# Prijenos obilježja slike primjenom konvolucijske neuronske mreže

**Šarčević, Matej**

**Master's thesis / Diplomski rad**

**2022**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:030103

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-19**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I**

**INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Diplomski sveučilišni studij Računarstvo**

# PRIJENOS OBILJEŽJA SLIKE PRIMJENOM KONVOLUCIJSKE NEURONSKE MREŽE

**Diplomski rad**

**Matej Šarčević**

**Osijek, 2022.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit**

**Osijek, 09.09.2022.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za diplomski ispit

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Matej Šarčević |
| **Studij, smjer:** | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | D-1165R, 13.10.2020. |
| **OIB studenta:** | 19679503712 |
| **Mentor:** | Izv. prof. dr. sc. Časlav Livada |
| **Sumentor:** | , |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | Izv. prof. dr. sc. Alfonzo Baumgartner |
| **Član Povjerenstva 1:** | Izv. prof. dr. sc. Časlav Livada |
| **Član Povjerenstva 2:** | Doc. dr. sc. Tomislav Galba |
| **Naslov diplomskog rada:** | Prijenos obilježja slike primjenom konvolucijske neuronske mreže |
| **Znanstvena grana diplomskog rada:** | **Umjetna inteligencija (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | U radu je potrebno opisati osnove konvolucijskih neuronskih mreža te ih primijeniti na postupak prijenosa obilježja slike s jedne na drugu. Detalji na: https://towardsdatascience.com/artistic-style-transfer-b7566a216431 Tema rezervirana za: Matej Šarčević |
| **Prijedlog ocjene pismenog dijela ispita (diplomskog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 2 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 2 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene od strane mentora:** | 09.09.2022. |

| | |
|---|---|
| Potvrda mentora o predaji konačne verzije rada: | *Mentor elektronički potpisao predaju konačne verzije.* |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 12.09.2022.**

| | |
|---|---|
| **Ime i prezime studenta:** | Matej Šarčević |
| **Studij**: | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | D-1165R, 13.10.2020. |
| **Turnitin podudaranje [%]:** | 9 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Prijenos obilježja slike primjenom konvolucijske neuronske mreže**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Časlav Livada

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

1. UVOD ..... 1
2. UVOD U NEURONSKE MREŽE ..... 2
2.1. Biološki neuron ..... 2
2.2. Umjetni neuron ..... 3
2.3. Aktivacijske funkcije ..... 4
2.3.1. Funkcija praga ..... 4
2.3.2. Funkcija identiteta ..... 5
2.3.3. Ispravljena linearna jedinica (ReLU) ..... 6
2.3.4. Sigmoid funkcija ..... 7
2.4. Učenje neuronske mreže ..... 8
2.4.1. Gradijentni spust ..... 9
2.4.2. Stohastički gradijentni spust ..... 10
2.4.3. Algoritam propagacije unatrag ..... 10
3. KONVOLUCIJSKE NEURONSKE MREŽE ..... 11
3.1. Konvolucijski sloj ..... 12
3.2. Sloj sažimanja ..... 13
3.3. Potpuno povezani sloj ..... 14
3.4. Primjeri konvolucijskih neuronskih mreža ..... 14
3.4.1. LeNet-5 ..... 14
3.4.2. VGG ..... 15
3.4.3. AlexNet ..... 16
4. PROGRAMSKO RJEŠENJE ZA PRIJENOS OBILJEŽJA SLIKE ..... 18
4.1. Neuronski prijenos stila ..... 18
4.2. Python ..... 22
4.3. Programsko rješenje ..... 22
5. ANALIZA REZULTATA ..... 27
6. ZAKLJUČAK ..... 42
LITERATURA ..... 43
SAŽETAK ..... 45
ABSTRACT ..... 46
ŽIVOTOPIS ..... 47

# 1. UVOD

Iako umjetna inteligencija i strojno učenje, kao grana umjetne inteligencije postoje već duže vremena, točnije od sredine 20. stoljeća, njihova popularnost naglo je porasla tek u posljednjih desetak godina. Razlog tomu je što su računalne performanse postale puno bolje nego prije i što je u današnje vrijeme količina podataka koja bi se koristila za treniranje neuronskih mreža puno veća, a samim time se i područje primjene strojnog učenja proširilo. Jedna od mogućih primjena neuronskih mreža bit će opisana kroz ovaj rad.

Prvi dio rada opisuje neuronske mreže i način na koji funkcioniraju. Kako bi se lakše shvatio koncept umjetnog neurona, napravljena je usporedba sa biološkim neuronom. Osim toga, opisan je i rad najkorištenijih aktivacijskih funkcija, te su također prikazani njihovi grafovi i formule. Na kraju prvog dijela navedene su metode učenja neuronskih mreža, njihova podjela te su objašnjeni optimizacijski algoritmi koji se često koriste.

Drugi dio rada daje uvid u način rada konvolucijskih neuronskih mreža. Objašnjeni su slojevi od kojih se neuronske mreže sastoje kao i sami postupak konvolucije. Navedene su najpoznatije arhitekture konvolucijskih neuronskih mreža i opisani su slojevi od kojih se one sastoje i način na koji obrađuju ulazne podatke.

U trećem dijelu rada objašnjen je pojam neuronskog prijenosa stila, metode koja korištenjem konvolucijske neuronske mreže prenosi stil jedne slike na drugu, referentnu sliku. Prikazan je prvi znanstveni rad u kojemu je korišten algoritam za neuronski prijenos stila. Navedene su tehnologije korištene za izradu programskog rješenja te je samo programsko rješenje detaljno objašnjeno.

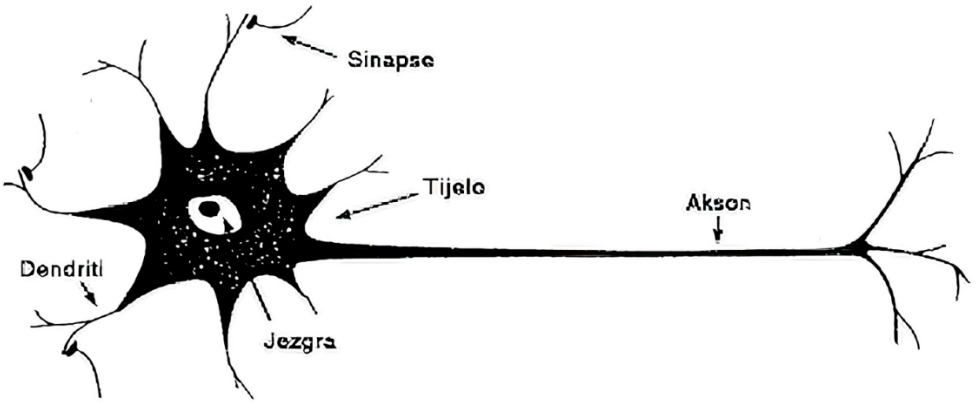
Za kraj, napravljena je analiza dobivenih rezultata gdje su prikazane generirane slike i slike koje su se koristile za generiranje istih.

# 2. UVOD U NEURONSKE MREŽE

Kako bi se lakše shvatio način rada konvolucijskih neuronskih mreža koje će biti korištene prilikom izrade programskog rješenja rada, kroz ovo poglavlje će biti obrađen uvod i osnovne stvari vezane za neuronske mreže. Pojam neuronska mreža može se odnositi na biološku neuronsku mrežu, kao ona u ljudskom mozgu ili na umjetnu neuronsku mrežu implementiranu na računalu, koja je motivirana biološkom neuronskom mrežom.

## 2.1. Biološki neuron

Zbog boljeg razumijevanja funkcioniranja umjetnih neurona, potrebno je upoznati se sa radom živčanog sustava koji je, kao što je spomenuto, poslužio kao inspiracija prilikom razvoja umjetnih neurona. Ljudski mozak sadrži otprilike $10^{11}$ neurona. Postoji oko 100 vrsta neutrona koji su raspoređeni prema definiranom rasporedu, ovisno o svojoj funkciji. Na slici 2.1. prikazan je biološki neuron.




Slika 2.1. *Prikaz biološkog neurona [1]*

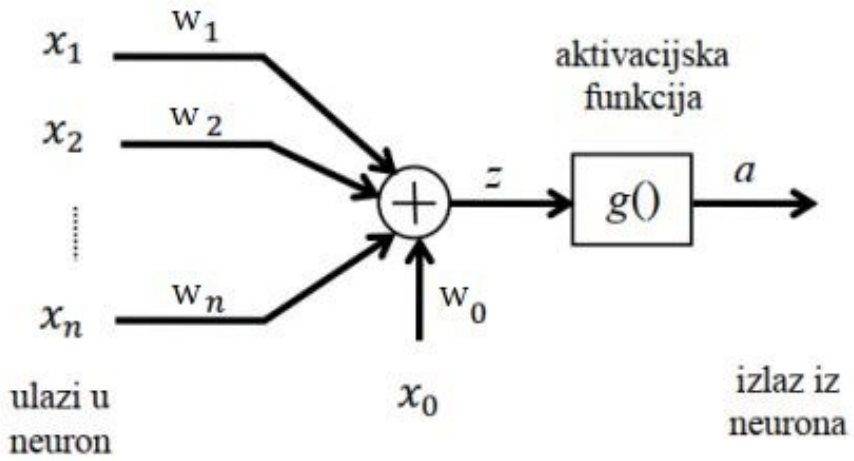
Neuroni se sastoje od tri dijela:

- Tijelo stanice
- Dendrit
- Akson

Tijelo stanice sadržava jezgru, odnosno nukleus u kojemu se nalaze informacije o nasljednim značajkama. Dendriti su kratke niti oko stanice koje služe za prijenos signala s drugih neurona. Aksoni su duge i tanke niti, a služe za prijenos signala drugim neuronima, prilikom čega se grana u vlakna.

## 2.2. Umjetni neuron

Umjetni neuron osnovni je element umjetne neuronske mreže. Dizajniran je tako da bude sličan biološkom neuronu, odnosno kako bi što vjernije oponašao funkcije biološkog neurona. Na slici 2.2. prikazano je kako izgleda umjetni neuron.




Slika 2.2. *Prikaz modela umjetnog neurona*

Ulazni signali označene su sa $x_1, x_2, \ldots x_n$, dok su težine označene sa $w_1, w_2, \ldots w_n$.

Težinska suma $z$ definirana je sa:

$$z = w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = \sum_{i=0}^{n} w_ix_i \qquad (2\text{-}1)$$

Izlaz α neurona je rezultat aktivacijske funkcije:

$$\alpha = f\left(\sum_{i=0}^{n} w_i x_i\right) = f(z) \tag{2-2}$$

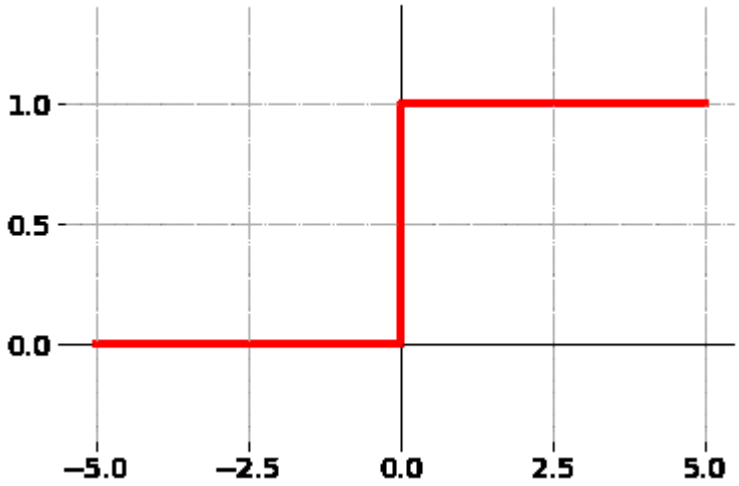
Aktivacijske funkcije koje se najčešće koriste opisane su u nastavku.

## 2.3. Aktivacijske funkcije

Aktivacijske funkcije koriste se jer unose nelinearnost u neuronsku mrežu te imaju značajan utjecaj na njeno učenje. Aktivacijska funkcija se odabire ovisno o arhitekturi neuronske mreže, o problemu koji ona rješava i o brzini izvođenja. Kada bi postojali samo linearni izlazi iz slojeva neuronske mreže, svi slojeve bi mogli biti zamijenjeni jednim slojem iz razloga što mreža ne bi mogla prepoznavati određene značajke zbog nedostatka nelinearnosti. Popularnije nelinearne aktivacijske funkcije su *sigmoid* i *ReLU* funkcije.

### 2.3.1. Funkcija praga

Funkcija praga jedna je od češće korištenih aktivacijskih funkcija u neuronskim mrežama. Funkcija proizvodi binarni izlaz. To je razlog zbog kojeg se naziva i binarnim korakom. Funkcija daje 1 (ili istinito) kada ulaz prijeđe granicu praga, dok se 0 dobije kada ulaz ne prijeđe prag. Zbog toga je funkcija praga vrlo korisna za binarnu klasifikaciju.




