat{G} = \sum_{(podskup\ od\ R)} G_{F_r}$ koji je znatno brži za izračunati od gradijenta G_F. Podskup se obično uzima nasumično za svaku iteraciju algoritma.

Stohastički gradijentni spust (SGD) obično zahtijeva više koraka kao bi pronašao minimum, ali zbog bržeg izračuna jedan je od najpopularnijih odabira.

4.3.2. ADAM

ADAM je nadogradnja na algoritam stohastičkog gradijentnog spusta gdje se gradijent računa na temelju prošlog koristeći tehniku $momenta$.

Do sada nam je pomak bio definiran kao $\Delta x = x_t - x_{t-1}$, sljedeći pomak je $\Delta x + 1 = -\eta G_F(x_t)$. U $moment$ [20] metodi korak je linearna kombinacija trenutnog gradijenta i prošlog koraka $\Delta x + 1 = -\eta G_F(x_t) + \alpha \Delta x$

ADAM metoda koristi prilagodljivu procjenu $momenta$ [9], odnosno linearnih kombinacija trenutnog gradijenta i prošlog koraka.

4.4. Funkcija gubitka

Kako bi mogli učiti neuronsku mrežu trebamo funkciju koja će odrediti koliko dobar rezultat daje mreža. Takvu funkciju zovemo funkcija gubitka. Funkcija gubitka uzima kao ulaz jednu ili više varijabli (npr. izlaz neuronske mreže), a na izlazu daje jedan realan broj koji opisuje "kvalitetu" rješenja. Cilj gore navedenih algoritama (SGD, ADAM) je da minimiziraju funkciju gubitka.

4.4.1. Srednja kvadratna pogreška

Srednja kvadratna pogreška ili MSE mjeri razlike između procjenitelja i onoga što se procjenjuje. MSE je mjera kvalitete procjenitelja.

Neka je $\hat{Y}$ vektor n predikcija, a Y vektor promatranih vrijednosti predviđene varijable, MSE se onda računa kao :

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{4.4}$$

Ako imamo 2 iste slike te jednoj promijenimo nijanse boje za neki mali faktor, slike naizgled ljudskome oku izgledaju skoro pa identično, ali srednja kvadratna pogreška može poprimiti velik broj. To je jedna od mana srednje kvadrante pogreške te razlog zašto se primarno ne koristi kod slika u boji. Moguća je uporaba kod monokromatskih slika bez dodatnih funkcija. Kako MSE nije pogodan za funkciju gubitka istražene su ostale opcije.

4.4.2. Strukturalna sličnost

Strukturalna sličnost ili SSIM indeks je metoda za predviđanja percipiranje kvalitete digitalnih slika. SSIM se koristi za mjerenje sličnosti između dvije slike. Mjerenje slike temelji se na početnoj ne komprimiranoj slici kao referenci. Metoda je osmišljena kako bi unaprijedila tradicionalne metode kao što su omjer vršnog signala i šuma (PSNR)[7] i srednje kvadratne pogreške (MSE)

Neka su x i y početna slika i slika dobivena nakon kompresije i dekompresije. Te neka su μ_x i μ_y srednje vrijednosti od x i y, σ_x^2 i σ_y^2 varijance od x i y te σ_{xy} kovarijanca. Onda je SSIM indeks :

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2\mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.5}$$

gdje su $c_i = (k_i L)^2$ varijable za stabilizaciju podjele, L je dinamički raspon vrijednosti piksela. Ako je x_i vrijednost pojedinog piksela onda se μ_x računa prema formuli :

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{4.6}$$

Koristimo standardnu devijaciju kao procjenu kontrasta signala slike. Neodređena procjena u diskretnom obliku računa se prema :

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)^2 \tag{4.7}$$

Kovarijanca σ_{xy} u diskretnom obliku definirana se kao :

$$\sigma_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y) \tag{4.8}$$

4.4.3. Prilagođena mjera gubitka

Neuronska mreža korištena u ovom radu koristi prilagođenu mjeru gubitka, odnosno funkciju C. Funkcija C je linearna kombinacija srednje kvadratne pogreške i strukturne različitosti. Strukturalna različitost (DSSIM) je metrička vrijednost koja proizlazi iz strukturne sličnosti.

$$DSSIM(x,y) = \frac{1 - SSIM(x,y)}{2} \tag{4.9}$$

Sada možemo definirati C kao :

$$C(x,y) = k_1 DSSIM(x,y) + k_2 MSE(x,y),\ k_1 + k_2 = 1 \tag{4.10}$$

Za mrežu korištenu u ovom radu odabrani su parametri $k_1 = 0.80$ i $k_2 = 0.20$ koji su pokazali najbolji omjer percipirane sličnosti slike i točnosti boja u testiranju parametara.

