UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Jelena Ž. Tasić

# Procesiranje slikovnih analogija neuronskim mrežama

Master rad

Beograd, 2012

UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Jelena Ž. Tasić

# Implementation of image analogies using neural networks

Master's thesis

Belgrade, 2012

Komisija:

dr Milan Tuba, redovni profesor, mentor
dr Miodrag Živković, redovni profesor
dr Vladimir Filipović, docent

Datum odbrane: 02.07.2012.

# Procesiranje slikovnih analogija neuronskim mrežama


## Apstrakt

U ovom radu predstavljeno je novo okruženje za procesovanje slika korišćenjem uzoraka – analogije slika (*image analogies*). Ovo okruženje se sastoji iz dve faze: faze učenja i faze primene. U fazi učenja se na osnovu originalne slike (*model)* i njoj odgovarajuće filtrirane slike (*master)*, uči *model-master* preslikavanje. Druga faza, faza primene je primena naučenog preslikavanja (filtera) na novoodabranu sliku. Jednom naučeni filter može da se primeni proizvoljan broj puta na novoizabrane slike. Za učenje preslikavanja odabrane su neuronske mreže, tehnika nadgledanog učenja, obzirom da su poznati ulazni podaci kao i cilj – filter koji želimo da bude naučen. Koristi se *feedforeward* arhitektura neuronskih mreža i *resilient propagation* algoritam učenja. Izborom različitih vrsta ulaznih slika (*model* i *master* slika) okruženje podržava učenje različitih vrsta filtera. Prva faza podržava i opciju odvajanja od modela koji pokušavamo da naučimo, gde korisnik nezavisno od modela evoluira sopstvene filtere.

# Implementation of image analogies using neural networks


## Abstract

This work presents a new framework for image processing by example – image analogies. The framework comprises two phases: a learning phase and an application phase. In the learning phase a *model-master* mapping is learnt, on the basis of an original image (*model)* and its corresponding filtered image (*master)*. The second phase, the application phase, is the implementation of the learnt mapping (filter) onto a newly selected image. Once learnt, the filter can be applied to newly selected images any number of times. For the learning of mapping neural networks have been chosen, a technique of supervised learning, considering that both the input data and the target (the filter we wish to be learnt) are given. The feedforward architecture of neural networks is used, as well as the resilient propagation learning algorithm. This framework supports learning of different kinds of filters through selection of different kinds of input images (a *model* and *master* image). The first phase supports in addition an option of diverging from the model which we are trying to learn, so that user can evolve own images independently from the model.

# Sadržaj

# 1 Uvod

Predloženi projekat istražuje mogućnost učenja odabranog filtera. Ideja je da korisnik izabere dve slike, originalnu i njoj odgovarajuću filtriranu sliku, koje predaje softveru kao ulazne podatke, na osnovu kojih se uči željeno preslikavanje – odabrani filter. Rezultat je fajl koji sadrži neuronsku mrežu novonaučenog filtera. Jednom naučeni filter može da se primeni na bilo koju drugu originalnu sliku i da proizvede efekat naučenog filtera.

Slike koje dobijamo korišćenjem fotoaparata su često lošeg kvaliteta. Umesto da se svaka slika obrađuje pojedinačno, bilo bi mnogo jednostavnije korišćenje filtera koji popravlja serije loše dobijenih fotografija, na osnovu jednog primera. Glavni cilj ovog projekta jeste učenje filtera koji su primena većeg broja transformacija na jednu sliku. Na taj način, bila bi omogućena popravka kvaliteta slika, onda kada uzrok loše dobijene slike nije poznat.

Naredna ideja je odvajanje od modela koji pokušavamo da naučimo, gde bi korisnik nezavisno od željenog modela evoluirao sopstvene filtere. Ovakvo rešenje bi omogućilo da je dizajn softvera namenjen zadovoljavanju određene estetike, vođen istom tom estetikom.

Implementacija ove generalne ideje zvuči kao težak problem. Olakšavajuća okolnost je da je program za obradu slika neophodno evoluirati samo jednom, posle čega on može da se kopira, distribuira i koristi proizvoljan broj puta. Kao rezultat, korisnici čija karijera zavisi od estetske originalnosti mogu da očekuju da će biti motivisani da ulože značajno vreme u evoluciju malih programa jer će njihovo neograničeno korišćenje vrlo brzo da potre to ulaganje.

Ovo istraživanje se u najvećoj meri bavi prvim delom predložene ideje, gde se na osnovu ulaznih slika (originalne i njoj odgovarajuće filtrirane slike) uči odabrani filter. Onog momenta kada je korisnik zadovoljan vizuelnim rezultatom naučenog filtera ili kada je postignuta zadata sličnost moći će da prekine program i sačuva neuronsku mrežu kao poseban fajl koji će kasnije moći da bude primenjen na bilo koju drugu sliku. Potrebno je vreme da bi se željeni filter naučio, ali nakon toga primena naučenog filtera se izvršava u realnom vremenu.

Ideja je da se na osnovu originalne slike – *model* i filtrirane slike – *master* korišćenjem neuronskih mreža nauči *model–master* preslikavanje, koje nakon dostignute zadovoljavajuće sličnosti – *fitness* može da se primeni na drugu nefiltriranu sliku. Definicija grafičkih analogija je da se za dati par slika $A$ i $A'$ (nefiltrirana i filtrirana slika respektivno) i za datu nefiltriranu sliku $B$ nađe slika $B'$ takva da je :

$$A : A' = B : B'$$

to jest, da su $A$ i $A'$ u istom odnosu kao i, $B$ i $B'$.

Drugi deo predložene ideje je delimično obrađen i daje iznenađujuće interesantne rezultate. Obzirom da je estetska vrednost subjektivna stvar, ono što je jednom

korisniku interesantno drugom ne mora da bude. Cilj druge ideje je da korisnik učestvuje u stvaranju filtera vođen ličnim ukusom, a da pri tom nema programersku stručnost.

Pošto je cilj predloženog projekta da proceni izvodljivost generalne ideje, eksperimentalni deo će se koncentrisati na uži izbor grafičkih efekata. Ova procena će biti propraćena istorijom radova u ovom polju, nizom eksperimentalnih rešenja, kao i problema predložene ideje.

Ovaj rad je organizovan na sledeći način. Poglavlje 2 se odnosi na fundamentalne koncepte vezane za digitalnu obradu slika, matematički pristup, sam pojam, kao i primenu. Navedeni su neki od osnovnih koraka digitalne obrade slika: restauracija slika, kompresija, segmentacija i poboljšanje slika. Obzirom na temu ovog rada poboljšanje slika je detaljnije objašnjeno. Opisani su i pojedini filteri kao što su: negativ, *threshold*, kontrast, prevođenje slike koja je u boji u crno belu sliku, izoštravanje ivica, detekcija ivica, zamućivanje ivica, kao i dva domena u kojima se tradicionalno ova obrada radi, spacijalni i frekventni domen. Za trening podatke uzete su slike dobijene korišćenjem ovog tradicionalnog pristupa digitalne obrade slike, pa je stoga ovo poglavlje jako važno za sam rad [1], [2], [3].

Za učenje analogije predloženi softver koristi neuronske mreže. Poglavlje 3 opisuje osnovne koncepte neuronskih mreža, njihov značaj, analogiju sa biološkim mrežama, tehnike kao i arhitekturu neuronskih mreža [4], [5]. Opisane su tri tehnike učenja: nadgledano, nenadgledano i *reinforcement* učenje. Softver koristi tehniku nadgledanog učenja obzirom da su poznati ulazni podaci kao i željeni cilj, filter koji softver pokušava da nauči. Prva verzija softvera koristi jednostavne rekurzivne neuronske mreže, kao test da li neuronskim mrežama može da se reši postavljeni problem, dok druga, završna verzija programa koristi *Encog feedforeward* neuronske mreže [6]. Tokom razvoja softvera bilo je jako važno odabrati aktivacionu funkciju koja će biti korišćena [6], [7], kao i algoritam učenja [6], [8]. Opisane su aktivacione funkcije, kao i algoritmi učenja koje podržavaju *Encog* neuronske mreže.

Poglavlje 4 opisuje metode koje koriste uzorke slika kao trening podatke za učenje pojedinih filtera. *Hertzmann*-ov algoritam [9] koji na osnovu jednog trening para slika originalne i njoj odgovarajuće filtrirane slike omogućava učenje različitih vrsta filtera kao što su tradicionalni filteri, sinteza teksture, super-rezolucija i umetnički filteri. Hertzmann koristi isto okruženje za analogije krive [10], učenje stila linija na osnovu datog primera. Pojednini algoritmi su skoncentrisani na uži spektar filtera u cilju dobijanja boljih rezultata, super-rezolucije [11], [12], [13], [14], [15], kolorizacije [16], [17], segmentacije slike [18] i automatskog popravljanja osvetljenja i kontrasta [19].

U Poglavlju 5 se ispituje mogućnost korišćenja neuronskih mreža i ideje analogije slika u cilju učenja odabranih filtera. Opisani su alati i radno okruženje koji su korišćeni za implementaciju softvera [20], [21], kao i metodologija i način korišćenja predloženog softvera. Prikazani su rezultati kao i problemi predloženog metoda. Pored učenja odabranog filtera u ovom poglavlju je opisana i implementacija ideje korisnički

vođenog stvaranja filtera. Na taj način omogućeno je korisniku da stvara ličnu paletu filtera vođen ličnim ukusom i da se na taj način razlikuje od svoje konkurencije.

Rezultati dobijeni u ovoj tezi predstavljeni su i objavljeni u *Proceedings of the Second International Students Conference on Informatics, Sibiu, Romania* [22].

# 2 Digitalna obrada slika

U ovom poglavlju je izložen tradicionalni pristup obrade digitalnih slika. Ovaj pristup koriste mnogi softveri za obradu digitalnih slika kao početni ili jedini način. Važnost ovog poglavlja u cilju ovog istraživanja, jeste to što su mnogi trening podaci, željeni filteri koje predloženi softver pokušava da nauči korišćenjem neuronskih mreža, dobijeni korišćenjem metoda tradicionalnog pristupa. Obzirom da je akcenat samog istraživanja na filterima koji se koriste za obradu slika, poboljšanje slika će biti detaljnije objašnjeno, dok će ostali aspekti digitalne obrade slika biti samo napomenuti bez daljeg zalaženja u detalje. Značajnost samih filtera kao i način na koji su isti prozvedeni su tema ovog poglavlja. U navedenoj literaturi koja je bila korišćena za ovo poglavlje, matematički dobijeni filteri su opisani kroz dva domena spacijalni i frekventni, tako da će taj pristup biti iskorišćen i ovde.

## 2.1 Digitalna obrada slika; Počeci i značaj

Digitalna obrada slika je skup algoritama za obradu digitalnih slika. Vid je najnaprednije naše čulo i obzirom da je veliki deo ljudskog mozga posvećen tome, nije iznenađujuće što slike imaju tako važnu ulogu. Iako je polje digitalne obrade slike zasnovano na matematičkim formulama i formulama verovatnoće presudnu ulogu u izboru jedne tehnike naspram druge ima čovek. Izbor može da se odnosi kako na zadovoljenje određene estetike tako i na poboljšanje slike u smislu dobijanja filtrirane slike koja je pogodnija za tumačenje određenih podataka na slici.