Slika 2.3. *Graf funkcije praga [2]*

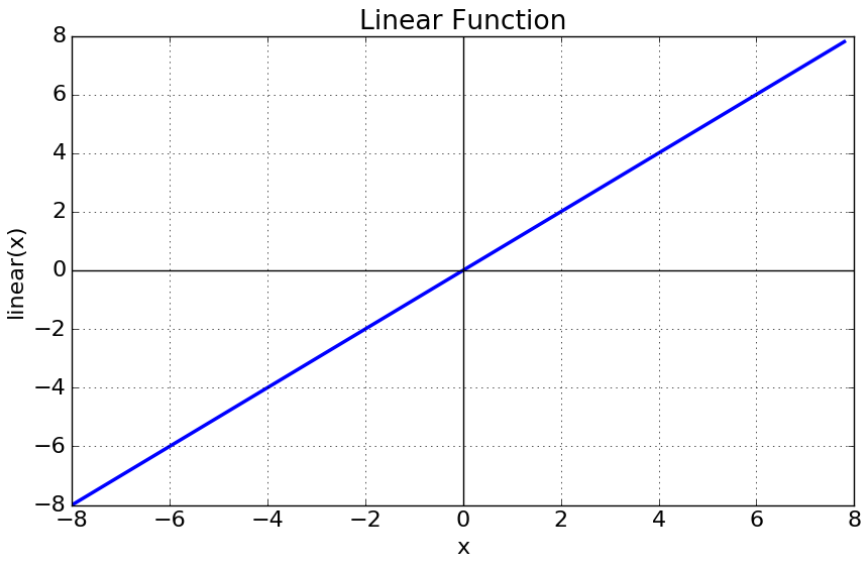
Funkcija praga definirana je formulom:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2-3}$$

Gradijent funkcije praga je nula što uzrokuje smetnju u procesu propagacije pogreške unatrag. To jest, kada se izračuna derivacija *f(x)* u odnosu na *x*, ispada da je 0. Gradijent se izračunava kako bi se ažurirale težine i pristranost tijekom propagacije pogreške unatrag. Budući da je gradijent funkcije nula, težine i pristranost se ne ažuriraju.

### 2.3.2. Funkcija identiteta

Funkcija identiteta je jednostavna, ali prilikom njenog korištenja neuronska mreža može naučiti samo linearne funkcije, te se mreža ne može koristiti na nelinearnim podatcima.




Slika 2.4. *Graf funkcije identiteta [2]*

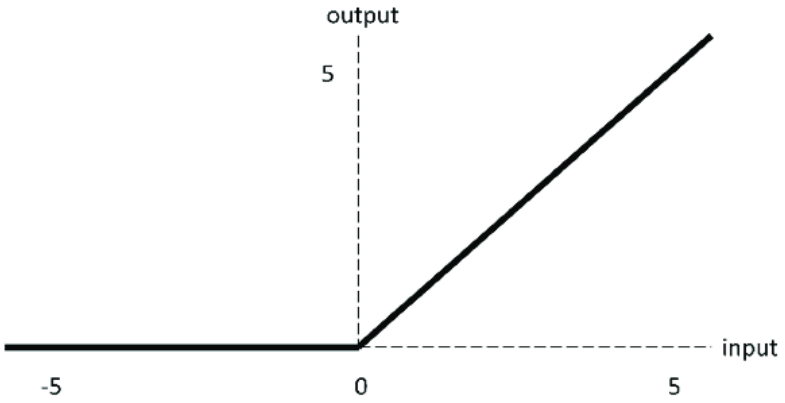
Funkcija identiteta definirana je formulom:

$$f(x) = x \tag{2-4}$$

Gradijent je u ovom slučaju konstanta iznosa 1 i ne ovisi o ulaznoj vrijednosti $x$. To znači da će se težine i pristranost ažurirati tijekom propagacije pogreške unatrag, ali će faktor ažuriranja uvijek biti isti. Zbog toga neuronska mreža neće poboljšati pogrešku jer je gradijent isti prilikom svake iteracije. Mreža se iz tog razloga neće moći dobro trenirati kada postoje složeni obrasci u podatcima.

### 2.3.3. Ispravljena linearna jedinica (ReLU)

Funkcija ispravljene linearne jedinice trenutno je najčešće korištena aktivacijska funkcija za optimizaciju neuronske mreže. Razlog tome je što se brzo računa, te što gradijent nikada ne ulazi u zasićenje, jer je uvijek 0 ili 1, neovisno o dubini neuronske mreže.




Slika 2.5. *Graf ReLU funkcije [2]*

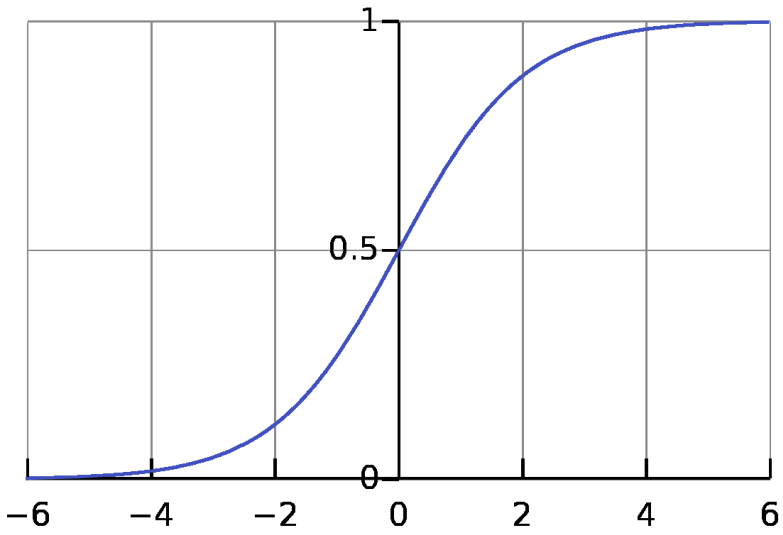
Funkcija ispravljene linearne jedinice definirana je formulom:

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} = \max(0, x) \tag{2-5}$$

Iako *ReLU* pokazuje dobre rezultate, ima i nekih nedostataka, od kojih je najizraženiji slučaj kada neuron nije aktivan. Tada gradijent iznosi 0, te se zbog toga težine neurona više ne mogu pomaknuti metodom gradijentne optimizacije. Ovaj problem može se riješiti tako da se pristranost postavi na vrlo malu pozitivnu vrijednost. Na taj način će svi neuroni na početku biti aktivirani.

### 2.3.4. Sigmoid funkcija

Sigmoidna funkcija je aktivacijska funkcija koja se često koristi u neuronskim mrežama. Kod sigmoidne funkcije vrijednosti na izlazu mogu biti u intervalu od 0 do 1, što ujedno predstavlja razlog zbog kojeg je ova funkcija među najkorištenijima.




Slika 2.6. *Graf sigmoidne funkcije [3]*

Sigmoid funkcija definirana je formulom:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2-6}$$

Kao što je prikazano na slici 2.6., sigmoidna funkcija je nelinearna, monotono rastuća funkcija, te je derivabilna na cijelom području. Sva ova obilježja čine ju pogodnom za korištenje u neuronskim mrežama za donošenje kompleksnijih odluka.

## 2.4. Učenje neuronske mreže

Najvažnijom karakteristikom neuronskih mreža smatra se sposobnost da uče od okoline. Učenje o okolini se odvija kroz ponavljajući proces podešavanja težina i pristranosti. Svakom iteracijom učenja neuronska mreža trebala bi imati više znanja o okolini. Metode učenja dijele se na metode prema algoritmu učenja i na metode prema paradigmi učenja.

Metode učenja po algoritmu učenja su:

- Učenje korekcijom pogreške
- Hebbovo učenje
- Kompetitivno učenje

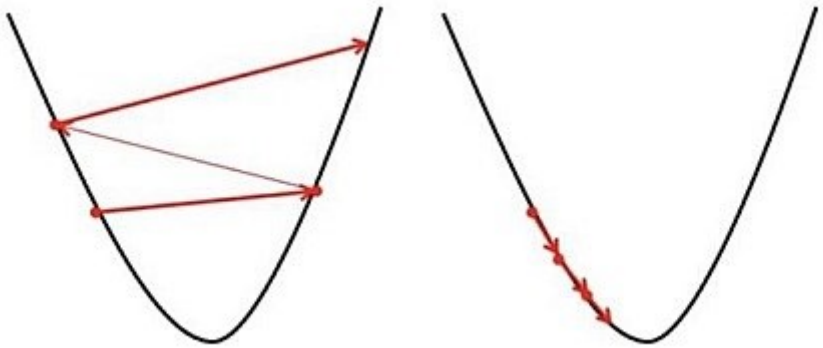
- Boltzmannovo učenje
- Thorndikeovo učenje

Metode po paradigmi učenja su:

- Nadzirano učenje
- Nenadzirano učenje
- Pojačano učenje (učenje podrškom)

### 2.4.1. Gradijentni spust

Gradijentni spust je optimizacijski algoritam koji se koristi za pronalazak globalnog minimuma funkcije. Ovaj algoritam traži težinske koeficijente neuronskoj mreži koji najviše odgovaraju modelu podataka koji se koristi. Svakim korakom vektor težine se mijenja u smjeru najvećeg spusta niz plohu pogreške. Postupak iteracije se provodi sve do dostizanja globalnog minimuma pogreške. Kriteriji konvergencije su dostizanje definiranog broja iteracija ili stagnacija u promjeni funkcije. Stopa učenja ne treba imati preveliku vrijednost, ali niti premalu jer može doći do toga da postupak oscilira ili divergira. U slučaju da funkcija nije striktno konveksna, postoji mogućnost da postupak ostane u lokalnom minimumu što ujedno predstavlja najvećih problema ovog optimizacijskog algoritma.



Slika 2.7. *Gradijentni spust sa velikim korakom (lijevo) i malim korakom (desno) [4]*

Još neki od problema gradijentnog spusta su moguća spora konvergencija ka minimumu i to što postoji mogućnost da se globalni minimum ne pronađe. Zbog toga, kao alternativa gradijentnom

spustu se koristi stohastički gradijentni spust, jer se kod njega izračun gradijenta obavlja nakon svakog uzorka i odmah ispravlja težine, za razliku od običnog gradijentnog spusta koji računa gradijent na temelju svih primjera i nakon toga radi korekciju težinskih koeficijenata.

### 2.4.2. Stohastički gradijentni spust

Stohastički gradijentni spust je manje zahtjeva što se tiče računalnih resursa i zbog toga je prikladan za korištenje kod velikih skupova. Osim gradijentnog spusta koriste se još neki postupci, a neki od njih su postupak konjugiranih gradijenata i Newton-Raphsonov postupak. Navedeni postupci su zahtjevniji što se tiče računalnih resursa, ali oni brže konvergiraju. Stohastički gradijentni spust je vrlo moćan optimizacijski algoritam koji se lako programski implementira.