5. Rezultati

Mreža je trenirana s 850 prolaza kroz skup za učenje. Nakon svake epohe tijekom učenja pratila se pogreška na skupu za učenje te na skupu za validaciju. Na slici 5.4 vidimo promjenu pogreške za vrijeme učenja nad skupom za učenje i validaciju. Vidimo da su linije za validacijski skup i skup za učenje jako bliza jedna drugoj. Time vidimo da naučena mreža dobro generalizira, te da nije došlo do prevelikog prilagođavanja na skup za učenje. Pogreška u zadnjoj iteraciji je 0.01.

Za usporedbu kvalitete kompresije koristit ćemo SSIM indeks, te sliku iz Kodak skupa slika PhotoCD PCD0992[10]. Također ćemo koristi PSNR[7] za usporedbu. Kako bi se mogli snalaziti u daljem tekstu mreža opisana u 3.4.3 nazvana je "M3IC"

Tablica 5.1: Usporedba različitih metoda kompresije na slici 5.1

Metoda	SSIM	PSNR [dB]	Veličina [KB]	Kompresija
Original	1		598	1
JPEG (100%)	0.9697787	41.376	109	5.48
JPEG (50%)	0.9152304	34.836	31.3	19.11
JPEG (20%)	0.8573988	31.963	15.9	37.61
WebP (100%)	0.9919465	48.231	152	3.92
WebP (50%)	0.8958998	34.641	19.3	30.98
M3IC	0.9342860	33.744	466	1.28

Slika 5.1: Slika iz skupa Kodak [10] korištena u mjerenju rezultata

Tipične vrijednosti za PSNR kod kompresije slika s gubitkom podataka su između 30 i 50 dB[7].

Vidimo kako WebP kod 100% kvalitete može kompresirati sliku na skoro $1/4$ veličine uz zanemarivo mali gubitak kvalitete slike.

Na slici 5.2 možemo vidjeti razliku između originale slike te slike dobivene nakon M3IC i JPEG s 20% kvalitete. Razlika je dobivena kao prosječna razlika svih kanala slike (R,G,B) nad svakim pojedinim parom piksela u obje slike. Nakon toga slika je invertirana te joj je kontrast jako povećan kako bi se vidjele razlike. Zanimljivo je primijetiti kako JPEG skoro pa nema nikakve razlike kod pozadine dok je kod oštrih rubova, kao što je kosa, iznimno vidljiva. Za razliku M3IC imam kontinuirani gubitak po cijeloj slici uz manje poteškoće s rubovima.

Zbog same arhitekture mreže usporedit ćemo još jednu sliku nativne rezolucije (64x64) istim postupkom kao za slike iz skupa Kodak [10]. Slika je dobivena računalnim alatima, te je svakome pikselu u slici dodijeljena slučajna boja.

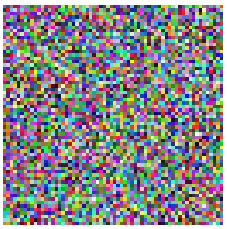
Tablica 5.2: Usporedba različitih metoda kompresije na slici 5.3

Metoda	SSIM	PSNR [dB]	Veličina [KB]	Kompresija
Original	1		29.3	1
JPEG (100%)	0.9965509	36.295	6.88	4.26
JPEG (50%)	0.9234026	22.417	1.84	15.92
JPEG (20%)	0.7016965	16.958	1.01	29.01
WebP (100%)	0.9997790	47.507	4.96	5.91
WebP (50%)	0.9892186	31.175	2.38	12.31
M3IC	0.8165394	18.321	2.92	10.03

