Uporaba konvolucijskih neuronskih mreža za kolorizaciju crno-bijelih slika

Gregorić, Marin

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
SMJER INFORMATIKA (NASTAVNIČKI)
Ak. god. 2022./2023.

Marin Gregorić

Uporaba konvolucijskih neuronskih mreža za kolorizaciju crno-bijelih slika

Diplomski rad

Mentor : prof. dr. sc. Sanja Seljan

Zagreb, kolovoz 2023.

Izjava o akademskoj čestitosti

Izjavljujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

Marin Gregorić

Sadržaj

1. Uvod

Razvojem tehnologije, snažna računala su sve dostupnija prosječnom čovjeku. Postavlja se pitanje je li danas, napretkom tehnologije, moguće stvarati i učiti sustave osnovane na neuronskim mrežama na osobnom računalu. Također, postavlja se pitanje mogu li se dizajnom vlastite arhitekture modela neuronskih mreža te korištenjem naprednih metoda učenja i dizajna modela postići rezultati generiranog (modelom stvorenog) sadržaja usporedivog sa suvremenim profesionalnim sustavima kao što je *DeOldify* sustav. U radu se kolorizacija postavlja kao problem računalnog vida te se pokušava dizajnirati model konvolucijskih neuronskih mreža koji će prepoznati sadržaj crno-bijele slike te će ju vjerno obojati. Također, kroz rad će se i izraditi poseban skup podataka za treniranje modela za zadatke kolorizacije. Cilj rada je dizajnirati, izraditi i trenirati model konvolucijskih neuronskih mreža za kolorizaciju na osobnom računalu, izraditi skup podataka za zadatak kolorizacije te usporediti dobivene rezultate s drugim konvolucijskim modelima za kolorizaciju.

Rad se sastoji od kratkog pregleda teorije neuronskih mreža kroz definiranje strojnog učenja, definiranje neuronskih mreža te definiranje načina učenja neuronskih mreža putem objašnjenja metoda i algoritama učenja, objašnjenja aktivacijskih funkcija i funkcija gubitka. U radu se, također, prikazuje nekoliko popularnih arhitektura modela dubokog učenja sa svrhom stvaranja temeljnog znanja za stvaranje projekta u kojem se izrađuje vlastiti model konvolucijskih neuronskih mreža za kolorizaciju crno-bijelih slika.

Kroz teorijski dio predstavljaju se umjetni neuroni kao gradivne jedinice neuronskih mreža i funkcije koje služe kao simulacije aktiviranja neurona (aktivacijske funkcije) poput *ReLU* (engl. *rectified linear unit*) i *sigmoidalne* (engl. *sigmoidal*) funkcije. Predstavljeno je i kako neuronske mreže mogu računati svoju pogrešku na određenom zadatku (funkcije gubitka) funkcijama poput MSE (engl. *mean squred error*) te kako uče iz pogrešaka postupkom optimiziranja svojih vrijednosti algoritmima za učenje neuronskih mreža poput stohastičkog gradijentnog spusta (engl. *stohastic gradient descent*) ili *Adam* optimizatora (engl. *optimizer*). Također, prikazat će se nekoliko osnovnih modela konvolucijskih neuronskih mreža

U projektnom dijelu se izrađuje poseban skup podataka za zadatke kolorizacije bez viška nepotrebnog sadržaja za treniranje kolorizacijskog modela. Kroz projekt se osmišljava i izrađuje model za kolorizaciju i sustav u kojem se trenira i djeluje. Pregledavaju se tehnike obrade slika za postizanje boljeg treniranja te se opisuje sam proces treniranja modela kao i potencijalna ograničenja.

2. Umjetna inteligencija i strojno učenje

Ovo poglavlje predstavlja kratki pregled strojnog učenja kao dijela umjetne inteligencije te se posebni naglasak stavlja na nadzirano strojno učenje.

2.1. Osnovne definicije

Ovaj rad bavi se izradom sustava osnovanog na neuronskim mrežama koje su jedna od metoda strojnog učenja koje pripada znanosti umjetne inteligencije. Za početak, potrebno je postaviti teorijsku podlogu kroz osnovne definicije. Leksikografski zavod Miroslav Krleža (*umjetna inteligencija*, bez dat.) nudi sljedeću definiciju umjetne inteligencije: „Umjetna inteligencija je dio računalstva koji se bavi razvojem sposobnosti računala da obavlja zadaće za koje je potreban neki oblik inteligencije."

Jedna od grana umjetne inteligencije je strojno učenje. Alypadin (2004) definira strojno učenje (engl. *machine learning*) kao programiranje računala za optimizaciju kriterija koristeći ponuđene podatke ili prošlo iskustvo. Također, navodi da je model (engl. *model*) strojnog učenja definiran određenim parametrima te on uči tako što računalni program optimizira parametre modela koristeći podatke za učenje ili prošlo iskustvo. Taj model može biti prediktivan (engl. *predictive*) i predviđati budućnost, ili opisni (engl. *descriptive*) i dobivati znanje iz podataka, a u nekim slučajevima može biti i oboje.

Russell i Norvig (2020), kako bi mogli definirati što je strojno učenje, prvo definiraju učenje - sposobnost agenta, u ovom slučaju računala, da promatranjem svijeta oko sebe poboljša svoje performanse. Potom definiraju strojno učenje kao sposobnost da taj agent (računalo) promatranjem podataka izgradi model osnovan na postojećim podacima, te da koristi taj model kao hipotezu o svijetu i kao softver koji može rješavati probleme.

Prema Russel i Norvig (2020), takav je proces moguće opisati kao postepenu modifikaciju funkcije $h(x)=\hat{y}$ s ciljem da aproksimira funkciju $f(x)=y$ dok se ne postigne $\hat{y}=y$ ili neki drugi kriterij gdje x označava ulazni podatak ili skup podataka (engl. *input*), h hipotezu ili funkciju koju tvori model, y označava temeljnu istinu (engl. *ground truth*), $\hat{y}$ označava generirani izlaz (engl. *output*), a f je originalna funkcija koju model pokušava aproksimirati učenjem. Učenje, također zvano i treniranje (engl. *training*), se odvija učitavanjem podataka iz skupa podataka za treniranje ili za učenje (engl. *traning dataset*) u model sve dok se ne postigne određeni rezultat modela na skupu podataka za testiranje, to jest, validaciju (engl. *test, validation*

dataset) ili sve dok se ne postigne neki drugi kriterij. Sustav je moguće i pretrenirati (engl. *overfitting*) jer je moguće da nakon previše prijelaza preko skupa podataka, model izgubi mogućnost generaliziranja i postane stručnjak za obradu podataka iz skupa za učenje na kojemu je treniran, dok nove podatke loše obrađuje. (Dalbelo Bašić i sur., 2008) Drugim riječima, pretreniranje je problem koji nastaje kod velikog broja prijelaza preko skupa podataka za treniranje čime model postaje vrlo dobar u obradi podataka iz skupa za treniranje, ali postaje loš u obradi novih podataka s kojima se još nije susreo.

Prema Russel i Norvig (2020), strojno učenje može se podijeliti kroz tri načina učenja ili optimizacije modela: nadzirano učenje (engl. *supervised learning*), nenadzirano učenje (engl. *unsupervised learning*) i učenje uz potporu ili potporno učenje (engl. *reinforcement learning*).

Od navedenih, najčešći oblik strojnog učenja je nadzirano strojno učenje (LeCun i sur., 2015). Nadzirano učenje je naziv za oblik strojnog učenja gdje agent (računalo) promatra odnose ulaznih i izlaznih podataka te pokušava stvoriti aproksimaciju funkcije $f(x)=y$ koja iz ulaza x tvori izlaz y (Russell i Norvig, 2020). U nadziranom učenju potrebni su označeni podaci (engl. *labeled data*) koji tvore ulaz i izlaz funkcije – što znači da su poznati i x i y u funkciji $f(x)=y$. Primjer nadziranog strojnog učenja bio bi slučaj gdje su modelu dani podaci o vremenskim uvjetima na određenom području kao ulaz, i podaci o poljoprivrednom urodu na istom području kao izlaz. Modelu je cilj naučiti odnos između ulaznih i izlaznih podataka kako bi mogao predvidjeti potencijalni urod s obzirom na nove vremenske uvjete. Nadzirano strojno učenje može se odnositi i na tekstualne podatke. Primjer nadziranog strojnog učenja nad tekstualnim podacima je, na primjer, model za razlikovanje zlonamjernih (phishing poruka) od legitimnih poruka, kao, na primjer, u radovima: Dunđer i suradnici (2023), Seljan i suradnici (2023), Kovač i suradnici (2022).

U nenadziranom učenju agent uči uzorke u podacima bez eksplicitnih povratnih informacija (Russell i Norvig, 2020). Cilj modela u takvome sustavu je prepoznati veze u neoznačenim podacima (engl. *unlabeled data*) - nije poznat odnos između x i y u funkciji $f(x)=y$, nego su samo dostupni podaci između kojih se, učenjem, stvaraju veze i odnosi među njima. Primjer sustava nenadziranog učenja je, na primjer, zadatak gdje model pronalazi sličnosti u skupu podataka kupaca za stvaranje boljih preporuka proizvoda kupcima.

Učenje uz potporu ili potporno učenje podrazumijeva agenta koji uči uz pomoć vanjskih podražaja poput nagrada i kazni (Russell i Norvig, 2020). Takvi sustavi uče na način da su im davani ili oduzimani bodovi s obzirom na njegovo djelovanje. Primjer takvoga sustava bio bi

sustav za igranje videoigara u kojem sustav biva nagrađivan za svoju sposobnost igranja videoigre čime model potencijalno stvara idealan način rješavanja zadataka videoigre.

2.2. Nadzirano strojno učenje

Nadzirano strojno učenje kao oblik strojnog učenja korištenog u ovome radu traži detaljniji pregled. Kao što je navedeno, nadzirano učenje osnovano je na učenju odnosa između skupova označenih podataka. Postoji više oblika nadziranog strojnog učenja, a dva najprisutnija su klasifikacija (engl. *classification*) i regresija (engl. *regression*) (Russell i Norvig, 2020).

Klasifikacija je naziv za oblik nadziranog strojnog učenja gdje je sustavu cilj otkriti odnose između ulaznih podataka i definiranih klasa ili kategorija, te s onime što je naučio odrediti gdje novi ulazni podatak pripada (Alpaydin, 2004). Primjer klasifikacije bio bi sustav za prepoznavanje slika, to jest, onoga što se na njima nalazi - sustav uči odnose između slika i njihovih oznaka (klasa) s ciljem da točno označi nove slike. U skupu podataka za treniranje, slika automobila bila bi označena kao član klase „automobil“ dok bi slika mačke bila označena kao, na primjer, član klase „životinje“. Tijekom treniranja model bi modificiranjem parametara učio odnose između slika i oznaka (engl. *label*) da bi po završetku treniranja mogao sam klasificirati, to jest, označiti slike koje nisu dio podataka s kojima se već susreo.

Regresija je također jedan od primjera nadziranog strojnog učenja gdje sustav uči odnose između raznih označenih podataka, no cilj sustava nije otkriti klasu ulaznih podataka kao u klasifikaciji, nego određenu vrijednost koju tvori funkcija iz ulaznih podataka (Alpaydin, 2004). Primjer regresije bio bi prethodno navedeni primjer s poljoprivrednim urodom na određenom području.

3. Umjetne neuronske mreže i duboko učenje

U ovome poglavlju bit će predstavljena osnovna teorijska podloga neuronskih mreža i dubokog učenja kao nastavka razvoja sustava neuronskih mreža s posebnim naglaskom na konvolucijske neuronske mreže.

3.1. Osnove neuronskih mreža

Jedna od suvremenih metoda dizajniranja modela strojnog učenja su umjetne neuronske mreže (engl. *artificial neural networks*). Dalbelo Bašić i suradnici (2008) navode kako je umjetna neuronska mreža „u širem smislu riječi umjetna replika ljudskog mozga kojom se nastoji simulirati postupak učenja.“ Drugim riječima, neuronska mreža je skup međusobno povezanih jednostavnih procesnih elemenata čija se funkcionalnost temelji na biološkom neuronu.

Dalbelo Bašić i suradnici (2008) također objašnjavaju i kako je biološki neuron stanica u ljudskome mozgu sastavljena od četiri osnovna dijela: soma (tijelo), skupova dendrita (ogranaka), aksona (cjevčice koje prenose električne poruke) i niza završnih članaka. Biološki neuron sadrži informaciju u obliku električnog potencijala između unutrašnjeg i vanjskog dijela stanice na koju utječu prethodni neuroni s kojima komunicira putem dendrita i sinapsa, dok on sam utječe na daljnje neurone putem aksona i završnih članaka. Taj put informacija je jednosmjeran i ovisi o aktivaciji ili paljenju neurona.

Na sličan način radi i umjetni neuron. Na ulazu se ulazni signali množe s težinskim faktorima (engl. *weights*) te se potom sumiraju i uspoređuju s određenim pragom u umjetnom tijelu neurona, a ako je sumirana vrijednost iznad određenog praga - neuron se aktivira i šalje informacije dalje (Dalbelo Bašić i sur., 2008). Drugi naziv za takvu jednostavnu aproksimaciju neurona kao osnovnog procesnog elementa je perceptron koji čini najjednostavniji oblik umjetnog neurona (Alpaydin, 2004). Funkcija koja definira aktivaciju neurona zove se prijenosna, transfer ili aktivacijska funkcija (engl. *activation funciton*) (Dalbelo Bašić i sur., 2008). Ipak, Russell i Norvig (2020) navode da je sličnost s biološkim neuronima samo površinska. Neuronske mreže je možda najbolje zamisliti kao pregledan način određivanja fleksibilnog skupa funkcija izgrađenih od računskih blokova zvanih neuronima (Roberts i sur., 2021).

Za rješavanje nelinearnih problema (problema gdje ulaz i izlaz nisu u linearnom odnosu) poput nelinearne regresije, zamišljene su višeslojne neuronske mreže ili višeslojni perceptron (engl.

multilayer perceptron - MLP). Višeslojne neuronske mreže sastoje se od više slojeva (engl. *layers*) neurona između ulaznog i izlaznog sloja, a takvi slojevi zovu se skriveni slojevi (engl. *hidden layers*) (Alpaydin, 2004). U klasičnoj višeslojnoj umjetnoj neuronskoj mreži, poput višeslojnog perceptrona, signali propagiraju jednosmjerno od ulaza mreže ka izlazu, zbog čega se često zovu i jednosmjerne ili acikličke mreže (engl. *feedforward network*) (Roberts i sur., 2021).

Mreže s više slojeva su predmet istraživanja dubokog učenja (engl. *deep learning*) gdje naziv „duboko“ dolazi iz činjenice da se obično radi o mnogo slojeva između ulaza i izlaza iz mreže koji time stvaraju dubinu modela (Russell i Norvig, 2020). Duboko učenje je grana strojnog učenja koja koristi neuronske mreže kao aproksimatore funkcija s naglaskom na slaganje mnogo slojeva strukturalno sličnih komponenti (Roberts i sur., 2021). Dodavanjem slojeva i jedinica u tom sloju duboka mreža može reprezentirati sve kompleksnije funkcije (Goodfellow i sur., 2016). Danas se nazivi duboke mreže i neuronske mreže često koriste naizmjenično.

3.2. Funkcije gubitka

Neuronske mreže uče podešavanjem težinskih faktora kroz iterativni postupak predočavanja ulaznih podataka i eventualno očekivanih izlaza sve dok izlaz iz mreže ne bude zadovoljavajući. Taj postupak zove se učenje ili treniranje neuronske mreže (Dalbelo Bašić i sur., 2008). Odnos između očekivanog izlaza i stvarnog izlaza naziva se pogreška ili gubitak (engl. *loss*).

Taj odnos može se kvantificirati funkcijama gubitka (engl. *loss function*). Russell i Norvig (2020) definiraju funkciju gubitka $L(x, y, \hat{y})$ kao količinu izgubljene korisnosti (engl. *utility*) predviđanjem $h(x)=\hat{y}$ dok je točan odgovor $f(x)=y$ gdje je x ulaz, y temeljna istina, h hipoteza, a $\hat{y}$ generirani izlaz.

Yathish (2022) navodi kako je funkcije gubitka moguće podijeliti po vrsti nadziranog učenja za koje se koriste - na funkcije gubitka za regresiju i funkcije gubitka za klasifikaciju.

Popularne regresijske funkcije gubitka su:

„prosječna apsolutna pogreška“ ili *MAE* (engl. *Mean Absolute Error*), ponekad zvana i *L1* funkcija gubitka

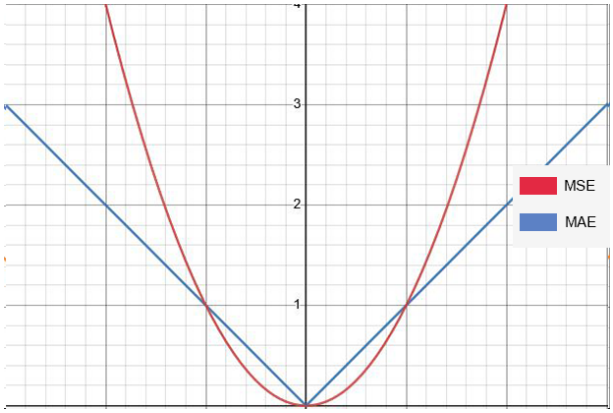
„prosječna kvadratna pogreška“ ili *MSE* (engl. *Mean Squared Error*), ponekad pisana kao *L2* gubitak (Yathish, 2022).

MAE i *MSE* funkcije računaju prosječnu razliku između rezultata i cilja, no razlikuju se u osjetljivosti na neuobičajene podatke (engl. *outliers*). *MSE* funkcija razliku između izlaza i istine kvadrira, čime je osjetljivija od *MAE* funkcije (Yathish, 2022). Formula koju navodi Seif (2022), a čija notacija je modificirana kako bi formula odgovarala notaciji u ostatku rada, funkcije MAE i MSE mogu se zapisati kao:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

Slika 1 prikazuje graf *MAE* (plave boje) i *MSE* (crvene boje) funkcija čime se može vidjeti način mapiranja vrijednosti izlaza modela i temeljne istine. Iz slike se može iščitati kako je *MSE* osjetljiviji na veće razlike između y i $\hat{y}$ te je manje osjetljiv na male razlike, dok je *MAE* linearan. Može se vidjeti da je graf *MSE* funkcije strmiji u prostoru iznad 1 i glađi u prostoru ispod 1 naspram *MAE* funkcije, čime se može vidjeti njegova veća osjetljivost na velike razlike i njegova manja osjetljivost na male razlike između y i $\hat{y}$.



Slika 1 *MAE i MSE funkcije - graf izrađen prema Seif (2022)*

Prema Seif (2022) jedna od funkcija gubitka koja pokušava riješiti nedostatke *MSE* i *MAE* funkcija je *Huberov* gubitak (engl. *Huber Loss*) koja nudi prednosti u odnosu na *MSE* i *MAE* funkcije tako što ih spaja u jednu funkciju, gdje posebno definirani parametar δ kontrolira njihov utjecaj (*HuberLoss — PyTorch 2.0 documentation*, bez dat.). Koristeći formulu koju navodi Seif (2022), a čija notacija je modificirana kako bi odgovarala notaciji u ostatku rada, *Huberov* gubitak može se zapisati kao:

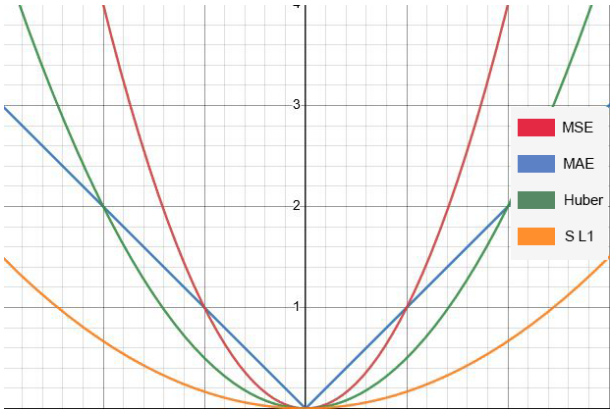
$$HuberLoss = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}\frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2, & ako\ \left|y^{(i)} - \hat{y}^{(i)}\right| \leq \delta \\ \delta\left(\left|y^{(i)} - \hat{y}^{(i)}\right| - \frac{1}{2}\delta\right), & inače\end{cases}$$

Pregledom formule, može se zaključiti da u slučaju kada je parametar δ veći ili jednak vrijednosti *MAE* gubitka, tada se izvršava modificirana *MSE* funkcija. U suprotnom se vrši varijacija *MAE* gubitka - time se stvara funkcija gubitka koja nije osjetljiva na neuobičajene

podatke poput *MSE*, dok s druge strane nije niti pretjerano neosjetljiva kao *MAE*. Varijacija *Huberovog* gubitka je „*glatki*“ *L1 gubitak* (engl. *Smooth L1 Loss*) koji također spaja *MSE* i *MAE* funkcije, no zbog razlika u formuli, povećanjem parametra (u ovome slučaju zvan β) ta funkcija konvergira nuli dok *Huberov* gubitak konvergira prema *MSE* funkciji gubitka. *Glatki L1 gubitak* se koristi iz istih razloga kao i *Huberov* gubitak (*SmoothL1Loss — PyTorch 2.0 documentation*, bez dat.). Prema formuli navedenoj na izvoru *SmoothL1Loss* iz Pytorch 2.0 dokumentacije, s prilagođenom notacijom kako bi odgovara notaciji u ostatku rada, *glatki L1* gubitak može se zapisati kao:

$$SmoothL1 = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}\dfrac{\frac{1}{2}\left(y^{(i)}-\hat{y}^{(i)}\right)^2}{\beta}, & ako\ \left|y^{(i)}-\hat{y}^{(i)}\right| \le \beta \\ \left|y^{(i)}-\hat{y}^{(i)}\right| - \frac{1}{2}\beta, & inače\end{cases}$$

Slika 2 prikazuje grafove svih navedenih funkcija. Zelena boja označava graf *Huberove* funkcije gubitka, a narančaste je boje funkcija *glatkog L1* gubitka. Na slici je moguće vidjeti kako istom vrijednosti parametara δ i β (kojima je vrijednost, u ovom slučaju, 3) funkcije idu u različitim smjerovima. Kako je navedeno, Huberov gubitak se približava *MSE* funkciji, a *glatki L1* gubitak nuli.