### 2.4.3. Algoritam propagacije unatrag

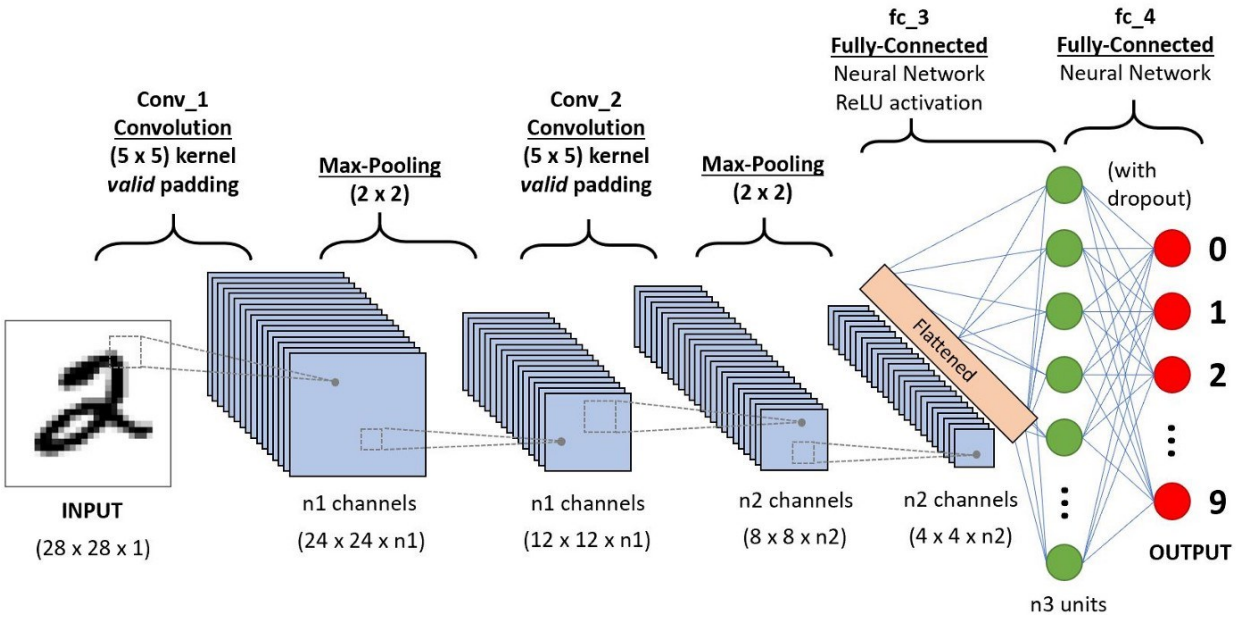
Propagacija unatrag predstavlja metodu za izračun gradijenta ključnog za pronalazak težinskih koeficijenata u algoritmima strojnog učenja. Najčešće se koristi kod dubokih neuronskih mreža, odnosno onih mreža koje sadrže više od jednog skrivenog sloja. Metoda propagacije unatrag zahtijeva da derivacija funkcije gubitka u odnosu na izlaz bude poznata. Navedeno znači da se metoda uglavnom koristi kod nadgledanog učenja. Kod acikličkih neuronskih mreža, u prvom koraku inicijaliziraju se težinski koeficijenti. Koeficijentima se često pridjeljuju vrijednosti slučajno odabirih realnih brojeva koji su vrijednošću blizu 0, ali su različiti od 0. Na ulazni sloj dolazi prvi uzorak, a zatim prolazeći uzorkom od ulaznog sloja kroz skrivene slojeve i pripadne težinske koeficijente dolazi se do izlaznog sloja u kojem dobivamo predikciju. Budući da znamo željeni rezultat jer se radi o nadgledanom učenju, na temelju predikcije i stvarnog rezultata vraća se natrag do ulaznog sloja. Prilikom povratka, ažuriranje težinskih koeficijenata se vrši u isto vrijeme. Za stohastički gradijentni spust ovaj postupak se ponavlja dok ne bude više uzoraka za treniranje. Kod običnog gradijentnog spusta razlika je to što se ažuriranje težina vrši nakon obrade svih uzoraka. Jedan prolazak između svih uzoraka zove se epoha. Poželjno je obaviti treniranje neuronske mreže u više epoha.

# 3. KONVOLUCIJSKE NEURONSKE MREŽE

Konvolucijske neuronske mreže se od drugih neuronskih mreža razlikuju po svojim vrhunskim performansama sa ulazima slike, govora ili audio signala. Glavne vrste slojeva su:

- Konvolucijski sloj
- Sloj sažimanja
- Potpuno povezan sloj

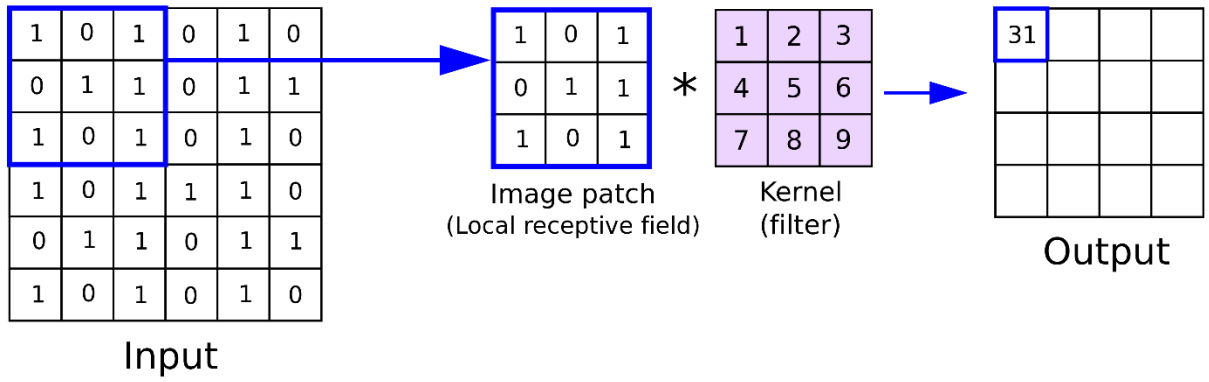
Konvolucijski sloj prvi je sloj konvolucijske mreže. Sa svakim slojem konvolucijska neuronska mreža postaje sve složenija, identificirajući veće dijelove slike. Raniji se slojevi usredotočuju na jednostavne značajke, kao što su boje i rubovi. Kako slikovni podatci prolaze kroz slojeve konvolucijske neuronske mreže, mreža počinje prepoznavati veće elemente ili oblike objekta sve dok u potpunosti ne izvrši identifikaciju željenog objekta.




Slika 3.1. *Primjer konvolucijske neuronske mreže [5]*

## 3.1. Konvolucijski sloj

Konvolucijski sloj je jezgra građevnog materijala konvolucijske neuronske mreže i tu se događa većina izračuna. Zahtjeva nekoliko komponenti, a to su ulazni podaci, filter i mapa značajki. Pretpostavimo da će ulaz biti slika u boji, koja je predstavljena kao matrica piksela u 3D. To znači da ulaz ima tri dimenzije - visinu, širinu i dubinu - koje odgovaraju RGB-u na slici. Imamo i detektor značajki, također poznat kao jezgra ili filter, koji se kreće po receptivnim poljima slike i provjerava je li tražena značajka prisutna. Opisani proces poznat je kao konvolucija. Konvolucija predstavlja matematičku funkciju nastalu integriranjem umnoška dvije funkcije po intervalu njihove definicije gdje su te funkcije ravnopravne tako da svaka infinitezimalna promjena jedne funkcije utječe na drugu funkciju u cijelome intervalu definicije. Detektor značajki je dvodimenzionalni (2D) niz težina, koji predstavlja dio slike. Iako se mogu razlikovati po veličini, veličina filtra je tipično matrica 3x3. To također određuje veličinu receptivnog polja. Filtar se zatim primjenjuje na područje slike, a skalarni umnožak izračunava se između ulaznih piksela i filtra. Taj se skalarni umnožak zatim unosi u izlazni niz. Nakon toga, filter se pomiče, ponavljajući postupak sve dok jezgra ne prođe cijelu sliku. Konačni izlaz iz serije skalarnih umnožaka iz ulaza i filtera naziva se mapa značajki, aktivacijska mapa ili konvolvirana značajka.

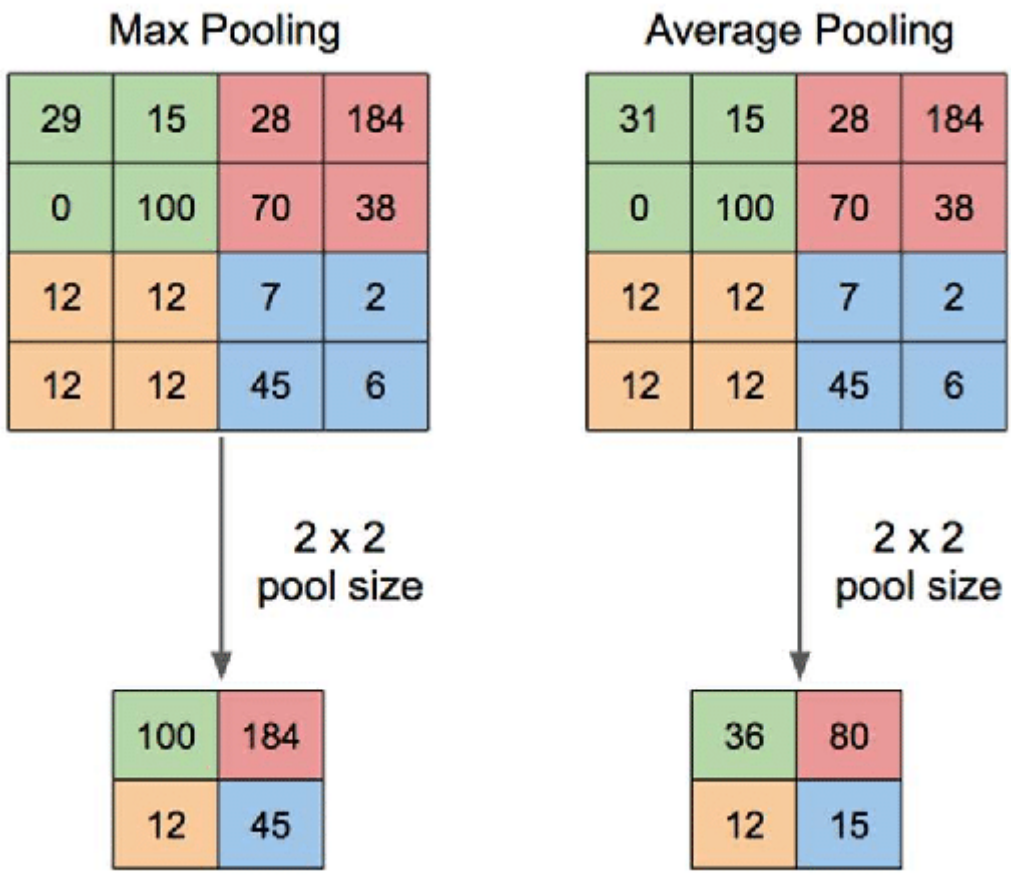



Slika 3.2. *Princip rada konvolucijskog sloja [6]*

## 3.2. Sloj sažimanja

U slojevima sažimanja smanjuje se dimenzionalnost na način da se smanjuje broj parametara u ulazu. Slično kao kod konvolucijskog sloja, operacija sažimanja pomiče filtar po cijelom ulazu, ali razlikuje se po tome što ovaj filter nema nikakvu težinu. Umjesto toga, jezgra primjenjuje funkciju agregacije na vrijednosti unutar receptivnog polja i na taj način popunjava izlazni niz. Vrste sažimanja koje se najčešće koriste su:

- Sažimanje po maksimalnoj vrijednosti- Pomicanjem filtera po ulazu bira se piksel s najvećom vrijednošću za slanje u izlazni niz. Ovakav pristup se više koristi u odnosu na sažimanjem po prosječnoj vrijednosti.
- Sažimanje po prosječnoj vrijednosti- Pomicanjem filtera po ulazu računa se prosječna vrijednost unutar receptivnog polja za slanje u izlazni niz.




Slika 3.3. *Usporedba sažimanja po maksimalnoj i sažimanja po prosječnoj vrijednosti [7]*

Iako se velik broj podataka izgubi u sloju sažimanja u konvolucijskoj neuronskoj mreži, on također donosi brojne prednosti. Pomaže kod smanjenja složenosti, poboljšava učinkovitost i ograničava rizik od prenaučenosti.

## 3.3. Potpuno povezani sloj

Naziv potpuno povezani sloj najbolje objašnjava o kakvom sloju se radi. Izlazni sloj nije direktno povezan s vrijednostima piksela ulazne slike u djelomično povezanim slojevima, dok se u potpuno povezanom sloju, svaki čvor u izlaznom sloju direktno povezuje sa čvorom u prethodnom sloju.

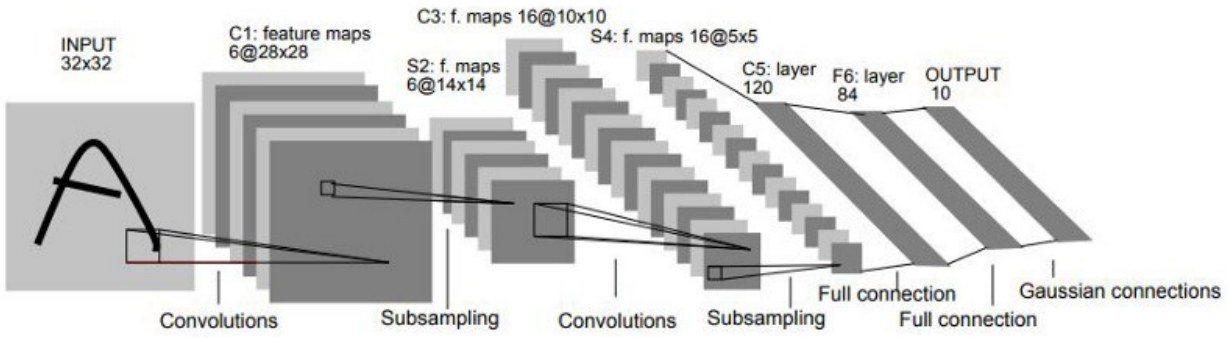
U potpuno povezanom sloju obavlja se zadatak klasifikacije na temelju značajki izdvojenih kroz prethodne slojeve i njihovih različitih filtera. Za razliku od konvolucijskog sloja i sloja sažimanja koji koriste *ReLu* aktivacijsku funkciju, kod potpuno povezanih slojeva najčešće se koristi aktivacijska funkcija *softmax* za odgovarajuću klasifikaciju ulaza, stvarajući vjerojatnost od 0 do 1.

