



UNIVERZITET U NOVOM
SADU
FAKULTET TEHNIČKIH
NAUKA U NOVOM SADU




Nikola Zubić

Korišćenje neuralnog prenosa stila uz optimizaciju očuvanja boja za generisanje umjetničkih radova

Diplomski rad
- Osnovne akademske studije -

Novi Sad, 2020.

	UNIVERZITET U NOVOM SADU FAKULTET TEHNIČKIH NAUKA 21000 NOVI SAD, Trg Dositeja Obradovića 6	Datum:
	ZADATAK ZA IZRADU DIPLOMSKOG (BACHELOR) RADA	List:
		1/1

(Podatke unosi predmetni nastavnik - mentor)

Vrsta studija:	Osnovne akademske studije
Studijski program:	Softversko inženjerstvo i informacione tehnologije
Rukovodilac studijskog programa:	vanr. prof. dr Miroslav Zarić

Student:	Nikola Zubić	Broj indeksa:	SW 3/2016
Oblast:	Soft kompjuting		
Mentor:	vanr. prof. dr Jelena Slivka		

NA OSNOVU PODNETE PRIJAVE, PRILOŽENE DOKUMENTACIJE I ODREDBI STATUTA FAKULTETA IZDAJE SE ZADATAK ZA DIPLOMSKI RAD, SA SLEDEĆIM ELEMENTIMA:

- problem – tema rada;
- način rešavanja problema i način praktične provere rezultata rada, ako je takva provera neophodna;
- literatura

NASLOV DIPLOMSKOG (BACHELOR) RADA:

Korišćenje neuralnog prenosa stila uz optimizaciju očuvanja boja za generisanje umjetničkih radova

TEKST ZADATKA:

Zadatak ovog rada je generisanje slike G koja kombinuje sadržaj slike C sa stilom slike S. Rješenje ovog problema može se iskoristiti kao dio nekog složenijeg softvera ili bilo koje aplikacije sa primjenom filtera, šire gledano podsistema koji ima veze sa *photo editing*-om. Za rješavanje ovog problema koristi se konvolutivna neuronska mreža (model: VGG-19) kao polazna tačka.

Rukovodilac studijskog programa:	Mentor rada:

Primerak za: ☐ - Studenta; ☐ - Mentora

SADRŽAJ

1.	UVOD	7
2.	PREGLED ORIGINALNOG RJEŠENJA	9
3.	TEORIJSKI POJMOVI I DEFINICIJE	13
3.1	Vještačke neuronske mreže.....	13
3.2	Konvolutivne neuronske mreže.....	15
3.2.1	Konvolutivni sloj.....	16
3.2.2	<i>Pooling</i> sloj	19
3.2.3	Normalizacioni sloj	21
3.2.4	Neuroni i aktivacione funkcije	21
3.2.5	<i>VGG-19</i> arhitektura.....	22
3.2.6	Rekonstrukcija sadržaja i stila.....	25
4.	SPECIFIKACIJA I IMPLEMENTACIJA RJEŠENJA	27
4.1	Korišćeni alati	27
4.2	Arhitektura rješenja.....	28
4.2.1	Standardni neuralni prenos stila	28
4.2.2	Optimizovani neuralni prenos stila	31
5.	EVALUACIJA RJEŠENJA I REZULTATI.....	35
5.1	Uticaj različitih slojeva CNN-a.....	37
5.2	Inicijalizacija <i>gradient descent</i> -a	38
6.	ZAKLJUČAK	41
7.	LITERATURA.....	43
8.	BIOGRAFIJA	47
	KLJUČNA DOKUMENTACIJSKA INFORMACIJA	49
	KEY WORDS DOCUMENTATION	51

1. UVOD

Neuralni prenos stila (engl. *Neural Style Transfer*) [1] je tehnika dubokog učenja (engl. *Deep Learning*) [2] u kojoj ulaz sistema čine dvije slike. Jednu sliku nazvaćemo sadržajnom (engl. *content*) slikom (označićemo je sa C), a drugu sliku stilskom (engl. *style*) slikom (označićemo je sa S). Cilj ovog rada jeste generisati sliku (označićemo je sa G), tako da G kombinuje sadržaj slike C sa stilom slike S . Na taj način možemo da, na primjer, generišemo naš *selfie*¹ u stilu umjetničkog rada nekog poznatog umjetnika, ali da pri tome ne koristimo tehnike za eksplicitnu transformaciju piksela (kao što je, na primjer, *non-photorealistic rendering*²) nego koristeći konvolutivne neuronske mreže (engl. *Convolutional Neural Networks*, CNN) [3] „učimo“ piksele za G sliku.

Rješenje ovog problema može se iskoristiti kao dio nekog složenijeg softvera (kao što je, na primjer, *FotoSketcher*³) ili bilo koje aplikacije sa primjenom filtera. Šire gledano, rješenje se može iskoristiti kao dio aplikacije ili podsistema koji je povezan sa *photo editing*-om (manipulacija i izmjena slika).

Naučni rad u kome je predložen algoritam neuralnog prenosa stila [1] koristi prethodno treniranu CNN i pomoću nje rješava dati problem. Koristeći *Transfer Learning* [4], iskorišćena je *VGG-19* mreža [5] kao polazna tačka. Kompletan pristup je, suštinski, zasnovan na minimizaciji funkcije greške: $J(G) = \alpha * J_{content}(C, G) + \beta * J_{style}(S, G)$, gdje se kroz $J_{content}$ teži ka tome da G slika što manje odstupa od sadržaja C (fokus na strukturu), a kroz J_{style} da G što manje odstupa od stila S (fokus na teksturu, luminozitet i slično).

Problem kod standardnog pristupa neuralnog prenosa stila [1] jeste činjenica da generisana slika G nema očuvane boje od C slike. Dakle, kada imamo kao ulaz C i S slike, kada se izgeneriše G slika, na njoj će biti očuvan sadržaj C slike, ali će biti preuzete dominantne boje iz S slike, što u nekim situacijama nije poželjno. Korišćenjem *Color Preservation* optimizacije [6] se može popraviti ovaj problem, tako da na kraju G očuvava sadržaj C slike, kao i boje prisutne na njoj, dok stil slike (tekstura, luminozitet...) bude isti kao stil S slike.

¹ Autoportret-fotografija na kojoj je osoba fotografisala sebe ili sebe s drugim osobama

² https://en.wikipedia.org/wiki/Non-photorealistic_rendering

³ <https://fotosketcher.com/>

Color Preservation optimizacija se može izvršiti koristeći jedan od dva pristupa:

- *Color Histogram Matching*
- *Luminance – only Transfer*.

Oba pristupa očuvanja boja su implementirana u ovom radu.

Evaluacija je čest problem kod soft kompjuering i fazi sistema (pogotovo u *Visual Computing*⁴-u), odnosno teško je definisati adekvatnu metriku za procjenu njihovog kvaliteta. U ovom radu je zato izvršeno poređenje rezultata sa ostalim naučnim radovima koji se tiču ovog problema i sličnih optimizacija (uključujući radove autora originalnog rada).

Glavna motivacija za rješavanje problema neuralnog prenosa stila ogleda se u tome da se, koristeći duboke neuronske mreže, proces generisanja umjetničkih radova izvede koristeći sisteme zasnovane na vještačkoj inteligenciji. Sveobuhvatno gledano, cilj vještačke inteligencije jeste oponašanje ljudskog ponašanja, pa i simuliranje kompleksnih radnji, čime je u neku ruku postala jedna od fundamentalnih naučnih disciplina. Ovakvi sistemi nisu u stanju da sami proizvedu unikatno umjetničko djelo kao što to čini čovjek, ali koristeći postojeće resurse, pokazuju impresivne rezultate u kreiranju umjetničkih radova.

U narednom poglavlju dat je pregled originalnog rješenja ovog problema gdje je ujedno objašnjen i koncept kompletnog rješenja kroz matematičke formalizme zasnovane na minimizaciji funkcije greške koja uzima u obzir sadržaj jedne i stil druge slike. Treće poglavlje uvodi teorijske pojmove potrebne za razumijevanje detalja i implementacije predloženog rešenja (uopšteno o CNN [3] i opis korišćene *VGG-19* [5] arhitekture, te njena namjena za zadatak ovog rada). Specifikacija i implementacija rješenja u programskom okruženju *PyTorch* [7] opisana je u četvrtom poglavlju. Potom slijedi poglavlje o evaluaciji rješenja i dobijenim rezultatima, te na kraju poglavlje zaključka koje predstavlja sumarizaciju cijelog rada i predlaže pravce daljeg istraživanja.

⁴ Generički naziv za sve discipline u računarstvu čiji predmet izučavanja su slike i 3D modeli (oblasti: računarska grafika, procesiranje slika, vizualizacija, računarska vizija, virtualna i augmentovana realnost, video procesiranje)

2. PREGLED ORIGINALNOG RJEŠENJA

Rješenje predstavljeno u radu koji se bavi neuralnim prenosom stila [1] koristi VGG-19 [5] neuronsku mrežu koja je pokazala odlične rezultate u dubokim vizuelnim reprezentacijama [8] u računarskoj viziji. Prostor obilježja (engl. *feature space* [9]) ove mreže čini 16 konvolutivnih i 5 *pooling* slojeva. Bitno je napomenuti da se u neuralnom prenosu stila ne koriste potpuno povezani (engl. *fully connected*) slojevi, zato što treniramo piksele generisane slike kao parametre, a nije nam cilj dobiti neki konkretan krajnji izlaz iz same mreže.

U radu [1], pokazano je da *average pooling* pristup [3] funkcioniše bolje od *max pooling* [3] pristupa. Kao pokazatelj ovog zaključka uzima se činjenica da se dobija bolji protok gradijenata (engl. *gradient flow*), odnosno, iz iteracije u iteraciju mreža sve suptilnije uči promjene (u parametrima) i ne dolazi do zamrzavanja mreže. Ovo znači da na višem nivou apstrakcije, uopšteno rečeno, nemamo problem sa konvergencijom jer mreža adekvatno ažurira parametre i dolazi do nekog (sigurno zadovoljavajućeg) rješenja.

Pristup neuralnog prenosa stila se zasniva na minimizaciji funkcije greške koja se dobija sabiranjem funkcije greške za sadržajnu i stilsku sliku, pri čemu se preko parametara α i β konfiguriše da li veći značaj (tj. težinu) dajemo sadržaju ili stilu slike. Recimo, u slučaju da parametar β ima znatno veći udio nego parametar α , tada bismo dobili sliku koja bi bila mnogo nejasnijeg izgleda i fokus bi bio na stilu, dok bi se malo pazilo na sadržaj. Generisana slika bi u ovom slučaju ličila na neki impresionistički umjetnički rad. Fino podešavanje odnosa parametara α i β može se pronaći u radu koji se bavi kontrolom faktora percepcije [10] (odnosa sadržaja i stila) slike u ovom algoritmu.