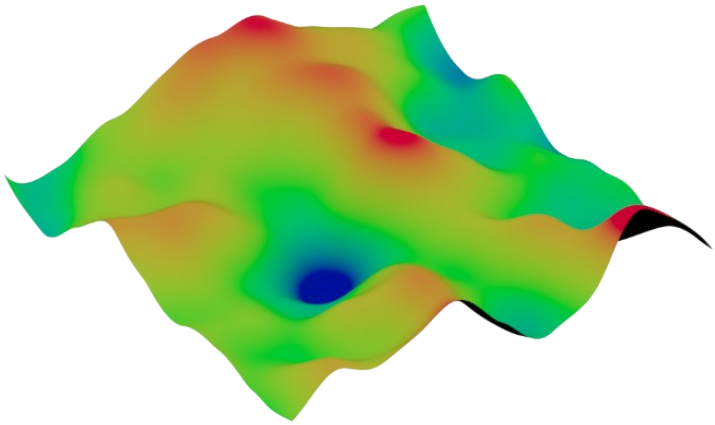
Slika 2 *MAE, MSE, Huber i glatki L1 gubitak – graf izrađen prema Seif (2022), HuberLoss-Pytorch 2.0 documentation (bez dat.) i SmoothL1Loss-Pytorch 2.0 documentation (bez dat.)*

Ne postoji jedna funkcija gubitka koja bi predstavljala idealno rješenje za sve zadatke i sustave, zato je potrebno odvojiti vremena i pronaći funkciju koja daje najbolje rezultate na određenom zadatku za koji se trenira sustav. Slikom 2 pokušavaju se prikazati grafovi popularnih funkcija gubitka za lakši izbor funkcije za određeni zadatak tako što pokazuje kako mapiraju vrijednosti. Važno je napomenuti da funkcije gubitka čine velik dio istraživanja neuronskih mreža te da postoji još mnogo funkcija gubitka kao i algoritama za računanje tog gubitka te da je u ovom dijelu bio opisan samo mali dio regresijskih funkcija gubitka.

Nakon što je prikazano na koje načine je moguće kvantificirati razliku između izlaza modela i temeljne istine, valja prikazati kako se gubitak koristi za učenje modela. Prema Dalbelo Bašić i suradnicima (2008) „Učinkovita i popularna metoda učenja višeslojnih neuronskih mreža je 'algoritam sa širenjem pogreške unazad' (engl. *backpropagation algorithm*).“ Zbog jednostavnosti, u nastavku rada, za navedeni algoritam koristiti će se izraz „back-propagacijski“ algoritam. Učenje duboke neuronske mreže pomoću back-propagacijskog algoritma podrazumijeva postepeno minimiziranje gubitka modificiranjem parametara (težinskih faktora) neuronske mreže unazad, to jest, od izlaza modela prema početku (ulaza u mrežu) kroz sve slojeve i jedinice mreže (LeCun i sur., 2015).

Gubitak je moguće vizualizirati kao hiper-površinu koja sadrži minimum (mjesto na hiper-površini gdje je gubitak najmanji) ili potencijalno više lokalnih minimuma (Dalbelo Bašić i sur., 2008). Cilj je smanjiti gubitak ili pogrešku dolaskom do teoretskog globalnog minimuma (engl. *global minimum*) prostora vrijednosti koje može stvoriti neuronska mreža gdje je gubitak najniži te je važno ne „zaglaviti“ u lokalnom minimumu (engl. *local minimum*). U praksi, loši lokalni minimumi su rijetko problem u velikim mrežama (LeCun i sur., 2015).

Slika 3 prikazuje teoretsku hiper-površinu gubitka u prostoru vrijednosti koje može poprimiti funkcija, gdje je crvenom bojom označen najveći gubitak, a plavom najmanji. U plavoj boji moguće je vidjeti globalni minimum, a u zelenoj se vidi nekoliko lokalnih minimuma. U slučaju koji prikazuje hiper-površina u slici 3, cilj sustava je doći do najniže točke u tamnoplavoj boji koja simbolizira najniži teoretski gubitak. Važno je naglasiti da u procesu učenja ovakva potpuna slika hiper-površine nije vidljiva te da je ovo samo prikaz zamišljene površine. Slika je samo ilustrativna te je stvorena korištenjem *Blender* softvera.

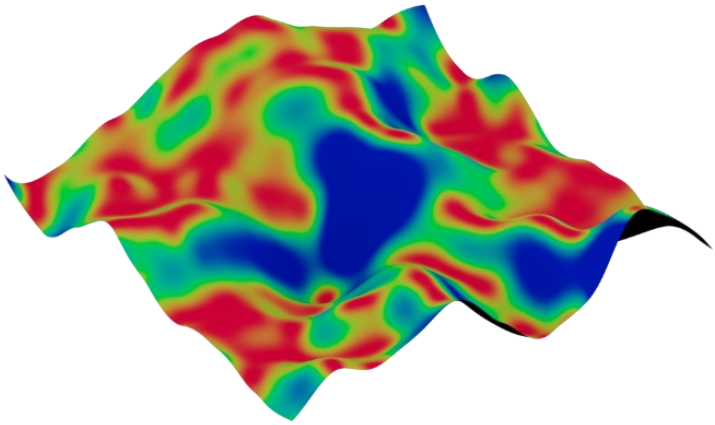


Slika 3 *Teoretska hiper-površina gubitka (autorska slika)*

Jedna od metoda kojom se dolazi do minimuma gubitka je gradijentni spust (engl. *gradient descent*). Dalbelo Bašić i suradnici (2008) navode kako je gradijentni spust iterativni postupak kojim se pokušava minimizirati gubitak tako da parametre mreže, o kojima ovisi gubitak, mijenja back-propagacijom s obzirom na gradijent (engl. *gradient*) funkcije.

Kwiatowski (2022) opisuje gradijent funkcije kao „nagib“ ili „strminu“ krivulje funkcije na određenoj točki koja opisuje brzinu promjene funkcije, to jest, derivaciju po određenoj točki ili, u slučaju višedimenzionalnih funkcija, vektor derivacija za određenu točku.

Slika 4 vizualizira aproksimaciju gradijenata hiper-površine. Crvenom bojom označene su točke s malim gradijentom, dok su plavom označeni najveći gradijenti (veliki gradijenti omogućuju lakši odabir smjera za algoritam gradijentnog spusta). Iz slike je moguće vidjeti da svi minimumi (globalni i lokalni) imaju mali gradijent (blizu nule) što znači da postoji teoretska mogućnost da gradijentni spust „zaglavi“ u minimumu koji nije idealno rješenje u prostoru vrijednosti modela. Bitno je napomenuti da je slika samo ilustrativna i ne prikazuje prave gradijente na površini nego se radi o aproksimaciji stvorenoj u *Blender* softveru za 3d modeliranje.



Slika 4 *Gradijent hiper-površine (autorska slika)*

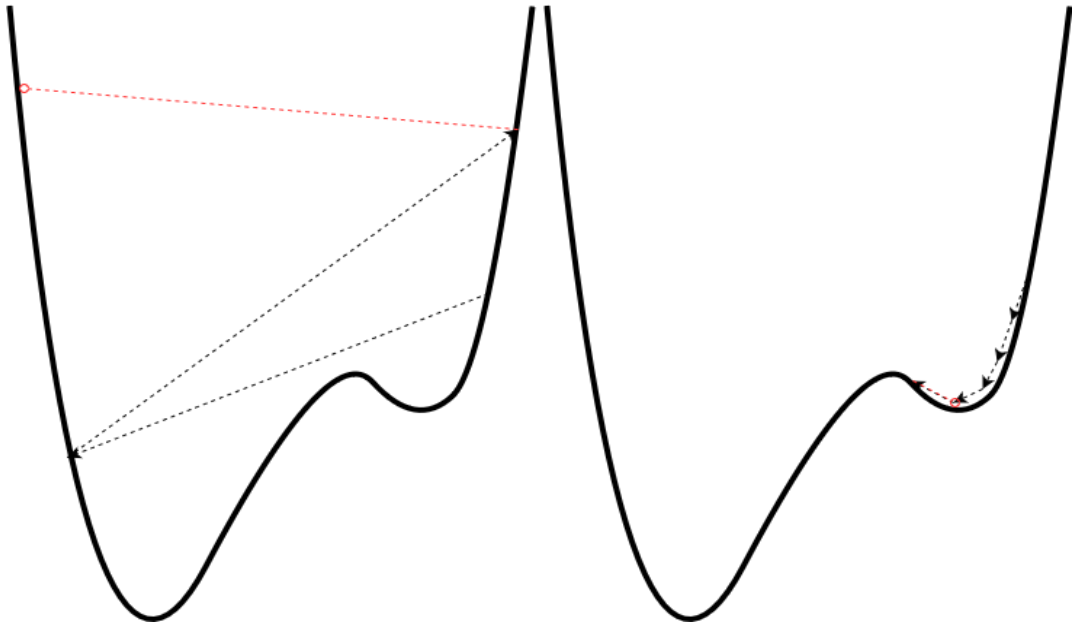
Algoritam gradijentnog spusta odabire početnu točku (inicijalni gubitak), računa gradijent na toj točki, napravi korak (skalira gradijent) u najstrmijem smjeru, oduzme dobivenu vrijednost gradijenta (cilj je smanjiti gubitak, doći do minimuma – gradijent je inače usmjeren prema većem gubitku) te s obzirom na veličinu gradijenta modificira parametre mreže back-propagacijom sve dok se ne postigne određeni kriterij - na primjer, dok se ne dođe do minimuma (Kwiatkowski, 2022; Russell i Norvig, 2020).

Dio postupka učenja koji završava s modificiranjem težina ili parametara mreže back-propagacijom naziva se iteracija (engl. *iteration*) (Dalbelo Bašić i sur., 2008).

Veličinu koraka koju algoritam radi po hiper-površini gubitka u svakoj iteraciji određuje takozvana stopa učenja (engl. *learning rate*) (Kwiatkowski, 2022). Jednostavnije, stopa učenja određuje koliki će korak napraviti algoritam gradijentnog spusta u jednoj iteraciji, to jest, koliko će skalirati gradijent u najstrmijem smjeru.

Slika 5 prikazuje dvodimenzionalni presjek zamišljene hiper-površine gubitka gdje je cilj doći do najniže točke. Strelice prikazuju korake koje algoritam radi po prostoru hiper-površine, a crvena linija pokazuje zadnji korak u ovom zamišljenom slučaju. Slika 5 pokazuje kako stopa učenja može utjecati na gradijentni spust – ako je prevelika iskočiti će iz minimuma, a u slučaju

da je premala, treniranje će trajati duže te postoji veća mogućnost zaglavljenja u lošem minimumu. Lijevi i desni slučaj prikazuju dvije krajnosti. Lijeva grafika pokazuje što se događa u slučaju da je stopa učenja prevelika - model tokom učenja nije sposoban doći do minimuma jer su koraci algoritma gradijentnog spusta, koje definira stopa učenja, preveliki. Desna grafika prikazuje što se događa u slučaju da je stopa učenja premala - model tokom učenja nije sposoban doći do minimuma jer su koraci koje radi algoritam gradijentnog spusta premaleni. Idealna stopa učenja bila bi između te dvije krajnosti gdje su koraci dovoljno veliki da algoritam ne zaglavi u lokalnom minimumu i dovoljno mali da ne iskoči iz globalnog minimuma.



Slika 5 *Utjecaj stope učenja na gradijenti spust (autorska slika)*

3.3. Metode učenja

Postoji više metoda kojima je moguće optimizirati put kroz hiper-površinu gubitka za učenje back-propagacijom. Takvi algoritmi za optimizaciju nazivaju se algoritmima učenja (engl. *learning algorithms*) ili optimizatori (engl. *optimizers*).

Prema Roberts i suradnicima (2021) najpopularniji optimizator je stohastički gradijentni spust (engl. *Stochastic Gradient Descent*) ili *SGD*. Stohastički gradijentni spust podrazumijeva učitavanje nekoliko primjera iz skupa podataka, računanje izlaza i gubitka, računanje prosječnog gradijenta za tih nekoliko primjera te modificiranje težina s obzirom na gradijent. Naziv stohastički dolazi iz šuma (engl. *noise*) gradijenta, u odnosu na prosječni gradijent svih podataka u skupu, koji nastaje obradom manje količine podataka (LeCun i sur., 2015).

Skupina primjera iz skupa podataka koji se zajedno učitavaju nazivaju se grupa (engl. *batch*), dok se potpuni prijelaz kroz cijeli skup podataka za učenje naziva epohom (engl. *epoch*) (Dalbelo Bašić i sur., 2008). U literaturi se često izmjenjuju nazivi grupni (engl. *batch*), mini-grupni (engl. *minibatch*) i stohastički gradijentni spust. Roberts i suradnici (2021) navode grupni i mini-grupni gradijentni spust kao varijaciju *SGD*-a, dok Russel i Norvig (2020) razdvajaju grupni gradijentni spust i *SGD*. Navedeni autori definiraju grupni gradijenti spust kao funkciju koja sumira sve primjere iz skupa podataka po iteraciji, to jest, kao takozvani deterministički gradijentni spust (engl. *deterministic gradient descent*). Isti autori definiraju *SGD* kao gradijentni spust u kojemu se nasumično izabire mini-grupa primjera iz skupa podataka po svakoj iteraciji. U ovome radu koristiti će se naziv *SGD* za gradijentni spust koji prima nasumičnu grupu (neovisne veličine) primjera iz skupa podataka po svakoj iteraciji treniranja.

Prema Dalbelo Bašić i suradnicima (2008) jedan od optimizacijskih algoritama (engl. *optimization algorithm, optimizer*) bio bi *gradijentni spust s momentom inercije* (engl. *momentum-based gradient descent*). Gradijentnim spustom s momentom inercije pokušavaju se riješiti problemi lokalnog minimuma gdje je gradijent blizu nule i algoritam neće modificirati težinske faktore. Uvođenjem parametra koji pokušava simulirati inerciju tijekom gradijentnog spusta potencijalno se izbjegava zaglavljenje u lokalnom minimumu (Dalbelo Bašić i sur., 2008), dok se i dalje omogućuje ostajanje u globalnom minimumu što je nedostatak nekih drugih metoda poput pristupa u kojemu se povećava stopa učenja za iskakanje iz minimuma.

Brownlee (2017) predstavlja napredniji optimizacijski algoritam zvan *Adam*. Naziv *Adam* dolazi od prilagodljive procjene momenta (engl. *adaptive moment estimation*). Mijenjanje stope učenja tijekom treniranja može spriječiti iskakanje iz globalnog minimuma. *Adam* algoritam sam mijenja stopu učenja tijekom treniranja računanjem pomičnog prosjeka (engl. *moving average*) čiji se parametri mogu kontrolirati, a čime se razlikuje od drugih naprednih algoritama poput *RMSProp* (engl. *root mean square propagation*) algoritma koji računa samo srednju vrijednost (engl. *mean*) za optimizaciju stope učenja (Brownlee, 2017). Računanjem pomičnog prosjeka *Adam* potencijalno bolje optimizira stopu učenja od *RMSPropa* jer pomični prosjek bolje prikazuje trenutni prostor hiper-površine gubitka (Ruder, 2017).

3.4. Aktivacijske funkcije

Kao što je prije navedeno, dodavanjem slojeva duboka mreža može reprezentirati kompleksnije funkcije, no dubina nije dovoljna. Dalbelo Bašić i suradnici (2008) navode: „Kako bi neuronska mreža mogla predstaviti visoko nelinearne funkcije, potrebno je da prijenosna ili aktivacijska funkcija njezinih procesnih elemenata i sama bude nelinearna funkcija svojih ulaza.“ Uspješnost neuronskih mreža ovisi o količini skrivenih slojeva, ali još su važnije aktivacijske funkcije koje se koriste (Sharma i sur., 2020).

Aktivacijske funkcije odlučuju kada će se neuron aktivirati. Osnovna aktivacija je *TLU* (engl. *treshold logic unit*) koju, na primjer, koristi perceptron. Ta aktivacija daje binarni izlaz (0 i 1, -1 i 1) iz neurona (Dalbelo Bašić i sur., 2008). Da bilo moguće aktivaciju prikazati u formuli definirana je *f(t)* kao funkcija aktivacije, a t kao suma težina ili težinskih faktora neurona. Prema formuli koju su naveli Dalbelo Bašić i suradnici (2008), a koja je prilagođena ovom radu prema navedenoj notaciji, *TLU* aktivacija može se zapisati kao:

$$f(t) = \begin{cases} 1, ako\ t < 0 \\ 0, \quad inače \end{cases}$$

Sharma i suradnci (2020) prezentiraju *sigmoidalnu* aktivacijsku funkciju kao često korištenu funkciju zbog svoje nelinearnosti. *Sigmoidalna* aktivacija transformira, to jest, mapira sve ulazne vrijednosti na raspon između 0 i 1 te ih nelinearno modificira. Formula *sigmoidalne* aktivacije koju navode Sharma i suradnici (2020), a koja je prilagođena notaciji u ovom radu, glasi:

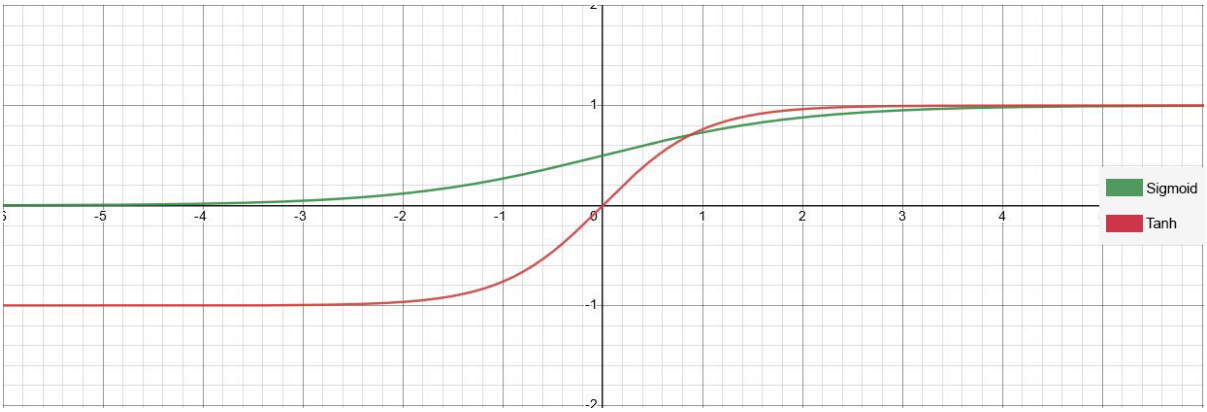
$$f(t) = \frac{1}{1 + e^{-t}}$$

Sharma i suradnici (2020) navode *tanh* funkciju ili funkcija hiperboličnog tangensa (engl. *Hyperbolic Tangent function*) kao funkciju sličnu *sigmoidalnoj*, ali je simetrična oko 0 (centrirana je na nuli) što znači da daje vrijednosti između -1 i 1 te je preferirana u odnosu na *sigmoidalnu*. Problemi koji nastaju kod *sigmoidalne* i *tanh* aktivacijske funkcije su nestajući gradijenti (engl. *vanishing gradient*) koji mogu otežati učenje neuronskih mreža, osobito

dubokih (Clevert i sur, 2016). Nestajući gradijenti su problem dubokih neuronskih mreža gdje u dubokim slojevima težinski faktori mreže konvergiraju prema nuli. Čest su problem kod navedenih funkcija jer one limitiraju velik raspon vrijednosti na raspon između -1 i 1 ili 0 i 1 zbog čega će velika promjena na ulazu u funkciju rezultirati malom promjenom na izlazu, stoga konstantnim množenjem tih malih vrijednosti kroz slojeve dolazi do smanjivanja gradijenta na ulazu u mrežu (radi se o back-propagaciji – težine se ažuriraju od kraja mreže prema početku) (Wang, 2019). Formula za *tanh* aktivacijsku funkciju prilagođena notaciji u ovome radu, a koju su naveli Sharma i suradnici (2020), glasi:

$$f(t) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}}$$

Slika 6 prikazuje graf *sigmoidalne* (zelene boje) i *tanh* (crvene boje) funkcije. Iz grafa se mogu iščitati njihovi rasponi i transformacije vrijednosti. Može se vidjeti kako obje funkcije mapiraju sve vrijednosti iznad 1 na 1 čime ograničuju potencijalni raspon vrijednosti. Može se uočiti i njihova razlika u djelovanju na vrijednosti ispod 0 gdje *sigmoidalna* funkcija mapira te vrijednosti na 0, a *tanh* propušta vrijednosti sve do -1.