## 3.4. Primjeri konvolucijskih neuronskih mreža

Ovo poglavlje opisuje najpoznatije konvolucijske neuronske mreže i njihovu arhitekturu. Bit će objašnjeno kako mreža radi kao i raspored slojeva u mreži.

### 3.4.1. LeNet-5

LeNet-5 je jedna od prvih konvolucijskih neuronskih mreža u kojoj se koristila metoda dubokog učenja. Modelirao ju je Yann LeCun i suradnici 1989. godine, a koristila se za prepoznavanje rukom pisanih znamenaka.




Slika 3.4. *Arhitektura LeNet-5 mreže [8]*

Na slici 3.4. može se vidjeti da je ulaz u mrežu slika dimenzija 32x32 piksela na kojoj se nalazi rukom napisana znamenka. Nakon primjene konvolucijskog sloja dobiva se 6 izlaznih mapa značajki koje su dimenzija 28x28. Mape značajki tada prolaze kroz sloj sažimanja u kojemu se dimenzije smanjuju na 14x14. Nakon toga opet slijedi konvolucijski sloj koji generira 16 mapa značajki koje su dimenzija 10x10 i još jedan sloj sažimanja koji mape značajki svoji na dimenzije 5x5. Zatim se primjenjuju 2 potpuno povezana sloja, jedan od 120 i drugi 84 neurona. Zadnji sloj sadrži 10 neurona koji označavaju klasu znamenaka od 0 do 9.

### 3.4.2. VGG

VGG arhitektura osmišljena je kako bi se povećala dubina konvolucijske neuronske mreže u svrhu povećanja performansi modela. Postoje VGG16 i VGG19. Razlika je u tome što se VGG16 sastoji od 16 slojeva, dok se VGG19 sastoji od 19 slojeva. Mreža su predstavili Andrew Zisserman i Karen Simonyan u znanstvenom radu naziva "*Very Deep Convolutional Networks for Large-Scale Image Recognition*.". Trenirana je nekoliko tjedana koristeći grafičke kartice Nvidia Titan Black. Kao set podataka za treniranje korištena je ImageNet baza podataka, koja sadrži više od 14 miliona slika podijeljenih u oko 1000 klasa.

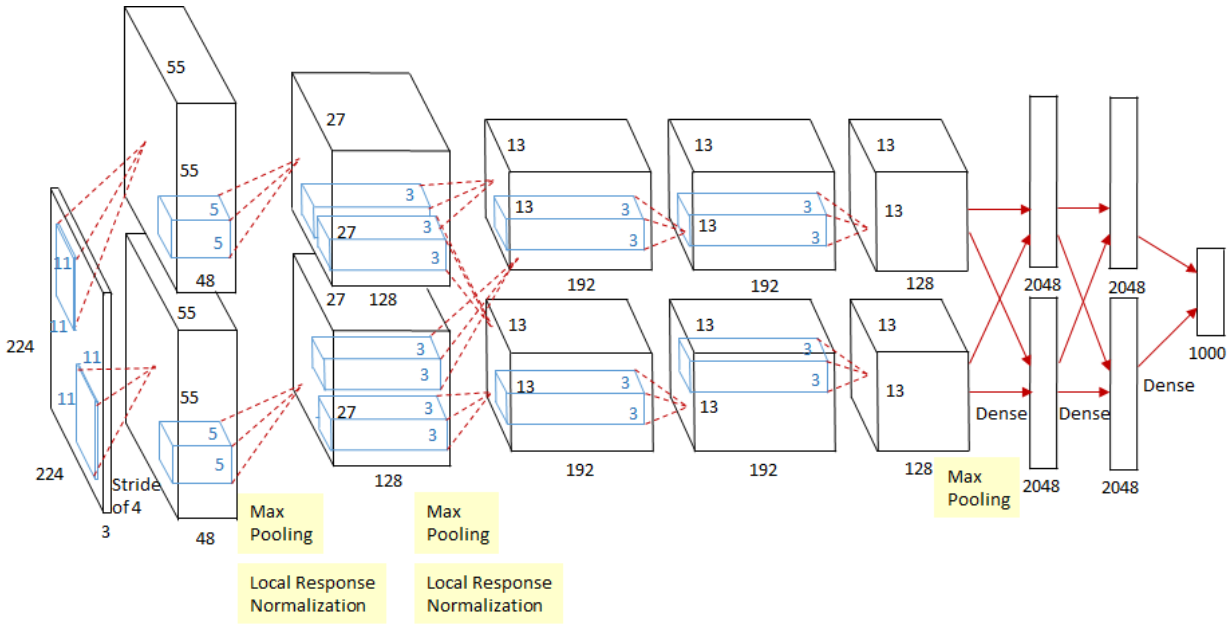
Slika 3.5. *Usporedba VGG16 i VGG19 arhitekture [10]*

VGG19 ima 3 sloja više u odnosu na VGG16 mrežu. Kao ulaz u mrežu uzima se slika dimenzija 224x224. Nakon toga slijedi konvolucijski sloj u kojem se na sliku primjenjuje filter dimenzija 3x3. Kao aktivacijska funkcija koristi se *ReLU*. U VGG mreži postoje 3 potpuno povezana sloja, prva dva sastoje se od 4096 kanala, dok se treći sastoji od 1000 kanala, po 1 za svaku klasu.

### 3.4.3. AlexNet

AlexNet je arhitektura konvolucijske neuronske mreže razvijena 2012. godine od strane Alexa Krizhevsky, Ilye Sutskevera i Geoffreya Hintona. Na natjecanju „*ImageNet Large Scale Visual Recognition Challenge*“ AlexNet arhitektura ostvarila je stopu pogreške 15,3%. Za usporedbu, drugoplasirana arhitektura imala je pogrešku od 26,1%. Prilikom treniranja mreže korištena je

Nvidia GTX 580 grafička kartica, koja je imala 3 GB memorije, ali zbog veličine, odnosno dubine mreže, bilo je potrebno podijeliti posao na dvije grafičke kartice.




Slika 3.6. *Arhitektura AlexNet konvolucijske neuronske mreže [9]*

Mreža se sastoji od 5 konvolucijskih slojeva i na kraju mreže se nalaze još 3 potpuno povezana sloja. *ReLU* aktivacijska funkcija koristi se poslije svakog konvolucijskog sloja i potpuno povezanog sloja. Ulazna slika je dimenzija 224x224x3. Na ulaznu sliku primjenjuje se filter dimenzija 11x11. AlexNet ima 60 milijuna parametara, a kako bi se izbjegla pretreniranost, koristi se isključivanje neurona.

# 4. PROGRAMSKO RJEŠENJE ZA PRIJENOS OBILJEŽJA SLIKE

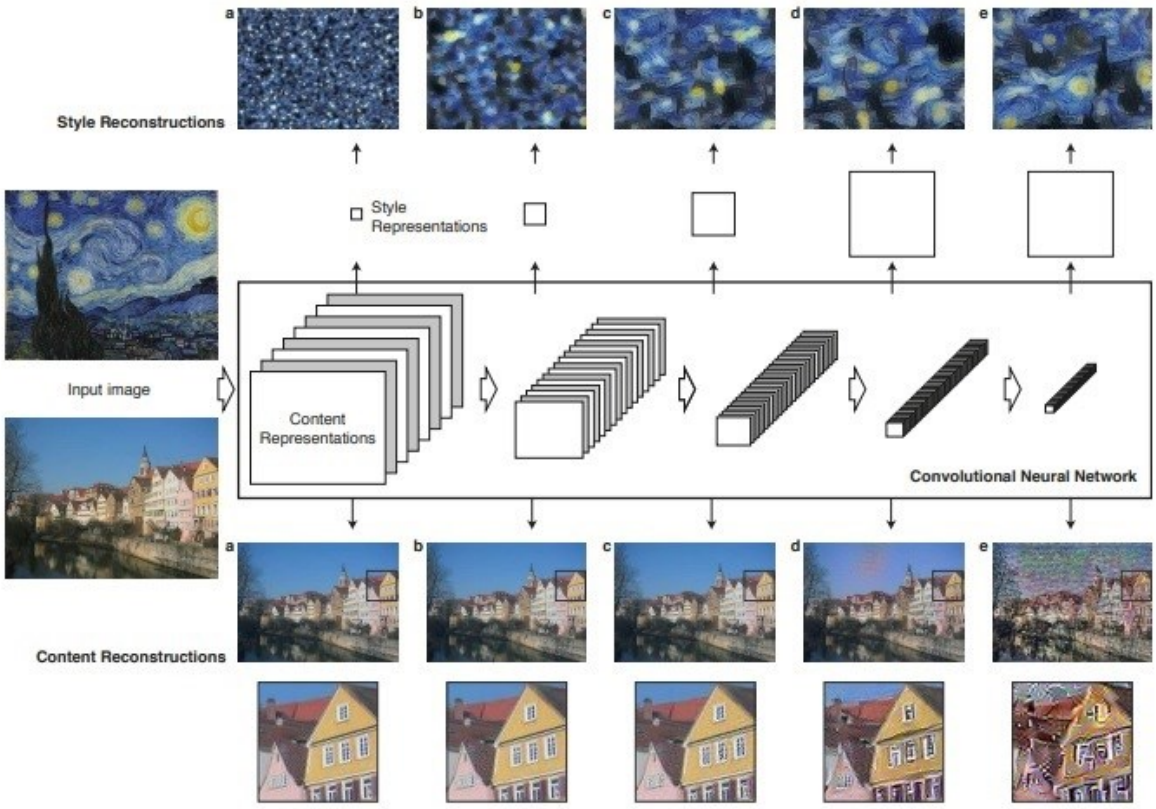
Kroz ovo poglavlje detaljnije će biti opisan praktični zadatak ovog rada, a to je prijenos obilježja slike korištenjem konvolucijske neuronske mreže. Bit će opisana teorijska podloga izrađenog rješenja, informacije o korištenim tehnologijama kao i detaljan opis programskog koda.

## 4.1. Neuronski prijenos stila

Neuronski prijenos stila pokazao je vrlo zanimljive rezultate koji omogućuju nove oblike manipulacije slikama. To je optimizacijska tehnika koja se koristi na način da se uzimaju dvije slike, jedna je slika sadržaja sa koje se uzima obilježje, dok druga slika predstavlja referencu stila koji će se primijeniti. Cilj je da se te dvije slike spoje u jednu, novu sliku koja će poprimiti obilježja slike sadržaja, ali će i poprimiti stil slike koja je služila kao referenca stila. Navedeno se postiže korištenjem konvolucijske neuronske mreže, gdje se iz određenih slojeva mreže ekstrahiraju značajke povezane sa obilježjem, odnosno stilom slike.

Prvi algoritam korišten za neuronski prijenos slike opisan je u znanstvenom radu „*A Neural Algorithm of Artistic Style*“, originalno objavljenom na ArXiv repozitoriju 2015. godine. Autori znanstvenog rada su Leon A. Gatys, Alexander S. Ecker i Matthias Bethge.

U njihovom radu korištena je VGG arhitektura konvolucijske neuronske mreže koja je predtrenirana za prepoznavanje objekata koristeći ImageNet bazu podataka. Kada se konvolucijske neuronske mreže treniraju za prepoznavanje objekata, tada razvijaju prikaz slike koja informacije o objektu čini sve eksplicitnijim u hijerarhiji procesiranja. Zbog toga, kroz hijerarhiju procesiranja mreže, ulazna slika se pretvara u prikaz koji više odgovara stvarnom sadržaju slike nego o točnim vrijednostima svakog piksela.




Slika 4.1. *Prikaz slike kroz slojeve konvolucijske neuronske mreže [11]*

Na slici 4.1. ulazna slika prikazana je kao skup filtriranih slika u svakoj fazi procesiranja kroz konvolucijsku neuronsku mrežu. Dok se broj filtera povećava kroz postupak procesiranja, veličina filtriranih slika se smanjuje, za što bi primjer bio sažimanje po maksimalnoj vrijednosti, što dovodi do smanjenja broja neurona po sloju u mreži. Slojevi koji sadržavaju značajke stila slike su *'conv1_1'*, *'conv2_1'*, *'conv3_1'*, *'conv4_1'* i *'conv5_1'* sloj. Korištene su značajke iz 16 konvolucijskih slojeva i 5 slojeva sažimanja. Potpuno povezani slojevi nisu korišteni u radu. [11] Ukupni gubitak sadržaja definiran je formulom:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2 \tag{4-1}$$

Gdje je : - $\vec{p}$ - originalna slika

- $\vec{x}$ - generirana slika

- $l$ - sloj u konvolucijskoj neuronskoj mreži

- $F^l, P^l$ - prikaz značajki slika u sloju $l$.