Digitalna obrada slika potiče još od početka dvadesetih godina. Prva upotreba bila je za kodiranje slika koje su prenošene kablovskom vezom preko Atlantskog okeana. Rekonstrukcija slike vršila se na prijemnoj strani. Već sredninom dvadesetih godina značajno je poboljšan kvalitet slike. Novi način reprodukcije baziran je na fotografskim tehnikama i na taj način je povećan broj tonova reprodukovane slike. Daljim usavršavanjem računarske opreme i početkom istraživanja svemira došlo je do naglog napretka ove oblasti. 1964. godine je prvi put računar iskorišćen za popravku kvaliteta slike Meseca dobijene iz svemirske sonde Ranger 7. Godinama je bila simbol za veoma skupu tehnologiju, tek razvojem računarske tehnike i elektronike počinje da prodire i u druge oblasti. 1970. godine digitalna obrada slika počinje da se koristi i u medicini, 1979. godine *Hounsfield* i *Cormack* su dobili Nobelovu nagradu za medicinu za otkriće kompjuterizovane tomografije, radiološke metode za snimanje koja pored rendgen zračenja primenjuje i matematičke procedure za obradu snimka. Metoda digitalne geometrijske obrade snimka koristi seriju dvodimenzionih snimaka za generisanje trodimenzionalnog prikaza.

Od 1980. godine pa do danas digitalna obrada slika se izučava i koristi u mnogim oblastima. Koristi se u medicini, fizici, astronomiji, kriminalistici, arheologiji, robotici, umetnosti, filmu i mnogim drugim oblastima. Pomoću teleskopa Habl koji je lansiran 1990. godine mogu da se dobiju slike jako udaljenih objekata, međutim ogledalo

teleskopa čini slike beskorisnim, tako da se digitalna obrada slika ovde koristi za popravku kvaliteta ovog efekta. Umetnički filteri se koriste kako bi sliku predstavili vizuelno privlačnijom dodavanjem specijalnih efekata. Geografski informacioni sistemi digitalnu obradu slika koriste za manipulaciju satelitski dobijenih snimaka, klasifikacija terena kao i prognoza vremena. Za svrhu sprovođenja zakona digitalna obrada slika se koristi za utvrđivanje brzine uz pomoć radara, prepoznavanje otisaka prstiju.

Iako se koristi u različitim oblastima, mnogi algoritmi za digitalnu obradu slika su isti.

## 2.2 Digitalna slika

Digitalne slike se danas koriste svuda i važna karakteristika je da se mogu lako procesirati korišćenjem različitih matematičkih metoda. Polazna tačka je projekcija 3-dimenzionalnog sveta na 2–dimenzioni pravougaonik, odnosno ekran. Ovaj pravougaonik je podeljen horizontalno i vertikalno na male pravougaonike fiksne veličine, obično kvadrate koji se zovu pikseli. Pikseli treba da budu mali tako da ih ljudsko oko ne identifikuje kao posebne elemente. Raster slike čuvaju informaciju o osvetljenju i boji za svaki piksel. U zavisnosti od broja predstavljenih boja, svaki piksel će biti kodiran sa odgovarajućim brojem bitova. Za crno-bele slike potreban je jedan bit (0 za belu i 1 za crnu), za veći broj nijansi sive potreban je veći broj bitova (za 256 nijansi sive potrebno je 8 bita). Slike u boji zahtevaju više memorije. Boja može da se predstavi preko RGB kolor modela sa tri boje koje predstavljaju nijanse crvene, zelene i plave komponente. Broj bitova po elementu slike koji je potreban da bi se opisala boja naziva se dubina boje. Slike se obično čuvaju red po red, s leva na desno i odozgo na dole. Redovi slike se nazivaju sken linije. Podaci potrebni za rekonstrukciju zabeleženih linija su: dužina sken linija, broj sken linija i dubina boje.

Konverzija analogne slike u digitalnu uključuje dve važne operacije sampliranje i kvantizaciju. Sampliranje je proces merenja osvetljenja samo u diskretnim spacijalnim lokacijama. Neprekidna funkcija slike *f(x,y)* može biti samplirana korišćenjem diskretne mreže sampliranih tačaka u toj oblasti. Kvantizacija je tehnika koja se postiže kompresijom opsega vrednosti u jednu kvantnu vrednost. Prednosti digitalnih slika naspram analognih jesu te što je procesovanje slika brže, digitalne slike mogu efikasno da se čuvaju i prenose sa jednog mesta na drugo. Kopiranje digitalnih slika je jednostavnije i iskopirana slika je istog kvaliteta kao i originalna čak i ako je kopirana više puta. Digitalna obrada slika omogućava korišćenje mnogo kompleksnijih algoritama za obradu koje bi bilo nemoguće primeniti na analognim slikama. Problem sa digitalnom tehnologijom je to što reprezentacija slike u digitalnom formatu generiše veliku količinu podataka. Obično veličina digitalnih slika može da bude nekoliko desetina megabajta. Jedno rešenje za ovaj problem je da se koriste tehnike kompresije.

U fotografiji i računarstvu digitalne *grayscale* slike su slike kod kojih se za svaki piksel čuva samo jedna vrednost, to je informacija o intenzitetu osvetljenosti. Ovakve slike se takođe zovu i crno-bele slike koje se sastoje od nijansi sive u rasponu od crne

najslabijeg intenziteta do bele jačeg intenziteta. Razlikuju se od jednobitnih slika gde je svaki piksel obojen belom ili crnom bojom. Najjednostavniji način konverzije slike u boji u *grayscale* sliku ako je slika predstavljena RGB modelom vrši se sabiranjem sve tri komponente (crvene, zelene i plave) i deljenjem sa tri. Bolji način je uzimanje u obzir činjenice da je ljudsko oko više osetljivo na zelenu i crvenu nego na plavu, pa se uzima 30% crvene komponente, 59% zelene i 11% plave.

## 2.3 Fundamentalni koraci u digitalnoj obradi slika

Oblast digitalne obrade slike obuhvata mnoge podoblasti. Ovde će biti navedene neke od njih kao što su: poboljšanje slike, restauracija slike, kompresija i segmentacija.

Poboljšanje slike je jedna od najjednostavnijih i najprivlačnijih oblasti digitalne obrade slika. Osnovni cilj je da se promene neke karakteristike slike kako bi slika bila pogodnija za prikaz i detaljnu analizu. Metodi za poboljšanje slike su brojni. Kvalitet slike jako zavisi od uslova pod kojim je slika dobijena, kao i od primene odgovarajućih filtera za poboljšanje slike. Važno je imati na umu da je poboljšanje slike jako subjektivno. Problem tradicionalnih tehnika koje se koriste kod poboljšanja slike je taj što proizvođači filtera nisu korisnici, pa je na neki način korisnik prinuđen da koristi samo paletu filtera koje proizvođač ponudi. Iz ovih razloga, uz mnoge filtere za poboljšanje slike su ponuđeni i paremetri čije vrednosti mogu da se menjaju ali ne i vrsta filtera.

Restauracija slike je oblast koja se takođe bavi poboljšavanjem izgleda slike. Za razliku od prethodne oblasti, koja je se zasniva na subjektivnom sudu, restauracija slike je objektivna oblast u smislu da se tehnike restauracije koriste u slučajevima kada je degradacija slike poznata ili kada može da se formira model degradacije. Za razliku od metoda za poboljšanje slike, metode za restauraciju se ne izvršavaju u realnom vremenu.

Kompresija slike se bavi fundamentalnim tehnikama koje se bave smanjenjem prostora za skladištenje slike ili propusnim opsegom (*bandwidth*) potrebnim za prenos. Tehnike vezane za skladištenje su napredovale poslednjih godina, dok to nije slučaj i sa kapacitetom za prenos. Naročito je to slučaj sa Internetom, gde ima značajno mnogo slikovnog sadržaja. S obzirom na značaj i potrebu za kompresijom podataka na slici razvijen je veliki broj metoda kompresije. Jedna grupa metoda kompresije se odnosi na reprezentaciju slike bez ikakvog gubitka informacije, dok se druga grupa metoda odnosi na kompresiju uz gubitak beznačajne količine informacija. Stepen kompresije korišćenjem druge metode je značajno veći.

Metode segmentacije dele sliku na njene sastavne delove ili objekte. Ovo je jedan od najtežih zadataka u digitalnoj obradi slika. Cilj segmentacije slike je da pojednostavi ili promeni reprezentaciju slike u nešto što je jednostavnije i lakše za analizu. Segmentacija slike se obično koristi za lociranje objekata i granica (linije, krive i sl.) na slici. Tačnije, segmentacija je proces označavanja svakog piksela slike tako da ti pikseli dele određene vizuelne karakteristike. Rezultat segmentacije slike je skup segmenata

koji zajedno čine celu sliku, ili set kontura izdvojenih iz slike – izdvajanje ivica. Svi pikseli jednog regiona imaju slične karakteristike ili osobine kao što su boja, intezitet ili tekstura. Susedni regioni su bitno drugačiji. Kada se segmentacija primeni na veliki broj slika, obično rezultujuće konture dobijene nakon segmentacije mogu da se iskoriste kako bi se napravila 3D rekonstrukcija.

U daljem tekstu rada detaljnije će biti objašnjeno poboljšanje slike u frekventnom i spacijalnom domenu.

## 2.4 Poboljšanje slika u spacijalnom (prostornom) domenu

Slika može da se definiše kao dvodimenzionalna funkcija *f(x,y),* gde su *x* i *y* spacijalne koordinate. Vrednost funkcije *f* za bilo koji par koordinata *(x,y)* naziva se intezitet ili nivo sive u toj tački. Kada su *x*, *y* i vrednost funkcije *f* diskretne veličine tada za sliku kažemo da je digitalna. Termin spacijalni domen se odnosi na samu sliku, pristupi u ovom domenu se zasnivaju na direktnoj manipulaciji sa pikselima slike za razliku od frekventnog domena gde se slika Furijeovim transformacijama prvo prebaci u frekventni domen, gde neki procesi mogu lepše i lakše da se izvedu. Često jedna ista stvar može da se uradi i u frekventnom i u spacijalnom domenu. Procesi u spacijalnom domenu se označavaju sa:

$$g(x,y) = T[f(x,y)]$$

gde je *f(x,y)* ulazna slika, *g(x,y)* procesovana slika, a *T* je operator funkcije *f*, definisan preko susednih piksela *(x,y)*. Najjednostavnija forma operatora *T* je kada *g* zavisi samo od vrednosti funkcije *f* u tački *(x,y)*, odnosno kada je okolina veličine 1x1, jedan piksel. Efekat ove transformacije može da bude slika većeg ili manjeg kontrasta, osvetljenja, *threshold* ili negativ. Tada T nazivamo transformacijama nivoa sive. Veći broj suseda pruža veću fleksibilnost.

Opšti pristup je korišćenje vrednosti funkcije *f* u prethodno definisanoj okolini suseda *(x,y)* da bi se izračunala vrenost funkcije *g(x,y)*. Susedi piksela *(x,y)* mogu da se predstave pomoću matrice koja se još zove i filter ili kernel. Primer, 3x3 okolina može da se predstavi kao matrica:

$$\begin{bmatrix} f(x-1,y-1) & f(x,y-1) & f(x+1,y-1) \\ f(x-1,y) & f(x,y) & f(x+1,y) \\ f(x-1,y+1) & f(x,y+1) & f(x+1,y+1) \end{bmatrix}$$

Za razliku od uobičajenog Dekartovog pravouglog koordinatnog sistema, za obradu slike se koristi koordinatni sistem koji vodi poreklo od televizije. Položaj koordinatnog početka je u gornjem levom uglu. Koordinatne ose su zamenile svoje uloge. Koordinatna osa *x*, je usmerena na dole.

Kod metoda za poboljšanje slike osnovni cilj je da se promene neke karakteristike slike tako da slika bude pogodnija za prikaz i analizu. Poboljšanjem se ne popravlja

informacioni sadržaj slike već se olakšava korišćenje postojećih informacija. Za ove metode je karakteristično da su u najvećem broju slučajeva jednostavne tako da se mogu izvršiti u realnom vremenu. Vizuelna procena kvaliteta slike je krajnje subjektivan proces, što definiciju "jako dobre slike" čini problematičnom u smislu poređenja algoritama.