Slika 6 *Sigmoidalna i tanh funkcija – graf prema Sharma i sur. (2020)*

Sharma i suradnici (2020) također navode i *ReLU* aktivacijsku funkciju. *ReLU* aktivacijska funkcija (engl. *Rectified linear unit*) je vrlo popularna nelinearna funkcija koja mapira vrijednosti u rasponu između 0 i beskonačnosti. Često se koristi zbog efikasnosti koja proizlazi

iz toga što se ne aktiviraju svi neuroni istovremeno - ako je vrijednost nula, neuron se ne aktivira i njegove težine nisu modificirane u back-propagacijskom procesu. No možda je čak i veća prednost *ReLU* to što je jedna od funkcija koja pokušava eliminirati probleme nestajućih gradijenata u dubokim dijelovima modela (Clevert i sur., 2016) propuštanjem vrijednosti iznad 1 čime se ne limitiraju velike vrijednosti. Upravo suprotno, kod *ReLU* aktivacije pak postoji suprotni problem zvan eksplodirajući gradijenti (engl. *exploding gradients*). Eksplodirajući gradijenti su suprotnost nestajućima, gdje velike vrijednosti parametara modela uzrokuju eksploziju vrijednosti do ulaza modela (Paik & Choi, 2023). Formula za *ReLU* aktivacijsku funkciju kako ju zapisuju Sharma i suradnici (2020), čija notacija je prilagođena notaciji u ovome radu, glasi:

$$f(t) = \begin{cases} t, & ako\ t \geq 0 \\ 0, & inače \end{cases}$$

Prema Sharma i suradnicima (2020) *ELU* funkcija (engl. *Exponential linear unit*) je varijanta *ReLU* funkcije s dodanim parametrom α za definiranje krivulje (engl. *slope*) za vrijednosti ispod nule. Parametar α dodan je kako bi se prosjek aktivacija preselio bliže nuli u odnosu na *ReLU* aktivacijsku funkciju što bi trebalo osigurati brži proces učenja (Clevert i sur., 2016). Ako je parametar α jednak 1 onda funkcija mapira vrijednosti u rasponu od -1 do beskonačnosti gdje je, kao u *ReLU*, prostor iznad 0 linearan, a u slučaju da se za parametar α izabere vrijednost 0, *ELU* se ponaša isto kao i *ReLU*. Formula za *ELU* funkciju prema Sharma i suradnicima (2020), a prilagođena notaciji u ovome radu, glasi:

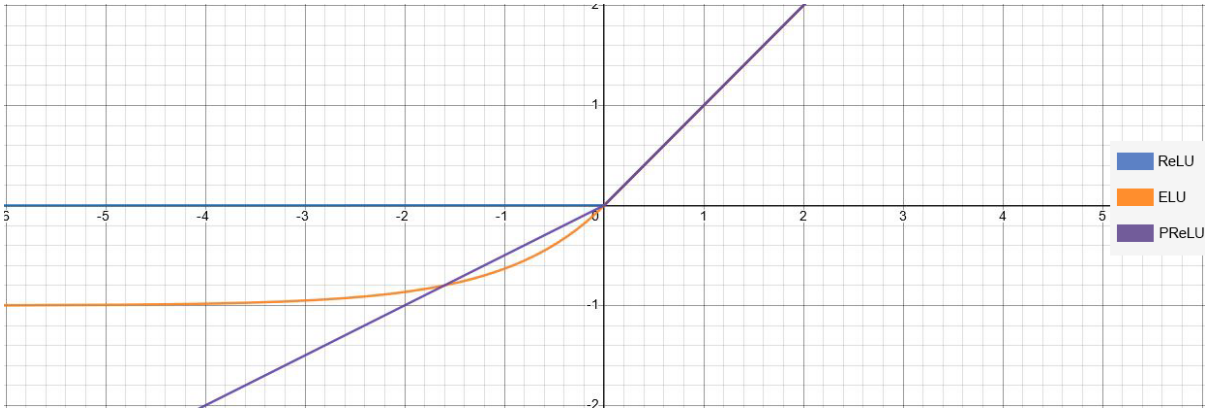
$$f(t) = \begin{cases} t, & ako\ t \geq 0 \\ \alpha(e^t - 1), & inače \end{cases}$$

He i suradnici (2015) navode unapređenje *ReLU* funkcije kao *PReLU* (engl. *Parametric Rectified Linear Unit*). Uvodi se novi parametar α koji se sam modificira u procesu učenja (engl. *learnable parameter*). Parametar se optimizira tijekom treniranja zajedno s ostalim elementima neuronske mreže s ciljem da se postigne idealna aktivacijska funkcija za određeni problem što dovodi do, teoretski, boljih performansa modela. Kao u *ReLU*, vrijednosti iznad 0

su linearno mapirane, a učenjem se parametar α modificira čime se određuje mapiranje funkcije za prostor ispod 0. Formula za *PReLU* aktivacijsku funkciju prema He i suradnicima (2015), s notacijom prilagođenom ovome radu, glasi:

$$f(t) = \begin{cases} t, & ako\ t > 0 \\ \alpha t, & inače \end{cases}$$

Slika 7 prikazuje grafove funkcija *ReLU* (plava), ELU (narančasta) i *PReLU* (ljubičasta). Na slici je moguće vidjeti da vrijednosti ispod 0 *ReLU* ne propušta, *ELU* ih propušta ovisno o parametru α (u ovom slučaju je 1) dok kod *PReLU* propuštanje ovisi o parametru koji se uči tokom treniranja (u ovom slučaju je 0.5). Sve funkcije se poklapaju u prostoru iznad 1 (sve su linearne u tom prostoru).



Slika 7 *ReLU, ELU i PReLU aktivacijske funkcije – graf prema Sharma i sur. (2020) i He i sur. (2015)*

3.5. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže su oblik neuronskih mreža koji se najčešće koristi za obradu vizualnog sadržaja. Ovo poglavlje čini pregled dijelova jednostavnih konvolucijskih neuronskih mreža tako što se definiraju konvolucijski slojevi i operacije u njima. Također, predstavljeni su *ResNet* i *U-net* modeli kao kompleksniji modeli osnovani na konvolucijskim neuronskim mrežama.

3.5.1. Osnove konvolucijskih neuronskih mreža

Prema LeCun i suradnicima (2015) konvolucijske neuronske mreže (engl. *convolutional neural networks*; skraćeno CNN) predstavljaju napredak u odnosu na klasične neuronske mreže (poput višeslojnog perceptrona) time što mogu učitavati kompleksnije oblike podataka u samu mrežu, kao što su polja (engl. *array*) ili matrice (engl. *matrix*). S obzirom na to da se slike mogu reprezentirati kao matrice gdje individualni pikseli predstavljaju vrijednosti boje (Compton i Ernstberger, 2020), slike je moguće koristiti kao podatke za treniranje konvolucijskih neuronskih mreža.

LeCun i suradnici (2015) navode da su zbog mogućnosti obrade slika direktno u mreži, što prijašnji oblici neuronskih mreža nisu nudili kao mogućnost, konvolucijske neuronske mreže dovele do revolucije u polju računalnog vida (engl. *computer vision*). Ukratko, *CNN* je vrsta neuronske mreže koja može izvući karakteristike kompleksnih podataka koristeći konvolucije (Li i sur., 2022).

Prema Weisstein, (*Convolution,* bez dat.) konvolucija je matematička operacije nad dvije funkcije kojom se tvori treća funkcija kao spoj prve dvije. U kontekstu obrade vizualnog sadržaja i konvolucijskih neuronskih mreža, konvolucija podrazumijeva korištenje male matrice, takozvane konvolucijske matrice (engl. *convolution matrix*) ili konvolucijske jezgre (engl. *kernel*) za konvoluciju između jezgre i određene matrice - u ovom slučaju slike (*Glossary - Convolution*, bez dat.).

Li i suradnici (2022) navode kako je arhitektura (dizajn, oblik) *CNN*-a inspirirana vizualnom percepcijom u smislu da biološki neuron zamjenjuje umjetni neuron, biološke receptore reprezentiraju konvolucijske jezgre, a aktivacije bioloških neurona simuliraju aktivacijske funkcije.

Prema LeCun i suradnicima (2015) arhitektura osnovne konvolucijske neuronske mreže sastoji se od dva tipa sloja - od konvolucijskih slojeva (engl. *convolutional layers*) i slojeva sažimanja

(engl. *pooling layers*). Svaki konvolucijski sloj se sastoji od jedinica organiziranih u mape značajki (engl. *feature maps*) koje su povezane s drugim mapama značajki u drugim konvolucijskim slojevima putem skupova težina (engl. *filter bank*). Svaki sloj može sadržavati više mapa značajki gdje svaka ima svoj različiti skup težina kojima se povezuje s drugim mapama značajki. Matematičke operacije koje se izvršavaju u konvolucijskim slojevima nazivaju se diskretne konvolucije (engl. *discrete convolution*) od čega je i nastao naziv konvolucijske neuronske mreže.

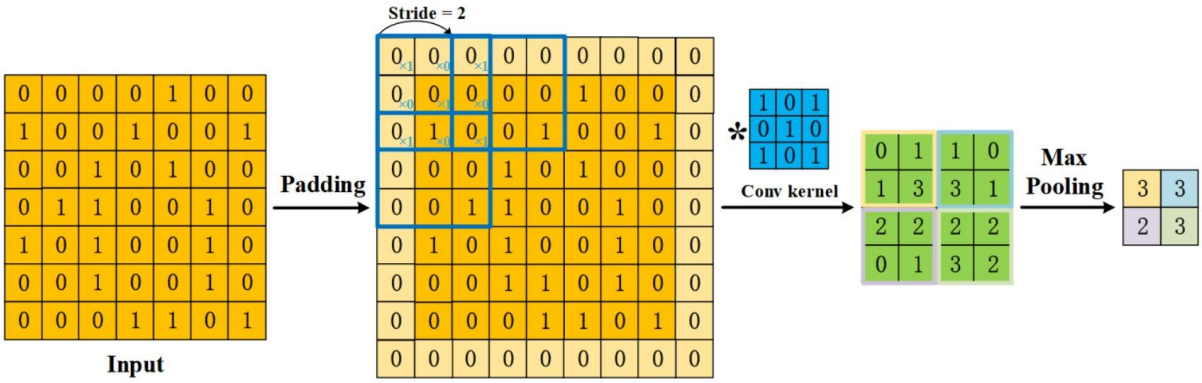
Li i suradnici (2022) tvrde da mape značajki nastaju kao rezultati konvolucija u konvolucijskom sloju. Konvolucije se vrše množenjem određene, takozvane, ulazne matrice s konvolucijskom jezgrom. Drugim riječima, konvolucijske jezgre stvaraju nove matrice (mape značajki) horizontalnim i vertikalnim kretanjem po ulaznoj matrici te sumiranjem skalarnog umnoška (engl. *dot product*) vrijednosti ulazne matrice i vrijednosti konvolucijske jezgre.

Li i suradnici (2022) za veličinu jezgre konvolucije navode da određuje prostor ulazne matrice koji će se konvulirati matricom jezgre u jednom koraku (horizontalnim ili vertikalnim pomicanjem kroz prostor ulazne matrice) - najčešća veličina matrice jezgre konvolucije je 3x3.(Ding i sur., 2022)

Prema Li i suradnicima (2022) veličina koraka (engl. *stride*) definira čestotu prijelaza preko elemenata ulazne matrice (velik korak čini raspršen prijelaz) tako što određuje koliko elemenata ulazne matrice zaobilazi jezgra konvolucije jednim korakom.

Li i suradnici (2022) navode da dopunjavanje (engl. *padding*) dodaje vrijednosti na rubove ulazne matrice s ciljem sprječavanja gubitka informacija tokom prijelaza jezgre, te ju time indirektno povećava.

Slika 8 prikazuje operacije u konvolucijskom sloju. Narančaste boje je ulazna matrica, svijetlo narančasta je dopuna, plave boje je konvolucijska jezgra, dok je zelene boje izlazna matrica (mapa značajki).



Slika 8 *Operacije u konvolucijskim neuronskim mrežama (Li i sur., 2022)*

Slika 8 prikazuje i operaciju sažimanja, no sažimanje je često implementirano kao zaseban sloj nakon jednog ili više konvolucijskih slojeva.

Prema LeCun i suradnicima (2015) uloga sloja sažimanja je spajanje semantički sličnih karakteristika matrice u jedno. Tipični sloj sažimanja je sloj koji računa lokalnu maksimalnu vrijednost matrice (engl. *max pool*), no ponekad se koristi i prosječna vrijednost (engl. *average pool*) (LeCun i sur., 2015; Li i sur., 2022). Važno je naglasiti način na koji slojevi sažimanja spajaju slične karakteristike. To se provodi tako što smanjuju (engl. *downsampling*) ulaznu matricu (Li i sur., 2022) - najčešće je to smanjivanje na pola.

Skupina od nekoliko konvolucijskih slojeva, slojeva sažimanja i aktivacija ponekad se naziva i blokom kao što je moguće vidjeti u opisima nekih konvolucijskih modela (Li i sur., 2022).

3.5.2. Modeli konvolucijskih neuronskih mreža

Napredniji modeli osnovani na konvolucijskim mrežama sastavljeni su od više kompleksnih elemenata. Osim konvolucijskih slojeva te slojeva aktivacija i sažimanja, koriste se dodatni slojevi poput normalizacijskih slojeva (engl. *normalization layers*) i rezidualnih veza (engl. *residual connections*) (Li i sur., 2022).

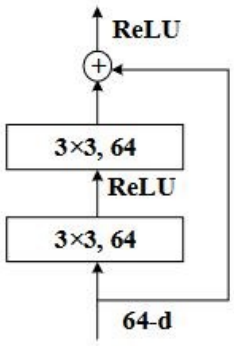
Li i suradnici (2022) navode da je među prvim kompleksnijim modelima *GoogLeNet* grupa modela koju čine *Inception* v1, v2, v3 i v4 moduli. *Inception v2* modul je među prvim mrežama koje koriste grupnu normalizaciju (engl. *batch normalization*) kako bi se riješio problem internog kovarijantnog pomaka (engl. *internal covariate shift problem*).

Prema Ioffe i Szagedy (2015) interni kovarijantni pomak je promjena u distribuciji aktivacija zbog promjene parametara mreže tijekom treniranja. Grupna normalizacija rješava navedeni

problem uvođenjem normalizacijskog koraka koji normalizira vrijednosti sloja s obzirom na učitanu grupu podataka. Također, za grupnu normalizaciju se vjeruje da ima regularizacijski učinak te smanjuje potrebu za drugim regularizacijskim tehnikama poput tehnike ispuštanja (engl. *droput*).

Prema Li i suradnicima (2022) jedan od naprednijih modela je *ResNet* koji je zamišljen da riješi problem nestajućih i eksplodirajućih gradijenata koji su čest problem dubokih dijelova dubokih neuronskih mreža. *ResNet* je sastavljen od uzastopnih blokova koji sadrže nekoliko konvolucijskih slojeva, slojeve sažimanja i slojeve aktivacijskih funkcija, čijom povezanosti pokušava riješiti probleme ostalih dubokih mreža. *ResNet* pokušava riješiti navedene probleme uvođenjem rezidualnih veza koje omogućuju nesmetan tok gradijenta kroz model. Rezidualne veze su veze koje omogućuju tok podataka između blokova koji nisu uzastopni. Problem nestajućih gradijenata uspješno je riješen rezidualnim vezama bez degradacije u dubokim dijelovima modela. Arhitektura *ResNeta* osnovana je na blokovima gdje je svaki uzastopni blok povezan s prethodnim tako da je izlaz prethodnog ujedno i ulaz trenutnog bloka, no dodaje se još jedna veza (rezidualna) koja je suma izlaza prethodnog i izlaza trenutnog bloka koja tvori ulaz u idući blok.

Slika 9 prikazuje shemu jednog bloka *ResNeta* za lakše razumijevanje rezidualnih veza. Na slici je vidljivo kako rezidualna veza zaobilazi blok čime se idućem bloku omogućuje pristup informacijama zaobiđenog bloka (direktna veza) i bloka prije (rezidualna veza). *ResNet* je sastavljen od više takvih blokova koji tvore lanac od ulaza do izlaza modela.



Slika 9 *ResNet blok (Li i sur., 2022)*

Jedan od naprednijih modela osnovanih na konvolucijskim neuronskim mrežama je *U-net*. Ronneberger i suradnici (2015) navode kako je *U-net* model osmišljen za obradu biomedicinskih slika, to jest, za njihovu segmentaciju. Segmentacija je naziv za zadatak

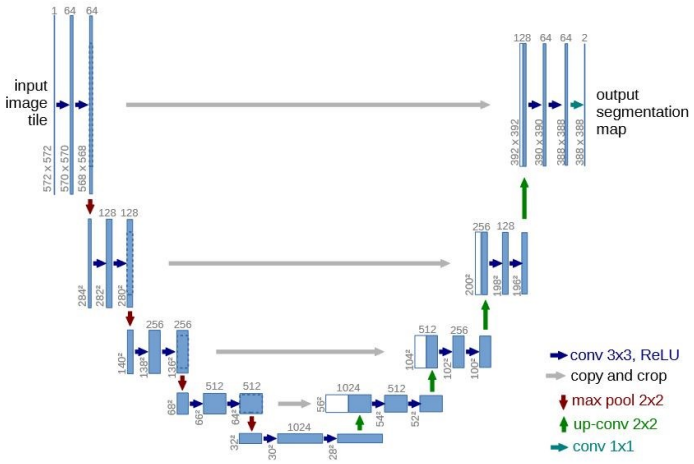
strojnog učenja gdje model pokušava odvojiti i klasificirati dijelove određene slike, poput pronalaženja tumora na MR slikama (Menze i sur., 2015).

U-net se sastoji od dva glavna dijela - dijela sažimanja i dijela širenja.

Ronneberger i suradnici (2015) opisuju kako se dio sažimanja sastoji od blokova koji sadrže dva konvolucijska sloja koje prate *ReLU* aktivacija i sloj sažimanja po maksimalnoj vrijednosti. Svaki uzastopni blok prepolovljuje veličinu ulazne slike i duplicira broj mapa značajki, to jest, kanala značajki (engl. *feature channels*).

Ronneberger i suradnici (2015) navode da se dio širenja sastoji od blokova koji također sadrže po dva konvolucijska sloja praćena *ReLU* aktivacijom, ali u ovom slučaju se koristi operacija povećanja čime se povećava razlučivost slike dok se prepolovljuje broj kanala značajki.