Prvo ćemo opisati kako se u originalnom radu definiše funkcija greške (engl. *loss function* [11]) koja će predstavljati sadržaj generisane slike. Ovdje je sloj (engl. *layer*) mreže sastavljen od N_l različitih filtera i ima samim tim N_l različitih *feature* mapa od kojih je svaka veličine M_l . M_l predstavlja proizvod veličine i širine odgovarajuće *feature* mape. Na osnovu toga, odgovore u sloju l možemo čuvati u matrici $F^l \in \mathbb{R}^{N_l \times M_l}$ gdje $F_{i,j}^l$ predstavlja aktivaciju i -tog filtera na poziciji j u sloju l . Neka su dati vektori \vec{p} i \vec{x} (originalna i generisana slika), i neka su P^l i F^l njihove *feature* reprezentacije u sloju l . Tada funkciju greške za sadržajni dio generisane slike zapisujemo na sledeći način:

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

Funkcija greške za sadržaj nam ukazuje na to koliko data slika odstupa od ciljane slike u pogledu sadržaja u nekom sloju l .

Izvod funkcije greške po aktivacijama u sloju l je:

$$\frac{\partial L_{content}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij} & \text{ako } F_{ij}^l > 0 \\ 0 & \text{ako } F_{ij}^l < 0 \end{cases}$$

na osnovu koga se može izračunati gradijent koristeći propagaciju unazad (engl. *backward propagation* [12]). Na osnovu ovoga mijenjamo inicijalnu nasumičnu sliku (najčešće inicijalizovanu kao *random noise* [13]) predstavljenu vektorom \vec{x} , sve dok ona ne generiše isti odgovor u nekom sloju l naše CNN kao i originalna slika \vec{p} .

Reprezentacija stila računa korelacije između različitih odgovora filtera. Ove korelacije u osobinama se računaju koristeći *Gram* matricu [14] $G^l \in \mathbb{R}^{N_l \times N_l}$, gdje je $G_{i,j}^l$ skalarni proizvod između vektorizovanih *feature* mapa i i j u sloju l . Dakle, za dati sloj l dobijamo:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

gdje iteriramo po k koji predstavlja konkretnu poziciju unutar *feature* mape, odnosno, brojčanu vrijednost.

Neka su dati vektori \vec{a} i \vec{x} tako da budu redom originalna i generisana slika, sa stilovima reprezentacije A^l i G^l u nekom sloju l . Uticaj konkretnog sloja na funkciju greške iznosi:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

Konačno, stilska greška se računa na sledeći način:

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l$$

gdje su w_l težinski faktori koji nam govore koliko koji sloj utiče na ukupnu grešku.

Izvod E_l po aktivacijama iz nižih slojeva neuronske mreže se može izračunati na sledeći način:

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} ((F^l)^T (G^l - A^l))_{ji} & \text{ako } F_{ij}^l > 0 \\ 0 & \text{ako } F_{ij}^l < 0 \end{cases}$$

Konačno, ako su dati vektori \vec{p} , \vec{a} i \vec{x} koji redom predstavljaju sadržajnu, stilsku i generisanu sliku, konačna funkcija greške glasi:

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x})$$

gdje se trudimo da sadržaj i stil generisane slike budu što sličniji sadržajnoj i stilskoj slici.

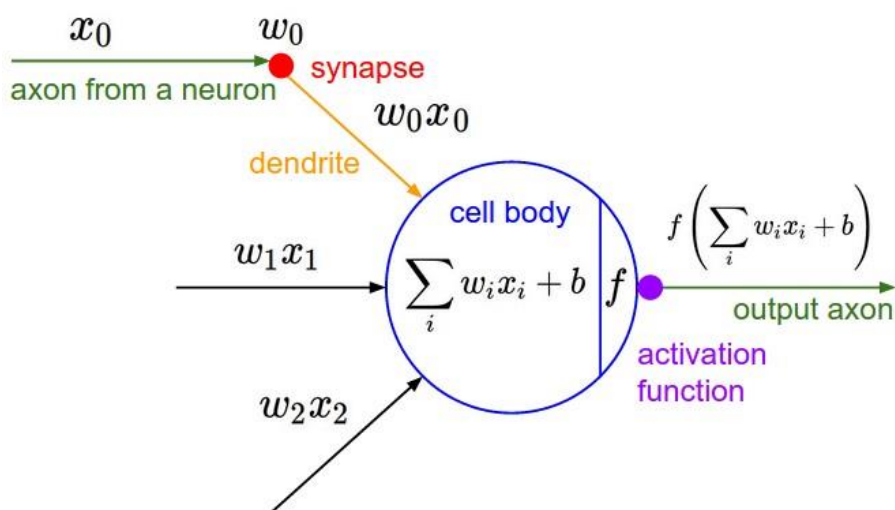
3. TEORIJSKI POJMOVI I DEFINICIJE

Za izvođenje ovog zadatka koriste se konvolutivne neuronske mreže (CNN) [3], a pristupi korišćeni za optimizaciju problema će biti detaljno opisani u narednom poglavlju. U ovom poglavlju cilj je objasniti standardne vještačke neuronske mreže (poglavlje 3.1) koje predstavljaju osnovu za razumijevanje CNN-ova, koje su predstavljene u poglavlju 3.2.

3.1 Vještačke neuronske mreže

Vještačke neuronske mreže (engl. *Artificial Neural Networks* - ANN) su računarski sistemi koji su dobrim dijelom inspirisani biološkim neuronskim mrežama od kojih su sačinjeni životinjski mozgovi [15]. Takvi sistemi „uče“ tj. progresivno popravljaju performanse razmatrajući primjere, uopšteno gledano bez specifičnog programiranja za datu primjenu.

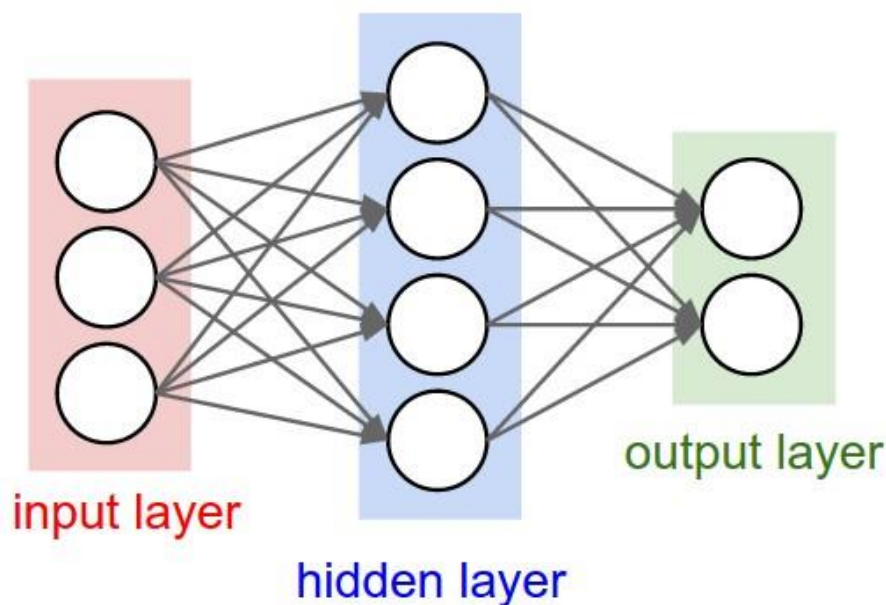
ANN se sastoji iz niza neurona i veza između njih koje imaju određene težine (engl. *weights*). Zadatak jednog neurona je da na osnovu vrijednosti njegovih ulaza izračuna vrijednost izlaza, kao na slici 3.1, i prosljedi je dalje kroz mrežu.



Slika 3.1 Prikaz prvog vještačkog neurona, tzv. McCulloch-Pitts-ov neuron [16]. x_0 , x_1 i x_2 predstavljaju ulaze neurona, a w_0 , w_1 i w_2 težine dodijeljene odgovarajućim neuronima. Funkcija f je aktivaciona funkcija čiji zadatak je uvođenje nelinearnosti.

Pored navedenog, svaki vještački neuron najčešće ima jedan dodatni ulaz koji se naziva *bias*. Ovo se obično postiže time što svaki sloj ima jedan dodatan neuron koji, poput neurona iz ulaznog sloja, nema ulazne veze, nego za ulaze koristi predefinisane vrednosti (slika 3.1, oznaka *b*). Nije važno koja se vrijednost odabere za *bias*, već su važne težine njegovih izlaznih veza. Težine *bias* veza, zajedno sa težinama ostalih veza, predstavljaju parametre modela koji se obučavaju i utiču na krajnji rezultat.

Arhitektura neuronske mreže predstavlja specifično povezivanje neurona u jednu cjelinu. Stuktura same neuronske mreže se razlikuje po broju slojeva. Prvi sloj se naziva ulazni (engl. *input layer*), a posljednji izlazni (engl. *output layer*), dok se slojevi između nazivaju skriveni slojevi (engl. *hidden layer*) [17]. Prvi sloj, tj. ulazni je jedini sloj koji prima podatke iz spoljašnje sredine. Sledeći (skriveni) slojevi prosljeđuju podatke sve do izlaznog sloja i u njima se vrši memorijski i vremenski najzahtjevnije računanje. Na izlazu izlaznog sloja dobijamo konačan rezultat. Slojevi su međusobno potpuno povezani. Grafički prikaz prethodno opisanog modela vještačke neuronske mreže dat je na slici 3.2.



Slika 3.2 Primjer arhitekture dvoslojne vještačke neuronske mreže - ulazni sloj se ne broji. Slojevi su međusobno potpuno povezani (engl. *fully connected*).

Model ANN-a „učí“, odnosno, trenira se koristeći *backpropagation* algoritam [12]. *Backpropagation* algoritam, odnosno, algoritam propagacije

unazad podrazumijeva računanje gradijenata u računarskom grafu mreže idući od izlaza ka ulazu mreže i ažuriranje parametara modela na osnovu njih. Slično kao kod *gradient descent* [18] algoritma, upotreba gradijenata dovodi do toga da se parametri modela ažuriraju proporcionalno njihovom uticaju na grešku rezultata i time, nakon određenog broja iteracija, dobijamo model koji je naučio određene paterne koje prije nije znao i samim tim daje bolje predikcije, odnosno, izlaze za datu primjenu.