### 2.4.1 Procesi na jednoj tački

Najjednostavniji filteri su operacije na jednoj tački. Definisani su kao funkcija koja se obavlja na svakom pikselu slike, nezavisno od ostalih piksela na toj slici.

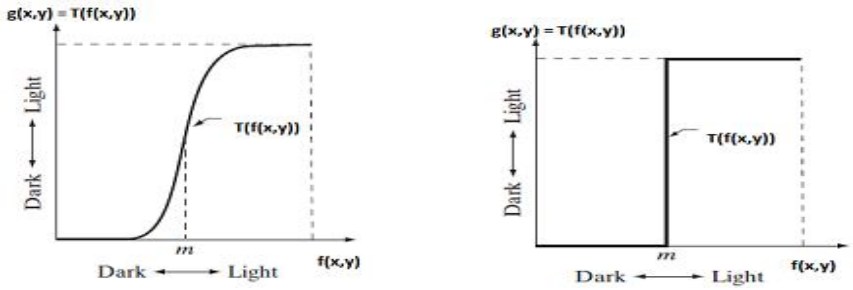
$$I(x,z) = f(I(x,y))$$

Funkcija *f* zavisi samo od vrednosti piksela, ali ne i od lokacije piksela. Treba da bude jednoznačna i monotono rastuća. Ako je vrednost piksela originalne slike u intervalu [0, 255] onda u tom intervalu treba da bude i rezultujuća slika. Prvi uslov je potreban zbog očuvanja redosleda skale nivoa sive, a drugi uslov obezbeđuje da se zadrži isti opseg skale. Nije neophodno da oba uslova budu zadovoljena, jer postoje korisna preslikavanja kod kojih neki od ova dva uslova nije zadovoljen.

Povećanje osvetljenja slike je proces dodavanja konstante svakom pikselu slike:

$$g(x,y) = \begin{cases} f(x,y) + k, & ako\ je\ rezultat \leq 255 \\ 255, & inače \end{cases}$$

Smanjenje osvetljenja je sličan proces samo se konstanta oduzima.




Slika 1: (a) povećanje kontrasta (b) *threshold*

Kontrast je razlika vizuelnih osobina koje čine da se objekti na slici razlikuju od drugih objekata pozadine. Množenjem funkcije konstantom smanjuje se i pojačava kontrast slike. Na Slici 1 (a) prikazan je efekat koji proizvodi sliku većeg kontrasta od originalne slike, ono što je tamno postaje još tamnije, a ono što je svetlo postaje još svetlije. Na Slici 1 (b) je prikazan granični slučaj - *threshold* .To je najjednostavniji metod segmentacije slike. Koristi se da bi se napravila binarna slika:

$$g(x,y) = \begin{cases} 255, & ako\ je\ f(x,y) > T \\ 0, & ako\ je\ f(x,y) \leq T \end{cases}$$

Ključni parametar za *threshold* proces je izbor parametra T. Postoji više metoda za izbor ovog parametra, korisnik može sam da izabere parametar T ili može da se iskoristi neki od algoritama za automatsko određivanje parametra za *threshold*. Najjednostavniji algoritam bi bio da se uzme srednja vrednost. Ako su pikseli svetliji od pozadine, treba da budu svetliji i od proseka.

Negativ slike se dobija:

$$g(x,y) = 255 - f(x,y)$$

Invertovanjem nivoa inteziteta slike proizvodi se ekvivalent fotografskog negativa. Ova vrsta procesovanja je specijalno pogodna za poboljšanje belih i sivih detalja okruženih tamnim regionima slike, specijalno kada su crne površine dominantne.

### 2.4.2 Detekcija ivica

Osnovni cilj detektovanja ivica je izvlačenje bitnih informacija iz slike pomoću kojih može da se izvrši računarska interpretacija kao i analiza slike. Primeri ovakvih sistema za analizu slike su brojni: sistemi za obradu čekova, razvrstavanja pošte, detekciju ciljeva, navođenje projektila i mnogi drugi.

Ivice odgovaraju značajnim lokalnim promenama inteziteta na slici. Intuitivno, ivica je skup povezanih piksela koji leže na granici između dva regiona. Promene inteziteta su prouzrokovane različitim fizičkim promenama uključujući boju, teksturu, refleksiju i senke. Detekcija ivica na slikama sa šumom je jako teška, zato što i šum i ivice sadrže visokofrekventni sadržaj, štaviše ne sadrže sve ivice postepene promene inteziteta.

Postoji mnogo metoda za detekciju ivica. Mnoge od postojećih mogu biti grupisane u dve kategorije: metode koje su zasnovane na prvom izvodu funkcije i metode koje su zasnovane na drugom izvodu funkcije. Metode koje koriste izvod prvog reda proizvode deblje ivice, dok metode koje koriste drugi izvod imaju jači odziv na fine detalje, kao što su tanke linije i izolovane tačke.

Metode koje koriste prvi izvod zasnovane su na različitim aproksimacijama 2-D gradijenta. Gradijent slike *f(x,y)* na lokaciji *(x,y)* je definisan kao vektor:

$$\nabla = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Magnituda ovog vektora je data:

$$\nabla f = \mathrm{mag}(\nabla) = [Gx^2 + Gy^2]^{\frac{1}{2}} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial x}\right)^2\right]^{\frac{1}{2}}$$

Magnituda gradijenta pokazuje na vrednost razlike okolnih piksela – snagu ivice. Jednačina za izračunavanje magnitude je teška za implementaciju i zbog toga se u praksi uzima približna vrednost:

$$\nabla f \approx |Gx| + |Gy|$$

U diskretnom slučaju to se svodi na razlike:

$$Gx = f(x+1,y) - f(x,y) \qquad Gx = [-1 \quad 1]$$
$$Gy = f(x,y+1) - f(x,y) \qquad Gy = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Pored ovog osnovnog metoda postoje i drugi kao što su Robertsov operator:

$$Gx = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, Gy = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Sobelov operator:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Previt:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, Gy = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

i drugi.

### 2.4.3 Ublažavanje ivica (*smoothing*)

Filteri za ublažavanje ivica koriste se za zamućivanje (*blurring*) slike i za redukciju šuma. Izlaz je prosek vrednosti piksela sadržanih u okolini zadatoj preko matrice – maske. Oni takođe referišu na filtere koji propuštaju niske frekvencije – *lowpass filters*. Ideja je da se vrednosti svakog piksela na slici, zamene prosekom nivoa osvetljenja njegove okoline, koja je definisana maskom:

$$g(x,y) = \sum_{k,l \in S} w(k,l) f(x-k, y-l)$$

gde *S* predstavlja skup koordinata tačaka koje pripadaju lokalnoj okolini, a *w(k,l)* su elementi koji predstavljaju spacijalnu masku. Ove matrice su obično kvadratnog oblika sa neparnim brojem elemenata. Neke od njih su:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix}0 & 1 & 0\\ 1 & 4 & 1\\ 0 & 1 & 0\end{bmatrix} \qquad \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Ublažavanjem ivica se gubi oštrina slike jer se oštri prelazi – ivice ublažavaju. Ovaj negativni efekat može da se smanji modifikacijom ovog postupka. Prvo, za svaki piksel se odredi vrednost ublažavanja, a zatim se zamenjuju samo one vrednosti piksela koje su znatno promenjene pod uticajem šuma. Ovaj metod je pogodniji kada ivice na slici treba da se zadrže, a šum redukuje.

Ublažavanje ivica je pogodno za uklanjanje Gausovog šuma iz slike, ako intezitet šuma nije mnogo veliki. U slučaju da jeste jako veliki – ovaj efekat se naziva "*salt and pepper*" ("so i biber"), ovim metodom se znatno menjaju vrednosti piksela koje nisu promenjene šumom i na taj način se kvari kvalitet slike.

## 2.5 Poboljšanje slika u frekventnom domenu

U kompjuterskoj grafici kao i u elektronici, statistici i mnogim drugim disciplinama, frekventni domen se koristi za opisivanje domena za analizu matematičkih funkcija ili signala pre nego spacijalni.

Dobro poznati način za reprezentaciju slike je piksel po piksel kodiranje, zadržavanjem informacije o svakoj komponenti ponaosob, bilo da je to RGB kolor model ili neki drugi. To je reprezentacija u spacijalnom domenu i veoma je korisna za prikazivanje slike na ekranu ili štampanje. To je način na koji popularni BMP fajl format čuva podatke o slici – piksel po piksel, memorisanjem tri bajta (ili neki drugi broj zavisno od broja podržanih boja) za svaki piksel. Za mnoge algoritme, međutim, spacijalni domen nije adekvatan. Osnovni razlog je što analiziranjem slika, kolekcije piksela koje se u spacijalnom domenu posmatraju kao proizvoljne i kao takve obrađuju, zapravo nisu nasumične. Većina slika predstavlja prirodu, nebo, ljude, more i slično. Posmatranjem, mi automatski uočavamo ivice i objekte. To je zato što su veći delovi slike obično identični ili veoma slično obojeni. Na primer nebo je sivo ili plavo, ali to je identično (ili skoro identično) obojeno, verovatno sa nekim oblacima.

Reprezentacija slika u frekventnom domenu je korisna zbog smanjenja prostora za čuvanje, ali i za druge svrhe. U frekventnom domenu jednostavnije je da se uoče neke važne osobine slike kao što je detekcija ivica. Nepravilnosti mogu biti uočene takođe. Ako je u pitanju slika sa šumom, posmatranjem slike u frekvenknom domenu i filtriranjem visokih frekvencija ove nepravilnosti mogu lako biti uklonjene. Pored jednostavnih, postoji i mnogo sofisticiranih tehnika koje se koriste, kao kombinacija različitih filtera. Jedan od ovih pristupa je kombinacija nelinearnih visoko i niskofrekventnih filtera. Ovaj pristup smanjuje šum i poboljšava ivice.

Furijeova transformacija je opšti način za konvertovanje podataka iz spacijalnog domena u frekventni domen, ali za manipulaciju sa slikama, to može da se izvede jednostavnijom transformacijom, obzirom da su podaci slike diskretne vrednosti. Koristi se diskretna Furijeova transformacija. Da bi se primenio filter u frekventnom domenu, Furijeova transformacija se prvo primeni na funkciju, pomnožena filter funkcijom, a onda se inverznom Furijeovom transformacijom vrati u spacijalni domen. Frekventni filteri mogu da se primene i u spacijalnom domenu izračunavanjem jednostavnog kernela (matrice) za željeni efekat, premda je filtriranje u frekventnom domenu prikladnije ako prava matrica ne može da se nađe u spacijalnom domenu, a nekad i mnogo efikasnije.

### 2.5.1 Niskofrekventni filteri (*lowpass filters*)

Niskofrekventni filteri propuštaju niskofrekventni sadržaj slike, dok visokofrekventi sadržaj slabi. Ovi filteri su dobri za smanjenje šuma na slici.
Savršeni niskofrekventni filter je veoma oštar filter koji seče sve visokofrekventne komponente Furijeove transformacije:

$$H(u,v) = \begin{cases} 1, & D(u,v) \le D_0 \\ 0, & D(u,v) > D_0 \end{cases} \qquad D(u,v) = [\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2]^{\frac{1}{2}}$$

gde je $D_0$ određena nenegativna veličina, tačka prelaza između $H(u,v)=1$ i $H(u,v) = 0$, a $D(u,v)$ je rastojanje od tačke $(u,v) = (\frac{M}{2}, \frac{N}{2})$, gde je veličina slike $MxN$.
Ovaj filter se ne koristi puno, češće se koriste sofisticiraniji filteri kao *Butterworth,* ili *Gaussian* filteri.