Ukupni gubitak stila definiran je formulom:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} \omega_l E_l \tag{4-2}$$

Gdje je: - $\vec{a}$ - slika stila

- $E_l$ - doprinos sloja ukupnom gubitku

- $\omega_l$ - težinski faktor doprinosa svakog sloja ukupnom gubitku stila

Doprinos sloja ukupnom gubitku stila računa se po formuli:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{4-3}$$

Gdje je : - $N_l$ - broj mapi značajki

- $M_l$ - veličina mape značajke

- $G_{ij}, A_{ij}$ - prikaz stila originalne i generirane slike u sloju $l$

Za generiranje slike koja je spoj slike sadržaja i slike stila minimiziraju se udaljenost bijelog šuma iz prikaza sadržaja slike u jednom sloju mreže i prikaz stila slike u slojevima mreže. Funkcija za funkciju gubitka koja se minimizira je:

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \tag{4-4}$$

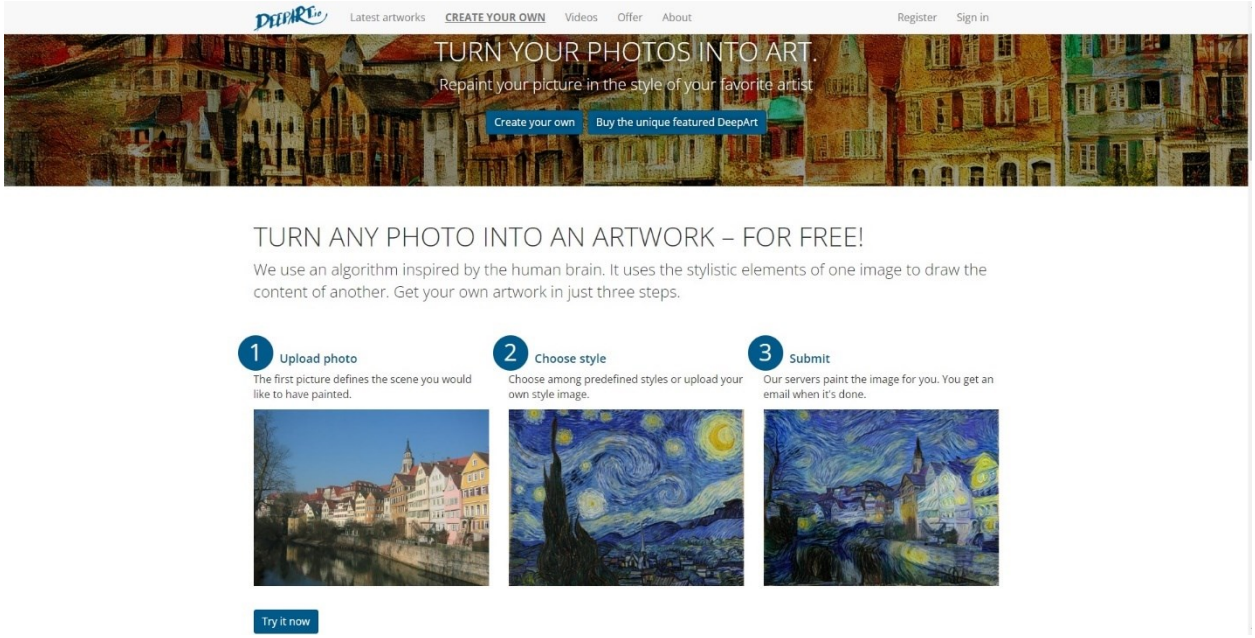
Gdje je: - $\vec{p}$ - slika sadržaja

- $\vec{a}$ - slika stila

- $\vec{x}$ - generirana slika

- α, β - težinski faktori za sadržaj i stil

Njihovo rješenje može se isprobati putem web aplikacije koja se nalazi na linku https://deepart.io gdje se putem forme odabire slika na koju se želi primijeniti određeni stil, te slika sa koje će se uzeti stil i primijeniti na prvu sliku. Izgleda web aplikacije prikazan je na slici 4.2.




Slika 4.2. *Prikaz deepart.io web aplikacije*

## 4.2. Python

Python je programski jezik kojega je stvorio Guido van Rossum 1991. godine. S razvojem Pythona Guido van Rossum počeo je 1989. godine te ga je objavio 1991. Python podržava proceduralni, objektno-orijentirani i funkcionalni stil programiranja, koji daje programerima slobodu da sami odaberu pristup koji odgovara njihovom problemu. Baš ta fleksibilnost čini Python sve popularnijim. Najviše se koristi na Linuxu, ali postoje verzije i za druge operacijske sustave. Python karakterizira jasna i jednostavna sintaksa te je iz tog razloga on savršen za početnike u programiranju. Međutim, zbog svoje jednostavnosti i prilagodljivosti ga koriste i velike kompanije poput Googlea, Yahooa, NASA-e i sl. Programski jezik Python je jezik visoke razine što bi značilo da je bliži govornom nego strojnom jeziku. Programi koji su pisani na Python programskom jeziku su jednostavniji i čitljiviji te se stoga povećava produktivnost programera. Za konstantni razvoj jezika zadužena je neprofitna organizacija Python Software Foundation (PSF) koja se bavi normiranjem koda te prikupljanjem donacija. Python je besplatan i dostupan svima. Koristi se za doista mnogo namjena, koriste ga znanstvenici za analizu gena, u matematičkim znanostima koristi se za simulacije, za umjetnu inteligenciju i slično. Za izradu programskog rješenja ovog rada korišten je Python biblioteka PyTorch. PyTorch se najčešće koristi u području računalnog vida, te obrade prirodnog jezika.

## 4.3. Programsko rješenje

Prvi dio programskog rješenja je uvoz potrebnih biblioteka

```
import torch
import torchvision.transforms as transforms
from PIL import Image
import torch.optim as optim
from torchvision.utils import save_image
from torchsummary import summary
from VGG import *
import click
import os
```

Slika 4.3. *Kod primjer*

U sljedećem dijelu koda postavlja se uređaj na kojem će se obavljati evaluacija modela. Ukoliko je dostupna grafička kartica, ona će biti korištena, a u suprotnom koristit će se centralna

procesorska jedinica. Zatim slijede „click“ komande pomoću kojih se mogu postavljati parametri prilikom pokretanja programa u konzoli. Neki od parametara koji se ovim putem mogu postaviti su broj epoha, stopa učenja, težine obilježja, odnosno stila te putanje do slike sadržaja i slike stila. Također, prikazana je i funkcija koja služi za učitavanje slika. Ona prilikom učitavanja slike postavlja dimenzije na 512x512, te sliku pretvara u tenzor. Ostatak koda odnosi se na sistemski dio gdje se uzima naziv slike obilježja i slike stila, te se kreira novi direktorij (ukoliko već ne postoji) u kojem će se spremati novo generirane slike.

```
device=torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")

@click.command()
@click.option('--epoch', default=1000, help='Number of epochs')
@click.option('--lr', default=0.005, help='Learning rate')
@click.option('--alpha', default=0.01, help='Content weight')
@click.option('--beta', default=1000, help='Style weight')
@click.option('--content', default='Images/castle.jpg', help='Path to content image')
@click.option('--style', default='Images/neb.jpg', help='Path to style image')


def main(epoch,lr,alpha,beta,content,style):

    def image_loader(path):
        image=Image.open(path)
        loader=transforms.Compose([transforms.Resize((512,512)),transforms.ToTensor()])
        image=loader(image).unsqueeze(0)
        return image.to(device,torch.float)

    original_image=image_loader(content)
    style_image=image_loader(style)

    basename_without_ext = os.path.splitext(os.path.basename(content))[0]
    basename_style_without_ext = os.path.splitext(os.path.basename(style))[0]

    dir_name = "output_" + str(basename_without_ext) + "_" + str(basename_style_without_ext)
    isExist = os.path.exists(dir_name)

    if not isExist:

      os.makedirs(dir_name)
      print("The new directory is created!")
```

Slika 4.4. *Kod primjer*

Nadalje, tenzor slike koja će se koristiti kao slika sadržaja kopira se u novu varijablu te se postavlja automatsko praćenje i računanje gradijenata za taj tenzor. Osim toga, u prikazanom isječku koda prikazane su funkcije u kojima se radi izračun gubitka sadržaja, gubitka stila kao i ukupnog gubitka sa za to predviđenim formulama. Navedeni gubitci koriste se prilikom generiranja nove slike, odnosno definiraju sadržaj koji će slika sadržavati, te primjenjuju stil druge slike na taj sadržaj.

```
generated_image=original_image.clone().requires_grad_(True)

def content_loss(gen_feat,orig_feat):
    loss=torch.mean((gen_feat-orig_feat)**2)
    return loss

def style_loss(gen,style):
    batch,channel,height,width=gen.shape

    G=torch.mm(gen.view(channel,height*width),gen.view(channel,height*width).t())
    A=torch.mm(style.view(channel,height*width),style.view(channel,height*width).t())

    loss=torch.mean((G-A)**2)
    return loss

def total_loss(gen_features, orig_feautes, style_featues):
    s_loss=0
    c_loss=0
    for generated,content,style in zip(gen_features,orig_feautes,style_featues):
        c_loss+=content_loss(generated,content)
        s_loss+=style_loss(generated,style)

    loss=alpha*c_loss + beta*s_loss
    return loss
```

Slika 4.5. *Kod primjer*

Nakon izračuna gubitaka, učitava se model pomoću VGG() funkcije, zatim se postavlja uređaj na kojem će se izvršavati evaluacija modela. Uređaji na kojima se izvršava evaluacija su procesor ili grafička kartica. U sklopu ovog rada, za evaluaciju modela korištena je grafička kartica GTX 1050 Ti 4GB. Kao optimizacijski algoritam korišten je LBFGS. Limited-memory BFGS (Broyden–Fletcher–Goldfarb–Shanno algoritam) je optimizacijski algoritam iz porodice Quasi-Newtonovih metoda koje se koriste za pronalaženje nula ili lokalnih minimuma i maksimuma funkcija. LBFGS

aproksimira BFGS algoritam koristeći ograničenu količinu računalne memorije, te je jako popularan za estimaciju parametara u strojnom učenju. Nakon definiranja algoritma optimizacije, slijedi treniranje modela kroz određeni broj epoha. Predefinirani broj epoha je 1000, no prilikom pokretanja programa moguće je unijeti željeni broj epoha. U svakoj epohi značajke generirane, originalne i slike stila koriste se za izračun gubitka u svrhu generiranja nove slike. Pomoću funkcije zero_grad() čiste se gradijenti iz posljednjeg koraka, dok funkcija backward() računa derivat gubitka koristeći algoritam propagacije unatrag. Funkcija step() omogućuje optimizatoru da napravi korak ovisno o gradijentima parametara. Nakon svakih 10 epoha generirana slika se pohranjuje kako bi se mogla pratiti promjena stila slike kroz epohe.

```
model=VGG().to(device).eval()
print(model)

optimizer=optim.LBFGS([generated_image],lr=lr)

for e in range (epoch):
    print(str(e) + "/" + str(epoch))

    def closure():
        generated_features=model(generated_image)
        original_features=model(original_image)
        style_features=model(style_image)

        t_loss=total_loss(generated_features, original_features, style_features)
        optimizer.zero_grad()
        t_loss.backward()
        return t_loss

    optimizer.step(closure)

    if(not (e%10)):
        print(str(e) + " epoch generated image saved.")
        save_image(generated_image,dir_name + "/generated"+str(e)+".jpg")
```

Slika 4.6. *Kod primjer*

Zadnji isječak koda prikazuje klasu VGG unutar koje se učitava predtrenirani VGG19 konvolucijske neuronske mreže. Navedena mreža sastoji se od 19 slojeva, te je trenirana na ImageNet bazi podataka koja sadrži preko milijun slika za treniranje. Iz modela je potrebno

ekstrahirati značajke slike koje sadržavaju stil slike. Slojevi koji sadržavaju značajke stila su '*conv1_1*', '*conv2_1*', '*conv3_1*', '*conv4_1*' i '*conv5_1*' sloj. Prolaskom kroz konvolucijsku neuronsku mrežu navedeni slojevi će se izdvojiti u posebno polje te će se vrijednosti iz tih slojeva, odnosno značajke koristiti prilikom generiranja nove slike.

```
import torch.nn as nn
import torchvision.models as models

class VGG(nn.Module):
    def __init__(self):
        super(VGG,self).__init__()
        # (0): block1_conv1 - Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        # (5): block2_conv1 - Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        # (10): block3_conv1 - Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        # (19): block4_conv1 - Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        # (28): block5_conv1 - Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))

        self.style_layers= ['0','5','10','19','28']
        self.model=models.vgg19(pretrained=True).features


    def forward(self,x):
        features=[]
        for layer_num,layer in enumerate(self.model):
            x=layer(x)
            if (str(layer_num) in self.style_layers):
                features.append(x)
        return features
```