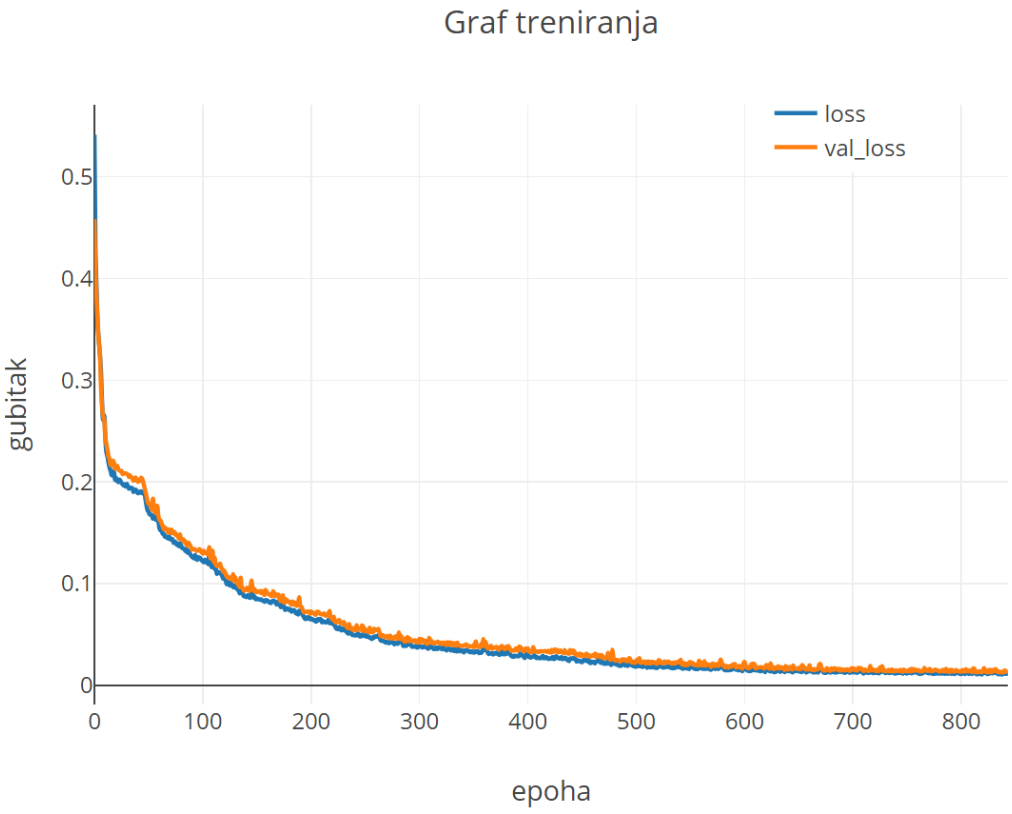
U ovom primjeru imamo puno bolju kompresiju M3IC mreže, ali dosta gubimo na kvaliteti slike. To je pretežito zbog činjenice da je mreža uvijek učena na pravim slikama, a ne računalno generiranim. Također je zanimljivo primijetiti kako opet WebP pobjeđuje u kompresiji i kvaliteti slike.

Usprkos vrlo malene pogreške tijekom treniranja i validacije mreža ne postiže impresivne rezultate. Djelomično tome pridonosi arhitektura mreže. Autoenkoderi se više koriste za određene kategorije te nisu najbolji za generalizaciju. Također moderni pristupi kao već obrađeni WaveOne[14] koriste više različitih slojeva, svaki sa svojom namjenom.

Slika 5.2: Razlika između originala i mreže M3IC (gore) te JPEG kod 20% kvalitete (dole)



Slika 5.3: Računalno generirana slika korištena u mjerenju rezultata



Slika 5.4: Promjena pogreške na validacijskom skupu (val_loss) i skupu za učenje (loss) tijekom učenja.

6. Zaključak

Prikazane su različite metode kompresije slika s naglaskom na kompresiju neuronskim mrežama. Proučene su neke specifičnosti neuronskih mreža i razrađeni su detalji algoritma gradijentnog spusta i stohastičkog gradijentnog spusta. Također je opisan model autoenkodera te razvijena implementacija konvolucijskog autoenkodera.