3.2 Konvolutivne neuronske mreže

Standardne duboke ANN uspješno mogu da izvrše klasifikaciju složenih reprezentacija znanja kao što su na primjer, slike i zvuk. Ovo je moguće zbog postojanja aktivacionih funkcija [19] koje omogućuju rješavanje nelinearno separabilnih problema. Za razliku od njih, perceptroni [20] mogu da rješe samo probleme kod kojih su reprezentovani podaci linearno separabilni. Međutim, čak i kod slika malih dimenzija potreban nam je veliki broj parametara. Kako bismo potkrijepili ovu tvrdnju uzmimo na primjer sliku u boji čije su dimenzije 32×32 piksela. Broj parametara potrebnih za ovu sliku je $32 \times 32 \times 3 = 3072$ za samo jedan neuron u prvom skrivenom sloju (32 za širinu, 32 za visinu, 3 za kanale RGB – *Red Green Blue* modela boja [21] u kojem je svaki piksel predstavljen sa tri vrijednosti). Posljedice ove činjenice su da ovakav model ima lošije performanse prilikom treniranja i prilikom izvršavanja. Takođe, povećan je rizik od preprilagođavanja [22] (engl. *overfitting* – pretjerano prilagođavanje modela trening podacima).

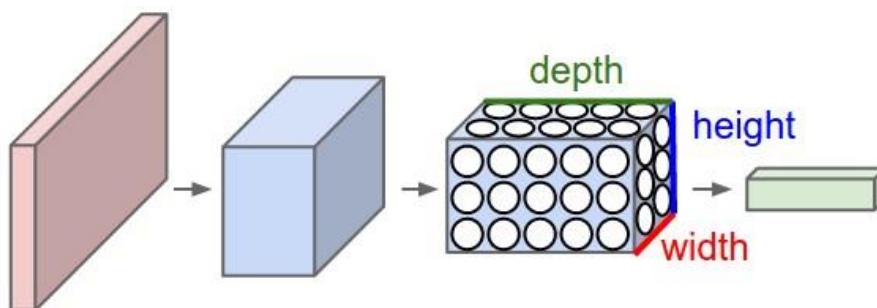
Upravo zbog ovog razloga, za rad sa slikama razvijen je specifičan model koji je zasnovan na ANN-ovima. Taj model nazvan je konvolutivnom neuronskom mrežom (engl. *Convolutional Neural Networks*, CNN) [3] i predstavljao je veliki pomak u generalnom izučavanju neuronskih mreža zato što je uspio da iskoristi neke ideje vezane za vizuelnu percepciju i spoji ih sa ANN-om. Ovaj model razvio je Žan LeKun (engl. *Yann LeCun*), računarski naučnik koji je jedan od tri dobitnika prestižne *Tjuringove* nagrade⁵ za oblast dubokog učenja [2] (engl. *Deep Learning*) koja se smatra *Nobelovom*⁶ nagradom za računarske nauke.

Kao i ANN, CNN se sastoji iz međusobno povezanih slojeva neurona. Glavna razlika je u tome što su neuroni u slojevima organizovani u

⁵ Nagrada koju jednom godišnje dodjeljuje američko *Udruženje za računarske nauke* za doprinose tehničke prirode na polju računarstva koji su trajnog karaktera i od velikog naučno-tehnološkog značaja

⁶ Internacionalna nagrada, koju godišnje dodjeljuju švedske i norveške institucije, kao znak priznanja za akademska, kulturna i naučna dostignuća

tri dimenzije – kvadar (kao, na primjer, u slučaju slika – širina, visina i dubina), što je prikazano na slici 3.3.



Slika 3.3 Prikaz neurona konvolutivne neuronske mreže organizovanih u trodimenzionalne slojeve.

Svaki sloj, primjenom određene diferencijabilne funkcije, na određeni način transformiše ulazni kvadar u izlazni kvadar različitih dimenzija. Pored potpuno povezanih slojeva uvedenih u ANN (koji se koriste na kraju mreže za dobijanje konačnih izlaza), unutar CNN-a, postoje i druge vrste slojeva. Osnovni specifični slojevi jednog CNN-a su: **konvolutivni** sloj (poglavlje 3.2.1), **pooling** sloj (poglavlje 3.2.2) i **normalizacioni** sloj (poglavlje 3.2.3). Izbor aktivacione funkcije neurona u CNN-u dat je u poglavlju 3.2.4. Poglavlja 3.2.5 i 3.2.6 opisuju specifičnu arhitekturu CNN-a koja se koristi kao osnova za rješavanje glavnog zadatka ovog diplomskog rada (VGG-19 mreža) i njenu primjenu u slučaju neuralnog prenosa stila [1].

3.2.1 Konvolutivni sloj

Konvolutivni sloj predstavlja osnovni i najspecifičniji sloj CNN-ova po kojem su one i dobile ime. Osnovna uloga ovog sloja jeste da transformiše odgovarajući ulaz čije su dimenzije $W_1 \times H_1 \times D_1$ u izlaz čije su dimenzije $W_2 \times H_2 \times D_2$ primjenom operacije konvolucije [23] većeg broja filtera (ili kernela) čije dimenzije su manje od ulaza (po širini i visini). Generalno gledano, u matematici, konvolucija je operacija koja se izvršava nad dvije funkcije (na primjer f i g) i formira treću funkciju $f * g$ koja opisuje kako se oblik jedne funkcije modifikuje drugom funkcijom. U našem slučaju, konvolucija između dvije matrice (označićemo ih sa M i N) čije su dimenzije jednake i iznose $F \times F$ se računa po formuli:

$$\sum_{i=1}^F \sum_{j=1}^F M_{ij} * N_{ij}.$$

Ako posmatramo u kontekstu kvadrova, za ulaz čije su dimenzije $W_1 \times H_1 \times D_1$, potreban je filter (ili kernel) čije su dimenzije $F \times F \times D_1$, tj. filter mora biti iste dubine kao i kvadar. Takođe, dimenzije filtera moraju biti takve da se mogu smijestiti unutar slike tj. $F < W_1$, $F < H_1$. Svakoј „slici“ ulaza (čije su dimenzije $W_1 \times H_1$, ukupno D_1 slika) odgovara jedna matrica filtera (čije su dimenzije $F \times F$, ukupno D_1 matrica). Operacija konvolucije se izvršava između odgovarajuće matrice filtera i svakog $F \times F$ dijela slike, „klizanjem“ od gore ka dole, sa lijeva na desno (engl. *sliding window*). Sve dobijene vrijednosti na istoj horizontalnoj i vertikalnoj poziciji (a po cijeloј dubini) se potom sumiraju u jednu vrijednost, poput spljoštavanja kvadra (po dubini) na pravougaonik. Rezultat ovoga je jedna slika čije su dimenzije $W_2 \times H_2$. Dubinu D_2 izlaznog kvadra dobijamo tako što primijenimo K različitih filtera, pri čemu važi uslov da je $D_2 = K$, odnosno koliko filtera primijenimo tolika će biti i dubina izlaznog kvadra.

U CNN-u svaki neuron iz izlaznog kvadra (ukupno $W_2 \times H_2 \times D_2$ neurona) nije, kao u slučaju potpuno povezanih slojeva, povezan sa svim neuronima iz prethodnog sloja, već samo sa određenom regijom neurona, tačnije sa $F \times F \times D_1$ neurona. Na taj način, jedan neuron iz izlaznog kvadra predstavlja „sumarizaciju“ te regije neurona iz ulaznog kvadra – ono što je filter u toј regiji naučio da „prepozna“. Ova osobina CNN-ova naziva se lokalna povezanost neurona (engl. *local connectivity*). Sa samim pojmom nastanka CNN-ova povezan je pojam receptivnog polja (engl. *receptive field*). Ideja da oponašaju način na koji funkcioniše mozak (u ovom slučaju vid) životinje je osnova za dizajn CNN-a [24]. Mozak mačke sastavljen je od kompleksno raspoređenih ćelija [25]. Ove ćelije su osjetljive na male regije vidnog polja, koje zovemo receptivna polja ili lokalni receptori. Oni su poredani tako da prekrivaju vidno polje [25]. Umjesto da svaki neuron u nekom sloju bude povezan sa svim neuronima u susjednom sloju, kao što je to slučaj sa ANN-ovima, ovdje se neuroni raspoređuju u trodimenzionalnu strukturu, pri čemu je svaki neuron povezan sa samo određenim regionom (receptivnim poljem).

Takođe, CNN-ovi imaju i osobinu tzv. dijeljenja parametara (engl. *parameter sharing*) – svi neuroni ulaznog kvadra koji se nalaze na istoj slici ($W_1 \times H_1$) koriste istu matricu. Odnosno, ista matrica filtera koristi se za operaciju konvolucije na svim pozicijama jedne slike, umjesto posebna matrica (posebni parametri za treniranje) za svaku od pozicija. Ideja iza ove osobine je sledeća: ukoliko je filter nešto prepoznao u određenom dijelu

slike (npr. ivicu ili dio određene boje), vjerovatno je korisno takve stvari tražiti i u ostalim dijelovima slike. Ova osobina čini model invarijantnim na pozicije i rotacije objekata (ili dijelova objekata) na slici.

Objek osobine utiču na smanjenje broja parametara modela, a istovremeno se zadržavaju najvažnije karakteristike iz prethodnog sloja (koje je filter naučio), dok se one manje važne odbacuju i time pokazuju zašto su za rad sa slikama bolji model nego ANN.

Korišćenje većeg broja filtera je opravdano činjenicom da svaki od filtera ima ulogu da prepozna određeno svojstvo na slici. Broj filtera tj. dubina konvolutivnog sloja je predstavljena sa hiperparametrom K .

Prilikom korišćenja *sliding window* algoritma, korak (engl. *stride*) najčešće ima predefinisanu vrijednost 1, ali on može biti i veći, i takođe predstavlja hiperparametar (S).

Hiperparametar (P) modela koji u CNN model uvodi konvolutivni sloj koristi se za proširivanje ivica slika ulaza nulama (engl. *zero-padding*) [3]. Ovaj parametar utiče na to koliko konvolucija se može izvršiti u jednoj slici, tj. koliko konvolucija „staje“ u jednu sliku. Prilikom modelovanja mreže ovo pruža veću kontrolu nad dimenzionalnošću izlaza (W_2 i H_2). Na primjer, moguće je očuvati da širina i visina izlaza budu jednake širini i visini ulaza.