Slika 4.7. *Kod primjer*

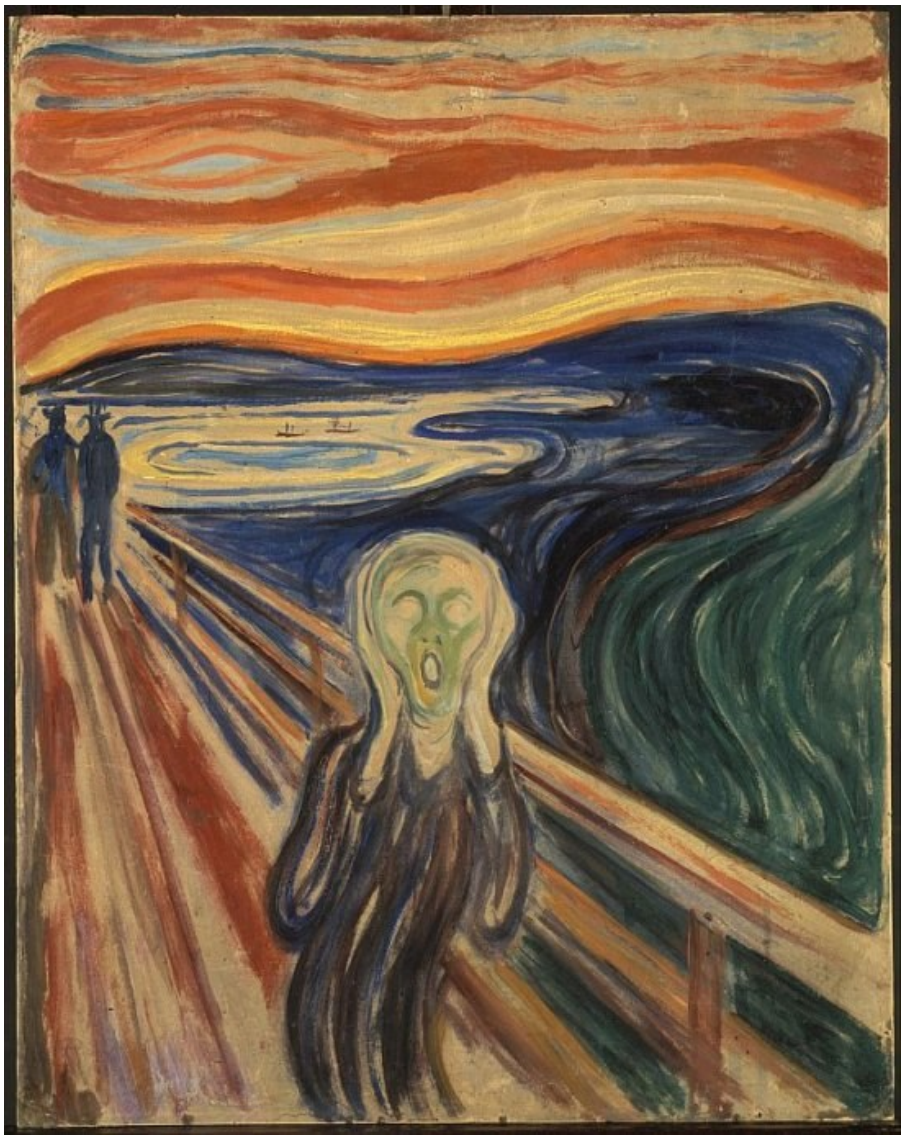
# 5. ANALIZA REZULTATA

Rezultat neuronskog prijenosa stila je generirana slika koja je zapravo spoj sadržaja jedne slike i stila druge slike. U ovom poglavlju bit će prikazan primjer korištenja programskog rješenja za neuronski prijenos stila. Bit će prikazane slike koje su korištene za generiranje nove slike, odnosno slika sadržaja i slika stila, zatim će biti prikazan razvoj generirane slike kroz epohe kao i graf ukupnog gubitka kroz epohe.

Na slici 5.1. prikazana je slika koja je korištena kao slika sadržaja. Slika prikazuje dvorac Neuschwanstein u Njemačkoj, a odabrana je iz razloga što sadrži dosta različitih tekstura i boja na koje će biti moguće primijeniti stil druge slike. Kroz poglavlje su prikazana tri primjera generiranih slika sa različitim stilovima.

Slika 5.1. *Dvorac Neuschwanstein [14]*

Slika 5.2. prikazuje umjetničko djelo koje je korišteno kao slika stila u prvom primjeru. Radi se o najpoznatijem djelu norveškog ekspresionista Edvarda Muncha, naziva „Krik“ iz 1893. godine. Slika prikazuje užasnutu figuru koja stoji na mostu nasuprot crvenog neba. Ova slika odabrana je zbog svoje popularnosti i zbog specifičnog stila koji je lako prepoznatljiv.



Slika 5.2. *„Krik“, Edvard Munch [15]*

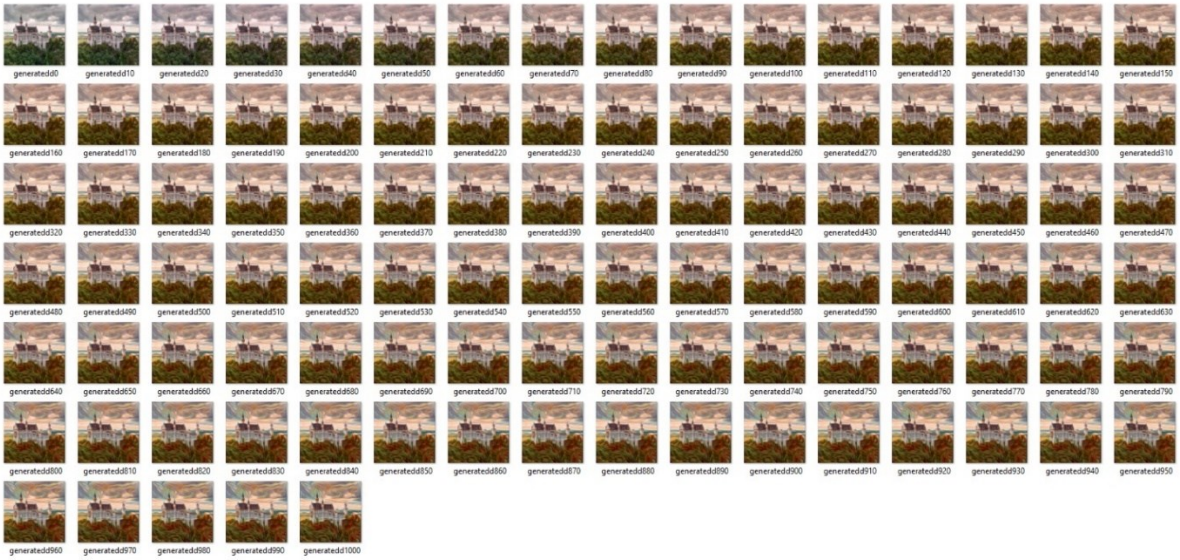
Na slici 5.3. prikazana je generirana slika nakon postupka neuronskog prijenosa stila. Na sliku sadržaja, koja je prikazana na slici 5.1. primijenjen je stil slike prikazane na slici 5.2. Kao što se može vidjeti, slika je zadržala izvorni sadržaj, ali boje i teksture je preuzela s umjetničkog djela koje se koristilo kao referenca za stil.

Parametri koji su korišteni kod evaluacije modela su:

- Broj epoha - 1000
- Stopa učenja - 0.005
- Optimizator - LBFGS
- Težina sadržaja - 1
- Težina stila - 10000

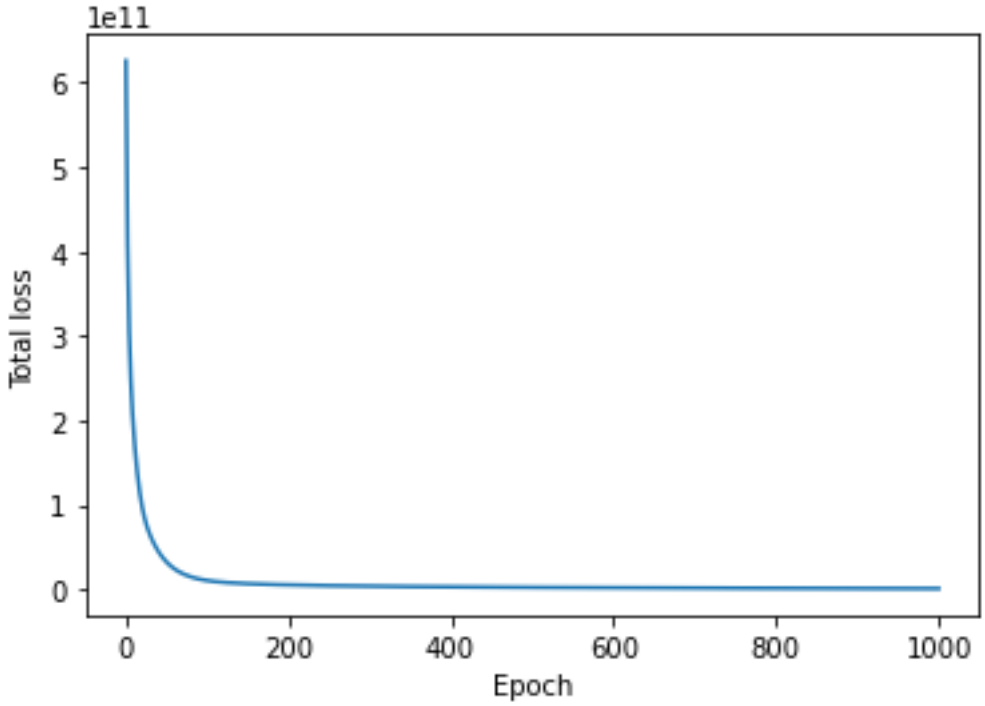
Slika 5.3. *Generirana slika*

Kao što je spomenuto u dijelu programskog rješenja, nakon svakih 10 epoha generirana slika se sprema, kako bi se pratio napredak kroz epohe. Na slici 5.4. prikazane su generirane slike kroz epohe.



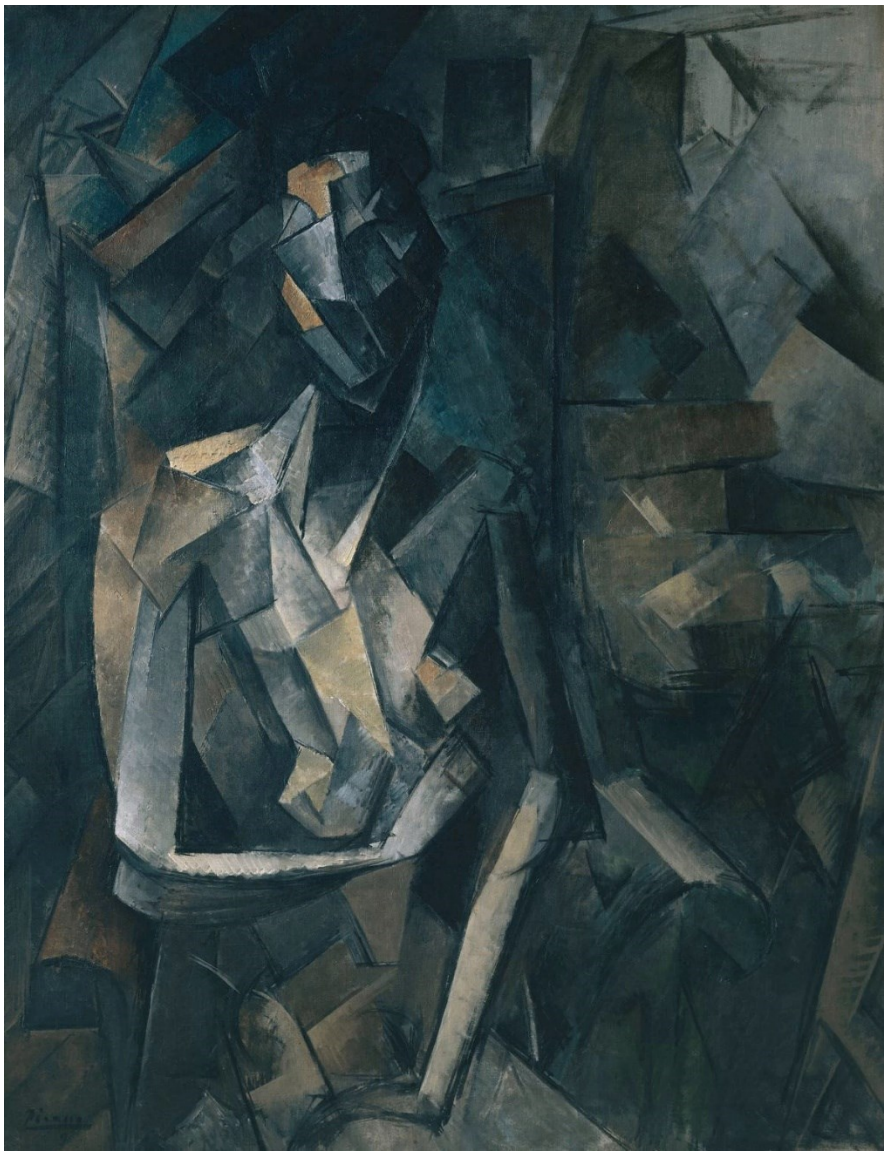
Slika 5.4. *Izgled generirane slike kroz epohe*

Na slici 5.5. prikazana je promjena ukupnog gubitka kroz epohe. Može se primijetiti da se ukupni gubitak značajnije mijenja kroz prvih 200 epoha, dok se kroz ostale mijenja puno sporije i za manji iznos. Navedeno je isto tako vidljivo na generiranim slikama kroz epohe, gdje se može vidjeti da slika najveće promjene ostvari kroz prvih 200 epoha.