### 2.5.2 Visokofrekventni filteri (*highpass filters*)

Niskofrekventni filteri potiskuju visoke frenkvencije, dok visokofrekfenktni filteri visoke frenkvencije ostavljaju nepromenjene, ali potiskuju niske frekvencije. Visokofrekventni filteri izoštravaju sliku, pošto su ivice i druge nagle promene visokofrekventne komponente. Isticanje ivica je u osnovi obrnuta operacija u odnosu na ublažavanje ivica (*smoothing*):

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

gde je $H_{lp}(u,v)$ funkcija koja odgovara funkciji niskofrekventnog filtera. Kada niskofrekventni filter nepropušta frekvecije, visokofrekventni filter propušta i obrnuto. Idealni visoko frekventni filter je definisan:

$$H(u,v) = \begin{cases} 0, & D(u,v) \le D_0 \\ 1, & D(u,v) > D_0 \end{cases}$$

# 3 Neuronske mreže

Postoje kategorije problema za koje ne može da se formuliše algoritam, problemi koji zahtevaju učenje da bi se dobili rezultati, ili problemi koji su eksponencijalne vremenske složenosti. Za rešavanje ovakvih vrsta problema koriste se neuronske mreže. Na prvi pogled izgleda idealno, ali je s druge strane jako kompleksno, zbog toga što su izračunavanja kompleksne prirode, pa je jako teško pronaći grešku. Obično se iz tog razloga testiraju na problemima za koje postoje egzaktni algoritmi ili se rešenje unapred zna.

Za rešenje problema analogije slika, za učenje izabranog filtera u cilju ovog istraživanja iskorišćene su neuronske mreže. Obzirom da je za izradu softvera bilo neophodno da se napravi izbor tehnike učenja, arhitekture neuronskih mreža kao i aktivacione funkcije koja će se koristiti, u ovom poglavlju će ove teme biti detaljnije obrađene.

## 3.1 Neuronske mreže; Počeci i značaj

Početak neuro-računarstva se najčešće vezuje za 1943. godinu i neuro-psihologe *Warrena McCulloch*-a i *Waltera Pitts*-a, koji su proizveli prvi veštački neuron, ali tehnologija dostupna u to vreme nije im dozvolila da urade nešto više na tom polju. Koncept neuronske mreže je prvi predložio *Alan Turing* u svom radu "Inteligentna mašina" 1948. godine. U periodu od 1950. do 1960. godine objavljivani su mnogi radovi i knjige na ovu temu. Sredinom 1960. godine pristup rešavanja problema korišćenjem neuronskih mreža je dokazan kao pogrešan. *Minsky* i *Papert* su u knjizi "Perceptron" dali dokaz kako neuronska mreža ne može da nauči XOR operaciju, uz pretpostavku da ako se doda više slojeva neurona taj problem neće moći da se prevaziđe i na taj način usporili razvoj ove oblasti. Njihov dokaz je kasnije opovrgnut, za malo složeniju mrežu od nekoliko neurona, to je jednostavan zadatak. Iako postoje od četrdesetih godina, značajnu praktičnu primenu imaju od osamdesetih godina kada su algoritmi postali dovoljno dobri za upotrebu. Danas imaju široku primenu u prepoznavanju oblika, rukopisa, govora, predviđanju cena na tržištu, kriminološkim i vojnim istraživanjima, analizi medicinskih testova, psihijatrijskim procenama, pronalaženju optimalnih rešenja, upravljanju robotima, analiziranju podataka, vremenskoj prognozi i mnogim drugim oblastima. Veliki potencijal neuronskih mreža ogleda se u mogućnosti paralelne obrade podataka.

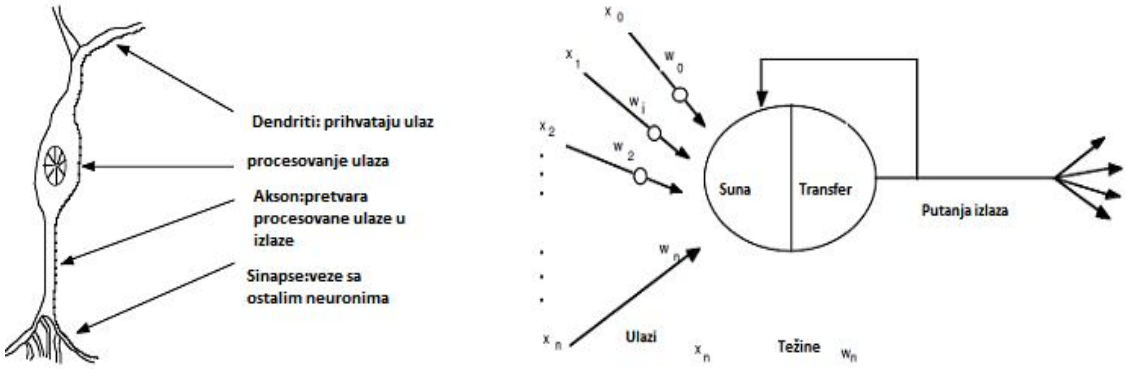
## 3.2 Analogija sa biološkim mrežama

Tačan rad ljudskog mozga je i dalje misterija, ali neki aspekti ovog neverovatnog procesa su poznati. Ljudski mozak se sastoji od 100 biliona neurona i svaki od njih može da bude povezan sa do 200 000 drugih neurona. Sami neuroni su takođe komplikovani, oni imaju bezbroj delova, podsistema i kontrolnih mehanizama. Postoji preko stotinu različitih klasa neurona u zavisnosti od klasifikacije metoda koji koriste.

Zajedno neuroni i njihove konekcije formiraju proces koji nije ni binaran, ni stabilan, ni sinhron. Ukratko, ništa kao trenutno dostupni računari, pa čak ni veštačke neuronske mreže.

Veštačke neuronske mreže su po strukturi, funkciji i obradi informacija slične biološkim, mada su biološke mreže daleko komplikovanije od svojih matematičkih modela. One pokušavaju da ponove samo najjednostavnije elemente ovog komplikovanog i moćnog sistema. Iako to rade na primitivan način, kao takve, jako su uspešne u rešavanju mnogih problema.

Osnovni element neuronskih mreža je neuron. U suštini, biološki neuron prima ulaze od okruženja kombinuje ih na neki način, generalno primenjujući nelinearne operacije, a zatim izbacuje krajnji rezultat. Na Slici 2 (a) prikazan je jednostavan biološki neuron.




Slika 2: (a) biološki neuron (b) veštački neuron

Kod ljudi postoji mnogo varijacija ove jednostavne vrste neurona. Na Slici 2 (b) prikazan je veštački neuron. Veštačke neuronske mreže koriste neurone koji se sastoje od ove četiri osnovne funkcije biološkog neurona navedene na Slici 2 (a). Ulazi u mrežu predstavljeni su matematičkim simbolom *x(n)*, svaki od ovih ulaznih podataka je pomnožen svojom težinom. Težine su označene sa *w(n)*. Najjednostavniji slučaj je da se ovi proizvodi saberu i proslede funkciji transfera (aktivacionoj funkciji) koja generiše rezultat, a zatim to prosleđuje kao izlaz. Ova implementacija je moguća i sa drugim mrežama koje koriste drugačije funkcije sumiranja, kao i drugačije transfer funkcije.
Funkcija sumiranja:

$$I = \sum w_i \, x_i$$

Funkcija transfera:

$$Y = f(I)$$

U zavisnosti od problema funkcija sumiranja ne mora da bude samo zbir, može da bude i maksimum, minimum, srednja vrednost, može da koristi i logičke operatore kao što su ∧,∨. Ova vrednost se prosleđuje aktivacionoj funkciji. Aktivaciona funkcija može

da bude binarna funkcija, sigmoidna, hiperbolički tangens i sl. Krajnji rezultat je izlaz aktivacione funkcije. Ovaj izlaz je obično ulaz drugih elemenata procesa-neurona, ili spoljna konekcija, zavisno od strukture mreže.

Neuronske mreže paralelno obrađuju podatke, njihove komponente su međusobno nezavisne. One se ne programiraju već se treniraju, pa je potrebno dosta vremena pre nego što mogu da se koriste. Na osnovu ulaznih podataka podešavaju se koeficijenti veza između neurona. Neuroni uče uz pomoć primera i poseduju svojstvo generalizacije. Učenje je u stvari, ažuriranje težinskih koeficijenata da bi se u narednoj generaciji dobio bliži rezultat. Neuronska mreža nakon završetka učenja pamti težinske koeficijente.

Neuronsku mrežu čine arhitektura mreže, aktivaciona funkcija i zakoni učenja. Tim redom će u daljem tekstu biti i objašnjeni.

## 3.3 Arhitektura neuronskih mreža

Arhitekturu neuronskih mreža predstavlja uređenje i povezivanje neurona u obliku mreže. Neuronske mreže se razlikuju po broju slojeva kao i prema načinu povezivanja neurona. Sastoje se iz tri vrste slojeva: ulazni sloj, izlazni (poslednji sloj), a između su skriveni slojevi. Slojevi su međusobno povezani. Oni međusobno komuniciraju tako što su izlazi svakog neurona iz prethodnog sloja povezani sa ulazima narednog sloja neurona.

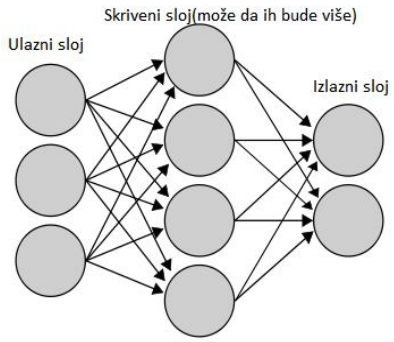
Iako postoje korisne mreže koje sadrže samo jedan sloj, ili čak jedan element, većina mreža sadrži najmanje tri sloja, dok složenije neuronske mreže imaju više skrivenih slojeva. Postoji veliki broj različitih realizacija neuronskih mreža, pa stoga i veliki broj podela. Jedna od već navedenih podela je prema broju slojeva, na jednoslojne i višeslojne. Danas se najviše izučavaju višeslojne neuronske mreže koje pored ulaznih i izlaznih slojeva sadrže i skrivene slojeve.

Vrsta veza između neurona može da bude slojevita gde se na ulaz jednog neurona dovode izlazi svih neurona prethodnog sloja, a njegov izlaz na sve neurone narednog sloja. Predstavnik ove vrste je *backpropagation* algoritam. Potpuno povezane su one kod kojih je izlaz jednog neurona ulaz svih neurona u mreži. Hopfildova neuronska mreža je predstavnik ovog tipa.

Na osnovu smera prostiranja informacija dele se na *feedforeward* – nepovratne i *feedback* – rekurzivne ili povratne neuronske mrže.

### 3.3.1 Nepovratne (*feedforeward*) neuronske mreže

U suštini, sve neuronske mreže imaju sličnu strukturu ili topologiju prikazanu na Slici 3. Za *feedforeward* mreže važi da viši slojevi ne vraćaju informacije nižim slojevima. Prostiranje informacija je samo u jednom smeru, od ulaza ka izlazu.



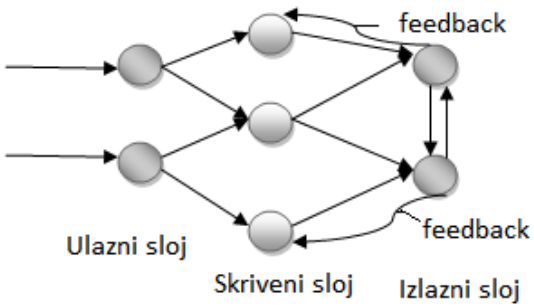

Slika 3: jednostavan primer nepovratne neuronske mreže

Neuronske mreže su više od gomile neurona. Neka ranija istraživanja su proučavala mogućnost nasumičnog povezivanja neurona, ali bez uspeha. Danas je poznato da je čak i mozak puža struktuiran. Jedan od najjednostavnijih dizajna strukture mreže jeste stvaranje slojeva elemenata. Grupisanje neurona u slojeve, konekcije između slojeva funkcije sumiranja i aktivacione funkcije definišu funkcionisanje neuronskih mreža. Ovo su karakteristike koje su zajedničke za sve neuronske mreže.