Hiperparametri su međusobno zavisni, tako da vrijednost jednog hiperparametra može direktno uticati na drugi hiperparametar ili hiperparametre tj. nisu sve kombinacije F , K , S i P validne. Širina, visina i dubina izlaznog kvadra ($W_2 \times H_2 \times D_2$) mogu se izračunati primjenom narednih formula:

$$\begin{aligned} W_2 &= \frac{W_1 - F + 2 * P}{S} + 1 \\ H_2 &= \frac{H_1 - F + 2 * P}{S} + 1 \\ D_2 &= K, \end{aligned}$$

pri čemu je obavezno da sve dobijene vrijednosti budu cijeli brojevi. U suprotnom, treniranje modela nije ostvarivo.

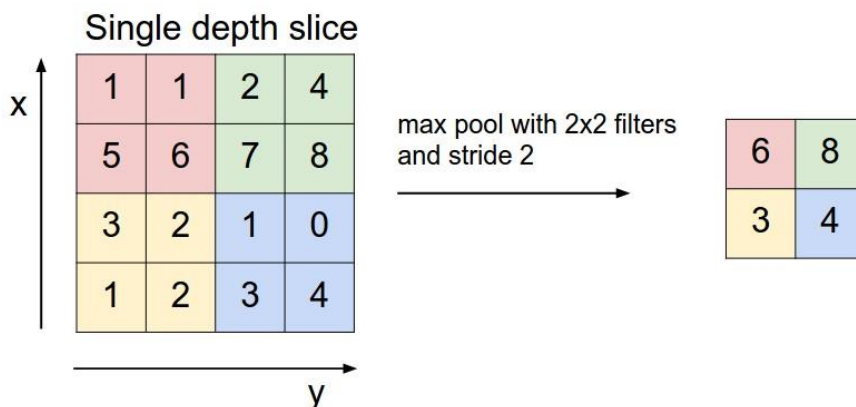
Prilikom treniranja CNN-a, vrijednosti matrica filtera predstavljaju parametre koji se obučavaju. Na slici 3.4 dat je grafički prikaz filtera [26] nakon treniranja modela. Tu se uočava 96 konvolutivnih kernela čije su dimenzije $11 \times 11 \times 3$ koji su naučeni od strane prvog konvolutivnog sloja nad slikama čije su dimenzije $224 \times 224 \times 3$. Gornjih 48 kernela je naučeno od strane prve grafičke procesorske jedinice (engl. *Graphics processing unit* – GPU), a donjih 48 od strane druge GPU.



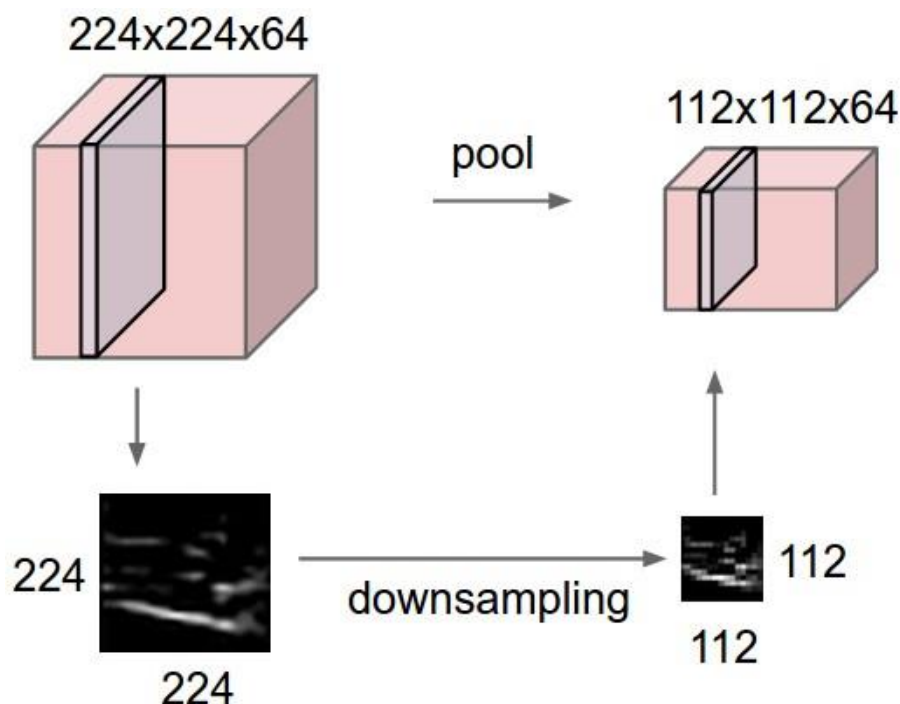
Slika 3.4 Grafički prikaz matrica filtera konvolutivnih slojeva nakon treniranja modela [26].

3.2.2 Pooling sloj

Između konvolutivnih slojeva može se nalaziti *pooling* sloj. Njegova uloga jeste da ulaz čije su dimenzije $W_1 \times H_1 \times D_1$ transformiše u izlaz čije su dimenzije $W_2 \times H_2 \times D_2$, pri čemu su širina i visina izlaza manje ($W_2 < W_1$, $H_2 < H_1$), a dubina jednaka ($D_2 = D_1$). To se postiže klizanjem filtera čije su dimenzije $F \times F$ (hiperparametar) po svakoj slici ulaza (čije su dimenzije $W_1 \times H_1$, ukupno D_1 slika) sa korakom S (hiperparametar). Na svakoj poziciji filtera, od svih vrijednosti sa dijela slike dimenzija $F \times F$, primjenom odgovarajuće funkcije, bira se samo jedna vrijednost. Najčešće je u pitanju funkcija maksimuma, a koristi se i funkcija prosjeka. Na slici 3.5 dat je primjer primjene *max pooling* operacije nad jednom matricom, a na slici 3.6 rezultat *pooling* sloja nad cijelim ulaznim kvadrom.



Slika 3.5 Primjena *max pool* operacije nad kvadratnom matricom.



Slika 3.6 Grafički prikaz primjene *pool* operacije nad slojem CNN-a.

Pooling sloj ne uvodi dodatne parametre za treniranje modela. Širina, visina i dubina izlaza ($W_2 \times H_2 \times D_2$) na osnovu širine, visine i dubine ulaza $W_1 \times H_1 \times D_1$ *pooling* sloja računaju se na osnovu sledećih formula:

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1$$

$$D_2 = D_1,$$

gdje su F i S hiperparametri za dimenziju filtera i korak (engl. *stride*).

Pooling sloj se još naziva i sloj sažimanja, a njegovom primjenom kao što vidimo takođe smanjujemo broj parametara, a samim tim i broj izračunavanja unutar mreže. Smanjenje broja parametara koristeći sloj sažimanja nam pomaže kod izbjegavanja pretreniranosti (engl. *overfitting*) [22] mreže.

3.2.3 Normalizacioni sloj

Postoji više tipova normalizacionih slojeva. Jedni od najpoznatijih su *batch* normalizacioni slojevi [27]. *Batch* normalizacioni sloj podrazumijeva normalizaciju izlaza prethodnog sloja nakon svake iteracije (odnosno *batch*-a – svih uzoraka; ili *minibatch*-a – dijela uzoraka) na osnovu trenutnih vrijednosti, po formuli:

$$\begin{aligned} norm &= \frac{input - mean(input)}{\sqrt{epsilon + var(input)}} \\ output &= gamma * norm + beta, \end{aligned}$$

gdje je *input* izlaz prethodnog sloja, *mean* i *var* su funkcije srednje vrijednosti i varijanse, *epsilon* je konstanta koja se proslijeđuje, *gamma* i *beta* su parametri za treniranje, a *output* predstavlja vrijednost nakon normalizacije.

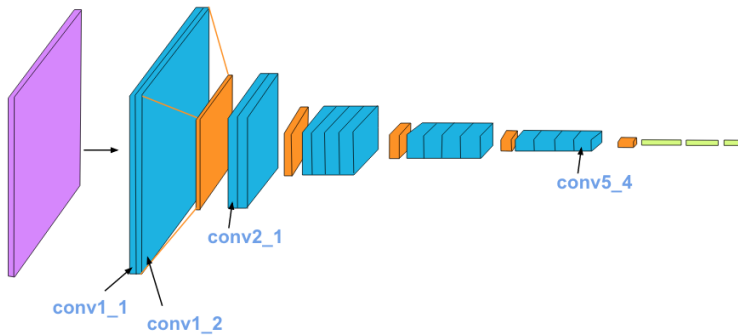
Glavna prednost primjene *batch* normalizacije se ogleda u činjenici da je moguće koristiti veći korak učenja (engl. *learning rate*) prilikom treniranja, jer nijedna vrednost neće biti previše mala ili previše velika (biće normalizovana). Kao posljedica ovoga, omogućeno je znatno brže treniranje modela. Takođe, *batch* normalizacija na neki način dodaje „šum“ na izlaz prethodnog sloja, što smanjuje preprilagođavanje mreže.

3.2.4 Neuroni i aktivacione funkcije

Neuroni u CNN-ovima su identični kao i neuroni u ANN-ovima. Jedna od najkorišćenijih aktivacionih funkcija trenutno je ReLU (engl. *Relu* – *Rectified Linear Unit*) [28], odnosno funkcija oblika $f(x) = \max(0, x)$. Njena prednost u odnosu na neke ranije široko korišćene funkcije (kao što su na primjer *sigmoid* [29], hiperbolički tangens itd.) su performanse (*max* funkcija je brža od množenja, dijeljenja, stepenovanja i sl.), što značajno ubrzava treniranje modela. Takođe, s obzirom na to da kodomen ReLU funkcije nije ograničen (sa gornje strane), ona ne pati od *vanishing gradient* problema [30] prilikom kojeg gradijenti tokom treniranja modela postanu suviše mali za uspješno napredovanje obučavajućeg algoritma. Neki autori eksplicitno u svojim arhitekturama navode ReLU kao poseban sloj CNN-a, iako se to zapravo samo odnosi na korišćenu aktivacionu funkciju.

3.2.5 VGG-19 arhitektura

VGG-19 je varijanta VGG modela koji je razvijen na Univerzitetu u Oxford-u⁷. Sastavljena je od 19 slojeva: 16 konvolutivnih, 3 potpuno povezana sloja, 5 *pooling* slojeva – koristeći *max pooling* pristup



Slika 3.7 Prikaz arhitekture VGG-19 neuronske mreže (*feature space* sa 16 konvolutivnih i 5 *pooling* slojeva) [5].

i jednog *softmax* sloja. U broju slojeva, računamo konvolutivne i potpuno povezane slojeve, dok *pooling* obično gledamo kao dodatne međuslojeve koji popravljaju postojeće performanse spriječavajući pretreniranost mreže. *Softmax* sloj se zapravo odnosi na korišćenje same funkcije za određivanje izlaza. Postoje i druge VGG arhitekture kao što su VGG-11 [31], VGG-16 [32] i ostale modifikacije, a VGG-19 [5] ima najbolje performanse od svih.