Mreža je najprije razvijena na konceptualnoj razini kao autoenkoder koji koristi konvolucijske slojeve. Zatim je trenirana na malom skupu testnih podataka kako bi utvrdili vrijednost koncepta. Nakon toga mreža je unapređivana 15 puta na razina arhitekture uz konstantno povećanje testnog skupa podataka. U svakoj iteraciji učenja mreže cijeli proces učenja je pomno praćen i analiziran. Iz analize konačne mreže je zaključeno da je generalizacija mreže skoro pa odlična jer je razlika između gubitka tijekom učenja i validacije minimalna kroz cijeli proces učenja. Rezultat od 0.02 gubitka (korištenjem funkcije gubitka opisane u 5.4) na skupu za validaciju je, iako izvrstan, ograničen na slike specifične rezolucije.

Model je razvijen da se može pokrenuti na osobnom računalu. Ako računalo ima grafičku karticu model će koristiti mogućnosti kartice. Model postiže otprilike 2500 kompresija slika, rezolucije 64x64 piksela, u minuti na grafičkoj kartici nižeg ranga što je zadovoljavajuće.

U daljnjem radu se preporučuje stvaranje arhitekture prilagodljive svakoj rezoluciji te mogućnost kompresije video zapisa. Također se preporučuje prelazak s autoenkodera na rekurentne neuronske mreže (RNN) koje pokazuju bolje rezultate [14] [8].

LITERATURA

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

[2] Eirikur Agustsson i Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. U *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[3] François Chollet et al. Keras. `https://keras.io`, 2015.

[4] Computer Vision Lab of ETH Zurich. Cvpr 2018 dataset. `http://www.compression.cc/challenge/`, 2018. [Online].

[5] Chao Dong, Chen Change Loy, Kaiming He, i Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[6] Google. A new image format for the web. `https://developers.google.com/speed/webp/`, 2016. [Online; accessed 24-May-2018].

[7] Alain Hore i Djemel Ziou. Image quality metrics: Psnr vs. ssim. U *Pattern recognition (icpr), 2010 20th international conference on*, stranice 2366–2369. IEEE, 2010.

[8] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, i George Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. *arXiv preprint arXiv:1703.10114*, 2017.

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

[10] Kodak. Kodak lossless true color image suite, 1999. URL `http://r0k.us/graphics/kodak/`. [Online].

[11] John Miano. *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.

[12] Richard O'Neil et al. Convolution operators and $l(p, q)$ spaces. *Duke Mathematical Journal*, 30(1):129–142, 1963.

[13] William B Pennebaker i Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[14] Oren Rippel i Lubomir Bourdev. Real-time adaptive image compression. U *International Conference on Machine Learning*, 2017.

[15] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, i Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, stranice 1874–1883, 2016.

[16] The Keras Blog. Building autoencoders in keras. `https://blog.keras.io/building-autoencoders-in-keras.html`, 2016. [Online; accessed 23-May-2018].

[17] Wikipedia contributors. Autoencoder — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=841771807`, 2018. [Online; accessed 25-May-2018].

[18] Wikipedia contributors. Data compression — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Data_`

compression&oldid=841580384, 2018. [Online; accessed 24-May-2018].

[19] Wikipedia contributors. Recurrent neural network — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=844462682, 2018. [Online; accessed 11-June-2018].

[20] Wikipedia contributors. Stochastic gradient descent — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Stochastic_gradient_descent&oldid=843200006. [Online; accessed 27-May-2018].

POPIS SLIKA

1.1. Usporedba kvalitete kompresije kod PNG (desno) i JPEG (lijevo) s 10% kvalitete na slici iz skupa podataka Kodak [10] 2
1.2. Izgled jednostavne umjetne neuronske mreže 2
1.3. Primjer 2d konvolucije s 3x3 filterom 3

2.1. Usporedba 4 modela kompresije podataka[14] 6

3.1. Primjer autoenkodera koji koristi slike [16] 8
3.2. Izgled početne verzije mreže 9
3.3. Izgled konačne verzije mreže 10

5.1. Slika iz skupa Kodak [10] korištena u mjerenju rezultata 15
5.2. Razlika između originala i mreže M3IC (gore) te JPEG kod 20% kvalitete (dole) 17
5.3. Računalno generirana slika korištena u mjerenju rezultata 17
5.4. Promjena pogreške na validacijskom skupu (val_loss) i skupu za učenje (loss) tijekom učenja. 18