U većini mreža svaki neuron skrivenog sloja prima signale svih neurona prethodnog sloja, obično ulaznog sloja. Nakon primene funkcije neuroni prosleđuju svoje izlaze svim neuronima narednog sloja, proizvodeći *feedforeward* put do izlaza. Ovakva linija komunikacije je jako važan aspekt neuronskih mreža. *Backpropagation* algoritam je predstavnik ovog tipa mreža.

### 3.3.2 Rekurzivne (*feedback*) neuronske mreže

Primer *feedback* neuronske mreže je prikazan na Slici 4, gde je takva vrsta veze povratna veza od izlaznog do prethodnog sloja. Na slici je prikazan jednostavan primer ovakve vrste mreže:




Slika 4: jednostavan primer rekurzivne neuronske mreže

Kod *feedback* neuronskih mreža viši slojevi vraćaju informacije nižim slojevima. Mogu postojati i veze i između neurona istog sloja. Ovakva vrsta mreža sadrži petlje obzirom da kod njih informacija može da putuje u oba pravca. Hopfield-ove i Elman-ove neuronske mreže su predstavnici ovih vrsta mreža.

## 3.4 Tehnike učenja

Kada je mreža struktuirana, onda je ona spremna za trening. Za početak ovog procesa početne težine se obično biraju nasumice. Zatim sledi trening, odnosno učenje.

Teoretski, neuronske mreže mogu da uče tako što se razvijaju nove konekcije, brišu postojeće, menjaju se težine konekcija, brišu postojeći neuroni ili dodaju novi. Promena težina je najčešći postupak. Brisanje konekcija se može realizovati dodatno, vodeći računa da veze koje imaju težinu nula više ne postoje. Na sličan način mogu da se razviju nove konekcije postavljanjem nepostojeće konekcije na vrednost različitu od nule. Ovo su paradigme učenja koje se koriste za treniranje sinaptičkih težina. Mogućnost dodavanja novih ili brisanje neurona ne proizvode samo dobro prilagođene težine tokom treninga neuronske mreže, već i optimizaciju topologije mreže. Najčešće se realizuju korišćenjem procedure evolucije.

Postoji više paradigmi za učenje korišćenjem neuronskih mreža: nadgledano, *reinforcement* i nenadgledano učenje.

### 3.4.1 Nadgledano učenje

Metode nadgledanog učenja porede izlazne podatke sa očekivanim rezultatima. Razlika između dobijenih i očekivanih podataka se šalje proceduri učenja, koja koriguje težinske koeficijente mreže. Proces se ponavlja sve dok se ne prilagode težine koje kontrolišu mrežu. Set podataka koji omogućava učenje zove se trening set. Tokom treninga mreže, isti set podataka se procesuje više puta sve dok se ne usavrše težinski koeficijenti.

Problem sa neuronskim mrežama je taj, da neke mreže nikad ne nauče da reše problem. Razlog tome mogu da budu trening podaci koji ne sadrže specifične informacije za dobijanje željenog rezultata. Mreže takođe, ne konvergiraju ako nema dovoljno podataka da bi se obavilo učenje. U idealnom slučaju potrebno je da postoji dovoljno podataka i da su uzeti podaci relevantni za postizanje zadatog cilja. Na efikasnost neuronske mreže utiče i broj slojeva, kao i broj elemenata svakog sloja, veze između slojeva, izbor funkcije sumiranja, transfer funkcije. Promene ovih parametara utiču na stvaranje uspešne neuronske mreže.

Postoje mnogi algoritmi koji se koriste za prilagođavanje sinaptičkih težina. Najčešća tehnika nadgledanog učenja je *backpropagation* algoritam.

### 3.4.2 *Reinforcement* učenje

Metode *reinforcement* učenja pružaju povratne informacije, da li se mreža ponaša dobro ili loše. U *reinforcement* učenju mreža prima logičke ili realne vrednosti posle završene sekvence, što ukazuje na to da li je rezultat dobar ili loš. Ovo je vrsta koja se koristi u mašinskom učenju, gde agent za svoje akcije dobija povratni odgovor od strane okruženja. Sistem učenja ocenjuje akcije kao dobre - nagrada ili loše - kazna zasnovane na odzivu okruženja i shodno tome prilagođava svoje parametre. Generalno, parametar prilagođavanja se menja sve dok se ne dostigne stanje ravnoteže.

Koristi se u mnogim disciplinama kao što su teorija igara, teorija kontrole, operacije istraživanja, statistika, genetski algoritmi...

### 3.4.3 Nenadgledano učenje

Nenadgledano učenje je učenje kod koga su samo ulazni podaci dati, ali ne i željeni izlaz. Mreža pokušava da identifikuje slične obrasce i da ih klasifikuje u slične kategorije. Obzirom da podaci za učenje ne sadrže podatke rezultata, stoga nema ni greške, a ni nagrade signala za procenu potencijalnog rešenja. Ovo je glavna razlika u odnosu na nadgledano i *reinforcement* učenje.

Nenadgledano učenje obuhvata mnoge druge tehnike koje sumiraju ili objašnjavaju ključne karakteristike podataka. Pristupi nenadgledanog učenja uključuju: klasterovanje – algoritam k-sredina, hijerarhijsko klasterovanje; smanjenje dimenzionalnosti; razdvajanje signala.

## 3.5 Aktivacione funkcije

Aktivacione funkcije se koriste kod neuronskih mreža za skaliranje izlaznih podataka iz slojeva. Aktivacione funkcije neurona na skrivenim slojevima su neophodne da bi mreža bila u stanju da nauči nelinearne funkcije. Nelinearnost je jako bitna, jer bez nelinearnosti neuroni skrivenih slojeva ne bi imali veće mogućnosti od obične mreže koja se sastoji samo od ulaza i izlaza – perceptronska mreža. Iz tog razloga se na izlazu neurona koristi aktivaciona funkcija koja je nelinearna.

Ovde će biti objašnjene neke aktivacione funkcije, kao što su: binarna, logaritamska, hiperbolički tangens i sigmoidna aktivaciona funkcija. Sve aktivacione funkcije koje imaju prvi izvod mogu da se koriste u *trenning propagation* algoritmima.

### 3.5.1 Binarna aktivaciona funkcija

Izlaz binarane funkcije je 1 ili 0, u zavisnosti od zadatog *threshold*-a, $\theta$.

$$y = \begin{cases} 1, & ako\ je\ u \geq \theta \\ 0, & ako\ je\ u < \theta \end{cases}$$

Ova funkcija se koristi kod perceptrona - najjednostavnija vrsta *feedforeward* neuronskih mreža, algoritam se ne zaustavlja ako podaci nisu linearno separabilni. Binarna funkcija je posebno korisna kod poslednjeg sloja u slučaju da treba da se izvede binarna klasifikacija ulaznih podataka.

### 3.5.2 Linearna aktivaciona funkcija

Linearna aktivaciona funkcija u stvari i nije aktivaciona fukcija, koristi se kada aktivaciona funkcija i nije potrebna. Predstavljena je jednačinom: y = f(x). Izvod ove

funkcije je 1, pa može da se koristi kod propagacionih treninga. Obično se koristi kod izlaznog sloja nepovratnih neuronskih mreža.

### 3.5.3 Konkurentna (*competative*) aktivaciona funkcija

Konkurentna aktivaciona funkcija se koristi da bi se forsirala određena grupa neurona. Pobednička grupa neurona je ona sa najvišom izlaznom vrednošću. Vrednosti svih neurona se čuvaju u nizu koji se predaje ovoj funkciji. Potrebno je definisati veličinu pobedničke grupe neurona, funkcija određuje grupu pobedničkih neurona, dok sve ostale neurone postavlja na vrednost nula. Svi pobednički neuroni dobijaju istu vrednost, koja je srednja vrednost vrednosti pobedničkih neurona.

Ovakva vrsta neurona se koristi za konkurentne neuronske mreže, kao što su samoorganizujuće mape, korisne za vizuelizaciju podataka koji sadrže više dimenzija preko manjeg broja dimenzija.

### 3.5.4 Aktivaciona funkcija blagi maksimum

Ova funkcija skalira sve ulazne vrednosti tako da je zbir jednak jedinici. Često se koristi kao aktivaciona funkcija skrivenog sloja. Koristi se često kod algoritama za klasifikaciju.

### 3.5.5 Logaritamska aktivaciona funkcija

Logaritamska aktivaciona funkcija:

$$f(x) = \begin{cases} log(1+x), & x \geq 0 \\ log(1-x), & u\ suprotnom \end{cases}$$

ulaz u intervalu $(-\infty, \infty)$ konvertuje u interval (-1, 1). Može da bude korisna za sprečavanje saturacije, gde je za dati set ulaznih podataka izlaz u većini slučajeva 1 ili -1. To može značajno da uspori trening. U tom slučaju se koristi logaritamska aktivaciona funkcija.

### 3.5.6 Aktivaciona funkcija hiperbolički tangens

Funkcija hiperbolički tangens:

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

Ulaz u intervalu $(-\infty, \infty)$ konvertuje u interval (-1, 1). Najčešći je izbor kod *feedforeward* neuronskih mreža i jednostavnih rekurentnih mreža.

### 3.5.7 Sigmoidna aktivaciona funkcija

Sigmoidna funkcija bi trebala samo da se koristi kada se kao izlaz očekuju samo pozitivne vrednosti. Ulaz u intervalu $(-\infty, \infty)$ konvertuje u interval (0,1).

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

Koristi se za feedforeward i rekurentne nervne mreže, ali samo onda kada se za trening podatke ne očekuju negativne vrednosti u suprotnom hiperbolički tangens daje bolje rešenje.

## 3.6 Algoritmi učenja

Trening je način na koji neuronske mreže uče, kako da se poboljšaju težine sinaptičkih veza kako bi se postigao željeni rezultat. Postoji veliki broj algoritama za treniranje neuronskih mreža.

Obzirom da softver za rešavanje problema analogije slika koristi *Encog* neuronske mreže, biće objašnjeni samo oni algoritmi koji su podržani od strane ovih mreža. *Encog* podržava dve tehnike učenja nadgledano i nenadgledano učenje. Ovde će biti navedeni algoritmi nadgledanog učenja u cilju pojašnjenja rada softvera.

*Propagation* treniranje može da se koristi kod nepovratnih neuronskih mreža i jednostavnih rekurzivnih mreža. Ova vrsta treniranja se koristi kod nadgledanog učenja. Algoritmu za treniranje se predaje trening set ulaznih podataka i idealni izlaz za svaki ulaz. *Propagation* trening algoritam prolazi kroz serije iteracija i nakon svake pokušava da umanji stopu greške za određeni stepen. Stopa greške predstavlja procenat razlike stvarnog izlaza i izlaza proizvedenog trening algoritmom. Algoritmi koriste diferencijalne funkcije, pa se stoga uzimaju aktivacione funkcije koje imaju prvi izvod. Računa se gradijent greške za svaku konekciju u neuronskoj mreži. Način na koji se ova vrednost koristi zavisi od samog algoritma.

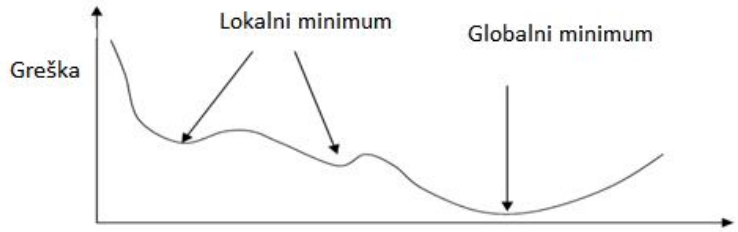
*Propagation* algoritmi koriste višeslojne neuronske mreže. Višeslojne neuronske mreže su podeljene u nivoe. Ne postoje veze između neurona unutar istog sloja, kao ni između ulaznog i izlaznog sloja, neuroni susednih slojeva su potpuno povezani. Broj neurona izlaznog sloja ne zavisi od broja neurona ulaznog sloja. Broj neurona skrivenih slojeva ne zavisi od broja neurona ostalih slojeva. Koliko je skrivenih slojeva i ukupno neurona potrebno, zavisi od mnogo faktora: broja ulaza i izlaza, aktivacione funkcije, kompleksnosti problema, kao i samog algoritma koji se koristi.