Ova arhitektura se može smatrati nasljednikom poznate AlexNet [26] mreže, a dobila je naziv po tome što je kreirana u grupi za vizuelnu geometriju (engl. *Visual Geometry Group* - VGG). Neke od ideja na kojima je zasnovana ova arhitektura su ostale iste kao i kod njenih mreža prethodnica, sa time da ona mnoge od njih unaprijeđuje [5] i koristi duboke konvolutivne slojeve, čime pokazuje još veću preciznost u eksperimentima sprovedenim nad raznim poznatim skupovima podataka.

2014. godine ova arhitektura je pobijedila na poznatom ImageNet⁸ takmičenju, gdje je zasjenila tadašnje *state-of-the-art*⁹ modele [5].

⁷ <https://www.ox.ac.uk/>

⁸ <http://www.image-net.org/>

⁹ Najveći stepen u razvoju, bilo uređaja, tehnike ili nekog naučnog polja, koji je postignut u određeno vrijeme, vrhunska tehnologija / pristup

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Slika 3.8 Kompletne konfiguracije VGG neuronskih mreža, izostavljeni su ReLu slojevi jer se striktno odnose na računanje funkcija [5].

Na slici 3.8 kolona *E* odnosi se na arhitekturu *VGG-19* mreže, dok ostale kolone predstavljaju druge varijante VGG modela.

Na ulaz *VGG-19* mreže prosljeđuje se RGB [21] slika čije su dimenzije 224×224 tako da je ulazna matrica bila dimenzija (224,224,3). Jedini vid pretprocesiranja jeste oduzimanje srednjih RGB vrijednosti od svakog piksela, što se računa prolaskom kroz cijeli trening skup podataka. Korišćenje kernela čije su dimenzije 3×3 sa korakom 1 je omogućilo ekstrakciju kompletnih informacija o slici (engl. *notion of the image* [5]). Korišćen je *spatial padding*, a *max pooling* je rađen koristeći *sliding window*-e čije su dimenzije 2×2 sa korakom 2. Sve ovo praćeno je ReLu funkcijama [28] čiji je cilj da uvedu nelinearnost u model i

omogućće bolju klasifikaciju, te unaprijede vrijeme izvršavanja problema zato što su prethodni modeli najviše koristili hiperbolički tangens ili sigmoidne [29] funkcije, koje su se pokazale sporijim.

Što se tiče potpuno povezanih slojeva, ova mreža je sastavljena od tri takva sloja pri čemu je veličina prva dva 4096 i nakon toga dolazimo do sloja sa 1000 kanala jer je cilj takmičenja bio klasifikacija u 1000 klasa, a izlazni sloj mreže je *softmax* funkcija.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	23.7	6.8	6.8
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	-	7.9
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	-	6.7
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

Slika 3.9 Poređenje rezultata između tadašnjih *state-of-the-art* modela iz 2014. godine, prezentovano na ILSVRC¹⁰. Slika, kao i prethodne u ovom poglavlju je preuzeta iz [5].

Kao što se može primijetiti na slici 3.9, duboki CNN-ovi su postigli bolje rezultate implementacijom više slojeva i znatno redukovali vrijeme potrebno za rješavanje odgovarajućeg problema uvođenjem konvolutivnih filtera čije su dimenzije 3×3 jer su redukovali broj parametara.

Sami autori navode da je glavna svrha projekta implementacije *VGG-19* mreže bila pobjeda na *ImageNet* takmičenju [5] i samim tim unaprijeđivanje postojećih *state-of-the-art* rješenja, a ova mreža se pokazala dobrom za rješavanje raznih problema. Neke od njenih značajnih primjena su [5]:

- Koristi se kao veoma moćna arhitektura za klasifikaciju nad različitim tipovima skupova podataka
- Može se iskoristiti kao osnova za mnoge zadatke u oblasti prepoznavanja lica – *transfer learning* [4]
- Izračunavanje sadržajne i stilske greške (što je posebno značajno bilo u zadatku kojim se bavi ovaj diplomski rad)

¹⁰ <http://www.image-net.org/challenges/LSVRC/>

3.2.6 Rekonstrukcija sadržaja i stila

Za rješavanje glavnog zadatka u ovom radu koristi se CNN [3], koji procesira ulaznu sliku, predstavljenu kao trodimenzionalni niz (*širina slike, visina slike, broj kanala* = 3) i pretvara je u specifičnu reprezentaciju koja je pogodna za dalju obradu. Slika se propagira kroz slojeve mreže i idući dublje po slojevima, u stanju smo da naučimo sve složenije funkcije, koje nam omogućuju da „učimo“ složenije osobine (kao što su, na primjer, konture/elementi na slici). Možemo direktno vizualizovati informacije koje je naša mreža naučila u tekucem sloju tako što rekonstruišemo sliku do tog nivoa, koristeći *feature mapping* [33].

Početni slojevi mreže na početku treniranja za dati piksel reprodukuju zapravo isti takav piksel, a kasnije počinju da zamućuju pojedinačne piksele, da bi na kraju taj proces bio proširen na kompletne regione. Za reprezentaciju sadržajne slike koristimo više slojeve, jer su oni već stekli uvid u to kako su neki karakteristični regioni raspoređeni na slici (na primjer na nekoj slici se nalaze dvije osobe i neka pozadina). Za sliku stila koristimo prostor osobina (engl. *feature space*) koji će da ekstrahuje informacije vezane za teksturu slike. Prolazeći kroz mrežu, postepeno „učimo“ i sadržajnu i stilsku sliku.

Sadržajna rekonstrukcija, što idemo dalje i dalje kroz mrežu, sve više mijenja informaciju o pikselima, a sadržaj slike na visokom (apstraktnijem) nivou se čuva. Stilaska rekonstrukcija koristeći prostor osobina računa korelacije između različitih osobina u različitim slojevima CNN-a. To stvara slike koje se poklapaju sa datom ulaznom slikom po stilu, isključujući informaciju o rasporedu elemenata na slici. Suština ove ideje jeste da su reprezentacije sadržaja i stila u CNN-u separabilne. Izlazna slika jeste takva slika koja simultano uzima u obzir i sadržajnu reprezentaciju jedne i stilsku reprezentaciju druge slike.

Stil je moguće definisati i do određenog sloja u mreži, čime dobijamo drugačiji vizuelni „osjećaj“ koji je u manjem broju slučajeva bolji za neke stilove. Kada vršimo sinteziranje slike koja kombinuje sadržaj jedne, te stil druge slike, često nećemo dobiti izlaznu sliku koja na savršen način ispunjava istovremeno oba ograničenja. Koristeći funkciju greške, minimizujemo dvije funkcije koje se tiču sadržaja i stila sa ciljem da odlučimo šta želimo više naglasiti (da li sadržaj ili stil). Ove dvije funkcije su jasno razgraničene.

Non-photorealistic rendering je bila motivišuća tehnika iz računarske vizije koja je korišćena za kreiranje umjetničkih radova od postojećih slika. Međutim, problem sa ovom tehnikom jeste činjenica da je riječ o neparametarskoj tehnici gdje smo morali direktno da manipuliramo pikselima neke slike. Neuralni prenos stila [1], koristeći duboke CNN, uči

funkcije sadržaja i stila kako bi se pronašao način kako da se rekonstruiše slika, bez da ekspert eksplicitno mora da definiše kako će se pojedini pikseli mijenjati.



Slika 3.10 Prikaz konvolutivnih slojeva po redovima, a po kolonama odnosa između parametara α i β . Slika preuzeta iz [1].

Kao što se vidi na slici 3.10, reprezentacija stila dobija više na značaju u dubljim slojevima što se tiče veličine i kompleksnosti kada uzmemo u obzir složenosti stilova (kao elemenata u prostoru stanja) u višim slojevima mreže. Ovo se objašnjava činjenicom da dolazi do povećanja veličina receptivnih polja i kompleksnosti osobina što idemo dublje kroz mrežu. Što je odnos parametara α i β veći, to je sadržaj slike više naglašen.

4. SPECIFIKACIJA I IMPLEMENTACIJA RJEŠENJA

U ovom poglavlju govorićemo o korišćenim alatima (programima, bibliotekama i okruženjima koji su bili neophodni za rješavanje zadatka i pokretanje i korišćenje samog programa) – poglavlje 4.1 i o samoj arhitekturi rješenja (osnovni i optimizovani pristup) – poglavlje 4.2.

4.1 Korišćeni alati

Zadatak je urađen u programskom jeziku *Python*. Programi i biblioteke koje su neophodne da bi projekat mogao da se pokrene su:

- *Pillow*¹¹ – poznata *Python* biblioteka za rad sa slikama
- *Numpy*¹² – fundamentalna biblioteka koja se koristi za naučno-tehničke proračune u *Python*-u
- *Matplotlib*¹³ – biblioteka koja se zajedno sa *NumPy* koristi za kreiranje grafika (u *2D* i *3D* prostoru), što je korišćeno prilikom posmatranja vrijednosti funkcije greške kroz iteracije
- *OpenCV*¹⁴ – biblioteka koja sadrži moćne funkcije koje su veoma korisne prilikom rješavanja problema računarske vizije u realnom vremenu
- *PyTorch*¹⁵ – uz *Tensorflow* i *Keras*, najpopularnija biblioteka iz mašinskog i dubokog učenja, ima primjenu prilikom rješavanja problema iz računarske vizije, obrade prirodnih jezika (engl. *Natural Language Processing*), radom sa zvukom itd.

Specifikacija računara na kojem su trenirani modeli je sledeća: procesor *Intel Core i7-7700HQ @ 2.80GHz* sa 4 jezgra i 8 logičkih procesora (engl. *CPUs*), grafička kartica *NVIDIA GeForce GTX 1070* sa ukupno 16 GB memorije, od čega je 8 GB *VRAM (display memory)*, 16 GB *RAM* memorije.

¹¹ <https://pillow.readthedocs.io/en/stable/>

¹² <https://numpy.org/>

¹³ <https://matplotlib.org/>

¹⁴ <https://opencv.org/>

¹⁵ <https://pytorch.org/>

4.2 Arhitektura rješenja

Rješenje se može podijeliti na dva logička dijela – inicijalni model kojim je predstavljen osnovni pristup (poglavlje 4.2.1) i optimizovani model (poglavlje 4.2.2).