Dodatak A
Dio izvornog koda za početnu mrežu

```
#Encoder
encoder_input = Input(shape=(input_shape))
encoder_output = Conv2D(64, (2,2), activation='relu', padding='same', strides=2)(encoder_input)
encoder_output = Conv2D(64, (2,2), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(128, (2,2), activation='relu', padding='same', strides=2)(encoder_output)
encoder_output = Conv2D(128, (2,2), activation='relu', padding='same', strides=2)(encoder_output)

#Decoder
decoder_output = UpSampling2D((2, 2))(encoder_output)
decoder_output = Conv2D(128, (2,2), activation='relu', padding='same')(decoder_output)
decoder_output = UpSampling2D((2, 2))(decoder_output)
decoder_output = Conv2D(64, (2,2), activation='relu', padding='same')(decoder_output)
decoder_output = Conv2D(64, (2,2), activation='relu', padding='same')(decoder_output)
decoder_output = UpSampling2D((2, 2))(decoder_output)
decoder_output = Conv2D(3, (2,2), activation='relu', padding='same')(decoder_output)

autoencoder = Model(inputs=encoder_input, outputs=decoder_output)

autoencoder.compile(optimizer='adam', loss='mse')
```

Dodatak B
Dio izvornog koda za konačnu mrežu

```
#Encoder
encoder_input = Input(shape=(input_shape))
encoder_output = Conv2D(64, (2,2), activation='relu', padding='same', strides=2)(encoder_input)
encoder_output = Conv2D(128, (2,2), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(128, (2,2), activation='relu', padding='same', strides=2)(encoder_output)
encoder_output = Conv2D(256, (2,2), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(256, (2,2), activation='relu', padding='same')(encoder_output)
encoder_output = Conv2D(256, (2,2), activation='relu', padding='same', strides=2)(encoder_output)

#Decoder
decoder_output = Conv2D(128, (2,2), activation='relu', padding='same')(encoder_output)#
decoder_output = SubPixelUpscaling(scale_factor=2)(decoder_output)
decoder_output = Conv2D(64, (2,2), activation='relu', padding='same')(decoder_output)#
decoder_output = SubPixelUpscaling(scale_factor=2)(decoder_output)
decoder_output = Conv2D(32, (2,2), activation='relu', padding='same')(decoder_output)
decoder_output = Conv2D(16, (2,2), activation='relu', padding='same')(decoder_output)
decoder_output = Conv2D(3, (2, 2), activation='tanh', padding='same')(decoder_output)#
#decoder_output = SubPixelUpscaling(scale_factor=2)(decoder_output)

autoencoder = Model(inputs=encoder_input, outputs=decoder_output)

loss_func = DSSIMObjective()

def custom_loss(y_true, y_pred):
            loss1=loss_func(y_true,y_pred)
            loss2=losses.mean_squared_error(y_true, y_pred)
            return 0.85*loss1+0.15*loss2

adam = keras.optimizers.Adam(lr=0.01, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.001, amsgrad=False)
autoencoder.compile(optimizer='adam', loss=custom_loss)
```

Sažimanje slika neuronskim mrežama

Sažetak

U ovom radu prikazane su različite metode kompresije slika s naglaskom na kompresiju neuronskim mrežama. Proučene su neke specifičnosti neuronskih mreža i razrađeni su detalji algoritma gradijentnog spusta i stohastičkog gradijentnog spusta. Također je opisan model autoenkodera te razvijena implementacija konvolucijskog autoenkodera.

Ključne riječi: neuronske mreže, konvolucijske neuronske mreže, autoenkoder, kompresija slike, gradijentni spust, strojno učenje

Image compression with neural networks

Abstract

In this paper different methods of image compression with an emphasis on image compression with neural networks are shown. Some specifics of neural networks have been studied and details of the gradient descent and stochastic gradient descent algorithms have been elaborated. Also described is the autoencoder model and a conventional autoencoder model has been implemented.

Keywords: neural networks, convulutional neural networks, autoencoder, image compression, gradient descent, machine learning