Slika 5.5. *Ukupni gubitak kroz epohe*

U drugom primjeru kao slika stila korišteno je umjetničko djelo Pabla Picassa naziva „*Figure dans un Fauteuil"*. Slika je naslikana 1910. godine, a na njoj je prikazana figura koja sjedi u fotelji s visokim naslonom. Slika predstavlja rani primjer kubizma. Kubizam je smjer slikarstva koji proizlazi iz tvrdnje Paula Cezannea da sve što se nalazi u prirodi može biti likovno prikazano pomoću osnovnih oblika kocke, kugle, stošca i valjka. Prvi radovi izloženi su 1907.godine, a izložili su ih Georges Braque i Pablo Picasso. U njihovim radovima sve vidljivo na slici je preneseno u stilizirane oblike geometrijskih tijela. Slike je odabrana kako bi se vidjelo hoće li se i koliko uspješno osnovni oblici prenijeti na sliku sadržaja u postupku neuronskog prijenosa stila.



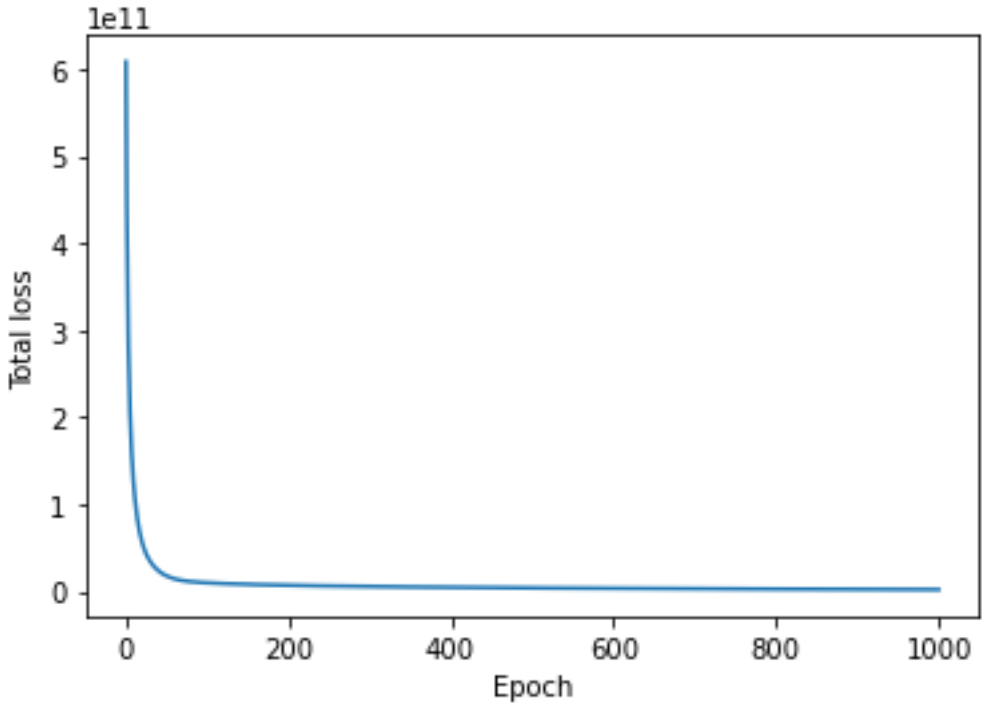
Slika 5.6. „*Figure dans un Fauteuil", Pablo Picasso [21]*

Slika 5.7. prikazuje generiranu sliku u kojoj se slika 5.1. koristila kao slika sadržaja, a slika 5.6. se koristila kao slika stila. Vidljivo je da je generirana slika u većem dijelu poprimila obilježja slike stila, odnosno kubizma. Obilježja kubizma na slici se posebnu primjećuju u donjem dijelu slike, gdje su stabla i u gornjem dijelu, gdje se nalaze oblaci. Također, spektar boja u generiranoj slici se smanjio u odnosu na sliku sadržaja. Kod evaluacije modela korišteni su isti parametri kao u prvom primjeru.

Slika 5.7. *Generirana slika*

Na slici 5.8. prikazan je ukupni gubitak kroz epohe za drugi primjer. Kao u prvom primjeru, veći dio promjena vezanih za teksture na slici događa se kroz prvih 200 epoha. Kroz ostale epohe je smanjenje ukupnog gubitka manje u odnosu na prvih 200 epoha, jer se u njima uglavnom mijenjaju samo boje na slici. Razlika u odnosu na prvi primjer je također ta što se u drugom primjeru ukupni gubitak značajno smanjio već kroz prvih 100 epoha. Razlog je taj što slika stila iz drugog primjera

sadrži manje i pravilnije teksture, te manji spektar boja nego što je slučaj u prvom primjeru. Navedeno također rezultira nešto bržim generiranjem slike iz drugog primjera.




Slika 5.8. *Ukupni gubitak kroz epohe*

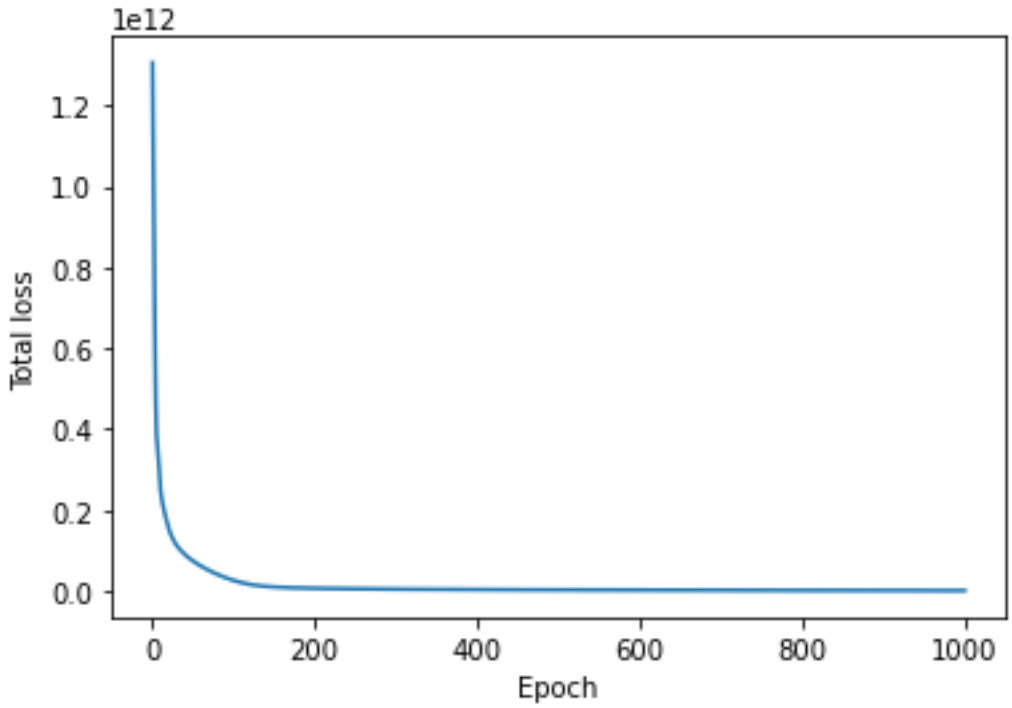
U trećem primjeru kao slika stila korišteno je umjetničko djelo prikazano na slici 5.9. Radi se o umjetničkom djelu naziva „Zvjezdana noć“, slavnom remek-djelu Vincenta van Gogha iz 1889. godine koje pripada postimpresionizmu. Slika prikazuje ogromne čemprese na brdu, polumjesec i zvijezde žute boje, te selo dolini koje na prvi pogled ne dolazi toliko do izražaja pokraj ostalih elemenata slike. Slika je odabrana zbog zanimljivih boja i tekstura, te naravno zbog popularnosti ovog umjetničkog djela.

Slika 5.9. *„Zvjezdana noć“, Vincent van Gogh*

Kao i u prva dva primjera, obilježja slike stila su uspješno prenesena na sliku sadržaja u generiranim slikama. Prilikom evaluacije modela uočeno je da proces prijenosa stila traje duže nego u prethodna dva primjera.

Slika 5.10. *Generirana slika*

Na početku procesa neuronskog prijenosa stila u trećem primjeru ukupni gubitak je veći nego u prva dva primjera. Vrijednost ukupnog gubitka je veća zbog toga što je umjetničko djelo korišteno kao slika stila u trećem primjeru kompleksnije od umjetničkih djela korištenih u prva dva primjera. Slika 5.11. prikazuje graf ukupnog gubitka kroz epohe za treći primjer.



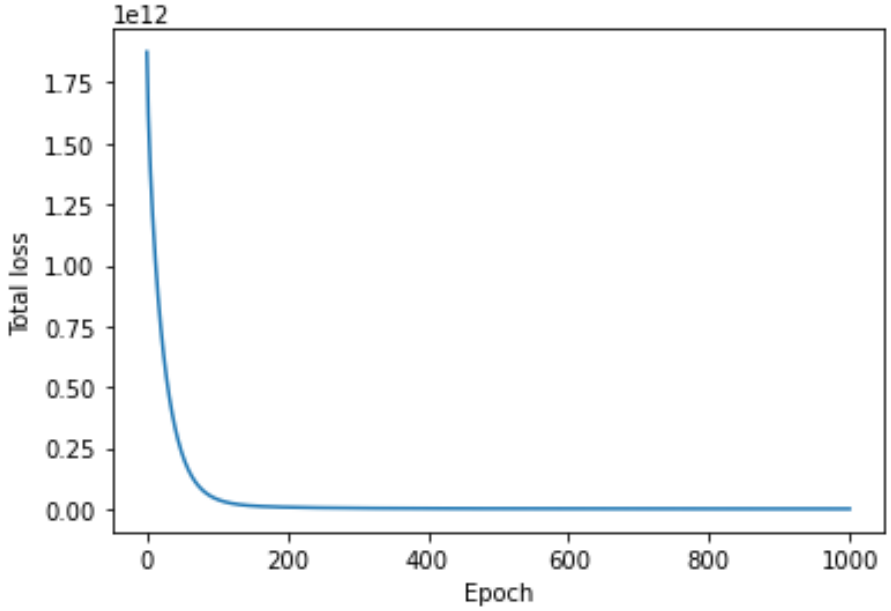

Slika 5.11. *Ukupni gubitak kroz epohe*

U nastavku je prikazano još primjera generiranih slika gdje se za slike stila koriste tri prethodno opisane slike. U ovom slučaju za sliku sadržaja korištena je slika na kojoj je prikazana Tvrđa u Osijeku. Ova slika također sadrži dosta detalja na kojima je moguće primijeniti stil sa slika stila, te nešto veći spektar boja u odnosu na sliku korištenu kao sliku sadržaja u prethodnim primjerima. Nakon prikaza generiranih slika i grafova ukupnih gubitaka napravljena je usporedba rezultata sa rezultatima iz prethodnih primjera i osvrt na brzine izvođenja.