4.2.1 Standardni neuralni prenos stila

Prva stavka koju je bilo potrebno ispuniti u projektu jeste reprezentovati sadržajnu (C), stilsku (S) i izlaznu (generisanu - G) sliku na numerički način. Koristeći *Pillow* biblioteku izvršeno je učitavanje slika nad datom putanjom u fajl (engl. *file*) sistemu. Slika je učitana i predstavljena u RGB formatu [21]. Ukoliko je veća od 800 piksela po visini ili širini, smanjena je na veličinu od 800 piksela po toj osi. Razlog za ovo jeste činjenica da performanse ovog algoritma koristeći CNN omogućuju najviše rad sa rezolucijama od oko 800 ili 900 piksela po osi. Ukoliko bismo htjeli da radimo sa visoko-rezolucionim slikama, morali bismo koristiti još složenije (u ovom slučaju pomoćne) arhitekture neuronskih mreža (kao što su, na primjer, generativne suparničke mreže [34] – *Generative Adversarial Networks*). Te arhitekture bi omogućile održavanje visoke rezolucije i za izlaznu sliku.

Sledeći korak u rješenju je da se, koristeći *PyTorch compose*¹⁶ funkciju ulanča niz operacija koje dovode do transformacije slike u željenu reprezentaciju.

Svi pretrenirani *PyTorch* modeli (pa i *VGG-19* mreža) očekuju da ulazne slike budu normalizovane na isti način: formiranje *mini-batch*-eva od trokanalnih RGB slika čije su dimenzije $3 \times \text{visina} \times \text{širina}$ slike, gdje se očekuje da su visina i širina slike makar 224 piksela. Slike se moraju učitati unutar intervala $[0, 1]$ i potom normalizovati koristeći srednju vrijednost $[0.485, 0.456, 0.406]$ i standardnu devijaciju $[0.229, 0.224, 0.225]$. Na kraju, kao vid optimizacije može se ukloniti *transparency* kanal, zato što ovdje nećemo imati nikakve providne elemente na slici, pa ćemo uštediti na vremenu jer nema potrebe da se gleda taj kanal.

Nakon što smo učitali sliku, potrebno je formirati *feature* reprezentacije (reprezentacije obilježja) za C, S i G sliku za ovaj problem. To podrazumijeva mapiranje odgovarajućih *torch*¹⁷ slojeva na rječnik čiji je ključ naziv sloja koji je korišćen u originalnom radu (uvedeno radi

¹⁶

<https://pytorch.org/docs/stable/torchvision/transforms.html#torchvision.transforms.Compose>

¹⁷ <https://pytorch.org/docs/stable/nn.html>

konzistentnosti), a vrijednost je *torch* tenzor koji predstavlja numeričku reprezentaciju date slike na odgovarajućem sloju:

- Nulti *torch* sloj se mapira na sloj *conv1_1*
- Peti *torch* sloj se mapira na sloj *conv2_1*
- Deseti *torch* sloj se mapira na sloj *conv3_1*
- Devetnaesti *torch* sloj se mapira na sloj *conv4_1*
- Dvadeset i prvi *torch* sloj se mapira na sloj *conv4_2*
- Dvadeset i osmi *torch* sloj se mapira na sloj *conv5_1*

Definisana je posebna klasa koja posjeduje statičku metodu koja za dati konvolutivni sloj K izračunava funkciju sadržajne greške između *feature* reprezentacije C i G slike. Funkcija greške između C i G slike, čije su *feature* reprezentacije smještene u matrice F (za G sliku) i P (za C sliku) izračunata je po sledećoj formuli:

$$L_{sadržaj}(C, G, K) = \frac{1}{2} \sum_{i,j} (F_{ij}^K - P_{ij}^K)^2$$

gdje su F_{ij}^K i P_{ij}^K aktivacije i -tog filtera na poziciji j u sloju K .

Potom je unutar klase koja se bavi stilskom greškom izračunata greška između S i G slike koristeći *Gram* matrice [14]. Zasnovana je na korelacijama između različitih odgovora filtera (*feature* korelacija). Te *feature* korelacije su predstavljene uz pomoć *Gram* matrica $M^K \in \mathbb{R}^{N_K \times N_K}$, gdje M_{ij}^K predstavlja skalarni proizvod između vektorizovanih *feature* mapa i i j u sloju K , a to izračunavamo na sledeći način:

$$M_{ij}^K = \sum_k F_{ik}^K F_{jk}^K,$$

gdje iteriramo po k koji predstavlja konkretnu poziciju unutar *feature* mape, odnosno, brojčanu vrijednost.

Potrebno je sumirati grešku na svim slojevima koji su nam reprezentativni za ekstrakciju stilskih karakteristika kako bi se dobila ukupna stilska greška. Nije cilj da upamtimo stil samo iz jednog sloja mreže, nego da spojimo funkcije greške za stilove iz više različitih slojeva mreže.

Upravo zbog toga, računamo prvo uticaj U sloja K na ukupnu stilsku grešku:

$$U_K = \frac{1}{4N_K^2 M_K^2} \sum_{i,j} (M_{ij}^K - N_{ij}^K)^2,$$

gdje su M i N redom stilske reprezentacije (*feature* matrice) G i S slika.

U radu [1] je eksperimentalno utvrđeno da nemaju svi slojevi istu težinu, te da se najbolja ekstrakcija teksture i stilskih elemenata dobija upravo dodijeljujući posebne težine odgovarajućim slojevima.

Upravo zato, korišćićemo posebne težine za slojeve mreže kao što je opisano u tom radu [1]. Potom ćemo sumirati stilske greške za različite slojeve i dobiti konačnu stilsku grešku na sledeći način:

$$L_{stil}(S, G, K) = \sum_{k=0}^K w_k U_k$$

gdje su w_k težine gore pomenutih slojeva, koje govore koji sloj više utiče na ukupnu stilsku grešku. Težine su preuzete iz rada [1]:

- $w_0 = 1.0$ za *conv1_1*
- $w_1 = 0.75$ za *conv2_1*
- $w_2 = 0.2$ za *conv3_1*
- $w_3 = 0.2$ za *conv4_1*
- $w_4 = 0.2$ za *conv5_1*

dok se *conv4_2* koristi za sadržajnu grešku i izostavlja prilikom računanja stilske greške.

Rješenje prikazano u radu je implementirano tako da je korisniku moguće da podešava $ratio = \frac{\alpha}{\beta}$, koji predstavlja odnos uticaja sadržaja i stila na generisanu sliku. Što je odnos veći, samim tim je i vrijednost parametra α veća i na G slici dajemo veći uticaj očuvanju sadržaja.

Glavni program funkcioniše tako da se specificira broj iteracija algoritma (podrazumijevano je stavljen na 8000) i potom se prvo za odgovarajuću G sliku formira *feature* reprezentacija. Bitno je napomenuti da je G slika inicijalno postavljena da bude jednaka C slici (za razliku od originalnog rada gdje smo na početku imali *white noise* sliku).

U svakoj iteraciji, računa se sadržajna greška između G i C slike (na početku su jednake) ali samo nad slojem *conv4_2* jer je on pokazan kao najuticajniji sloj što se tiče reprezentacije sadržaja [1] (na primjer, raspored elemenata na slici).

Stilska greška se na početku svake iteracije inicijalizuje na nulu, a potom se prolazi kroz sve gore navedene slojeve relevantne za reprezentaciju stila. Za svaki sloj (na primjer, *conv1_1*) računa se stilska greška između *Gram* matrica [14] G i S slike za dati sloj i ukupna greška se uvećava za tu grešku.

Kada se prođe kroz sve slojeve relevantne za stilsku grešku, od *conv1_1* do *conv5_1*, potom se računa ukupna greška za tu iteraciju:

$$L_{ukupna\ greška}(C, S, G) = \alpha L_{sadržaj}(C, G) + \beta L_{stil}(S, G)$$

i vrši se ažuriranje G slike koristeći odgovarajući optimizator (u ovom slučaju *Adam* [35]). Ovo se vrši na način da se isprazne svi baferi za gradijente za sve parametre (kako ne bi ostale vrijednosti za prethodnu iteraciju), potom se izvrši propagacija unazad [29] koja je automatizovana

kroz okruženje *PyTorch* [7] i na kraju se ažuriraju parametri u datom koraku preko optimizeroze metode *step*¹⁸ koja je, takođe, ugrađena u okruženje. Na kraju, nakon odgovarajućeg broja iteracija (podrazumijevanih 8000) dobija se izlazna slika. U programu je nakon svakih 1000 iteracija generisana slika kako bi se mogla porediti stanja.

4.2.2 Optimizovani neuralni prenos stila

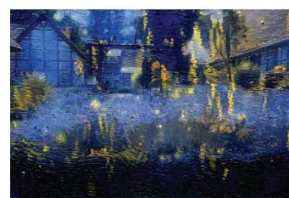
Glavni problem sa standardnim pristupom jeste činjenica da generisana slika G nema očuvane boje od sadržajne slike C (kao što se vidi na slici 4.1). Dakle, kada imamo kao ulaz C i S slike, kada se izgeneriše G slika, na njoj će biti očuvan sadržaj C slike, ali će biti preuzete dominantne boje iz S slike, što najčešće nije poželjno.



Sadržajna slika



Stilska slika



Dobijeni izlaz koristeći standardni pristup

Slika 4.1 Prikaz problema sa očuvanjem boja kod standardnog pristupa neuralnog prenosa stila. Slika koja je korišćena kao stilska jeste „Zvezdana noć iznad Rone“, autora Vinsenta Van Goga.

¹⁸ <https://pytorch.org/docs/stable/optim.html#optimizer-step>



Sadržajna slika



Stilska slika

Dobijeni izlaz korišćenjem *Color Histogram Matching* optimizacijeDobijeni izlaz korišćenjem *Luminance-only Transfer* optimizacije

Slika 4.2 Prikaz dobijenog rješenja korišćenjem optimizovanog pristupa.

Korišćenjem *Color Preservation* optimizacije [6] se može popraviti ovaj problem (što se vidi na slici 4.2), tako da na kraju G očuvava sadržaj C slike, kao i boje prisutne na njoj, dok stil slike (tekstura, luminozitet...) bude isti kao stil S slike. Korišćenje oba pristupa podrazumijeva da se kao ulaz iskoriste C i S slika, te da se formira S' kao nova stilska slika koja će na neki način odgovarati C slici i zajedno sa njom biti ulaz u standardni neuralni prenos stila. Ovu optimizaciju moguće je izvršiti koristeći jedan od dva pristupa [6]:

- *Color Histogram Matching*
- *Luminance-only Transfer*.

Color Histogram Matching pristup.

Za početak, potrebno je učitati C i S slike uz pomoć funkcije *imread*¹⁹ koja je dio *Python*-ove *OpenCV* biblioteke. Nakon toga, vrši se normalizacija nad slikama (predstavljene su kao multidimenzionalni nizovi pri čemu je svaki piksel posebna lista tri vrijednosti: *Red*, *green* i *blue*). Normalizaciju vršimo tako što svaku vrijednost dijelimo sa brojem 256 jer su moguće vrijednosti za crvenu, zelenu i plavu vrijednost od 0 do 255.