Ovde će biti objašnjena tri algoritma: *backpropagation*, Menhetn pravilo ažuriranja i *resilient propagation*.

### 3.6.1 *Backpropagation* algoritam

Ovo je jedna od najstarijih metoda nepovratnih neuronskih mreža. Ovaj algoritam koristi višeslojne neuronske mreže. Problem kod višeslojnih neuronskih mreža je problem pronalaženja težinskih veza između skrivenih slojeva, kada greška može da se izračuna samo za neurone izlaznog sloja. *Backpropagation* se sastoji iz dve faze: *foreward pass* i *backward pass*. Mreža se prvo inicijalizuje postavljanjem svih težinskih koeficijenata na male proizvoljne vrednosti između -1 i 1. Ulazni šablon se primeni i izračuna izlaz. *Foreward pass* čuva vrednosti izlaza na osnovu ulaza i trenutne konfiguracije mreže, dok *backward pass* računa grešku. Izračunavanjem se dobija rezultat koji je različit od ciljnog, dokle god su težine proizvoljne. Računa se greška za svaki neuron. Greška se koristi za promenu težinskih koeficijenata. Težine se menjaju u cilju minimizacije greške. Smanjenjem greške izlaz postaje bliži željenom cilju. Odstupanje izlaznih vrednosti od željenih se prenosi kroz mrežu od izlaza do ulaza. Težine ne menjaju svoju vrednost sve dok se ne završe obe faze. Ovaj proces se ponavlja sve dok se ne postigne minimalna greška.

Jedan od problema *backpropagation* algoritma je problem lokalnog minimuma. Ovaj problem nastaje zbog toga što se algoritam uvek menja u smislu smanjenja greške. To može da dovede do zaglavljivanja na lokalnom minimumu. Problem je što nakon malog povećanja greške može da usledi pronalaženje minimalne greške - globalnog minimuma, kao na Slici 5:




Slika 5: lokalni i globalni minimum funkcije

Postoji više načina da se ovaj problem reši. Jedan od najlakših načina je da se počne ispočetka sa novim nasumičnim postavljanjem težina i treniranjem ispočetka. Druga solucija je dodavanje momentuma na promenu težine. To znači da promena težine u iteraciji ne zavisi samo od trenutne greške, već i od prethodnih promena. Momentum parametar je manji od jedinice, kako bi prethodne promene imale manji uticaj nego promene izračunate u trenutnoj iteraciji.

### 3.6.2 Algoritam - Menhetn pravilo ažuriranja

Jedan od problema *backpropagation* algoritma je stepen promena težina sinaptičkih mreža. Gradijent često pravi prevelike promene na matrici težina. Menhetn pravilo ažuriranja i resilient *propagation* algoritam ne koriste magnitudu gradijenta, već samo znak gradijenta, da li je pozitivan ili negativan. Za Menhetn pravilo ažuriranja

magnituda se koristi samo da odredi kako da se promene vrednosti matrice težina. U slučaju da je vrednost magnitude blizu nule ne prave se nikakve promene, ako je vrednost pozitivna vrednost težina se povećava za određenu vrednost, a ukoliko je negativna vrednost težina se smanjuje za određenu vrednost. Ta vrednost je definisana određenom konstantom.

### 3.6.3 *Resilient propagation* algoritam

*Resilient propagation* trening algoritam je najefikasniji algoritam *Encog* neuronskih mreža nadgledanog učenja za *feedforeward* neuronske mreže.

*Algoritam:*

$\forall i,j: \Delta_{i,j}(t) = \Delta_0$

$\forall i,j: \frac{\partial E}{\partial w_{ij}}(t-1) = 0$

**do**

Računaj gradijent $\frac{\partial E}{\partial w}(t)$

Za sve težine {

**if** $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w}(t) > 0)$ **then**{

$\Delta_{i,j}(t) = minimum(\Delta_{ij}(t-1) * \eta^+, \Delta_{max})$

$\Delta w_{ij}(t) = -sign\,(\frac{\partial E}{\partial w_{ij}}(t) * \Delta_{ij}(t))$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

$\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$

}

**else if** $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w}(t)) < 0$ **then**{

$\Delta_{i,j}(t) = maximum(\Delta_{ij}(t-1) * \eta^-, \Delta_{min})$

$\frac{\partial E}{\partial w_{ij}}(t-1) = 0$

}

**else if** $(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w}(t)) = 0$ **then**{

$\Delta w_{ij}(t) = -sign\,(\frac{\partial E}{\partial w_{ij}}(t) * \Delta_{ij}(t))$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

$\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$

}

}

**while** (konvergira)

Prednost ovog algoritma je što ne zahteva postavljanje parametara za korišćenje. Nema postavljanja parametara kao što su stopa učenja, vrednost momentuma ili

konstante ažuriranja koja treba da se odredi. To je dobro iz razloga što je ponekad teško utvrditi parametre koji daju optimalne rezultate. Ovo čini ovaj algoritam lakim za korišćenje. Algoritam je sličan Menhetn pravilu ažuriranja. Razlikuju se u tome što se ne koristi fiksirana konstanta za ažuriranje vrednosti težina, već mnogo precizniji pristup gde se te vrednosti menjaju tokom procesa treniranja.

Ne postoji globalni parametar ažuriranja, već za svaku vrednost matrice težina postoji posebna delta-vrednost. Ove vrednosti su u početku proizvoljno inicijalizovane na jako male vrednosti. Posle svake iteracije ove vrednosti se ažuriraju u skladu sa delta-vrednostima. Magnituda gradijenta se koristi da bi se utvrdilo kako delta-vrednosti treba da se menjaju. Na ovaj način svaka matrica težina može individualno da bude trenirana.

Reprezentacija ulaznog vektora je $x^p$, a ciljnog $t^p$. *Feedforeward* izračunavanjem dobija se rezultet $s^p$. Porede se rezultati $s^p$ i $t^p$, računa se rastojanje ovih vektora $E = \frac{1}{2}\sum_p \sum_n |t^p - s^p|^2$, gde je n broj neurona izlaznog sloja. Opadajući $\eta^- = 0.5$ i rastući $\eta^+ = 1.2$ faktor. Ograničenja: $\Delta_{max} = 50.0$ i $\Delta_{min} = 10^{-6}$ gornje granice. Inicijalna vrednost $\Delta_o = 0.1$.

# 4 Analogije slika

Digitalna obrada slika je bila tema prvog poglavlja, gde je opisan tradicionalni način poboljšanja slika. U ovom poglavlju je opisan drugačiji način obrade digitalnih slika, procesovanje slika korišćenjem uzoraka - analogije slika. U cilju postizanja željenog efekta algoritmi za obradu slika koriste uzorke slika: originalne slike i njima odgovarajuće filtrirane slike. Ovde će biti akcenat na metodama koje za učenje određenog filtera koriste samo jedan primer.

Tehnike koje koriste analogije slika kao novi način obrade digitalnih slika koriste se za učenje različitih vrsta filtera. Izborom različitih vrsta trening slika kao ulaznih, analogije slika mogu da podrže raznovrsne efekte. Neke od primena su: učenje tradicionalnih filtera, sinteza tekstura, kolorizacija, super-rezolucija, umetnički filteri i druge.

## 4.1 Analogije slika definicija i osnove

Analogija je osnovni proces rezonovanja. Ljudi često koriste analogije, a da toga obično nisu ni svesni. Koriste ih za predviđanje ili rešavanje različitih vrsta problema. Generalizacija na osnovu seta poznatih primera je centralni problem mašinskog učenja. Iz ovog razloga cilj veštačke inteligencije još od početka razvoja ove oblasti je bio da se napravi sistem koji bi bio sposoban da rezonuje korišćenjem analogije.

Ovde će biti opisana istraživanja upotrebe analogije u smislu učenja odabranih filtera. Termin analogije slika uvodi *Hertzmann* 2001. godine u svom radu "Analogije slika" ("*Image analogies*") [9]. Polazeći od odnosa:

$$A : A' = B : B',$$

postavlja se problem dobijanja slike $B'$ ako su poznate slike $A$, $A'$ i $B$. Ideja je da se na osnovu originalne i filtrirane slike nađe funkcija preslikavanja tako da nova originalna slika bude u istom odnosu sa svojom filtriranom slikom kao i prethodna. Ovde se kao trening podaci koriste dve slike: originalna slika $A$ i filtrirana slika $A'$. Opisana metoda se sastoje iz dve faze. Prva faza je faza dizajna gde se uči odabrani filter na osnovu trening skupa podataka. Druga faza je faza primene, gde se naučeni filter primenjuje na proizvoljno odabrane nove slike. Kasnije, 2002. godine *Hertzmann* koristi isto okruženje za analogiju krive [10], učenje stila linija na osnovu datog primera.

Prednost analogije slika je da obezbedi prirodne transformacije slika umesto izbora različitih filtera i njihovih podešavanja. Korisnik jednostavno može da izabere odgovarajući efekat i isti proizvede na novoj slici koristeći analogije slika. Uzimajući ovo u obzir filteri ne moraju individualno da budu pronađeni ili posebno programirani, idealno, isti mehanizam bi mogao da se iskoristi da se proizvede širok spektar efekata. Analogije slika jasno opisuju svoju svrhu, ali nije sasvim jasno kako mogu da se realizuju i dostignu željeni cilj.

Ključni aspekt kod analogije slika je problem izbora definicije sličnosti koja će da se koristi ne samo za odnose između svake nefiltrirane i njoj odgovarajuće filtrirane slike, već i za odnose između trening para slika i ciljnog para slika kao celine. Ovo pitanje je jako trikovito, u smislu da jedna metrika koja može da očuva prepoznatljivost osobina odnosa između slika *A* i *A'*, u isto vreme može da bude suviše loša da bi se primenila na sasvim drugačiju sliku *B*. Nije očigledno da li karakteristike trening skupa pokrivaju sve transformacije odabranog filtera.

Ne možemo da očekujemo da analogije slika rešavaju problem učenja svih mogućih filtera uzimanjem samo jednog trening para slika. Iznenađujuće je ipak da se korišćenjem analogija mogu naučiti mnoge vrste filtera.

## 4.2 Tehnike i primene

Ovde će biti navedeni metodi koji koriste uzorke slika u cilju poboljšanja digitalnih slika. Akcenat je na metodama koje za obradu digitalnih slika koriste jedan uzorak. Pojedini algoritmi daju rešenja za širi spektar učenja filtera, dok su pojedini skoncentrisani na pojedinačne efekte u cilju dobijanja boljih rezultata.

### 4.2.1 Analogija slika *Hertzmann*-ov algoritam

*Hertzmann* [9] kao ulaz u algoritam koristi tri slike, nefiltriranu izvornu sliku *A*, filtriranu izvornu sliku *A'* i nefiltriranu ciljnu sliku *B*, a kao izlaz daje filtriranu ciljnu sliku *B'*. Predloženi algoritam se sastoji iz dve faze. U prvoj fazi dizajna filtera, uzimaju se dve slike originalna i njoj odgovarajuća filtrirana slika kao trening podaci. U drugoj fazi primene naučeni filter se primenjuje na odabranu ciljnu sliku u cilju stvaranja analogije, filtriranog izlaza. Mera sličnosti je zasnovana na aproksimaciji Markovljevog slučajnog modela i koristi vrednosti piksela za učenje odabranog filtera. Za izračunavanje odnosa između izvornog i ciljnog para slika, koriste se statistike odabranih okolina piksela.