U-net sprječava „zaborav“ tokom treniranja vezama koje konkateniraju izlaze blokova sažimanja sa suprotnim blokovima širenja. Te veze se ponekad nazivaju i *preskočne* ili *skip* veze (engl. *skip connections*) jer preskaču dio modela. Preskočne veze u *U-netu* nisu iste kao rezidualne veze u *ResNetu*. Veze u *ResNetu* završavaju tako da se rezultati blokova sumiraju, dok se u *U-netu* vrši konkatenacija mapa značajki. I jedne i druge veze se često zovu istim imenima stoga će se ovdje naziv preskočne veze koristiti za veze koje zaobilaze veći dio modela, a rezidualnim vezama će se nazivati one koje zaobilaze samo jedan blok. Slika 10 prikazuje arhitekturu *U-net* modela.



Slika 10 *U-net arhitektura (Ronneberger i sur., 2015)*

4. Podloga istraživanja

Ovo poglavlje služi kao pregled osnovnih Python programskih biblioteka koje će se koristiti u istraživačkom dijelu rada. Prvenstveno će biti opisan Python kao programski jezik korišten u projektu te programske biblioteke *Pytorch*, *numpy*, *sickit-image* i *pillow*.

4.1. Prostori boja

Prema Hasting i Rubin (2012) prostor boja (engl. *color/colour space* ili *color model*) je način definiranja, stvaranja i vizualizacije boja. Zato što je proces percepcije boje sam po sebi varijabilan i subjektivan proces, prostorima boje želi se postići što objektivnije definiranje i standardiziranje boja kroz razne medije, znanosti i industrije. Prostori boje su podijeljeni u dvije grupe: na psihološke prostore i na geometrijske prostore. Psihološki prostor boja organizira boje po percepciji (kako se čine), to jest, s obzirom na eksperimentalne rezultate ljudskog vida. Geometrijski prostor boja je pak matematički izračunat okvir boja uređenih po određenim mjerenim svojstvima te je definiran u određenom prostoru. Zbog potreba rada, fokus će biti postavljen samo na geometrijske prostore boja, specifično na *RGB* i *CIELab* prostor.

Hasting i Rubin (2012) navode da je *RGB* (engl. *Red Green Blue*) prostor boja trodimenzionalni aditivni prostor boja kojega čine dimenzije ili vektori za crvenu, zelenu i plavu boju. Te dimenzije koje tvore prostor nazivaju se i kanalima ili komponentama boje. *RGB* prostor spada u aditivne prostore jer se boje stvaraju miješanjem komponenti, to jest, dodavanjem jedne boje drugoj.

Prema Hasting i Rubin (2012) *CIELab* (franc. *Commission International de l' Eclairage*, lightness ab) prostor boja je trodimenzionalni prostor kojega tvore dimenzija za svjetlinu (engl. *lightness*), dimenzije za boje pod nazivom *a* i *b* gdje *a* reprezentira boje između crvene i zelene, te *b* reprezentira boje između žute i plave. Iz tih dimenzija dolazi i naziv *Lab* (L od engl. *lightness* te a i b). Ponekad se kanali *CIELab* prostora pišu i kao *L**, *a** i *b**. *CIELab* prostor boja je jedan od najtočnijih u svojoj organizaciji boja, no teško ga je reproducirati digitalno.

4.2. Python i Pytorch

Python je programski jezik visoke razine i opće namjene. Može se koristiti u kontekstu više programskih paradigmi poput skriptiranja ili objektno-orijentiranog programiranja (Kuhlman, bez dat.). Ima široki raspon primjena poput web razvoja, edukacije i desktop aplikacija te je često korišten u znanstvenom kontekstu. Široki raspon primjena omogućen je time što Python podržava mnoštvo okvira (engl. *framework*) i programskih biblioteka (engl. *library*) za specifične primjene, poput *Django* okvira za web razvoj ili *Pandas* biblioteke za obradu podataka (*Applications for Python*, bez dat.). Zbog toga što nudi veliki izbor programskih biblioteka, Python je izabran kao jezik u kojemu će se trenirati sustav kroz istraživački dio rada.

Jedna od Python programskih biblioteka je i *Pytorch* - optimizirana biblioteka za duboko učenje korištenjem procesora (engl. *Central Processing* Unit - CPU) i grafičkih procesora (engl. *Graphical Processing Unit* - GPU). *Pytorch* kao glavni podatkovni objekt koristi tenzore (engl. *tensors*) (*PyTorch documentation — PyTorch 2.0 documentation*, bez dat.).

Prema Sveučilištu Cambridge (*What is a Tensor?*, bez dat.) tenzor je naziv za matematički objekt koji je, zapravo, samo generalizacija skalara (brojeva), vektora i matrica. Tenzor je n-dimenzionalni objekt koji može sadržavati proizvoljan raspon vrijednosti i dimenzija, stoga se koristi kao osnovni element suvremenih primjena modela dubokog učenja jer može reprezentirati širok raspon svojstava.

Biblioteka *Pytorch* olakšava proces izrade sustava strojnog učenja opsežnom listom ugrađenih funkcija koje eliminiraju potrebu za ručnim programiranjem kompleksnih (ali opće poznatih i dobro dokumentiranih) funkcija strojnog učenja i manipulacije podataka. Također, *Pytorch* dopušta uređivanje tih ugrađenih funkcija, što pak omogućuje modifikaciju određenih dijelova sustava poput učitavanja podataka za treniranje; zbog toga je moguć dizajn jednostavnih, ali i poprilično kompleksnih sustava strojnog učenja. Korisna značajka *Pytorcha* je i to što koristi *cuda* operacije koje drastično ubrzavaju izvođenje na Nvidia grafičkim karticama što pak ubrzava treniranje i rad sustava, pogotovo na novijim izdanjima koje koriste takozvane tenzor jezgre (engl. *tensor cores*) za česte operacije nad tenzorima (*PyTorch documentation — PyTorch 2.0 documentation*, bez dat.). *Pytorch* dobro „surađuje“ i s drugim programskim bibliotekama poput *pillow* biblioteke za obradu slika.

4.3. Alati za digitalnu obradu slika u Pythonu

Prema *pillow* dokumentaciji *(Image Module, bez dat.) pillow* biblioteka je unapređenje starije *PIL* (Python Imaging Library) biblioteke za obradu slika u Pythonu. *Pillow* svojim modulima dodaje mogućnost dublje obrade slika te podržava preko 30 različitih formata slikovnih datoteka kao, na primjer, JPEG, GIF i PNG, ali podržava i opskurnije formate poput BLP i PCX formata. *Pillow* prilikom učitavanja slike pretvara sliku u svoju posebnu vrstu podataka koji se potom može obrađivati te se prilikom spremanja pretvara u proizvoljni format. S obzirom na to da *Pytorch* koristi PIL za učitavanje, on sam po sebi dozvoljava učitavanje slikovnog sadržaja kroz *pillow* te pretvaranje slikovnog sadržaja u tenzore. To omogućuje nesmetanu obradu slika tokom učitavanja u sustav.

Scikit-image ili skraćeno *skimage* je također biblioteka za procesiranje slikovnog sadržaja, no bitna razlika od prethodne je to što operacije na sadržaju vrši kroz *numpy* tip podataka n-dimenzionalnih kompleksnih nizova (engl. *numpy n-dimensional array*) (*scikit-image's documentation — skimage 0.21.0 documentation*, bez dat.). *Numpy* je temeljni paket za znanstveno računalstvo (engl. *scientific computing*) u Pythonu zbog svojeg tipa podataka te ugrađenih, optimiziranih operacija za obradu *numpy* tipa podataka (*NumPy documentation — NumPy v1.25 Manual*, bez dat.). Zbog korištenja *numpy* formata za slikovni sadržaj, *skimage* lako radi s većinom drugih matematičkih i znanstvenih biblioteka, te su dostupne kompleksnije transformacije u usporedbi s *pillow* bibliotekom. Valja napomenuti da *Pytorch* ne može sam po sebi učitavati slike kroz *skimage*, ali pretvaranje *numpy* tipa podataka u tenzore je podržano tako da se slike prilikom učitavanja u sustav moraju prvo pretvarati u *numpy* tip podataka za uređivanje te po završetku obrade u tenzor.

Korištenjem ugrađenih funkcija iz navedenih programskih biblioteka moguća je duboka obrada slikovnog sadržaja. Moguće je mijenjati formate, obrezivati slike, dodavati filtere, raditi kompozite (slike sastavljene od elemenata drugih slika), ali moguće su i kompleksnije transformacije poput promjene prostora boja.

5. Istraživanje

Cilj istraživanja je izraditi sustav za kolorizaciju starih, crno-bijelih ili monokromatskih fotografija kroz dizajn i treniranje modela dubokog učenja na osobnom računalu. Za potrebe projekta osmišljen je i zaseban skup podataka za treniranje kolorizacijskih sustava.

Kolorizacija (engl. *colorization*) je, prema Hrvatskom jezičnom portalu (*kolorizacija*, bez dat.), bojanje starih crno-bijelih filmova uz pomoć kompjuterske tehnike. Termin kolorizacija se danas rijetko koristi samo za opis bojenja crno-bijelih filmova, već se kolorizacijom naziva i aktivna tema istraživanja u polju digitalne obrade slika gdje je cilj vjerno bojanje monokromatskih slika (onih koje sadrže samo jednu boju različitih vrijednosti zasićenosti i svjetlosti) i crno-bijelih slika, restauracija starog filma, bojanje monokromatskih animiranih filmova ili crteža i slično (Chen i sur., 2022). S obzirom na navedeno, može se reći da je kolorizacija naziv za digitalni proces bojanja crno-bijelog, monokromatskog ili sadržaja sivih tonova. U kontekstu ovog rada, kolorizacija se postavlja kao problem računalnog vida i generiranja vizualnog sadržaja kao jednih od problema strojnog učenja.

5.1. Skup podataka

Zadatak kolorizacije crno-bijelih fotografija zahtjeva specifičan skup podataka, stoga je za potrebe ovog rada stvoren novi skup podataka sastavljen od nekoliko različitih skupova fotografija. Skup podataka za ovaj projekt zamišljen je kao skup koji omogućuje treniranje na slabijim računalima te je stoga manjeg opsega od većine popularnih skupova koji su namijenjeni za treniranje na računalnim klasterima. Također, važnost se pridodaje i tematici slika u skupu podataka tako što se izabiru slike iz skupova koji motivima odgovaraju starim slikama.

Stvoreni skup sastoji se od nekoliko javno dostupnih skupova podataka koji sadrže vizualne materijale. Većina uključenih skupova poput *ImageNet* i *WIDER* skupa podataka prvotno je bila namijenjena za učenje klasifikacijskih sustava prepoznavanja sadržaja slika (engl. *image recognition*) te su prenamijenjeni za zadatak kolorizacije. Više o korištenim skupovima bit će navedeno u nastavku. Stvoreni skup sastoji se od 28,5 gigabajta sadržaja kojeg čine 213.522 slike JPG formata pogodne za treniranje osnovnih kolorizacijskih sustava. Stvoreni skup nije posebno podijeljen na podskupove za treniranje i testiranje, nego se ostavlja prostor za

samostalno definiranje omjera skupova. U slučaju ovog projekta, za treniranje se koristio cijeli skup zbog hardverskih ograničenja koja su onemogućila testiranje tokom treniranja.

5.1.1. Selekcija

Većina ostalih javno dostupnih skupova podataka s fotografijama sadrži mnoštvo fotografija koje nisu pogodne za treniranje kolorizacijskog sustava jer su monokromatske i crno-bijele, ili pak nisu odgovarajuće veličine. S obzirom na navedeno, selekcijski proces obuhvaćao je isključivanje fotografija koje su manje od 256 piksela širine i 256 piksela visine pošto je odlučeno da će se model većinom trenirati na slikama te veličine zbog nedostatka reprezentativnih slika u manjoj razlučivosti. Manje slike bi trebale biti uvećane da bi mogle biti učitane u model čime se potencijalno gubi još informacija iz fotografija već osrednje kvalitete. Kao prag nije odabrana veća veličina slike od 256x256 jer većina skupova podataka nije imala dovoljan broj velikih slika.

Drugi dio selekcijskog procesa obuhvaćao je isključivanje monokromatskih i crno-bijelih fotografija koje su detektirane kroz dva stupnja te su potom isključene iz skupa podataka.

Prvi stupanj detekcije bio je provjeravanje sličnosti po kanalima boje slike. Istinito crno-bijele slike imaju jednake vrijednosti svih kanala boja na određenom pikselu te je ta karakteristika crno-bijelih slika omogućila jednostavnu detekciju i odstranjivanje.

Drugi stupanj detekcije bio je usmjeren na pronalaženje monokromatskih slika poput sepije (engl. *sepia*) – smeđih monokromatskih slika, i cijanotipa (engl. *cyanotype*) – plavih monokromatskih slika. Slike su pronađene koristeći operaciju dekompozicije singularne vrijednosti ili *SVD* (engl. *singular value decomposition*). *SVD* se koristio tako što se uspoređivao postotak *PC1* varijance (varijance u prvom smjeru *PCA*) s određenim pragom. U ovome slučaju taj prag je bio 90%. Ovaj postupak bi trebao detektirati monokromatske slike računanjem postotka vrijednosti koje se poklapaju s određenim smjerom vrijednosti slike u trodimenzionalnom prostoru boja - monokromatske slike imaju jasno definiran smjer vrijednosti od jedne boje do druge, isto kao što to imaju i crno-bijele slike, no one imaju smjer vrijednosti samo između crne i bijele. Slike u boji pak nemaju jasno definirani smjer vrijednosti između dvije boje čime se razlikuju od monokromatskih i crno-bijelih slika.

5.1.2. Struktura skupa podataka

Skup podataka stvoren za potrebe ovog rada sadrži podatke iz sedam različitih skupova podataka: *ImageNet*, *Landcape Pictures*, *The Images of Groups Dataset*, *Oxford & Paris Buildings Dataset*, *Google Landmarks Dataset v2*, *The Oxford-IIIT Pet Dataset* te *HICO* (Humans Interacting with Common Objects) *dataset* i *WIDER dataset*.

ImageNet je opsežan skup podatak koji sadrži preko milijun fotografija. Treniranje na tako velikom skupu podataka je otežano na slabijim računalima. U slučaju ovog projekta, za jednu epohu (potpuni prelazak kroz cijeli skup podataka) bilo je potrebno više od dvadeset dana. Također, *ImageNet* sadrži mnogo monokromatskih ili crno-bijelih fotografija koje otežavaju proces učenja. Zbog navedenih problema, iz *ImageNet* skupa podataka Python skriptom je nasumično izabrano 150,000 fotografija što čini dovoljno sveobuhvatnu bazu za skup podataka. Nakon drugog stupnja selekcije (odbacivanje malih i monokromatskih slika) preostalo je 124,880 fotografija. *ImageNet* je dobar skup za treniranje kolorizacijskih sustava jer sadrži jako velik broj fotografija širokog raspona različitog sadržaja, stoga potencijalno čini dobru osnovu skupa za širok spektar problema računalnog vida što uključuje i kolorizaciju. *ImageNet* skup (Li i sur., bez dat.) je dostupan na poveznici https://www.image-net.org.

Landscape Pictures (Rougetet, bez dat.) skup podataka sadrži oko 3,900 fotografija krajolika. Navedeni skup je izabran zbog boljeg utvrđivanja osnovnih boja krajolika i prirodnih okruženja koja su česta pozadina na starim fotografijama. Skup je dostupan na https://www.kaggle.com/datasets/arnaud58/landscape-pictures.

The Images of Groups Dataset (Gallagher & Chen, bez dat.) sadrži fotografije skupina ljudi u različitim kontekstima. Navedeni skup je izabran zbog velike količine fotografija grupa ljudi, a grupe su relativno česta pojava u starim fotografijama. Više informacija o skupu podataka je dostupno na http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html.

Oxford & Paris Buildings Dataset (*Oxford & Paris Buildings Dataset*, bez dat.) sadrži oko 11,000 fotografija građevina i znamenitosti iz Pariza i Oxforda. Fotografije znamenitosti čest su motiv starih fotografija. Dostupan je na https://www.kaggle.com/datasets/skylord/oxbuildings.

Google Landmarks Dataset V2 (*Google Landmarks Dataset v2*, 2023) je skup od pet milijuna slika znamenitosti. Za potrebe ovog projekta preuzeto je prvih pet paketa što je oko pet gigabajta podataka koji sadržavaju 47,000 slika. Nakon obrade ostalo je oko 38,000 slika. Skup

je izabran iz istog razloga kao i *Oxford & Paris Buildings* skup podataka - sadrži brojne slike znamenitosti koje su česti motiv starih fotografija. Cjelokupni skup dostupan je na https://github.com/cvdfoundation/google-landmark.

Idući dodani skup podataka bio je *Oxford-IIIT Pet Dataset* (Omkar M i sur., bez dat.) s oko 7,000 fotografija pasa i mačaka. Nakon odbacivanja slika koje ne odgovaraju prethodno zadanim kriterijima, ostalo je oko 6,000 slika. Skup je izabran jer sadrži slike životinja koje su često prikazivane u ranim fotografijama. Skup podataka je dostupan na poveznici https://www.robots.ox.ac.uk/~vgg/data/pets/.

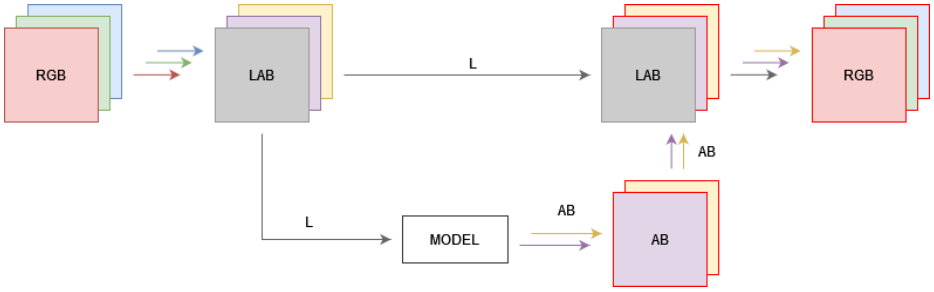
Uključeni skup podataka je i *HICO* (Yu-Wei i sur., bez dat.) skup podataka. *HICO* u podskupu za treniranje sadržava 38,000 slika ljudi u interakciji s čestim predmetima. Iz skupa je izdvojeno 15,000 slika, potom je iz njih odbacivanjem neodgovarajućih slika izvučeno oko 13,000 slika. *HICO* je izabran jer se sastoji od mnogo slika ljudi. Dostupan je na poveznici http://www-personal.umich.edu/~ywchao/hico/.

Posljednji uključeni skup je *WIDER* (*WIDER FACE: A Face Detection Benchmark*, bez dat.) skup podataka. Preuzet je samo podskup za treniranje koji se sastoji od 12.880 fotografija lica ljudi. Prvenstveno je dizajniran za detekciju lica čime tvori dobar skup za kolorizaciju lica. Nakon obrade izdvojeno je 12.138 slika. Dostupan je na poveznici http://shuoyang1213.me/WIDERFACE/

Zajedno navedeni skupovi tvore potencijalno dobar osnovni skup za treniranje kolorizacijskih modela na slabijim računalima zbog relativno malog broja fotografija (213.522 fotografije u 28,5 gigabajta) što omogućuje lakši višestruki prijelaz preko cjelokupnog skupa zbog sveobuhvatnosti sadržaja fotografija te zbog minimalne količine monokromatskih, crno-bijelih ili beskorisno malih fotografija. Važno je napomenuti da bi se mali broj fotografija mogao smatrati i kao nedostatak ovog skupa - modeli, trenirani na jačim računalnim sustavima, vjerojatno bi bolje mogli iskoristiti veći skup podataka jer s velikim brojem epoha (potpunih prijelaza kroz skup podataka) na malim skupovima podataka postoji veća mogućnost pretreniranja (engl. *overfitting*). Također, omjer motiva fotografija nije nužno u potpunosti reprezentativan u odnosu na stare fotografije.

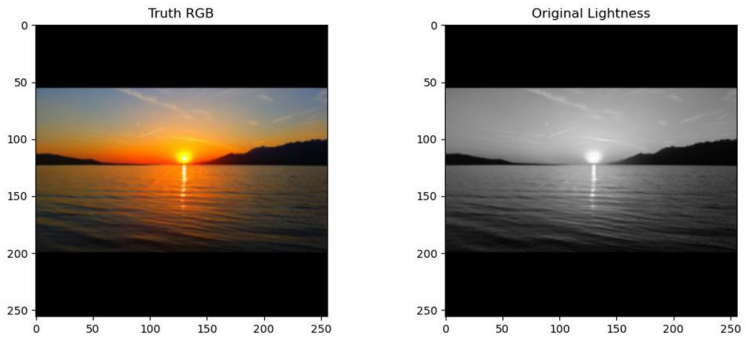
5.2. Sustav

Sustav radi tako što uzima fotografiju standardnog *RGB* prostora boja te ju pretvara u *CIELab* prostor boja nakon čega korištenjem L^* kanala model predviđa a^* i b^* kanale. Generirani a^* i b^* kanali pridružuju se originalnom L^* kanalu za stvaranje potpune fotografije koja se na kraju ponovno pretvara u standardni *RGB* prostor. Priložen je dijagram (Slika 11) za lakše razumijevanje cjelokupnog procesa. Svaki pravokutnik prikazuje jedan kanal boje - crvenom, zelenom i plavom bojom su označeni *RGB* kanali, dok su sivom, ljubičastom i žutom bojom označeni kanal svjetline, a^* i b^*. Crveni obrub označava onaj sadržaj koji je stvoren (predviđen) modelom ili je nastao iz stvorenog sadržaja. Strelice pokazuju put podataka.