¹⁹ <https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/>

Nakon što smo učitali i normalizovali C i S slike, potrebno je izvršiti transformaciju slike S tako da dobijemo sliku S' čija će srednja vrijednost i kovarijansa RGB vrijednosti da se poklapa sa C slikom. Inicijalno, slika S' je ista kao slika S.

Neka je $x_i = (R, G, B)^T$ piksel slike. Svaki piksel se transformiše na sledeći način: $x_{S'} \leftarrow Ax_S + b$, gdje je A matrica čije su dimenzije 3×3 , a b je vektor dužine 3.

Srednju vrijednost i kovarijansu piksela računamo preko sledećih formula:

$$\mu = \sum_i \frac{x_i}{N}$$

$$\delta = \sum_i \frac{(x_i - \mu)(x_i - \mu)^T}{N}.$$

Nakon što smo izračunali srednje vrijednosti i kovarijanse piksela, želimo da matrica A i vektor b budu takvi da zadovolje da je $\mu_{S'} = \mu_C$ i $\delta_{S'} = \delta_C$. To se zadovoljava uvodeći dva ograničenja:

$$b = \mu_C - A\mu_S \text{ i } A\delta_S A^T = \delta_C.$$

Postoji više načina da se pronađe familija rješenja za A (b se izračunava na osnovu A, kao što je dato u formuli iznad) koja zadovoljava ova ograničenja. Pristup koji je korišćen u ovom radu jeste dekompozicija matrica metodom Šoleskog (engl. *Cholesky decomposition*²⁰). Ovo se koristeći *Numpy* biblioteku uradi pozivom funkcije *numpy.cholesky*²¹ na osnovu čega dobijamo dekompozicije za C i S sliku koje ćemo označiti sa L_C i L_S . Konačno, vrijednosti matrice A izračunavamo na sledeći način:

$$A = L_C L_S^{-1},$$

čime na kraju dobijamo transformisanu S' sliku koja se može iskoristiti kao ulaz u standardni pristup jer odgovara po distribuciji boja slici C.

²⁰ https://en.wikipedia.org/wiki/Cholesky_decomposition

²¹ <https://numpy.org/doc/stable/reference/generated/numpy.linalg.cholesky.html>

Luminance-only Transfer pristup.

Kao i u prvom pristupu, prvo se izvrši učitavanje slike koja se, u ovom slučaju, konvertuje u logaritamski prostor boja. Nakon toga, izračunaju se srednje vrijednosti (Δ) i standardne devijacije (σ) luminoziteta za C i S slike.

Inicijalno se postavi da je $S' = S$, tj. slika S se prekopira u S' . Potom se prolazi kroz svaki piksel (iteracija po visini, širini i dubini) i ažurira njegova vrijednost na sledeći način:

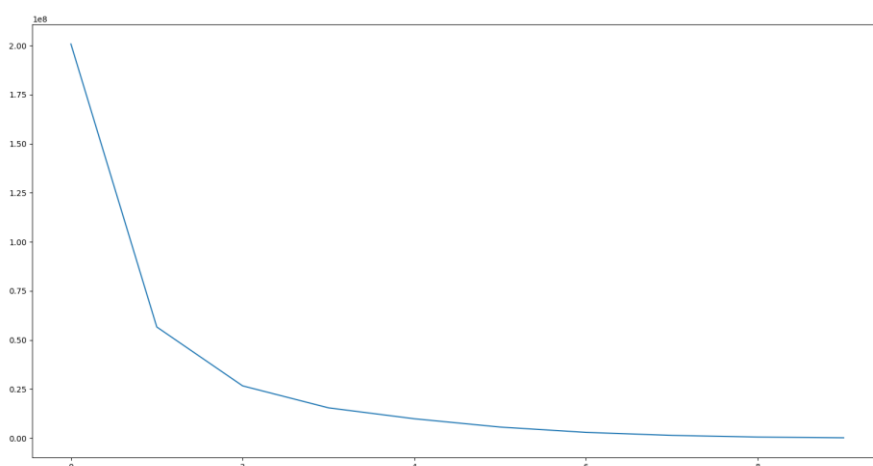
$$L_{S'} = \frac{\sigma_C}{\sigma_S} (L_S - \Delta_S) + \Delta_C ,$$

gdje je L_S tekući piksel stilske slike.

Na ovaj način dobili smo novu sliku S' koju konvertujemo u RGB prostor boja i koristimo kao ulaznu stilsku sliku za standardni neuralni prenos stila i dobijamo rješenje koje očuvava boje C slike zato što su sada C i S slika ekvivalentne po luminozitetu.

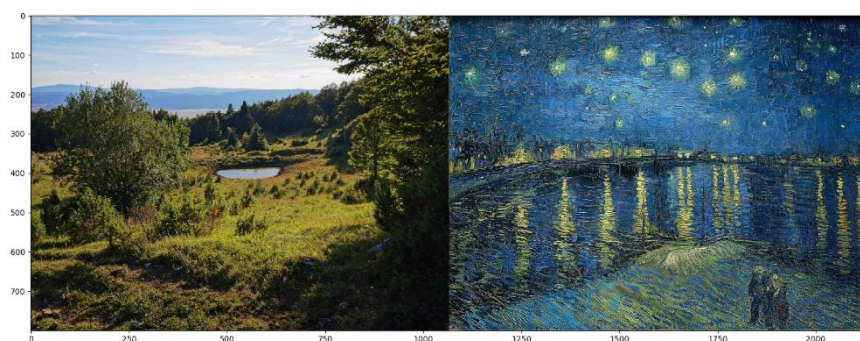
5. EVALUACIJA RJEŠENJA I REZULTATI

Evaluacija je čest problem kod soft kompjuting i fazi sistema, odnosno, teško je definisati adekvatnu metriku za procjenu njihovog kvaliteta. Upravo zato, izvršeno je poređenje rezultata implementacije date u ovom radu sa ostalim naučnim radovima (pa i originalnim radom) i poređenjem subjektivnom vizuelnom percepcijom utvrđeno je da nema razlika u kvalitetu izlaznih slika, bez obzira na to što su načini implementacija u izvjesnoj mjeri različiti (za inicijalnu generisanu sliku u ovom radu je uzeta sadržajna slika, dok je u ostalim uzeta *white noise* slika, potom korišćena su i različita okruženja itd.).



Slika 5.1 Prikaz funkcije greške. Na horizontalnoj osi nalaze se vremenski koraci jedinice 1000 iteracija (zato što se generiše slika na svakih 1000 iteracija, a podrazumijevano nakon 8000 završavamo sa iteriranjem), dok se na vertikalnoj osi nalazi vrijednost funkcije greške.

Kao što se može primijetiti na slici 5.1, funkcija greške (engl. *loss function*) [11] opada kroz vrijeme, što je pokazatelj da, usljed pravilne postavke problema, minimizujemo grešku između G slike tako da njen sadržaj što više odgovara sadržaju C slike, a stil stilu S slike.



Slika 5.2 Primjer zadatih ulaza u program. Lijevo se nalazi slika uslikana telefonom autora diplomskog rada (mjesto Blagaj, Kupres, RS, BiH), a desno se nalazi slika autora Vinsenta Van Goga, pod nazivom "Zvezdana noć iznad Rone" izmijenjena postupkom *Luminance-only transfer-a*.



Slika 5.3 Dobijeni izlazi sa ulaznih slika sa slike 5.2.

Na slici 5.3, u slučaju a) dat je veći značaj sadržaju slike, dok je u slučaju d) dat veći značaj stilu slike, te vidimo da izgleda mnogo više kao rad na ulju.

5.1 Uticaj različitih slojeva CNN-a

Veoma bitan faktor u procesu sinteze izlazne slike jeste i odabir slojeva neuronske mreže koji će se koristiti za sadržajnu i stilsku reprezentaciju (kao što se vidi na slici 5.4).



Slika 5.4 Uticaj različitih slojeva na sadržajnu reprezentaciju slike. Slika koja je korišćena kao stilski jeste "Jezuiti 3", autora Lajonela Fajningera.

Reprezentacija stila je višeslojna reprezentacija i pažljiv odabir slojeva mreže koji će se koristiti za stil će dovesti do najbolje rekonstrukcije stila (teksture, luminoziteta...). Vizuelno gledano, najbolje slike se dobijaju tako što se za stilsku reprezentaciju iskoriste viši slojevi mreže, te su upravo iz ovog razloga za reprezentaciju stila odabrani slojevi: *conv1_1*, *conv2_1*, *conv3_1*, *conv4_1* i *conv5_1*.

Kada govorimo o reprezentaciji sadržaja, na nižim slojevima mreže algoritam rekonstruiše detaljno informaciju o pikselima sadržajne slike i dobijena generisana slika je veoma slična sadržajnoj, te izgleda kao „zamućena“ sadržajna slika (kada odaberemo sloj *conv2_2*). U slučaju da odaberemo viši sloj *conv4_2*, tekstura stilske slike i sadržaj sadržajne slike su oboje prisutni na generisanoj slici. Na osnovu vizuelne percepcije, upravo zato je sloj *conv4_2* odabran kao najuticajniji konvolutivni sloj za računanje sadržajne greške.

5.2 Inicijalizacija *gradient descent-a*

U ovom radu, sve generisane slike su inicijalizovane tako da budu jednake sadržajnoj slici. Takođe, moguće je izvršiti inicijalizaciju slike tako da ona bude *white noise* slika, što je isto isprobano. Iako kada inicijalizujemo sliku kao sadržajnu, G nekim dijelom teži ka tome da očuva prostornu stukturu sadržajne slike, ali to uopšte nema velikog efekta po krajnji rezultat, kao što se može vidjeti na slici na sledećoj stranici (slika 5.5).



Slika 5.5 Različite *gradient descent* inicijalizacije. **A** - rješenje dobijeno iz inicijalne sadržajne slike. **B** - rješenje dobijeno iz inicijalne stilske slike. **C** - četiri primjera rješenja dobijena iz *white noise* slike sa različitim odnosima α/β .

6. ZAKLJUČAK

U ovom radu predstavljen je sistem za generisanje umjetničkih radova date ulazne slike i slike stila. Motivacija je bila što bi takav sistem mogao da se koristi kao dio nekog složenijeg softvera ili bilo koje aplikacije sa primjenom filtera. Šire gledano, rješenje se može iskoristiti kao dio aplikacije (ili podsistema) koji ima veze sa *photo editing*-om.