Boje na i oko bilo kog datog piksela *p* slike *A* odgovaraju bojama na i oko tog istog piksela *p* slike *A'*, filter koji treba da se nauči. Indeks *p* se koristi za piksele na slikama *A* i *A'*, a indeks *q* za slike *B* i *B'*. Ovaj metod ne uzima u obzir da su slike predstavljene samo preko *RGB* kolor modela, već se uzimaju u obzir i druge karakteristike kanala kao što je osvetljenje. Zajedno svi kanali uključujući i *RGB* čine vektor osobina (n-dimenzionalni vektor gde su osobine objekta predstavljene numerički) za svaki piksel *p*. Koristi se *A(p)* odnosno *A(p')* da se označi vektor karakteristika u pikselu *p*, slično vektor *B(q)* odnosno *B(q')* da označi vektor karakteristika u pikselu *q*. Ovaj vektor se koristi u procesu poklapanja, gde se pronalaze odgovarajući pikseli slike *A'* koji će biti iskorišćeni za stvaranje slike *B'*. Potrebno je voditi računa o poziciji *p* izvornog piksela koji se kopira u piksel *q* ciljne slike. Ove informacije se skladište u strukturu *s(.)*; npr. *s(q) = p*. Svaki od ovih pet parametara biće predstavljen i preko skale različitih nivoa rezolucije. Ako $A_l$ predstavlja izvornu sliku date rezolucije, onda $A_{l-1}$

predstavlja sliku sa duplo manjim brojem piksela u obe dimenzije. Maksimalna rezolucija $L$ uzima se kao maksimalni nivo rezolucije slike.

U prvoj fazi algoritma koristi se višeslojna reprezentacija slika $A$, $A'$ i $B$ (Gausova piramida), zajedno sa vektorom osobina kao i aproksimativnom metodom najbližih suseda (*approksimate nearest neighbour ANN*) za ubrzavanje procesa poklapanja. Na svakom nivou $l$, od najgrubljeg do najfinijeg nivoa rezolucije $l$, statistike koje se koriste za svaki piksel $q$ ciljnog para porede se sa statistikama svakog piksela $p$ izvornog para i na taj način se pronalazi najbolje poklapanje koje se skladišti u $s_l(q)$.

*Algoritam:*

***function*** *kreiranje_analogije_slika(A, A', B)*
    *računanje Gausove piramide za slike A, A' i B*
    *računanje osobina za A, A' i B*
    *inicijalizacija strukture pretrage (korišćen je ANN metod)*
    ***for*** *za svaki nivo l od najgrubljeg do najfinijeg* ***do***
        ***for*** *za svaki piksel* $q \in B'_l$ *u redosledu sken linija* ***do***
            $p \leftarrow$ *najbolje_poklapanje (A, A', B, B', s, l, q)*
            $B'_l(q) \leftarrow A'_l(p)$
            $s_l(q) \leftarrow p$
    ***return*** *B'*
***end***

Osnova ovog algoritma je funkcija *najbolje_poklapanje*. Ova funkcija pronalazi piksel $p$ u izvornom paru slika koji se najbolje poklapa sa pikselom $q$ slike $B'$. Koristi dve različite metode: aproksimativnu i koherentnu pretragu. Aproksimativna pretraga pokušava efikasno da pronađe najbliže piksele pokalapanja na osnovu vektora osobina $p$ i $q$ i njihovih suseda, dok koherentna pretraga pokušava da sačuva koherentnost sa susednim sintetizovanim pikselima. Što je veća vrednost parametra $k$, favorizovana je veća koherentnost prilikom izračunavanja. Da bi parametar koherencije bio koenzistentan u odnosu na različite skale rezolucije, vrednost je pomnožena faktorom $2^{l-L}$, jer su lokacije piksela na grubljoj skali više razmaknute od piksela na finijoj skali.

U ovom istraživanju pokazano je da korišćenjem ovog algoritma nije bilo dovoljno korišćenje *RGB* informacija za svaki piksel. Korišćenje ove informacije za vektor osobina pokazalo se nedovoljno za učenje mnogih filtera. Za vektor osobina iskorišćena je komponenta osvetljenja svakog piksela za izračunavanje metrike rastojanja. Razlog tome jeste činjenica da je ljudsko oko osetljivije na intezitet osvetljenja nego na boje. Osvetljenje može da se izračuna na različite načine, ovde je korišćen $Y$ kanal $YIQ$ prostor boja. Kanali $I$ i $Q$ su komponente za boju. Nakon završetka procesa gde se koristi osvetljenje, boja na slici se dobija kopiranjem komponenata $I$ i $Q$ slike $B$ na sliku $B'$, propraćeno konverzijom u *RGB*. Na ovaj način se ubrzava i celokupan postupak.

Nakon završetka faze dizajna filtera, filter se čuva u biblioteci, gde u drugoj fazi naučeni filter može da se primeni na druge izabrane slike. Izborom različitih izvornih

slika kao ulaznih, predloženi algoritam podržava učenje raznovrsnih vrsta filtera: tradicionalnih filtera kao što su *blurring* i *embossing*; super-rezolucije gde se uči dobijanje slike veće rezolucije na osnovu date slike koja je niže rezolucije; različiti umetnički filteri gde se uče različiti stilovi crtanja; sinteze tekstura koja se odnosi na konstrukciju velike slike na osnovu malog uzorka uzimanjem u obzir strukturu slike (može da se koristi za popunjavanje rupa na slici).

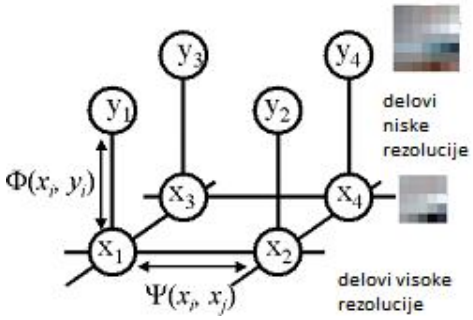
### 4.2.2 Algoritmi za izoštravanje slika

Interpretacija slike zavisi od njene oštrine, od mogućnosti izdvajanja informacija koje slika sadrži. Oštrije slike pokazuju više detalja. Međutim, nije lako definisati šta je oštra slika. Mogla bi da se definiše kao slika koja izgleda kao prirodna scena, ali ono što je ljudskom oku prirodno nije lako da se izračuna. Obično se za definiciju zamućene slike uzima definicija slike koja je izgubila visoko frekventne informacije. Slike koje dobijamo korišćenjem fotoaparata su često zamućene. Neki od razloga mogu da budu nenamerni pokreti kamere, pogrešna podešavanja ili loše osvetljenje. Zumiranjem ili kompresijom slika se često dobijaju slike lošije rezolucije.

Ako je u pitanju veliki broj slika koje treba da se obrade i da se od njih dobiju izoštrene slike, korišćenje bilo kog softvera za standardnu obradu slika bi bilo jako teško. Svaka slika bi pojedinačno morala da se obradi. Standardne metode kao što je izoštravanje ivica, ovaj problem ne mogu da reše. U principu nije moguće stvoriti informacije koje nedostaju i koje odgovaraju delovima koji ne mogu da se vide sa originalne slike. Međutim, moguće je da se pogode informacije koje nedostaju. Taj način bi omogućio povećanje rezolucije originalne slike – super rezolucija.

Jedan od pristupa za rešavanje ovog problema je već opisani *Hertzmann*-ov algoritam koji uzima dve slike kao trening podatke: originalnu sliku niže rezolucije i njoj odgovarajuću sliku više rezolucije. Korišćenjem analogija, učenja preslikavanja na osnovu primera, za novoizabranu sliku niže rezolucije, dobija se slika veće rezolucije.

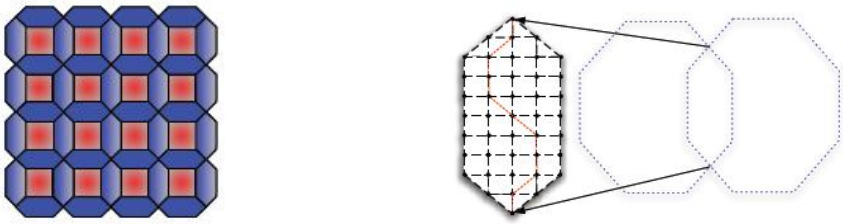
*Freeman* je predložio algoritam [11] koji objašnjava kako da se pogode opsezi visokih frekvencija koji nedostaju na slici. Algoritam uzima trening slike kao reference sa ciljem da se nauči izoštravanje slike. Za trening podatke algoritam koristi set slika niske rezolucije, obično 7x7 piksela i njima odgovarajuće slike visoke rezolucije, obično 5x5 piksela. Kao ulaz se uzima slika koju želimo da zumiramo, a koja je prethodno uvećana za određeni faktor, obično 2. Slika je podeljena na delove i za svaki deo se traže najbliži primeri iz trening skupa, delovi niže rezolucije i njima odgovarajući delovi više rezolucije. Trening skup se koristi da bi se izračunale matrice verovatnoće (Slika 6) $\Psi$ (horizontalni odnos između delova visoke rezolucije) i $\Phi$ (vertikalni odnos između delova niske i njima odgovarajućim delovima visoke rezolucije). Optimalni deo visoke rezolucije je onaj koji maksimizuje verovatnoću Markovljeve mreže. Algoritam je efikasan ali je spor.



Slika 6: horizontalni i vertikalni odnos slika visoke i njima odgovarajući delovi niske rezolucije

Za poboljšanje kvaliteta ličnih fotografija u algoritmu [12] kao trening podaci koristi se omiljeni skup ličnih fotografija. Sistem koristi detekciju lica kako bi se napravila razlika između dobrih i loših fotografija, tako da se osobine dobrih primera mogu iskoristiti u popravljanju loših fotografija. Algoritam se sastoji iz pet faza: automatskog detektovanja lica ciljne slike i trening slika, segmentacija lica ciljne i trening slika, dekompozicija slika na boju, teksturu i osvetljenje, globalna korekcija i korekcija koja se odnosi samo na lica.

Za razliku od *Hetzmann*-ove ideje za rešavanje problema analogije slika u algoritmu [13] se koristi *semi-supervised* tehnika učenja, Markovljev slučajni model da bi se obezbedila globalna koenzistentnost i *image quilting* tehnika da bi se obezbedila lokalna koenzistentnost. *Image quilting* algoritam [14] daje dobre rezultate u rešavanju problema sinteze teksture, gde se na osnovu malog uzorka formira veća slika naučene teksture. Naučena tekstura može da se primeni i na neki novi objekat. Algoritam za sliku *A* uzima sliku manje rezolucije, a za sliku *A'* istu sliku, samo veće rezolucije. Kada se nauči zadati filter konvertovanja slike niže rezolucije u sliku više rezolucije, tada isti filter može da se primeni na proizvoljan broj drugih slika i da proizvede efekat naučenog filtera. Za razliku od *Hertzmann-ovog* algoritma, ovaj algoritam nudi mogućnost, da ne mora cela slika *A'* da bude poznata već samo neki označeni deo slike *A* što rešava tehnika *semi-supervised* učenja. Ova tehnika se koristi kada je poznat mali broj označenih trening podataka, a kada je veliki broj neoznačen. Pored ovoga nefiltrirana ciljna slika *B* je dostupna za vreme procesa treniranja i može da se koristi kao izvor za neoznačene primere, za razliku od *Hertzmann-ovog* algoritma. Da bi se rešio problem lokalne koenzistencije, naročito oko granica, koristi se *image quilting* algoritam. Za datu teksturu *A* i ciljnu sliku *B*, algoritam deli sliku *B* na male preklapajuće delove (osmouglove u ovom slučaju) i proizvodi kandidate tekstura za ove delove, nasumičnim sampliranjem slike *A*. Ovi kandidati se spajaju dinamičkim programiranjem u cilju smanjenja graničnih grešaka. Na ovaj način se rešava problem lokalne koenzistencije, dok se struktura celokupne slike ignoriše.