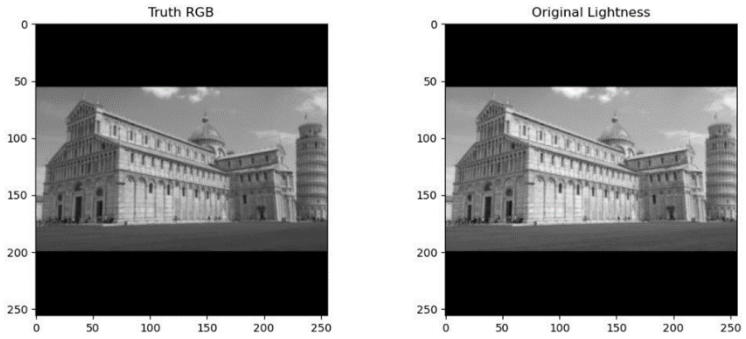


Slika 11 *Dijagram sustava (autorska slika)*

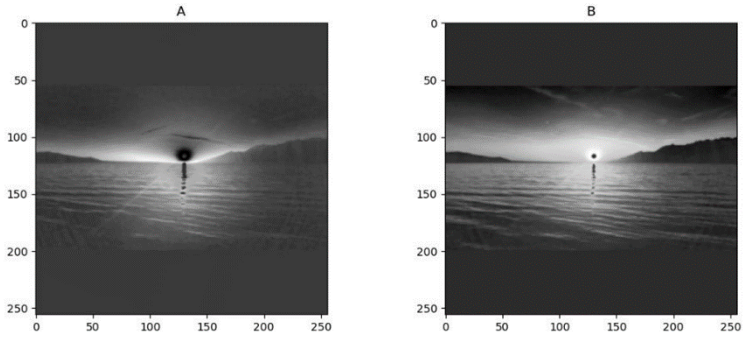
CIELab prostor boja je izabran zbog bržeg treniranja. Kao što je ranije navedeno, *CIELab* se sastoji od tri kanala gdje jedan predstavlja svjetlinu slike, a druga dva predstavljaju boju. Proces treniranja korištenjem *CIELab* prostora olakšan je kroz dva aspekta. Prvi je to što sustav mora učiti stvarati samo dva kanala (a^* i b^*) jer oni sadrže informacije o svim bojama u slici za razliku od tri kanala koja su potrebna u slučaju *RGB* prostora. Druga prednost je olakšano pretvaranje slika u crno-bijeli oblik time što L^* kanal već sam po sebi reprezentira crno-bijeli oblik slike. Slika 12 prikazuje originalnu *RGB* sliku te kanal svjetline dobiven iz konverzije slike u *CIELab*. Slika 13 prikazuje isti proces, ali je originalna slika crno-bijela. Slika 14 prikazuje a^* i b^* kanale *CIELab* prostora, ali u crno-bijelom obliku gdje je veća vrijednost svjetlija.



Slika 12 *Usporedba originalne RGB fotografije i L kanala (fotografija u boji) (autorska slika)*



Slika 13 *Usporedba originalne RGB fotografije i L kanala (crno-bijela fotografija) (autorska slika)*

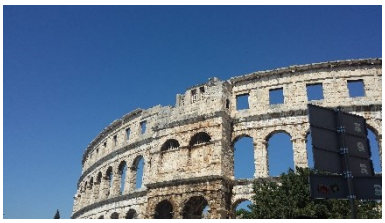


Slika 14 *a* i b* kanali LAB prostora (autorska slika)*

Sustav uči učitavanjem grupe slika, obradom učitanih slika te slanjem grupe u model. Obradom slika pokušava se poboljšati proces učenja modela ubacivanjem varijabilnosti podataka kroz epohe. Kroz proces obrade, slike se mogu: rezati, povećavati, zakretati i popunjavati. Također, sustav prihvaća nekoliko parametara o kojima ovisi proces obrade. Ti parametri su: veličina slike ili radna veličina, šansa za zakretanje slike, šansa za rezanje slike te parametar koji određuje vrijednosti kojima se popunjava rub slike. Valja napomenuti da je parametar veličina slike uveden jer model može učitavati i slike veće od 256x256 (koja je minimalna veličina u

skupu podataka) kao, na primjer, 512x512, a u slučaju da model želi učitati manju sliku od navedenog u parametru, tu sliku je potrebno dodatno obraditi.

Povećavanje je operacija gdje se slika poveća na radnu dimenziju bilinearnom interpolacijom zato što model ne može učitati slike različitih dimenzija od njegove radne veličine. Slika 15 prikazuje kako izgleda originalna slika učitana u sustav za uspoređivanje raznih operacija.



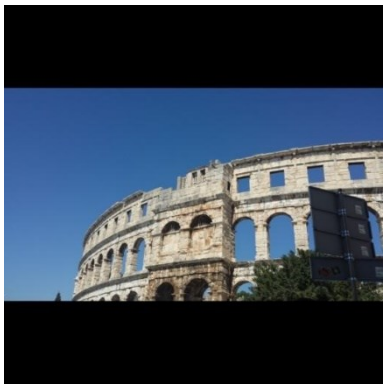
Slika 15 *Originalna slika (autorska slika)*

Operacija rezanja odnosi se na proces obrezivanja slike na određenu veličinu te odbacivanje ostalih vrijednosti slike. Rezanje se vrši na nasumičnom dijelu slike te je uvedeno zato što nudi varijabilnost u prelasku kroz skup podataka, a i povećava iskoristivost slika većih od radne veličine. Slika će rijetko biti obrezana na isti način kao i u prethodnoj epohi čime se umjetno povećava skup podataka i uvodi navedena varijabilnost u višestrukim prijelazima. Druga prednost je veća iskoristivost slika koja se postiže na način da se iz slike koja je veća od radne veličine u jednom prijelazu uzima samo jedan dio slike na kojemu model uči, dok drugi dijelovi potencijalno mogu biti korišteni u drugim prijelazima. U slučaju da se uzima cijela slika, potrebno ju je smanjiti na radnu veličinu čime se gube informacije u odnosu na uzimanje dijela slike. Slike 16 i 17 prikazuju kako izgleda operacija rezanja slike, te kako potencijalno povećava skup podataka različitim lokacijama rezanja.

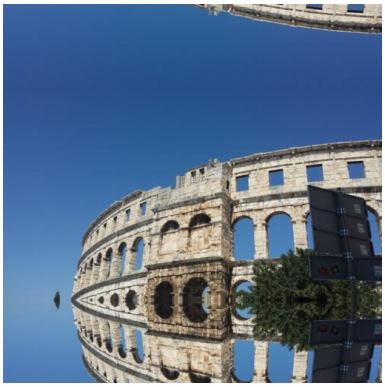
Slika 16 *Operacija rezanja slike (autorska slika)*

Slika 17 *Operacija rezanja slike, druga (autorska slika)*

Popunjavanje podrazumijeva popunjavanje ruba slike određenim vrijednostima u slučaju da je slika nejednakih duljina stranica (širine i visine). Model nije sposoban učitati slike različitih omjera, odnosno, učitava samo slike koje su jednakih dimenzija širine i visine, stoga je potrebno rubove slike popuniti da bi se zadovoljio taj kriterij – kriterij jednakih omjera. Vrijednosti kojima se popunjava i način na koji se popunjava rub određuje parametar za popunjavanje slike. Rubovi mogu biti popunjeni konstantnom vrijednosti koja je najčešće nula ili se može koristiti algoritam poput algoritma za zrcaljenje (engl. *reflect*) rubnih piksela. Slika 18 prikazuje popunjavanje slike konstantom vrijednosti za postizanje radne veličine, dok slika 19 prikazuje popunjavanje algoritmom za zrcaljenje. Slika 19 pokazuje nama neobičan oblik slike, no takav oblik može pomoći modelu da bolje nauči teksture i oblike bez da ga „zbunjuju“ crni rubovi.

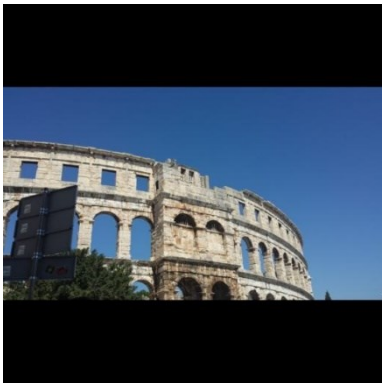


Slika 18 *Operacija popunjavanja (konstantna vrijednost) (autorska slika)*



Slika 19 *Operacija popunjavanja (reflektiranje) (autorska slika)*

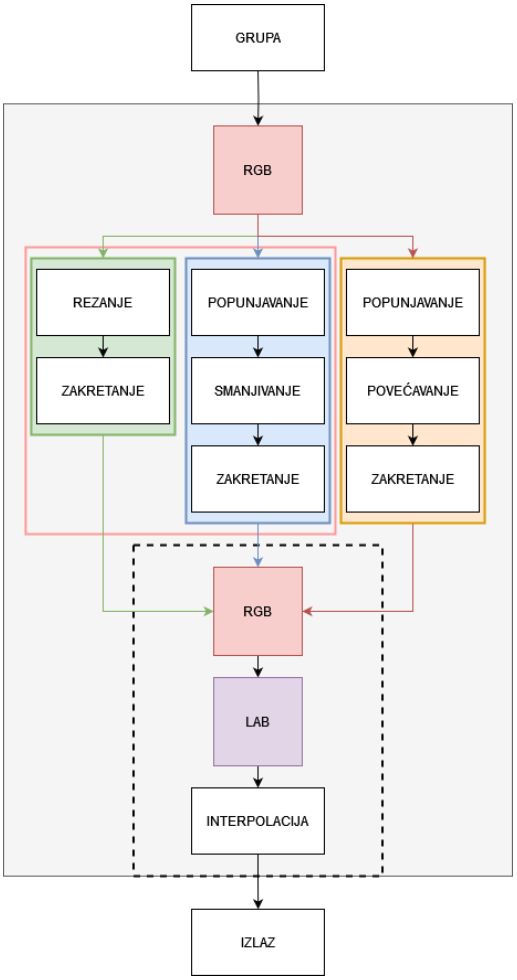
Zakretanje je u ovom slučaju naziv za operaciju horizontalnog zrcaljenja slike. Zakretanje je dodano u proces obrade slika zbog uvođenja varijabilnosti u proces učenja kao i umjetnog povećanja skupa podataka tako što postoji šansa za zrcaljenje slike čime se slika može pojaviti u dva različita (originalni i zakrenuti) oblika u različitim epohama. Slika 20 pokazuje kako izgleda operacija zakretanja na već popunjenoj slici.



Slika 20 *Operacija zakretanja (autorska slika)*

Slika 21 pokazuje tijek procesa obrade slika za vrijeme treniranja modela. Zeleni, plavi i narančasti pravokutnici reprezentiraju blokove operacija koje se zajedno izvršavaju sekvencijalno. Slika se učita u standardnom *RGB* prostoru boja, potom se, u slučaju da je slika manja od specificirane veličine u parametru veličina slike, izvršavaju operacije u narančastom bloku. To znači da se prvo izvršava popunjavanje nakon kojeg se slika poveća te zrcali, ovisno o parametru šansa zakretanja. U slučaju da je slika veća ili jednaka radnoj veličini, izvršava se jedan od blokova okruženih crvenim pravokutnikom, to jest, izvršavaju se zeleni ili plavi blok.

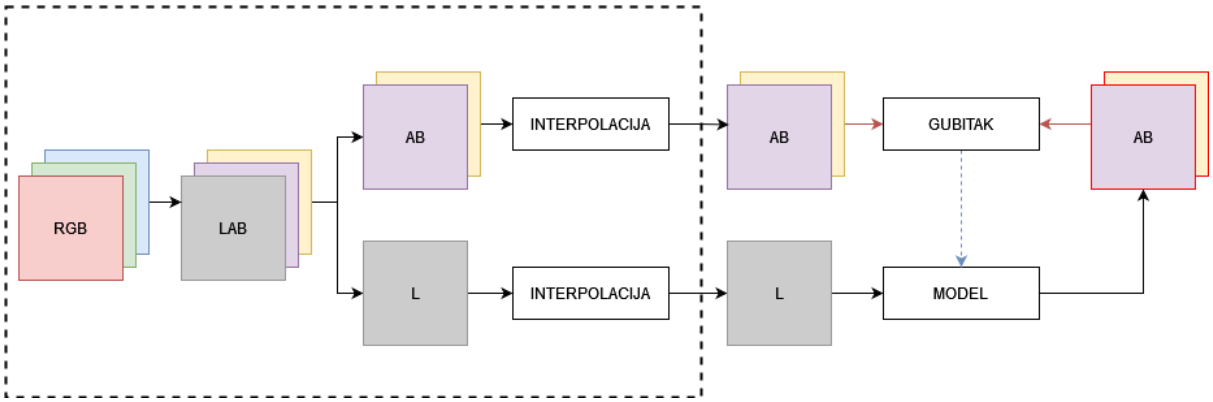
Na izvršenje zelenog ili plavog bloka utječe parametar šansa za rezanje. U slučaju da se izvršava zeleni blok, slika će se obrezati na veličinu koja je specificirana, a u slučaju plavog, slika će se interpolirati na specificiranu veličinu. Zadnji korak u oba bloka je zakretanje koje ovisi o parametru šansa za zakretanje. Nakon blokova obrade slijedi konverzija slike u *CIELab* prostor boja nakon koje dolazi interpolacija vrijednosti.



Slika 21 *Dijagram obrade slika tijekom treniranja (autorska slika)*

Slika 22 prikazuje podatke potrebne za treniranje modela te je ujedno i nastavak prijašnjeg dijagrama. Nakon što je slika pretvorena u *CIELab* prostor, razdvajaju se kanali na L^* te a^* i b^*. Kanali se zasebno linearno interpoliraju za lakše učenje modela. L^* se pretvara iz raspona od 0 do 100 na raspon između -1 i 1. Kanali a^* i b^* se interpoliraju iz raspona od -128 do 127 na raspon -1 i 1. Nakon tog koraka, L* kanal se učitava u model koji generira a^* i b^* kanale. Generirani a^* i b^* kanali se uspoređuju s originalnim kanalima a^* i b^* izabranom funkcijom

gubitka te taj rezultat tvori vrijednost gubitka kojom se modificiraju parametri modela kroz optimizacijski algoritam.



Slika 22 *Dijagram treniranja modela (autorska slika)*

5.3. Model

5.3.1. O modelu

Model je zamišljen kao modifikacija standardne *U-net* arhitekture s dodatkom rezidualnih blokova nalik na *ResNet* u najdubljem dijelu modela, ali s konkatenacijama mapa značajki umjesto sumiranjem rezultata. Za naziv modela izabrana je skraćenica *SRUNet* (Simple Residual U-net). Cilj je bio napraviti model koji je dovoljno kompleksan za generaciju sadržaja konvolucijskim neuronskim mrežama, ali ipak dovoljno jednostavan da se može trenirati na osobnom računalu što je postignuto korištenjem novijih metoda i optimizacija u procesu treniranja.

Dimenzije tenzora i slojeva će biti pisane kao *x*, *y* i z gdje je *x* širina, *y* visina, a *z* dubina ili broj kanala (mapa značajki) tenzora. Obično se nazivi dubina, širina i rezolucija modela koriste za opisivanje veličine ili opsežnosti modela gdje dubina opisuje količinu slojeva u modelu, širina opisuje broj kanala ili mapa značajki sloja (*z* dimenzija), a rezolucija opisuje veličinu sloja u *x* i *y* dimenziji (Tan & Le, 2020). Takav opis se pokazuje zbunjujućim u kontekstu modela nalik na *U-net* (modeli gdje se slojevi sažimaju do sredine pa povećavaju prema izlazu), stoga će za potrebe ovog rada veličina modela biti opisana na način da dubina modela definira *z* dimenziju sloja, širina definira broj slojeva, a rezolucija ostaje kao opis *x* i *y* dimenzija slojeva. To bi

značilo da je najdublji dio modela onaj koji ima najviše mapa značajki ili kanala, to jest, najveći je u *z* dimenziji.

5.3.2. Arhitektura

U ovom poglavlju biti će opisana arhitektura *SRUNet* modela. Sam model se kontrolira s nekoliko vanjskih parametara. Sadrži parametar za količinu ulaznih i parametar za količinu izlaznih kanala, parametre za veličine jezgri konvolucija u određenim dijelovima modela, parametre koji kontroliraju funkcije povećanja i sažimanja u slojevima te parametar za izbor aktivacijskih funkcija u aktivacijskom sloju.

Model je strukturiran u blokove i sastavljen je od tri glavna dijela – dio smanjenja, duboki rezidualni dio i dio povećanja. Ideja iza *SRUNet* modela bila je da model rastavi sliku i njezine značajke na osnovne elemente kroz dio smanjenja te da kroz dio povećanja rekonstruira tu sliku iz vrijednosti najdubljih slojeva gdje se nalaze osnovne značajke slika.

Model prihvaća tenzore proizvoljne veličine identičnih *x* i *y* dimenzija te proizvoljan broj ulaznih i izlaznih kanala (*z* dimenzija), no model će biti opisan kroz oblik ulaznih i izlaznih podataka gdje je *L** kanal učitane slike ulaz, to jest, crno-bijela inačica slike oblika tenzora 256x256x1, a izlaz je tenzor oblika 256x256x2 koji reprezentira *a** i *b** kanale *CIELab* prostora boja.

Dijagram (Slika 23) pokazuje tok operacija s lijeva na desno. Dimenzije blokova u dijagramu simboliziraju veličine tenzora u tom dijelu modela (visina bloka ovisi o *x* i *y* dimenziji, a širina o *z* dimenziji).

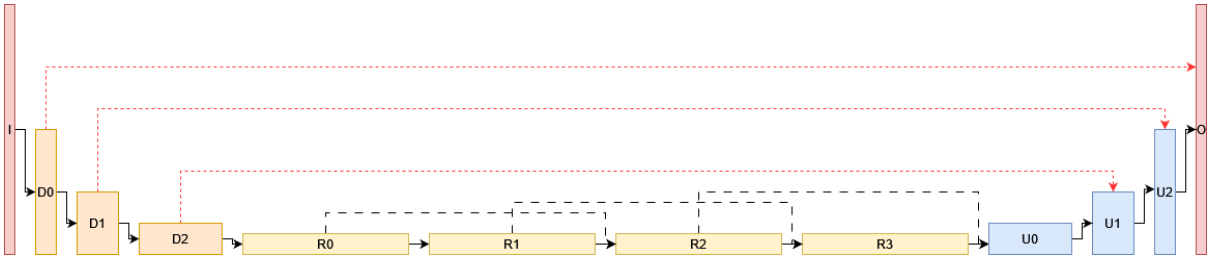
Crvenom bojom su označeni ulaz i izlaz - slovo I predstavlja ulaz (engl. *input*) dok slovo O predstavlja izlaz (engl. *output*). Narančastom bojom su označeni transformacijski blokovi smanjenja koji produbljuju - povećavaju broj mapa značajki, ali smanjuju *x* i *y* dimenzije tenzora. Tenzor se iz ulaznih dimenzija od 256x256x1, do kraja D2 bloka transformira u 32x32x256 gdje je 32 veličina mapa značajki u *x* i *y* dimenziji, a 256 je njihov broj.

Nakon tog dijela modela, slijede duboki blokovi (žute boje) po uzoru na *ResNet*. Taj dio čini najdublji dio modela koji se ponekad naziva i usko grlo (engl. *bottleneck*). Ovdje dimenzije tenzora ostaju iste (16x16x512) sve dok se ne dođe do prvog bloka povećanja.

Plavom bojom označeni su blokovi povećanja u kojima se vrše suprotne transformacije od blokova smanjenja. Tu se *x* i *y* dimenzije tenzora povećavaju dok se *z* dimenzija (broj mapa značajki) smanjuje sve do izlaznog bloka.