Glavni problem sa standardnim neuralnim prenosom stila [1] jeste što nije moguće efektivno očuvati boje na generisanoj slici koje potiču od sadržajne slike. Ukoliko su nam date sadržajna i stilska slika kao ulazi, kada generišemo izlaznu sliku, njen sadržaj je očuvan, ali su dominantne boje preuzete iz stilske slike.

Color Preservation optimizacija [6] nam omogućuje da očuvamo sadržaj sadržajne slike, ali takođe i boje, dok se samo stilske karakteristike (tekstura, luminozitet, itd.) preuzimaju iz stilske slike.

Optimizacija se može izvršiti koristeći jedan od navedena dva pristupa:

- *Color Histogram Matching* (rad sa distribucijama boja)
- *Luminance-only transfer* (vrši prenos stila samo u kanalu koji se tiče luminoziteta – ovo je motivisano činjenicom da zapažanja koja se tiču vizuelne percepcije imaju veću senzitivnost nad promjenama u luminozitetu nego u boji).

Pošto je problem standardnog neuralnog prenosa stila riješen koristeći *Color Preservation* optimizaciju, osvrnućemo se na pojedine njene nedostatke.

Color Histogram Matching (*) je prirodno ograničen time koliko dobro se odvije prenos boje sa sadržajne slike na stilsku sliku. Distribucija boja se ne može uvijek savršeno premapirati, što dovodi do određene neusklađenosti (ponekad neznatne) između boja u izlaznoj i sadržajnoj slici.

Luminance-only transfer (**) savršeno očuva boje sadržajne slike. Međutim, zavisnosti između luminoziteta [6] i RGB kanala [21] se gube u izlaznoj slici. Ovo je naročito očigledno za stilove gdje su prisutni istaknuti potezi debelim kistom. Kao rezultat, gubi se osjećaj o veličini korišćenog kista i zasebni potezi mogu imati mnogo više boja nego što to imaju pravi umjetnički radovi.

U budućnosti, ovaj problem se može dodatno optimizovati unifikacijom oba optimizujuća pristupa (* i **) i takođe eksperimentisanjem sa sofisticiranijim metodama koje su u stanju da bolje rade sa prenosom samih tekstura, zato što standardni pristup ima izvjesne probleme sa teksturama.

7. LITERATURA

- [1] Gatys, L.A., Ecker, A.S. and Bethge, M., 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.
- [2] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *nature*, 521(7553), pp.436-444.
- [3] LeCun, Y. and Bengio, Y., 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), p.1995.
- [4] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. and Liu, C., 2018, October. A survey on deep transfer learning. In *International conference on artificial neural networks* (pp. 270-279). Springer, Cham.
- [5] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [6] Gatys, L.A., Bethge, M., Hertzmann, A. and Shechtman, E., 2016. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*.
- [7] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (pp. 8026-8037).
- [8] Zhou, B., Bau, D., Oliva, A. and Torralba, A., 2018. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9), pp.2131-2145.
- [9] Kumar, V., Glaude, H., de Lichy, C. and Campbell, W., 2019. A Closer Look At Feature Space Data Augmentation For Few-Shot Intent Classification. *arXiv preprint arXiv:1910.04176*.
- [10] Gatys, L.A., Ecker, A.S., Bethge, M., Hertzmann, A. and Shechtman, E., 2017. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3985-3993).
- [11] Zhao, H., Gallo, O., Frosio, I. and Kautz, J., 2015. Loss functions for neural networks for image processing. *arXiv preprint arXiv:1511.08861*.
- [12] LeCun, Y., Touresky, D., Hinton, G. and Sejnowski, T., 1988, June. A theoretical framework for back-propagation. In *Proceedings of the 1988*

- connectionist models summer school* (Vol. 1, pp. 21-28). CMU, Pittsburgh, Pa: Morgan Kaufmann.
- [13] Yang, L., Chen, W., Liu, W., Zha, B. and Zhu, L., 2020. Random noise attenuation based on residual convolutional neural network in seismic datasets. *IEEE Access*, 8, pp.30271-30286.
 - [14] Hoyle, D.C. and Rattray, M., 2004, July. A statistical mechanics analysis of Gram matrix eigenvalue spectra. In *International Conference on Computational Learning Theory* (pp. 579-593). Springer, Berlin, Heidelberg.
 - [15] Van Gerven, M. and Bohte, S., 2017. Artificial neural networks as models of neural information processing. *Frontiers in Computational Neuroscience*, 11, p.114.
 - [16] McCulloch, W.S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), pp.115-133.
 - [17] Brightwell, G., Kenyon, C. and Paugam-Moisy, H., 1997. Multilayer neural networks: one or two hidden layers?. In *Advances in Neural Information Processing Systems* (pp. 148-154).
 - [18] Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
 - [19] Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S., 2018. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.
 - [20] Stephen, I., 1990. Perceptron-based learning algorithms. *IEEE Transactions on neural networks*, 50(2), p.179.
 - [21] Wilson, D.B., 2004. Red-green-blue model. *Physical Review E*, 69(3), p.037105.
 - [22] Hawkins, D.M., 2004. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1), pp.1-12.
 - [23] Ma, Y., Cao, Y., Vruthula, S. and Seo, J.S., 2018. Optimizing the convolution operation to accelerate deep neural networks on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(7), pp.1354-1367.
 - [24] Hubel, D.H. and Wiesel, T.N., 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1), p.106.
 - [25] McCann, S. and Reesman, J., 2013. Object Detection using Convolutional Neural Networks.
 - [26] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

- [27] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- [28] Nair, V. and Hinton, G.E., 2010, January. Rectified linear units improve restricted boltzmann machines. In *ICML*.
- [29] Han, J. and Moraga, C., 1995, June. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks* (pp. 195-201). Springer, Berlin, Heidelberg.
- [30] Hu, Y., Huber, A., Anumula, J. and Liu, S.C., 2018. Overcoming the vanishing gradient problem in plain recurrent networks. *arXiv preprint arXiv:1801.06105*.
- [31] Gan, Y., Yang, J. and Lai, W., 2019, December. Video Object Forgery Detection Algorithm Based on VGG-11 Convolutional Neural Network. In *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)* (pp. 575-580). IEEE.
- [32] Alippi, C., Disabato, S. and Roveri, M., 2018, April. Moving convolutional neural networks to embedded systems: the alexnet and VGG-16 case. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (pp. 212-223). IEEE.
- [33] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. and Lipson, H., 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*.
- [34] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).
- [35] Zhang, Z., 2018, June. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (pp. 1-2). IEEE.

8. BIOGRAFIJA

Nikola Zubić je rođen 13. decembra. 1997. godine u Banjoj Luci. Osnovnu školu „Sveti Sava“ u Brodu završio je 2012. godine, kao nosilac diplome „Vuk Karadžić“, te je bio učesnik mnogobrojnih takmičenja iz matematike, informatike, istorije i geografije. Opštu gimnaziju „Nikola Tesla“ u Brodu upisao je iste te godine, a završio 2016. godine kao učenik generacije i nosilac diplome „Vuk Karadžić“, te je bio učesnik takmičenja iz matematike i osvojio je specijalnu pohvalu na takmičenju „Inost mladih“ koji organizuje Savez Inovatora Republike Srpske. 2016. godine upisuje Fakultet Tehničkih Nauka u Novom Sadu, smjer Softversko Inženjerstvo i Informacione Tehnologije kao budžetski student. Tokom studija, nastavlja da aktivno i dodatno radi na sebi i stiče znanja iz oblasti vještačke inteligencije gdje je znanje proširio online specijalizacijama i studentskom praksom. Položio je sve ispite predviđene planom i programom sa prosječnom ocjenom 9.68.

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj, RBR:	
Identifikacioni broj, IBR:	
Tip dokumentacije, TD:	monografska publikacija
Tip zapisa, TZ:	tekstualni štampani dokument
Vrsta rada, VR:	diplomski rad
Autor, AU:	Nikola Zubić
Mentor, MN:	Vanr. Prof. Dr Jelena Slivka
Naslov rada, NR:	Korišćenje neuralnog prenosa stila uz optimizaciju očuvanja boja za generisanje umjetničkih radova
Jezik publikacije, JP:	srpski
Jezik izvoda, JL:	srpski / engleski
Zemlja publikovanja, ZP:	Srbija
Uže geografsko područje, UGP:	Vojvodina
Godina, GO:	2020
Izdavač, IZ:	autorski reprint
Mesto i adresa, MA:	Novi Sad, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6
Fizički opis rada, FO:	8 / 52 / 35 / 2 / 17 / 0 / 0
Naučna oblast, NO:	Softversko inženjerstvo i informacione tehnologije
Naučna disciplina, ND:	Soft kompjuting
Predmetna odrednica / ključne reči, PO:	soft kompjuting, konvolutivne neuronske mreže, računarska vizija
UDK	
Čuva se, ČU:	Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6, Novi Sad
Važna napomena, VN:	
Izvod, IZ:	Zadatak ovog rada je generisanje slike <i>G</i> koja kombinuje sadržaj slike <i>C</i> sa stilom slike <i>S</i> . Rješenje ovog problema može se iskoristiti kao dio nekog složenijeg softvera ili bilo koje aplikacije sa primjenom filtera, šire gledano podsistema koji ima veze sa <i>photo editing</i> -om.
Datum prihvatanja teme, DP:	
Datum odbrane, DO:	
Članovi komisije, KO:	
predsednik	
član	
mentor	Vanr. Prof. Dr Jelena Slivka
Potpis mentora	

KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	monographic publication
Type of record, TR :	textual material
Contents code, CC :	bachelor thesis
Author, AU :	Nikola Zubić
Mentor, MN :	Jelena Slivka, Associate Professor, PhD
Title, TI :	Usage of Neural Style Transfer with Color Preservation Optimization for generating art pieces
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian / English
Country of publication, CP :	Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2020
Publisher, PB :	author's reprint
Publication place, PP :	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, PD :	8 / 52 / 35 / 2 / 17 / 0 / 0
Scientific field, SF :	Software Engineering and Information Technologies
Scientific discipline, SD :	Soft Computing
Subject / Keywords, S/KW :	soft computing, convolutional neural networks, computer vision
UDC	
Holding data, HD :	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, N :	
Abstract, AB :	Main task of this thesis is to generate an image <i>G</i> which combines content of image <i>C</i> with the style of image <i>S</i> . Solution of this problem can be used as part of more complex software or any application with filter application, more comprehensively said any subsystem that deals with photo editing.
Accepted by sci. Board on, ASB :	
Defended on, DE :	
Defense board, DB :	
president	
member	
mentor	Jelena Slivka, Associate Professor, PhD
Mentor's signature	