Slika 5.12. *Tvrđa u Osijeku [23]*

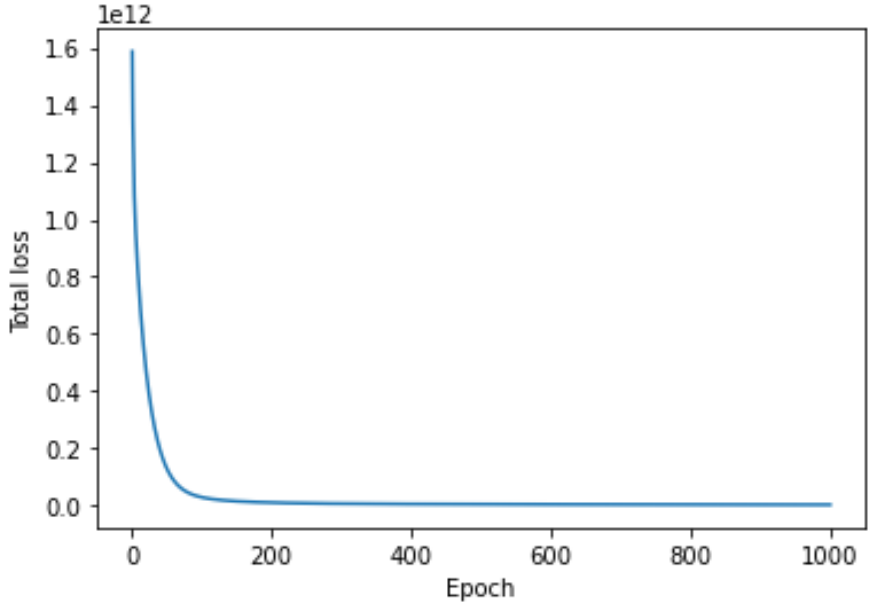
Na slici 5.12. se nalazi slika korištena kao slika sadržaja za generirane slike prikazane u nastavku, a kao slike stila biti će korištene slike 5.2, 5.6. i 5.9.

Slika 5.13. *Generirana slika*



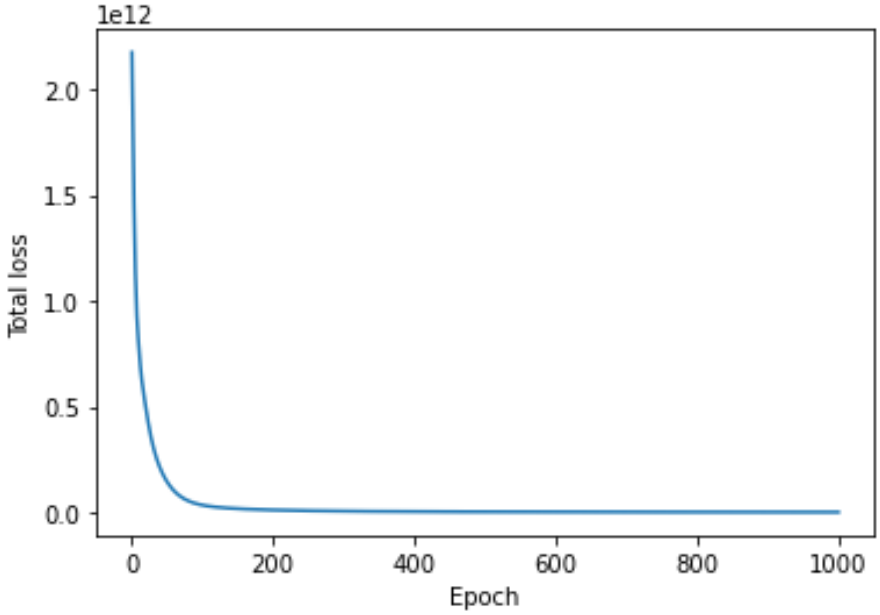

Slika 5.14. *Ukupni gubitak kroz epohe*

Slika 5.15. *Generirana slika*




Slika 5.16. *Ukupni gubitak kroz epohe*

Slika 5.17. *Generirana slika*




Slika 5.18. *Ukupni gubitak kroz epohe*

Kroz prikazane primjere primjećuje se da uz odgovarajuće parametre i kroz optimalan broj epoha model daje vrlo dobre rezultate. Na generiranim slikama se i dalje prepoznaje izvorni sadržaj, dok su teksture i boje na slici slične onima koje se pronalaze u umjetničkim djelima koja su korištena kao slike stila. Što se tiče računalnih performansi i efikasnosti, usporedbom rada modela u slikama generiranim koristeći sliku 5.1. kao sliku sadržaja i onima generiranima koristeći sliku 5.12. kao sliku sadržaja primjećuje se da je kroz 1000 epoha model brže generirao slike sa slikom 5.1. kao slikom sadržaja, nego u drugom slučaju. Isto tako, usporedbom grafova ukupnog gubitka vrijednosti su nešto veće u slučaju kada je korištena slika 5.12. kao slika sadržaja. Iz toga se zaključuje da je generiranje brže i manje računalno zahtjevno kada se na slikama koje se koriste kao slike sadržaja i slike stila ne nalazi puno zahtjevnih tekstura i veliki spektar boja, ali samim time i rezultati su manje primjetni i manje zanimljivi nego u slučaju kada se na slikama nalazi više detalja i boja.

# 6. ZAKLJUČAK

U ovom radu opisane su osnove konvolucijskih neuronskih mreža te je prikazana njihova praktična primjena u neuronskom prijenosu stila. Rezultati praktičnog dijela rada pokazuju da se kombiniranjem dvije slike mogu dobiti vrlo zanimljivi rezultati, odnosno nove slike koje su nalik na umjetnička djela. Budući da se radi sa slikama, cijeli postupak evaluacije mreže je dosta zahtjevan što se tiče računalnih resursa. Iako se proces neuronskog prijenosa stila i evaluacija modela može odraditi koristeći procesor, u tom slučaju će on puno duže trajati. Iz tog razloga svakako je preporuka da se koristi grafička kartica sa što boljim performansama, jer korištenjem grafičke kartice sve operacije koje se koriste unutar konvolucijske neuronske mreže se izvode puno brže, povećavajući time efikasnost rješenja. Isto tako, korištenjem grafičke kartice boljih performansi lakše i brže se obrađuju i slike većih dimenzija. Rješenje se može primijeniti na način da se implementira kao web aplikacija u kojoj bi bilo moguće odabrati proizvoljnu sliku na koju će se primijeniti stil druge slike i tako dobiti generirana slika, te da se na web stranici prikazuju dobiveni rezultati.

# LITERATURA


[1] „Umjetne neuronske mreže“ , Ž. Ujević Andrijić, https://hrcak.srce.hr/file/322233 [05.06.2022.]

[2] „Getting to know Activation Functions in Neural Networks“ , https://towardsdatascience.com/getting-to-know-activation-functions-in-neural-networks-125405b67428 [12.06.2022.]

[3] „Sigmoid function“ , https://en.wikipedia.org/wiki/Sigmoid_function [12.06.2022.]

[4] „Gradient Descent: An Introduction to 1 of Machine Learning’s Most Popular Algorithms“, Niklas Donges, https://builtin.com/data-science/gradient-descent [12.06.2022.]

[5] „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“, Sumit Saha , https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [14.06.2022]

[6] „Convolutional Neural Networks (CNNs)“, Anh H. Reynolds, https://anhreynolds.com/blogs/cnn.html [14.06.2022.]

[7] „Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail“ , Muhamad Yani, S, Si., M.T. Budhi Irawan, Casi Setianingsih , https://www.researchgate.net/publication/333593451_Application_of_Transfer_Learning_Using_Convolutional_Neural_Network_Method_for_Early_Detection_of_Terry's_Nail [16.06.2022.]

[8] „LeNet-5 - A Classic CNN Architecture“ , https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/ [16.06.2022.]

[9] „Review: AlexNet, CaffeNet — Winner of ILSVRC 2012 (Image Classification)“ , https://medium.com/coinmonks/paper-review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160 [16.06.2022.]

[10] „Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet“ , https://towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d [17.06.2022.]

[11] „A Neural Algorithm of Artistic Style“, Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, https://arxiv.org/pdf/1508.06576.pdf [18.08.2022.]

[12] „Image Style Transfer Using Convolutional Neural Networks“, Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf [20.08.2022.]

[13] „Controlling Perceptual Factors in Neural Style Transfer“, Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, Eli Shechtman, https://arxiv.org/pdf/1611.07865.pdf [21.08.2022.]

[14] https://www.pexels.com/photo/aerial-photo-of-white-castle-with-green-leafed-tree-under-white-cloudy-sky-772472/ [22.08.2022.]

[15] https://hr.wikipedia.org/wiki/Krik [22.08.2022.]

[16] „Neural Style Transfer: Everything You Need to Know“, Pragati Baheti, https://www.v7labs.com/blog/neural-style-transfer [22.08.2022.]

[17] „Linearni diskriminativni modeli“, Bojana Dalbelo Bašić, Jan Snajder, https://www.fer.unizg.hr/_download/repository/SU-8-LinearniDiskriminativniModeli.pdf [22.08.2022.]

[18] „Predviđanje ishoda učenja pomoću neuronskih mreža u okruženju konceptualnih mapa“, Hrvoje Ljubić, 2018., https://www.researchgate.net/profile/Hrvoje-Ljubic/publication/327282986_Predvidanje_ishoda_ucenja_pomocu_neuronskih_mreza_u_okruzenju_konceptualnih_mapa/links/5b86894d92851c1e123926b3/Predvidanje-ishoda-ucenja-pomocu-neuronskih-mreza-u-okruzenju-konceptualnih-mapa.pdf [22.08.2022.]

[19] PyTorch, https://pytorch.org [22.08.2022.]

[20] Python, https://hr.wikipedia.org/wiki/Python_(programski_jezik) [22.08.2022.]

[21] https://en.wikipedia.org/wiki/File:Pablo_Picasso,_1909-10,_Figure_dans_un_Fauteuil_(Seated_Nude,_Femme_nue_assise),_oil_on_canvas,_92.1_x_73_cm,_Tate_Modern,_London.jpg [31.08.2022.]

[22] Kubizam, https://www.znanje.org/i/i24/04iv07/04iv07040826/kubizam.htm [31.08.2022.]

[23] https://hr.wikipedia.org/wiki/Tvrđa#/media/Datoteka:Tvrđa_pogled.jpg [01.09.2022.]

# SAŽETAK

U ovom diplomskom radu opisane su osnove konvolucijskih neuronskih mreža. Kroz rad su opisane osnove i način rada neuronskih mreža, objašnjeni su sastavni dijelovi, odnosno slojevi mreže, te su predstavljeni primjeri najčešće korištenih konvolucijskih neuronskih mreža. Cilj praktičnog dijela rada je primjena konvolucijske neuronske mreže u postupku neuronskog prijenosa stila. To je postupak kojim se obilježja jedne slike prenose na drugu sliku. Opisana je teorijska strana i formule algoritma koje se koriste u algoritmu za neuronski prijenos stila. Programsko rješenje za navedeni postupak napisano je u programskom jeziku Python, točnije PyTorch biblioteci koja se koristi za razvoj i treniranje neuronskih mreža baziranih na dubokom učenju modela. Arhitektura konvolucijske neuronske mreže koja je korištena u rješenju je VGG19. Rezultati programskog rješenja prezentirani su u zadnjem dijelu rada gdje se može vidjeti kako izgleda generirana slika nakon prijenosa obilježja slike s jedne na drugu i kako kompleksnost slike utječe na generiranje novih slika.

Ključne riječi: konvolucijska neuronska mreža, obilježja, sadržaj, slika, stil

# ABSTRACT

**Title: Transmission of image features using a convolutional neural network**

This master's thesis describes the basics of convolutional neural networks. The paper describes the basics and way of working of neural networks, explains the constituent parts, the layers of the network and presents examples of the most commonly used convolutional neural networks. The purpose of the practical part is the application of a convolutional neural network in the process of neural style transfer. It is a process in which the features of one image are transferred to another image. The theoretical side and formula of the algorithm used in the neural style transfer algorithm are described. The program solution for the mentioned process was written in the Python programming language, more specifically the PyTorch library, which is used for developing and training neural networks based on deep learning models. The convolutional neural network architecture used in the solution is VGG19. The results of the solution are presented in the last part of this paper, where you can see how the generated image looks like after transferring image features from image to another and how the complexity of the image affects generating of new images.

Key words: convolutional neural network, features, content, image, style

# ŽIVOTOPIS

Matej Šarčević rođen je u Slavonskom Brodu 21. siječnja 1997. godine. Pohađao je osnovnu školu Sikirevci u Sikirevcima. Završetkom osnovne škole, 2011. godine upisuje se u srednju Tehničku školu u Slavonskom Brodu smjer Tehničar za računalstvo. Srednju školu završava 2015. godine, te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, Preddiplomski stručni studij Elektrotehnike, smjer Informatika. Preddiplomski stručni studij završava 2019. godine, nakon čega upisuje Razlikovne obveze, smjer Računarstvo u svrhu upisa na Diplomski sveučilišni studij. Završetkom programa Razlikovnih obveza 2020. godine, upisuje se na Diplomski sveučilišni studij Računarstvo, izborni blok Informacijske i podatkovne znanosti koji u trenutku pisanja ovog diplomskog rada još uvijek pohađa.