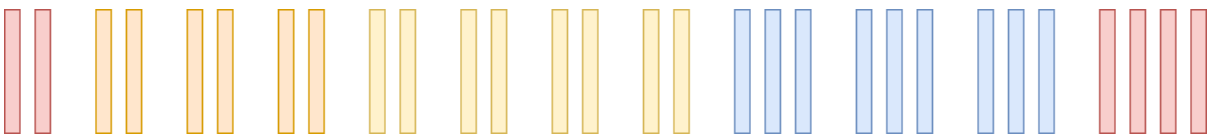
U izlaznom bloku se vrše završne transformacije za postizanje tenzora specificiranog oblika. U ovom slučaju taj je oblik dimenzija 256x256x2.

Linije u dijagramu simboliziraju tok podataka. Pune linije su klasične veze među blokovima bez ikakvih posebnih transformacija. Isprekidane linije crvene boje prikazuju preskočne veze nalik onima u *U-net* arhitekturi gdje se izlaz bloka smanjenja konkatenira s paralelnim blokom povećanja. Crne isprekidane linije simboliziraju rezidualne veze (u ovom slučaju konkatenacije) u najdubljem djelu modela gdje inače lako može doći do eksplodirajućih ili nestajućih gradijenata.



Slika 23 *Dijagram SRUNet modela (autorska slika)*

Dijagram konvolucijskih slojeva (Slika 24) prikazuje broj konvolucijskih slojeva u svakome bloku. Može se primijetiti da se u blokovima povećanja nalazi jedan sloj više koji je potreban za usklađivanje količine mapa značajki. Kod konkatenacije spajaju se mape značajki dva konvolucijska sloja što znači da je broj značajki na ulazu duplo veći, zato se u blokovima povećanja nalazi sloj više kako bi se uskladio broj tih mapa značajki, ali i da ostatak slojeva u bloku ostane simetričan blokovima smanjenja. Izlazni blok (crvene boje) ima više slojeva zbog postepenog pretvaranja tenzora u izlazni oblik.

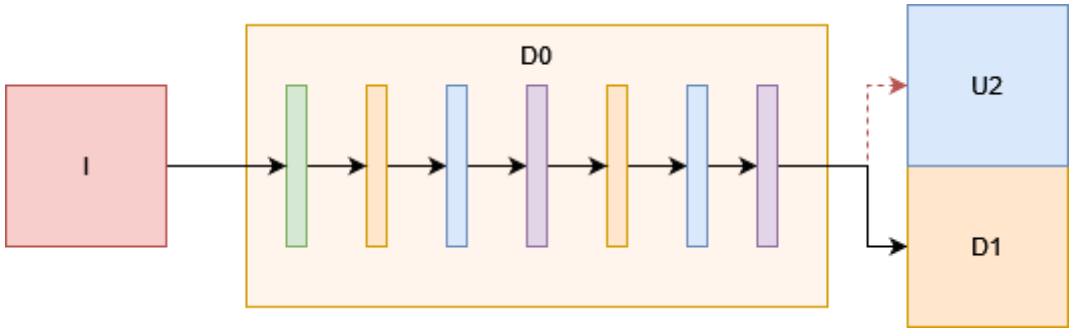


Slika 24 *Dijagram konvolucijskih slojeva (autorska slika)*

Detaljnijim pregledom bloka označenog u dijagramu (Slika 25) kao D0 možemo bolje opisati transformacije nad tenzorom. Zelenom bojom označen je sloj sažimanja po maksimalnoj vrijednosti, narančaste je boje konvolucijski sloj, plave boje normalizacijski sloj te je ljubičaste boje aktivacijski sloj. Model je dizajniran na način da se, ako je potrebno, operacije u sloju sažimanja mogu lako izmijeniti kao i aktivacijska funkcija u aktivacijskom sloju. Iz dijagrama je moguće zaključiti da se radi o dva konvolucijska sloja koje slijede normalizacijski i aktivacijski sloj.

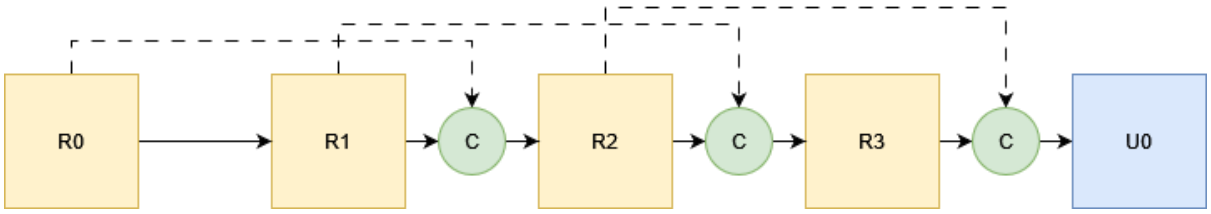
U ovome modelu koristi se normalizacija po instanci umjesto grupne normalizacije. Normalizacija po instanci zasebno normalizira vrijednosti po svakom ulazu u model, dok grupna normalizacija normalizira vrijednost s obzirom na cijelu učitanu grupu te je time ovisna o veličini grupe.

Ulaskom u blok, tenzor se smanjuje u sloju sažimanja, zatim prolazi kroz konvolucijski sloj gdje se u ovom slučaju konvulira s 32 mape na 64, potom se normaliziraju njihove vrijednosti prije ulaska u aktivacijski sloj. Taj se proces ponavlja u drugom dijelu bloka nakon kojeg tenzor odlazi u idući blok, ali i u paralelni blok dijela povećanja.



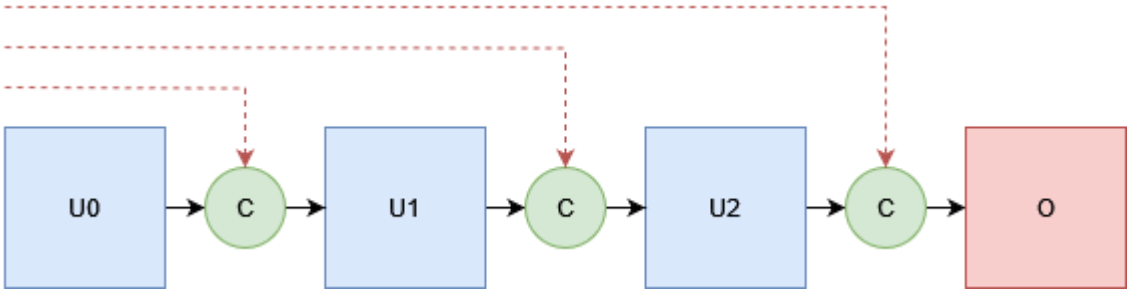
Slika 25 *Dijagram bloka smanjivanja (autorska slika)*

Pregledom najdubljeg djela modela (Slika 26) moguće je vidjeti proces konkatenacije tenzora. Žute boje su duboki blokovi, zelene boje su konkatenacije, a plavi je prvi blok povećanja. Tenzor iz bloka označenog kao R0 ulazi u blok R1, kopija tog tenzora preskače blok R1 nakon čega se spaja s izlazom tog bloka R1. Time se tvori duplo veći tenzor koji je učitan u R2 blok gdje se usklađuju mape značajki ili kanali tenzora te se proces ponavlja za iduće blokove sve do prvog bloka povećanja.



Slika 26 Dijagram najdubljeg dijela modela (autorska slika)

U zadnjem dijelu (Slika 27) model povećava tenzore sve dok ne poprime veličinu ulaznog tenzora ali jedino u *x* i *y* dimenziji jer veličina *z* dimenzije ili količina kanala ovisi o vanjskom parametru koji određuje broj izlaznih kanala. Proces spajanja se vrši na sličan način kao i u najdubljem djelu modela. Tenzori se spajaju operacijom konkatenacije s paralelnim tenzorom dijela smanjenja prije ulaska u blok gdje se *z* dimenzija usklađuje te se nastavljaju operacije kroz slojeve.



Slika 27 Dijagram djela povećanja (autorska slika)

Kao što je navedeno, arhitektura modela je inspirirana *U-netom* i blokovima *ResNeta*. Osim što predstavlja kombinaciju ta dva modela, što ga čini drugačijim od većine, arhitektura sustava koristi još nekoliko novijih tehnika poput velikih i, u ovom slučaju, kaskadnih jezgri konvolucija. Već duže vrijeme većina modela konvolucijskih neuronskih mreža koristi male jezgre konvolucije (do 5x5) za širok spektar zadataka (Ding i sur., 2022). Ding i suradnici (2022) tvrde kako korištenje velikih matrica jezgri konvolucija (veće od 5x5), umjesto mnogo malih, rezultira u boljim performansama konvolucijskih neuronskih mreža tako što se povećava učinkovito receptivno polje (engl. *effective receptive field*) neuronske mreže te čak mogu konkurirati najsuvremenijim metodama. Zbog navedenog, u ovome modelu korištene su velike konvolucijske jezgre. Korišteno je pet različitih veličina jezgri za različite dijelove modela tako što su najveće na ulazu i izlazu modela, a sa smanjivanjem *x* i *y* dimenzija slojeva se postepeno

smanjuju. Tako je i došlo do naziva kaskadne jezgre konvolucije. Najveća korištena matrica jezgre konvolucije je 13x13 koja se nalazi u ulaznom i izlaznom bloku, potom po dubini, to jest, po veličini slojeva u z dimenziji, slijede jezgre veličine 9x9, 7x7, 5x5 te u najužem i najdubljem dijelu modela se nalaze jezgre veličine 3x3.

5.3.3. Nedostaci modela

Potencijalni nedostatak modela je to što, zbog broja slojeva sažimanja, model može uzimati samo kvadratne slike u dimenzijama djeljivim s 32. To bi značilo da je minimalna veličina 32x32, te da je iduća dopuštena veličina tek 64x64, i tako dalje. Do navedenog problema dolazi u dijelu smanjenja zbog broja blokova, to jest, zbog slojeva sažimanja koji su krajevi blokova. Model se sastoji od pet blokova u dijelu smanjenja što znači da bi ulazna slika veličine 16x16 bila smanjena na 8x8 (slojevi sažimanja prepolovljuju veličinu slike) po izlazu iz prvog bloka, smanjena na 4x4 nakon drugog bloka, 2x2 nakon trećeg bloka i 1x1 nakon četvrtog bloka. U petom bloku model izbacuje grešku jer nije u mogućnosti prepoloviti sliku veličine 1x1.

Problemi kvadratnih i premalih slika riješeni su popunjavanjem obruba do tražene veličine ili obrezivanjem slike na potrebnu veličinu tijekom procesa obrade slika, prije ulaska u model. Tražena, to jest, potrebna veličina definirana je sustavnim parametrom za radnu veličinu navedenim u poglavlju o sustavu. Taj parametar određuje veličinu na koju će se slika preoblikovati prije ulaska u model, stoga njegova vrijednost mora biti djeljiva s 32. Dimenzija slike na ulazu u model je jednaka dimenziji slike na izlasku iz modela što znači da veličina generirane (izlazne) slike mora biti djeljiva s 32 barem u jednoj dimenziji. Na primjer, s vrijednost parametra od 512, slika veličine 800x500 će prije ulaza u model biti pretvorena u sliku veličine 512x512, proći će kroz model, te će tek kroz obradu nakon izlaza biti obrezana u sliku veličine 512x320 što nije ista veličina kao i originalna slika, no omjer ostaje isti.

5.3. Izrada i treniranje

Računalo na kojem je sustav izrađen i testiran sastoji se od:

Procesor (CPU) – AMD Ryzen 9 3900X (12 jezgri – 24 logičkih procesora)

Radna memorija (RAM) – 64 GB

Grafički procesor (GPU) – TU104-400 (Nvidia RTX 2080)

Grafička memorija – 8 GB

Sustav je izrađen na Windows 10 operacijskom sustavu u Python programskom jeziku koristeći *Pytorch* biblioteku za izradu i treniranje modela, *pillow* i *scikit-image* biblioteke za obradu i spremanje slika, *numpy*, *Pandas* i *scipy* biblioteke za operacije nad podacima i statistiku te *matplotlib* biblioteku za vizualizaciju.

Sustav je treniran koristeći optimizacijski algoritam *Adam*, funkcija gubitka bila je *glatki L1* gubitak, a aktivacijska funkcija bila je *ELU* funkcija. Treniranje je provedeno na grafičkoj kartici Nvidia RTX 2080 s 8 gigabajta grafičke radne memorije. *Adam* je izabran jer predstavlja jedan od naprednijih algoritama sa samostalnim modificiranjem stope učenja, čime se minimizira potreba za modifikacijom iste putem dodatnih algoritama tokom treniranja. *Glatka L1* funkcija gubitka je izabrana nakon preliminarnog testiranja na manjem skupu podataka gdje se pokazala kao dovoljno stabilna na nasumičnim slikama, a i dovoljno brza u svojem spustu. Iz istog razloga je izabrana i *ELU* aktivacijska funkcija.

Važno je napomenuti da metodologija preliminarnog testiranja nije bila kvalitetna. Preliminarno testiranje sastojalo se od treniranja na malom skupu podataka uz praćenje vrijednosti gubitka i kvalitativnog analiziranja rezultata kroz nekoliko epoha s različitim parametrima. Računao se prosjek tih rezultata te je time bilo odlučeno koji će se parametri koristiti za glavno treniranje. Problem nastaje kod inicijalizacije treniranja gdje je svaki testni krug započinjao s nasumično stvorenim težinama modela, a težine su trebale biti jednake u svakom testnom krugu, zbog toga dolazi do potencijalno iskrivljene slike utjecaja određenih parametara na treniranje.

5.3.1. Petlja za treniranje

Petlja za treniranje se sastojala od učitavanja grupe slika, obrade istih po definiranim parametrima za obradu slika te slanja obrađene grupe slika u grafičku memoriju. Izabrano je treniranje na grafičkoj kartici jer omogućuje puno brže izvođenje jednostavnijih operacija istovremeno za razliku od procesora koji je bolji u kompleksnijim, neparaleliziranim zadacima. Zbog bržeg treniranja korištene su i neke *Pytorch* ugrađene funkcije za optimizaciju procesa na grafičkim karticama poput automatske miješane preciznosti (engl. *automatic mixed precision*) koja samostalno pronalazi idealne tipove podataka u različitim dijelovima procesa učenja za ubrzanje istog. Na primjer, u nekim dijelovima koristi se *float16* tip podataka nad kojim je lakše vršiti operacije u odnosu na *float32*. Zbog korištenja *float16* tipa podataka u nekim dijelovima mora se koristiti i funkcija za skaliranje gradijenata (engl. *gradient scaling function*) da mali gradijenti ne bi bili stisnuti na nulu (*Automatic Mixed Precision package - torch.amp — PyTorch 2.0 documentation*, bez dat.). Takve optimizacije su bile potrebne jer se treniranje vršilo na grafičkoj kartici koja nije namijenjena treniranju neuronskih mreža, nego se radi o grafičkoj kartici potrošačke kvalitete (engl. *consumer grade*).

Nakon što je grupa slika učitana, slike se razdvajaju na L^*, a^* i b^* kanale, s L^* kanalom se generiraju novi a^* i b^* kanali kroz model koji se potom uspoređuju s originalnim a^* i b^* kanalima kroz *glatku L1* funkciju te se tako dobiva vrijednost gubitka kojom se optimiziraju težine modela back-propagacijom. Nakon što se izvrši proces optimizacije *Adam* algoritmom, prazni se predmemorija i kreće se sa sljedećom iteracijom. Petlja za treniranje podržava i spremanje modela te ispisivanje informacija poput trenutne epohe, gubitka u posljednjih nekoliko grupa, gubitka u zadnjoj grupi te informacije o trajanju treniranja i trenutačno vrijeme. Učestalost spremanja modela i ispisa informacija tijekom treniranja kontrolira se posebnim parametrom čestota ispisa. S obzirom da zbog slabosti grafičke kartice nije bilo moguće istovremeno provjeravati uspješnost modela na skupu podataka za testiranje, uveden je navedeni parametar koji određuje učestalost spremanja modela kako bi se performanse modela mogle ručno ocijeniti tokom treniranja na spremljenoj inačici modela. Isto tako, nije bilo moguće uživo stvarati graf gubitka modela stoga su se informacije o gubitku također spremale u posebnu datoteku za kasniju vizualizaciju.

5.3.2. Proces treniranja

Trenirano je nekoliko inačica modela s raznim vrijednostima parametara kroz duže vrijeme sve dok se nije postigla subjektivno zadovoljavajuća brzina i kvaliteta učenja sa *SRUNet* modelom, *Adam* optimizacijskim algoritmom, funkcijom gubitka *glatki L1* i *ELU* aktivacijom. Važni parametri za proces učenja bili su: veličina grupe, veličina slike, stopa učenja, šansa za rezanje, šansa za zakretanje te parametar za popunjavanje slike. Završno treniranje se izvodilo kroz tri etape. Prva etapa sastojala se od treniranja s parametrima:

veličina grupe – 16

veličina slike – 256

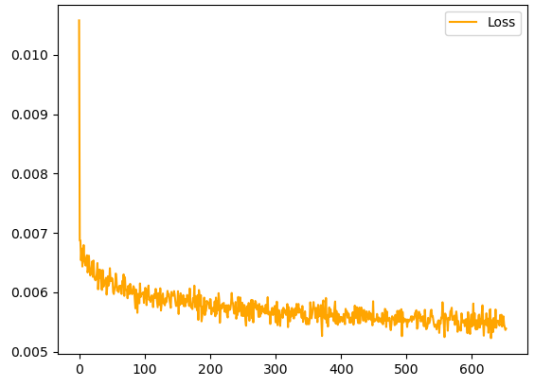
stopa učenja – 1e-5

šansa za rezanje – 50%

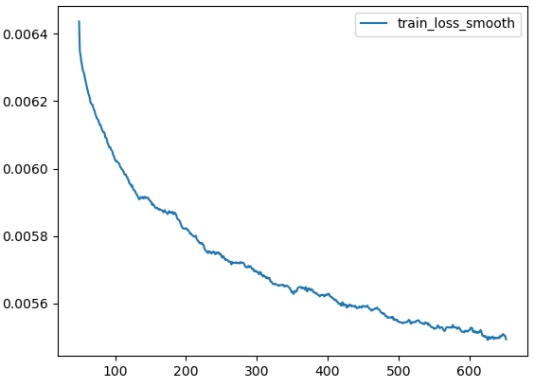
šansa za zakretanje – 50%

popunjavanje slike – konstantna vrijednost (0)

Treniranje u prvoj etapi je trajalo 25 sati i 27 minuta koliko je bilo potrebno za 13 epoha i 30% 14. epohe. Količina od 16 slika u grupi se pokazala kao maksimum koji može izdržati grafička memorija na veličini slike 256x256. Zbog tako male grupe slika, odlučeno je da će se u modelu koristiti normalizacija po instanci umjesto klasične grupne normalizacije – kako bi se izbjegla mogućnost da male grupe loše utječu na treniranje. Preliminarnim testiranjem zaključeno je da u kontekstu *SRUNet*-a za *Adam* optimizaciju najviše odgovara stopa učenja od 1e-5, to jest 0.00001. Šansa za rezanje i šansa za zakretanje nisu bili toliko važni parametri za samo treniranje, ali su imali utjecaj na rezultate modela. Parametar popunjavanja slike konstantom vrijednosti, u ovom slučaju nulom, imao je veliki utjecaj na završni gubitak i na performanse modela. Rubovi slika su se popunjavali nulama da bi se postigla potrebna veličina. U slici 28 moguće je vidjeti gubitak sustava tokom učenja modela, dok je u slici 29 moguće vidjeti isti gubitak, ali prikazan koristeći pomični prosjek (engl. *rolling average*) s prozorom (engl. *window*) od 50 kojim se može zagladiti graf za bolji prikaz tendencija.



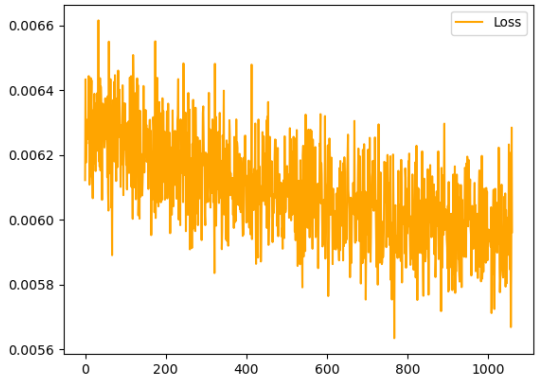
Slika 28 *Gubitak tokom treniranja prve etape (autorska slika)*



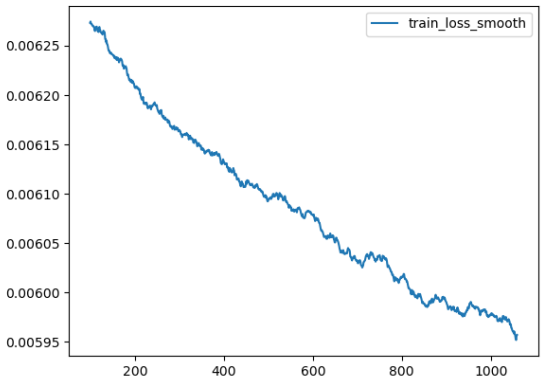
Slika 29 *Gubitak pomičnim prosjekom (autorska slika)*

Druga etapa treniranja sastojala se od istih parametara kao i prva, osim u načinu popunjavanja rubova slika. Trajala je 22 sata i 17 minuta čime je sustav prošao 10 punih epoha. U ovoj etapi umjesto da se rubovi pune nulama, rubovi su bili refleksija zadnjih nekoliko redova (koliko je bilo potrebno da se zadovolji radna veličina) piksela slike. To jest, rubne vrijednosti slike su bile zrcalno okrenute i kopirane sve dok se nije popunio prostor. Rezultati ove etape su već bili subjektivno zadovoljavajući. U trećoj etapi je promijenjena stopa učenja s 1e-5 na 1e-6 (0.000001). Ideja iza smanjivanja stope učenja je bila ta da se spriječe veće promjene gradijenata te da se model samo dodatno nauči. Treća etapa je također obuhvaćala 10 epoha i trajala je 21 sat i 55 minuta. Razlika u vremenu potrebnom da se završi druga etapa i da se završi treća vjerojatno je nastala zbog češćeg testiranja tokom treniranja u drugoj etapi. Na slici 30 vidi se gubitak u drugoj i trećoj etapi te se zbog nestabilnosti gubitka teško može iščitati, stoga je na slici 31 izračunat pomični prosjek s prozorom od 100 za lakše shvaćanje trenda.

Pregledom slike, može se vidjeti da ima još potencijalnog prostora za učenje jer je tendencija gubitka i dalje prema manjim vrijednostima, no mora se pripaziti na pretreniranje modela što nažalost nije bilo moguće grafički prikazati u ovom slučaju jer model nije konstantno testiran na skupu za testiranje.



Slika 30 *Gubitak tokom treniranja u drugoj i trećoj etapi (autorska slika)*



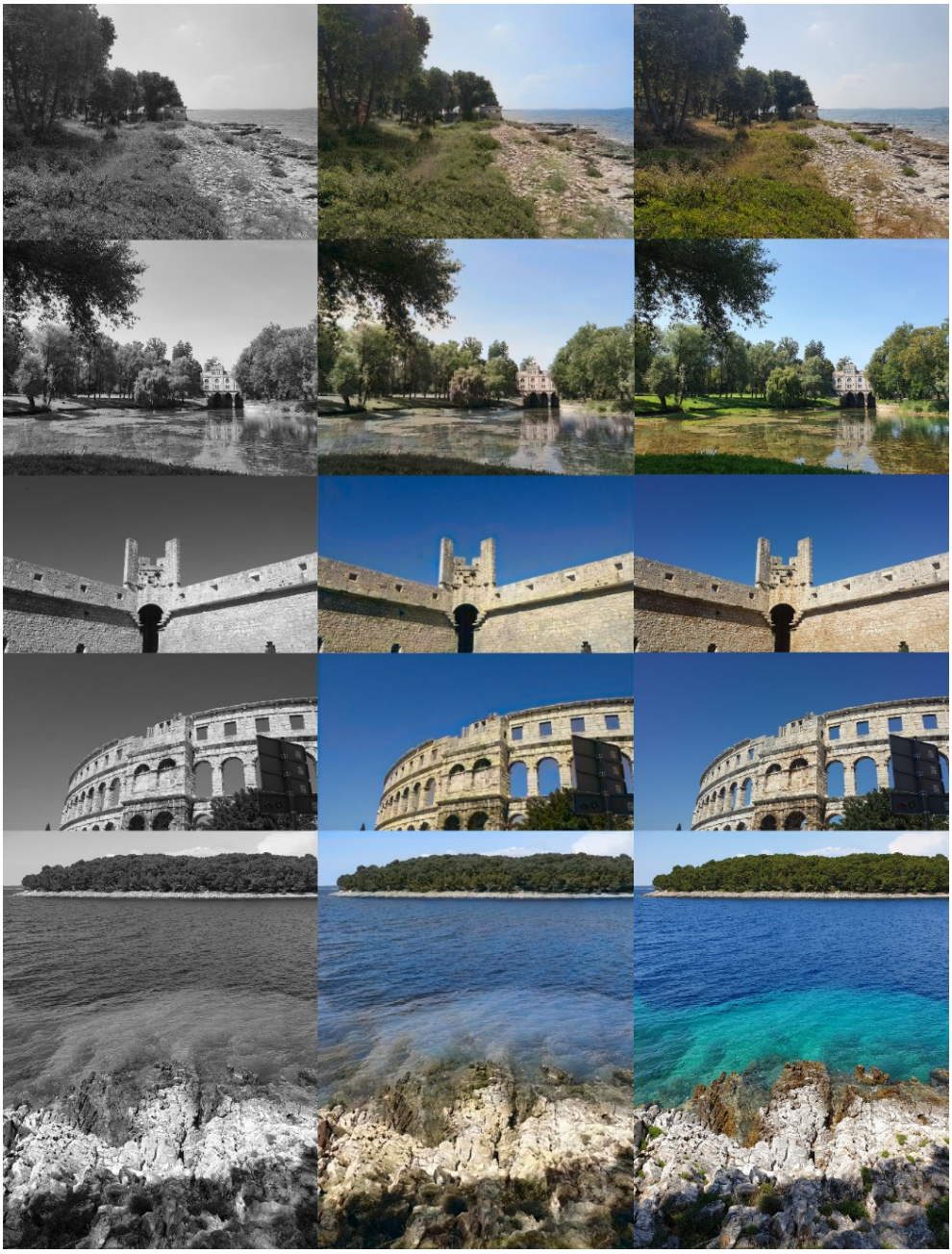
Slika 31 *Gubitak s pomičnim prosjekom u drugoj i trećoj etapi (autorska slika)*

5.4. Rezultati

U ovom slučaju kvaliteta generiranog sadržaja teško se može kvantificirati, zbog čega ovo poglavlje sadrži samo kvalitativne rezultate i analize.

5.4.1. Fotografije u boji

Prvo će se pregledati slike koje imaju svoju varijantu u boji. Slike u ovom dijelu su iz osobne kolekcije te model nije bio učen na njima. S lijeve strane se nalazi crno bijela varijanta slike, u sredini je *SRUNet*-ova halucinacija, a desno je temeljna istina, to jest, originalna fotografija.



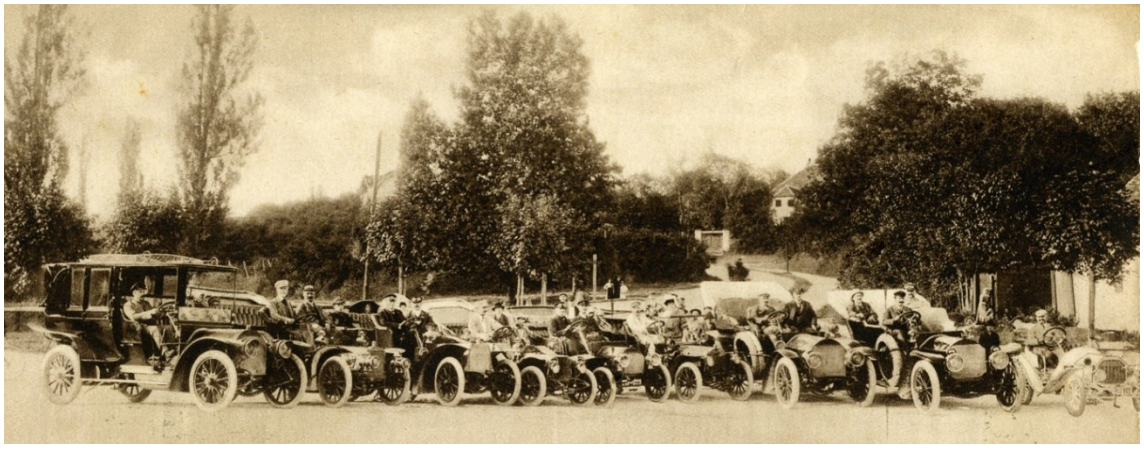
Slika 32 *Usporedbe SRUNet rezultata i stvarnosti (autorska slika)*

Slika 33 *Usporedbe SRUNet rezultata i stvarnosti (autorska slika)*

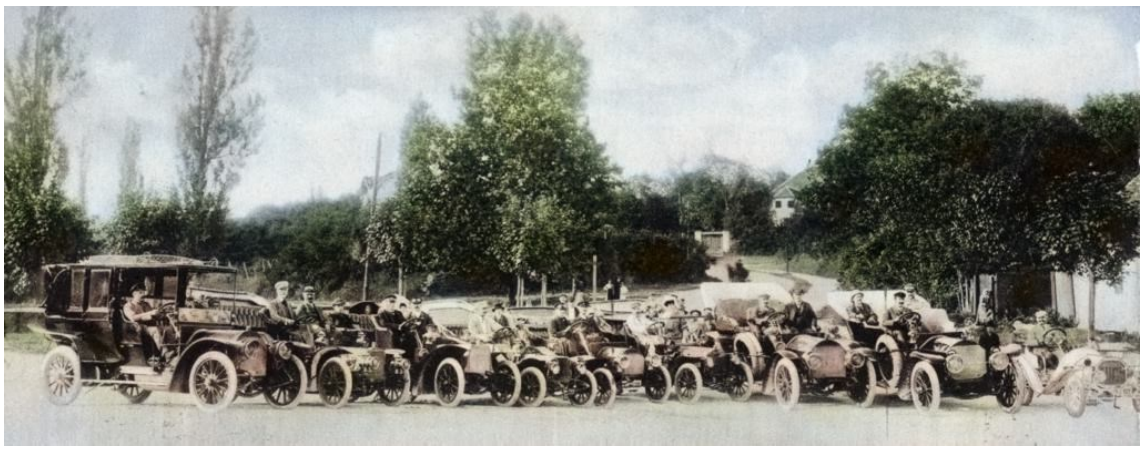
Iz slika 32 i 33, moguće je zaključiti da *SRUNet* većinom dobro kolorizira pejsaže - prepoznaje nebo i boji ga u plavo, prepoznaje bilje i boji ga u zeleno te prepoznaje vodu i boji ju u plavu, iako voda nije uvijek plava u stvarnosti. Valja napomenuti da su u ovoj sekciji sve slike učitane u model u veličini od 1024x1024 što je veličina slika na kojoj model nikad nije bio treniran – model je treniran na slikama veličine 256x256 što je četiri puta manje od slika koje je stvorio. To sugerira da treniranje na manjim slikama, s parametrima navedenima u poglavlju o treniranju, rezultira modelom koji može generirati veći sadržaj, to jest, da je skalabilan do određene mjere. Ovakvi rezultati vjerojatno su omogućeni treniranjem na obrezanim slikama koje modelu omogućuju bolje raspoznavanje teksture, a ne samo kompozicije slike.

5.4.2. Originalno crno-bijele i monokromatske fotografije

Iduća sekcija sadrži originalno crno-bijele i monokromatske slike. Model, također, nije imao pristup ovim slikama. Prva prikazana slika je slika automobila na ulici veličine 1024x1024. Slika je prikazana u svojem originalnom, monokromatskom obliku te nakon nje slijedi slika koju je model kolorizirao. Moguće je vidjeti dobro obojano nebo i drveće, dok automobili nisu toliko dobro obojani.



Slika 34 *Originalna slika (As4, bez dat.)*



Slika 35 *SRUNet rezultat (kolorizirana verzija slike As4, bez dat.,)*

Idući par prikazuje automobile ispred građevine. Također se radi o slikama veličine 1024x1024 gdje je prva slika originalna inačica, a druga je kolorizirana. Može se vidjeti da prilikom kolorizacije starih, mutnih slika s mnogo smetnji u originalu generirani sadržaj dovodi do slabijih rezultata kolorizacije, to jest, boje nemaju jednaku razinu zasićenosti kao u slikama bez smetnji.

Slika 36 *Originalna slika (As1, bez dat.)*

Slika 37 *SRUNet rezultat (kolorizirana verzija slike As1, bez dat.)*

Idući par fotografija stvoren je u veličini od 2048x2048 što ujedno čini najveću podržanu veličinu slika zbog hardverskih ograničenja računala na kojemu je model izvođen. Originalna inačica slike sadrži manju količinu smetnji, stoga je moguće primijetiti da su u koloriziranoj slici boje intenzivnije nego u prethodnom paru. Moguće je vidjeti da je model dobro obojao floru, nebo te čak na nekim dijelovima krovove i fasade građevina, no cesta je postala pretežito plava, što nije idealno rješenje.

Slika 38 *Originalna slika (Ponce de Leon Hotel, 1890)*

Slika 39 *SRUNet rezultat (kolorizirana verzija slike Ponce de Leon Hotel, 1890)*

5.4.3. Nedostaci

SRUNet dobro boji slike pejsaža i vedute, no veliki nedostatak modela čini loša kolorizacija ljudi i slika s atipičnim kompozicijama. Prve tri slike (slika 40, 41 i 42) predstavljaju koloriziranu inačicu slike koja je u originalu crno-bijela te ih model nikad prije nije vidio. Može se primijetiti da je model neuspješno obojao lica ljudi makar je uspješno obojao pozadinsku sliku drveća. Također, model je obojio odjeću muškarca u zelenu, vjerojatno jer ga teksturom podsjeća na travu (Slika 40). Druga fotografija (Slika 41) prikazuje malo bolji rezultat, ali i dalje nije idealan. Lice i pozadina su dobro obojani, no vrat i ruke su promašeni. Treća slika (Slika 42) pokazuje kako model ponekad prepoznaje lica, no ne svaki put. Lijevo lice je dobro obojano, ali desno nije.

Slika 40 *Prva slika lošijih rezultata (kolorizirana verzija slike Fritz Reuter, 1894)*

Slika 41 *Druga slika lošijih rezultata (kolorizirana verzija sličice iz Tanhofer, 1957)*

Slika 42 *Treća slika lošijih rezultata (kolorizirana verzija slike Eckert, 1882)*

Idući set slika pokazuje neuspješnost modela u kolorizaciji atipičnih kompozicija. Ove slike su iz osobne kolekcije te ih model nikad nije vidio. Prva slika je modelova halucinacija, a druga je temeljna istina. Iz slika 43 i 44 može se zaključiti da model prepoznaje gdje nebo prestaje, a gdje počinje prostor suncobrana, no svejedno ga nije vjerno obojao.

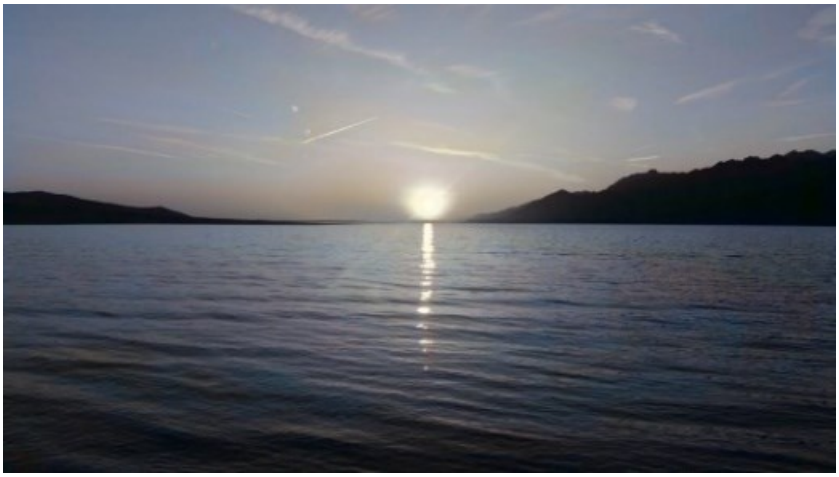
Slika 43 *SRUNet rezultat (autorska slika)*

Slika 44 *Temeljna istina (autorska slika)*

Iz slika 45 i 46 može se zaključiti da model nije sposoban vjerno obojati nebo i ostale objekte ako se radi o slici koja nije tipične kompozicije. Zanimljivo je i to što model nije sposoban prepoznati i točno obojati zalaske sunca (nedostaju tople boje) kao što možemo vidjeti iz slika 47 i 48.

Slika 45 *SRUNet rezultat (autorska slika)*

Slika 46 *Temeljna istina (autorska slika)*



Slika 47 *SRUNet rezultat (autorska slika)*

Slika 48 *Temeljna istina (autorska slika)*

5.4.4. Usporedba s drugim modelima

Prvi model s kojim se vrši usporedba je model iz rada „Colorful Image Colorization“ (Zhang i sur., 2016). godine kojeg se u ovom radu naziva *CIC2016* model. Vrše se tri usporedbe gdje je prva slika stvorena *SRUNet* modelom, a druga je rezultat *CIC2016* modela.

Slika 49 *SRUNet rezultat (kolorizirana verzija slike Adams, 1944)*

Slika 50 *CIC2016 rezultat (Zhang i sur., 2016)*

Slika 51 *SRUNet rezultat (kolorizirana verzija slike Cartier-Bresson, 1938)*

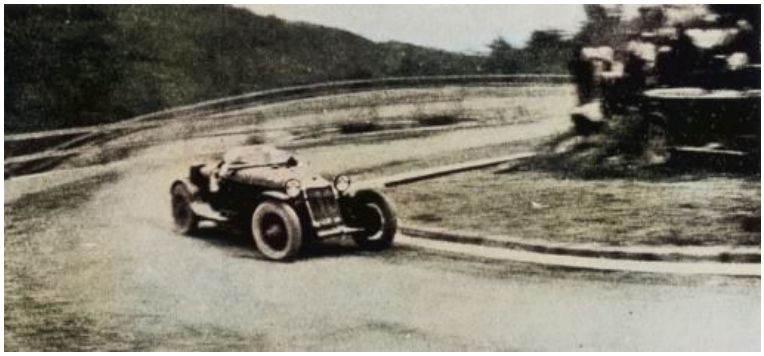
Slika 52 *CIC2016 rezultat (Zhang i sur., 2016)*

Slika 53 *SRUNet rezultat (kolorizirana verzija slike Cartier-Bresson, 1968)*

Slika 54 *CIC2016 rezultat (Zhang i sur., 2016)*

U slikama 49-54 može se vidjeti da *CIC2016* stvara mnogo intenzivnije boje od *SRUNet* modela stvorenog u ovom projektu. Zhang i suradnici (2016) su koristili posebnu tehniku klasifikacije kroz rebalans vrijednosti za dobivanje intenzivnih boja te se treniranje vršilo na kompletnom *ImageNet* skupu podataka čime se susreo sa širim rasponom motiva i brojnijim primjerima.

Drugi model s kojim je rađena usporedba je *DeOldify* model koji predstavlja novije modele za kolorizaciju korištenjem generativnih suparničkih mreža (engl. *generative adversarial networks*) (Antic, 2023). Usporedba je prezentirana na isti način kao i prethodna. Prva slika je rezultat *SRUNet* modela dok je druga rezultat modela s kojim se uspoređuje.



Slika 55 *SRUNet rezultat (kolorizirana verzija slike 000086-004, bez dat.)*

Slika 56 *DeOldify rezultat (kolorizirana verzija slike 000086-004, bez dat.)*

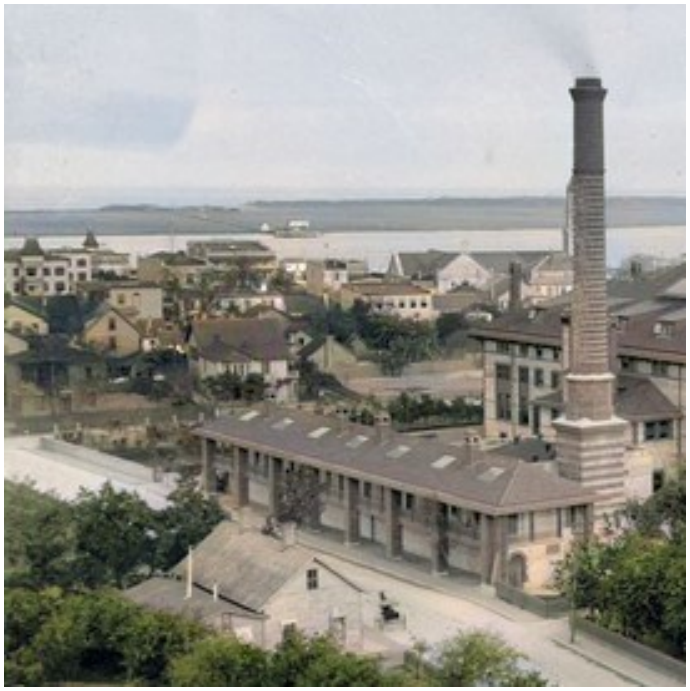
Slika 57 *SRUNet rezultat (kolorizirana verzija sličice iz Tanhofer, 1957)*

Slika 58 *DeOldify rezultat (kolorizirana verzija sličice iz Tanhofer, 1957)*

Slika 59 *SRUNet rezultat (autorska slika)*

Slika 60 *DeOldify rezultat (autorska slika)*

Slika 61 *Detalji u SRUNet rezultatu (kolorizirana verzija slike Ponce de Leon Hotel, 1890)*



Slika 62 *Detalji u DeOldify rezultatu (kolorizirana verzija slike Ponce de Leon Hotel, 1890)*

Kroz slike 55-62 može se vidjeti da *DeOldify* model većinom bolje boji sadržaj, a pogotovo je bolji u definiranju detalja (slike 61 i 62) i bojanju ljudi (slike 57 i 58). Ljudi su vjerno obojani, a boja se rijetko prelijeva iz objekta što je čest problem *SRUNet* modela. *SRUNet* model je dobro pogodio boje u slici 59, no zbog prelijevanja boja *DeOldify* rezultat izgleda bolje. Zanimljivo je to što ni *SRUNet* niti *DeOldify* nisu kvalitetno obojali sliku automobila (slike 55 i 56).

5.5. Nastavak istraživanja

Istraživanje je provedeno na posebnom skupu podataka sastavljenom od dijelova više različitih skupova podataka. Skup se sastojao od ukupno 213.522 fotografija u boji s motivima pejsaža, ljudi i građevina.

Model *SRUNet* je *U-net* tip modela s modifikacijama u najdubljem dijelu. Za konvolucije su korištene kaskadne, velike jezgre te se koristila normalizacija po instanci umjesto grupne normalizacije. Također, model koristi *ELU* aktivacijske funkcije umjesto popularnih *ReLU* aktivacijskih funkcija.

Sustav je treniran *Adam* metodom učenja na osobnom računalu kroz ukupno 33 epohe, koristeći *glatki L1* funkciju za računanje gubitka. Sustav je treniran na obrađenim slikama 256x256 veličine. Slike su bile nasumično izabrane iz skupa podataka u grupama od 16 slika te su bile individualno obrađene rezanjem, mijenjanjem veličine ili zakretanjem. Slike su bile pretvorene iz *RGB* prostora boje u *CIELab* prostor boja iz kojeg se *L** kanal koristio za stvaranje *a** i *b** kanala.

Osnovni nedostaci modela u usporedbi s drugim modelima za kolorizaciju osnovanim na konvolucijskim neuronskim mrežama su: loše bojanje ljudi, prelijevanje boja, loše bojanje slika atipičnih kompozicija, općenita slaba zasićenost stvorenih slika bojom te nužnost stvaranja slika čija je veličina djeljiva s 32.

S obzirom na navedene nedostatke modela potreban je nastavak istraživanja. Model *SRUNet* bi se vjerojatno mogao poboljšati dužim treniranjem, kao i treniranjem na većem skupu podataka poput *ImageNet* skupa. Preliminarnim testiranjem zaključeno je i da modelu odgovara učenje na većim radnim veličinama poput 512x512, no u ovom slučaju hardverska ograničenja su onemogućila treniranje na većim veličinama, kao i duže treniranje ili treniranje na većem skupu podataka. S obzirom na to da model može prihvatiti veće dimenzije slika, trebalo bi se istražiti kako bi model učio postepenim treniranjem gdje se u svakoj etapi povećava radna veličina i smanjuje se stopa učenja. Poboljšanje performansi modela potencijalno je moguće i kroz još opsežniju obradu slika poput dodavanja šuma na ulazne slike za bolji učinak na crno-bijelim slikama ili dodavanjem drugih augmentacija poput zakretanja slike za 90 stupnjeva za sprječavanje modelove neuspješnosti kolorizacije atipičnih kompozicija. Navedene eksperimente nije bilo moguće izvršiti u ovome istraživanju radi ograničenih hardverskih mogućnosti, no takva istraživanja bi upotpunila daljnje rezultate.

Osim navedenog, predlaže se i daljnja usporedba utjecaja *glatkog L1* gubitka i utjecaja neke druge funkcije poput Huberovog gubitka.

Daljnja poboljšanja su moguća kroz naprednije metode računanja gubitka poput perceptualnog gubitka (engl. *perceptual loss*) ili gubitka kroz generativne suparničke mreže (engl. *generative adversatial networks*). Obje spomenute metode računanja gubitka su bolje u računanju gubitka na slikovnom sadržaju od metoda korištenih u projektu (u projektu se koristio gubitak po pikselu slike funkcijom *glatkog L1* gubitka), ali su previše napredne za hardver na kojem je sustav treniran te su time izvan opsega ovog rada. Trebale bi se i bolje istražiti druge aktivacijske funkcije poput *PReLU* aktivacije obrađene u teorijskom dijelu, ali i novije funkcije poput *Mish* i *SiLU* funkcije.

Model *SRUNet* je fleksibilan u svojem dizajnu (arhitektura se djelomično može izmijeniti vanjskim parametrima) te ga je lako prenamijeniti za druge zadatke generacije slikovnog sadržaja zbog čega bi bilo dobro istražiti njegovu potencijalnu širu primjenu. Također, model se može primijeniti i na zadatke klasifikacije dodavanjem klasifikacijskog vektora te izmjenom funkcije gubitka funkcijom za klasifikaciju, poput funkcije za unakrsni gubitak entropije (engl. *cross-entropy loss*). Jednostavnost arhitekture modela omogućuje treniranje i pokretanje na slabijim računalima te time zahtjeva daljnje istraživanje njegovih primjena u slabijim sustavima.

6. Zaključak

Cilj ovog rada bio je izrada i kontekstualizacija sustava za kolorizaciju crno-bijelih slika na osobnom računalu kroz dizajn arhitekture, programiranje i treniranje modela konvolucijskih neuronskih mreža. Također, cilj je bio i sastaviti poseban skup podataka za kolorizaciju za brže treniranje na slabijim računalima.

U radu se prikazala teorija neuronskih mreža s naglaskom na konvolucijske neuronske mreže te se oprimjerila kroz projekt izrade sustava za kolorizaciju crno-bijelih slika. U teorijskom dijelu prikazane su neuronske mreže od svojih jednostavnih jednoslojnih oblika do dubokih mreža i mreža osnovanih na konvolucijskim slojevima. Predstavljene su i osnovne aktivacijske funkcije, funkcije gubitka te metode učenja modela neuronskih mreža.

U istraživanju je prikazano kako je moguće izraditi i trenirati kompleksni model konvolucijskih neuronskih mreža na osobnom računalu koristeći optimizacije dostupne u *Pytorch* programskoj biblioteci.

U istraživanju je provedeno treniranje samostalno dizajniranog i izrađenog modela i cijelog sustava za kolorizaciju crno-bijelih i monokromatskih slika. Treniranje se odvijalo koristeći skup podataka od preko 200.000 slika u boji koji je sastavljen od dijelova više skupova podataka s vizualnim materijalima. Treniranje sa sastojalo od ukupno 33 epohe, to jest, od 33 prelaska kroz čitav skup podataka.

Model predstavljen u radu, nazvan *SRUNet*, se pokazao kao dovoljan za zadatke kolorizacije pejsaža te je u nekim aspektima čak i sposoban konkurati ostalim, profesionalnim sustavima za kolorizaciju. Daljnjim istraživanjem i treniranjem modela vjerojatno bi se mogla postići razina na kojoj bi *SRUNet* bio sposoban rješavati i kompleksnije zadatke kolorizacije poput bojanja slika neuobičajenih kompozicija.

Kroz istraživanje sastavljen je i poseban skup podataka za lakše treniranje na slabijim sustavima poput osobnog računala izbacivanjem većine nepotrebnih slika za zadatak kolorizacije. Pokazao se kao dovoljno dobar skup koji se uspješno koristio za treniranje modela za kolorizaciju.

Potencijalna poboljšanja modela se većinom svode na rješavanje problema koji proizlaze iz hardverskih ograničenja, a uključuju drugačije načine računanja gubitka u modelu i produljenje procesa treniranja kao i povećanje radne veličine slika modela tokom treniranja. *SRUNet* model napravljen je na način da je ostavljen prostor za djelomičnu modifikaciju, čime se otvara i

potencijalno šira primjena modela na zadacima stvaranja slikovnog sadržaja poput povećavanja razlučivosti slika ili pak stvaranja potpuno novog sadržaja. Također, promjenama u izlaznom djelu modela i korištenjem drugih funkcija gubitka, model bi se mogao prenamijeniti za klasifikaciju.

7. Literatura

1. *000086-004*. (bez dat.). http://www.opatija.net/hr/povijest
2. Adams, A. (1944). *Mount Williamson, Sierra Nevada from Manzanar, CA* [Fotografija]. https://www.anseladams.com/mt-williamson-manzanar/
3. Alpaydin, E. (2004). *Introduction to Machine Learning*. MIT Press.
4. Antic, J. (2023). *DeOldify* [Python]. https://github.com/jantic/DeOldify
5. *Applications for Python*. (bez dat.). Python.Org. Preuzeto 07. kolovoz 2023., od https://www.python.org/about/apps/
6. *As1*. (bez dat.). [Fotografija]. https://www.otk-ferdinandbudicki.hr/index.php/knjiga
7. *As4*. (bez dat.). [Fotografija]. https://www.otk-ferdinandbudicki.hr/index.php/knjiga
8. *Automatic Mixed Precision package—Torch.amp—PyTorch 2.0 documentation*. (bez dat.). Preuzeto 07. kolovoz 2023., od https://pytorch.org/docs/stable/amp.html?highlight=amp#module-torch.amp
9. Brownlee, J. (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. *MachineLearningMastery.Com*. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
10. Cartier-Bresson, H. (1938). *On the banks of the Marne, France* [Fotografija]. https://www.icp.org/browse/archive/objects/on-the-banks-of-the-marne-france
11. Cartier-Bresson, H. (1968). *Brie, France* [Fotografija]. https://www.icp.org/browse/archive/objects/brie-france
12. Chen, S.-Y., Zhang, J.-Q., Zhao, Y.-Y., Rosin, P. L., Lai, Y.-K., & Gao, L. (2022). A review of image and video colorization: From analogies to deep learning. *Visual Informatics*, *6*(3), 51–68. https://doi.org/10.1016/j.visinf.2022.05.003

13. Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)* (arXiv:1511.07289). arXiv. http://arxiv.org/abs/1511.07289
14. Compton, E. A., & Ernstberger, S. L. (2020). *Singular Value Decomposition: Applications to Image Processing*.
15. Dalbelo Bašić, B., Čupić, M., & Šnajder, J. (2008). *Umjetne neuronske mreže*. Zavod za elektroniku, mikroelektroniku i inteligentne sustave, Fakultet elektrotehnike i računarstva.
16. Ding, X., Zhang, X., Zhou, Y., Han, J., Ding, G., & Sun, J. (2022). *Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs* (arXiv:2203.06717). arXiv. http://arxiv.org/abs/2203.06717
17. Dunđer, I., Seljan, S., & Odak, M. (2023). Data Acquisition and Corpus Creation for Phishing Detection. *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, 533–538. https://doi.org/10.23919/MIPRO57284.2023.10159904
18. Eckert, H. (1882). *Rudolf, Kronprinz von Österreich Gemeinsam mit Kronprinzessin Stephanie*. http://www.opatija.net/hr/povijest
19. *Fritz Reuter*. (1894). [Fotografija]. https://www.junipergallery.com/node/8360
20. Gallagher, A., & Chen, T. (bez dat.). *The Images of Groups Dataset* [dataset]. Preuzeto 07. kolovoz 2023., od http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html
21. *Glossary—Convolution*. (bez dat.). Preuzeto 23. kolovoz 2023., od https://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm
22. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org
23. *Google Landmarks Dataset v2*. (2023). [dataset]. Common Visual Data Foundation. https://github.com/cvdfoundation/google-landmark

24. Hasting, G., & Rubin, A. (2012). Colour spaces—A review of historic and modern colour models*. *African Vision and Eye Health*, *71*. https://doi.org/10.4102/aveh.v71i3.76
25. He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification* (arXiv:1502.01852). arXiv. http://arxiv.org/abs/1502.01852
26. *HuberLoss—PyTorch 2.0 documentation*. (bez dat.). Preuzeto 07. kolovoz 2023., od https://pytorch.org/docs/stable/generated/torch.nn.HuberLoss.html#torch.nn.HuberLoss
27. *Image Module*. (bez dat.). Pillow (PIL Fork). Preuzeto 07. kolovoz 2023., od https://pillow.readthedocs.io/en/stable/reference/reference/Image.html
28. Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* (arXiv:1502.03167). arXiv. http://arxiv.org/abs/1502.03167
29. *Kolorizacija*. (bez dat.). Hrvatski jezični portal. Preuzeto 07. kolovoz 2023., od https://hjp.znanje.hr/index.php?show=search_by_id&id=eltiXBg%3D&keyword=kolorizacija
30. Kovač, A., Dunđer, I., & Seljan, S. (2022). An overview of machine learning algorithms for detecting phishing attacks on electronic messaging services. *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, 954–961. https://doi.org/10.23919/MIPRO55190.2022.9803517
31. Kuhlman, D. (bez dat.). *A Python Book: Beginning Python, Advanced Python, and Python Exercises*.
32. Kwiatkowski, R. (2022). *Gradient Descent Algorithm—A deep dive*. Medium. https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21
33. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), Article 7553. https://doi.org/10.1038/nature14539

34. Li, F.-F., Jia, D., Olga, R., Alex, B., & Kai, L. (bez dat.). *ImageNet* [dataset]. Preuzeto 07. kolovoz 2023., od https://www.image-net.org/download.php

35. Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

36. Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., Lanczi, L., Gerstner, E., Weber, M.-A., Arbel, T., Avants, B. B., Ayache, N., Buendia, P., Collins, D. L., Cordier, N., … Van Leemput, K. (2015). The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, *34*(10), 1993–2024. https://doi.org/10.1109/TMI.2014.2377694

37. *NumPy documentation—NumPy v1.25 Manual*. (bez dat.). Preuzeto 07. kolovoz 2023., od https://numpy.org/doc/stable/

38. Omkar M, P., Andrea, V., Andrew, Z., & C. V., J. (bez dat.). *The Oxford-IIIT Pet Dataset* [dataset]. Preuzeto 07. kolovoz 2023., od https://www.robots.ox.ac.uk/~vgg/data/pets/

39. *Oxford & Paris Buildings Dataset*. (bez dat.). [dataset]. https://www.kaggle.com/datasets/skylord/oxbuildings

40. Paik, I., & Choi, J. (2023). *The Disharmony between BN and ReLU Causes Gradient Explosion, but is Offset by the Correlation between Activations* (arXiv:2304.11692). arXiv. http://arxiv.org/abs/2304.11692

41. *Ponce de Leon Hotel*. (1890). [Fotografija]. https://www.junipergallery.com/node/355

42. *PyTorch documentation—PyTorch 2.0 documentation*. (bez dat.). Preuzeto 07. kolovoz 2023., od https://pytorch.org/docs/stable/index.html

43. Roberts, D. A., Yaida, S., & Hanin, B. (2021). *The Principles of Deep Learning Theory*. https://doi.org/10.1017/9781009023405

44. Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation* (arXiv:1505.04597). arXiv. http://arxiv.org/abs/1505.04597

45. Rougetet, A. (bez dat.). *Landscape Pictures* [dataset]. Preuzeto 07. kolovoz 2023., od https://www.kaggle.com/datasets/arnaud58/landscape-pictures

46. Ruder, S. (2017). *An overview of gradient descent optimization algorithms* (arXiv:1609.04747). arXiv. http://arxiv.org/abs/1609.04747

47. Russell, S., & Norvig, P. (2020). *Artifical Intelligence: A Modern Approach* (4. izd.). Pearson.

48. *scikit-image's documentation—Skimage 0.21.0 documentation*. (bez dat.). Preuzeto 07. kolovoz 2023., od https://scikit-image.org/docs/stable/

49. Seif, G. (2022). *Understanding the 3 most common loss functions for Machine Learning Regression*. Medium. https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

50. Seljan, S., Tolj, N., & Dunđer, I. (2023). Information Extraction from Security-Related Datasets. *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, 539–544. https://doi.org/10.23919/MIPRO57284.2023.10159920

51. Sharma, S., Sharma, S., & Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, *04*(12), 310–316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

52. *SmoothL1Loss—PyTorch 2.0 documentation*. (bez dat.). Preuzeto 07. kolovoz 2023., od https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

53. Tan, M., & Le, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* (arXiv:1905.11946). arXiv. http://arxiv.org/abs/1905.11946

54. Tanhofer, N. (Redatelj). (1957). *Nije bilo uzalud*. Jadran Film.

55. *Umjetna inteligencija*. (bez dat.). Hrvatska enciklopedija. Preuzeto 07. kolovoz 2023., od https://www.enciklopedija.hr/natuknica.aspx?ID=63150

56. Wang, C.-F. (2019). *The Vanishing Gradient Problem*. Medium. https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

57. Weisstein, E. W. (bez dat.). *Convolution* [Text]. Wolfram Research, Inc. Preuzeto 23. kolovoz 2023., od https://mathworld.wolfram.com/

58. *What is a Tensor?* (bez dat.). Preuzeto 07. kolovoz 2023., od https://www.doitpoms.ac.uk/tlplib/tensors/what_is_tensor.php

59. *WIDER FACE: A Face Detection Benchmark*. (bez dat.). [dataset]. Preuzeto 07. kolovoz 2023., od http://shuoyang1213.me/WIDERFACE/

60. Yathish, V. (2022). *Loss Functions and Their Use In Neural Networks*. Medium. https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9

61. Yu-Wei, C., Wang, Z., He, Y., Wang, J., Law, H., Liu, Y., Liu, X., Zeng, H., & Deng, J. (bez dat.). *HICO & HICO-DET* [dataset]. Preuzeto 07. kolovoz 2023., od http://www-personal.umich.edu/~ywchao/hico/

62. Zhang, R., Isola, P., & Efros, A. A. (2016). *Colorful Image Colorization* (arXiv:1603.08511). arXiv. http://arxiv.org/abs/1603.08511

Popis slika

Uporaba konvolucijskih neuronskih mreža za kolorizaciju crno-bijelih slika

Sažetak

Cilj ovog rada je izraditi sustav strojnog učenja za kolorizaciju crno-bijelih slika na osobnom računalu, osnovan na konvolucijskim neuronskim mrežama. U teorijskom dijelu rada prezentira se teorijska podloga neuronskih mreža s naglaskom na aktivacijske funkcije, funkcije gubitka i algoritme učenja neuronskih mreža. U praktičnom dijelu prikazana je izrada, treniranje i dizajn modela te sustava za kolorizaciju crno-bijelih slika. Također, izrađen je i poseban skup podataka za treniranje sustava za kolorizaciju. Detaljno je opisana arhitektura modela, te je model kontekstualiziran usporedbom sa drugim kolorizacijskim modelima. Dokazuje se da je moguće izraditi i trenirati sustav za kolorizaciju na osobnom računalu uz optimizaciju procesa treniranja. Također, dokazuje se da je moguće izraditi model koji je usporediv sa drugim modelima konvolucijskih neuronskih mreža koristeći nove metode i elemente arhitekture modela.

Ključne riječi: strojno učenje, neuronske mreže, konvolucijske neuronske mreže, aktivacijske funkcije, funkcije gubitka, metode učenja neuronskih mreža, kolorizacija

Using convolutional neural networks for colorizing black and white images

Summary

The goal of this paper is to create a machine learning system based on convolutional neural networks for colorization of black and white images on a personal computer. I the theoretical part of this paper neural networks are presented with an emphasis on activation functions, loss functions and learning algorithms of neural networks. The practical part of this paper describes building, designing and training of the model, and system for colorization of black and white images. Also, a novel dataset has been developed for training of colorization systems. Model architecture is detailed, and the model is contextualized by comparing it to other colorization models. This paper proves that it is possible to build and train a colorization system on a personal computer by optimizing the training process. Also, it is proven that it is possible to build a model that is comparable with other models based on convolutional neural networks by using new methods and elements of model architecture.

Key words: machine learning, neural networks, convolutional neural networks, activation functions, loss functions, neural network optimization algorithms, colorization