Slika 7: (a) podela slike na osmougaone delove (b) minimalna granična greška nastala spajanjem dva osmougaona dela

Na Slici 7 (a) je prikazana podela ulazne slike na osmougaone delove. Pikseli mogu da pripadaju jednom osmougaonom delu – označeni su crvenom bojom, dok su plavom bojom označeni pikseli koji pripadaju preseku dva osmougaona dela. Da je slika podeljena na kvadrate pikseli bi pripadali jednom delu, bili u preseku dva ili četiri dela. Slika 7 (b) pokazuje minimalnu graničnu grešku proizvedenu spajanjem dva susedna dela dinamičkim programiranjem.

Markovljev slučajni model se koristi za rešavanje problema globalne koenzistencije, ovaj algoritam se pokazao kao dobar za rešavanje problema super-rezolucije. Markovljev slučajan model ili Markovljeva mreža je uopštenje Markovljevog lanca u više dimenzija. Kod Markovljevog lanca naredno stanje zavisi samo od prethodnog. Kod Markovljevih mreža svaka slučajna promenljiva zavisi od susednih promenljivih s kojima je povezana.

Digitalne aplikacije, kao što su *Google Earth* i *World Wind*, omogućavaju nam da istražujemo stvarne podatke o zemljinoj površini. Da bi se prikazali različiti detalji Zemljine površine, potrebno je da se slike koje su dobijene preko satelita izoštre tako da se dobiju slike veće rezolucije. U nekim slučajevima dobijanje visoke rezolucije satelitskih snimaka je jako skupo. U radu [15] opisan je metod koji koristeći primere tehnike super-rezolucije i analogije slika popravlja vizuelni kvalitet satelitskih snimaka. Primer slike visoke rezolucije i niske rezolucije istog mesta se koriste kao primer za formiranje filtera super-rezolucije.

### 4.2.3 Kolorizacija

Bojenje *grayscale* slika je težak problem. Jako je teško da se oboji slika ako prethodno nemamo informacije o toj slici. Isti predmeti mogu da budu različito obojeni, npr. plava i zelena lopta koje imaju isti oblik, strukturu i koje su napravljene od istog materijala. Problem je i sa bojenjem prirodnih objekata. Lišće je tokom proleća zeleno, dok je braon boje tokom jeseni. Mnoga rešenja za ovaj problem se sastoje od uzimanja informacija o boji od samog korisnika softvera. Korisniku je obično dozvoljeno da definiše boju pojedine oblasti, a potom se ta informacija koristi za bojenje cele slike.

*Levin* [16] predlaže metod koji se zasniva na jednostavnoj pretpostavci: susedni pikseli u prostoru koji imaju sličan intezitet, treba da budu i sličnih boja. Ovaj algoritam ne zahteva tačnu segmentaciju slike. Potrebno je da korisnik (umetnik) označi boju svakog regiona slike koji želi da oboji.

Za razliku od ovog pristupa *Irony-jev* [17] algoritam nalazi nekoliko tačaka na slici koje je algoritam pravilno obojio. Zatim se primenjuje *Levin-ov* pristup, kao da su to tačke koje je korisnik označio. Ovaj pristup se zasniva na predikciji koja je zasnovana na osnovu seta obojenih slika, delimično segmentiranih u regione od strane korisnika. Nova slika koja treba da se oboji se automatski deli na regione čija je tekstura slična sa regionom neke od prethodno odabranih slika. Ovaj metod smanjuje napor korisnika, ali je preprocesovanje i dalje manuelno.

### 4.2.4 Segmentacija slike

Segmentacija slike se odnosi na postupak podele slike na regione sa sličnim atributima. Od atributa se najčešće koristi osvetljenost ako su u pitanju *greyscale* slike, dok se boja koristi za slike u boji. U procesu segmentacije mogu da se koriste različite karakteristike kao što su mere teksture i ivice. Segmentacija je u analizi slika jedan od prvih i najvažnijih koraka, naročito u pretrazi multimedijalnog sadržaja. Koristi se i u medicinskoj obradi slike. Veliki broj postupaka segmentacije koji se koristi u praksi je heurističkog karaktera. Teorijski ne postoji parametar za kvantitativnu procenu koliko je neki postupak segmentacije dobar. Jedan od metoda za segmentaciju je određivanje praga (*threshold*) osvetljenosti. Za prag može da se odabere i više od jednog parametra. Drugi način je korišćenje tehnike nenadgledanog učenja kao što je klasterovanje.

U medicinskoj industriji se prikuplja veliki broj podataka, dok mali broj može pravilno da se analizira. Ne postoji pouzdan način da se brzo segmentuje velika količina podataka. U radu [18] koriste se analogije slika za rešavanje problema segmentacije slika. Kao tehnika učenja segmentacije korišćena je metoda nadgledanog učenja. Za trening podatke korišćene su medicinske slike koje su pregledane od strane stručnjaka, anatoma.

Za set slika $\{m_{0,\ldots\ldots,}m_n\}$ i segmentovanu sliku od strane anatoma, naivna aplikacija računa analogije:

$$m_0 : s_0 :: m_1 : s_1$$
$$m_0 : s_0 :: m_2 : s_2$$
$$\ldots$$
$$m_0 : s_0 :: m_n : s_n$$

Ovo daje jako loše rezultate, pa je ovaj metod unapređen korišćenjem progresivnog izračunavanja analogija:

$$m_0 : s_0 :: m_1 : s_1$$
$$m_1 : s_1 :: m_2 : s_2$$
$$\ldots$$
$$m_{n-1} : s_{n-1} :: m_n : s_n$$

### 4.2.5 Podešavanje tonova, osvetljenje i kontrast

Podešavanje tonova slike je važan aspekt u digitalnoj obradi fotografije. Automatsko podešavanje osvetljenja i kontrasta na fotografiji bi omogućilo da se na jednostavan način dobije fotografija koja liči na razglednicu. Manuelno podešavanje ovih atributa je zahtevan posao. Mnogi paketi pružaju automatsko podešavanje slika, kao što su histogrami istezanja i ekvalizacije. Takve jednostavne heuristike ne prave razlike između ključnih i manje važnih scena, kao ni scena sa pozadinskim osvetljenjem.

Za automatsko popravljanje osvetljenja i kontrasta u radu [19] predložen je metod koji koristi tehniku nadgledanog učenja. Kao i u svim pristupima koji se zasnivaju na učenju, kvalitet podataka koji se koristi za trening je od ključnog značaja. Kao trening podaci korišćeni su primeri slika koji pokrivaju širok spektar scena, tema i uslova osvetljenja. Fotografije koje se koriste kao trening su obrađene od strane umetnika. Takav par originalne i obrađene fotografije se koristi kao trening podatak za novoizabranu sliku.

# 5 Predloženo rešenje

Ovo poglavlje je spoj prethodna tri poglavlja. Tradicionalni način digitalne obrade slika iskorišćen je za dobijanje velikog broja trening podataka. Za učenje preslikavanja proizvoljno odabrane nefiltrirane slike – *model* i njoj odgovarajuće filtrirane slike – *master* koriste se neuronske mreže.

Priloženi softver istražuje mogućnost rešavanja problema analogije slika. U ovom poglavlju biće opisana metodologija i implementacija predloženog rešenja problema grafičkih analogija, radno okruženje i alati koji su korišćeni, kao i način korišćenja softvera kroz glavne funkcije programa. Na kraju će biti prikazani dobijeni rezultati. Mogućnost korisnički vođenog stvaranja filtera biće prikazana kroz jedan primer, više izbora novih slika kroz više iteracija.

## 5.1 Predloženo rešenje vs. tradicionalni pristup obrade slika

Tradicionalni pristup digitalne obrade slika se radi u dva domena spacijalni i frekventni domen (objašnjeno u poglavlju 2). U spacijalnom domenu slika se predstavlja kao dvodimenziona funkcija *f(x,y)*, gde *x* i *y* predstavljaju spacijalne koordinate. Vrednost funkcije *f* je intezitet sive u tački *(x,y)*. Obrada slike u spacijalnom domenu odnosi se na direktne manipulacije sa pikselima slike. Digitalna obrada slika u frekventnom domenu se vrši tako što se slika Furijeovim transformacijama prevede u prostor frekvencija gde se vrši obrada, a nakon toga inverznom Furijeovom transformacijom slika ponovo prevodi u spacijalni domen. U frekventnom domenu jednostavnije je da se uoče neke važne osobine slike kao što je detekcija ivica i otklanjanje šuma.

Za razliku od tradicionalne obrade slika u radu je predložena metoda za obradu slika korišćenjem analogija, kao i korisnički vođenog stvaranja filtera. Ideja je da korisnik ne mora da bude programerski stručan da bi stvorio svoju paletu filtera. Umesto izbora specifičnog filtera korisnik bira filter koji može da bude i kombinacija primene više filtera i predaje softveru koji uči željeno preslikavanje. Opcija, odvajanje od modela i stvaranje filtera vođenog ličnim ukusom omogućava korisniku da ne zavisi od proizvođača filtera. Ako je u pitanju veliki broj slika koje treba da se obrade i da se od njih dobiju obrađene slike koje uključuju primenu velikog broja filtera, korišćenje bilo kog softvera za standardnu obradu slika bi bilo jako teško. Svaka slika bi pojedinačno morala da se obradi. Prednost korišćenja predloženog softvera ogleda se i u činjenici da jednom naučeni filter može da se primeni na proizvoljan broj novoizabranih slika.

Kao trening podaci uzete su slike dobijene tradicionalnim pristupom obrade slika kako bi se ispitao kvalitet i pokazale mogućnosti predloženog rešenja.

## 5.2 Radno okruženje i alati

Projekat je napisan u *JRuby* programskom jeziku. *JRuby* je *Java* implementacija *Ruby* programskog jezika, što omogućava korišćenje svih *Java* biblioteka pisanjem *Ruby*

koda. Dinamičan je i objektno orijentisan programski jezik. Sintaksa *Ruby* jezika inspirisana je programskim jezicima *Perl* i *Smalltalk*. Korišćena je verzija 1.9.

Napisane su dve verzije programa. U prvoj verziji programa korišćene su jednostavne neuronske mreže i evolucija kao proba izvodljivosti generalne ideje. Druga verzija, čiji će rezultati u ovom radu biti izloženi koristi *Encog3* neuronske mreže.

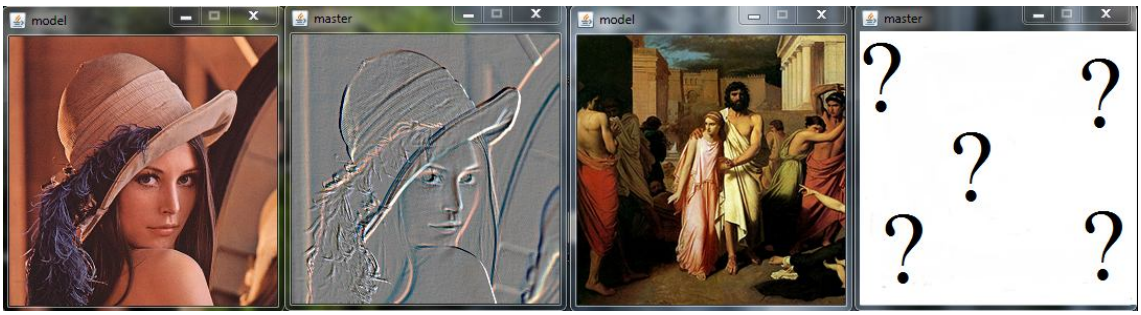
*Encog3* je napredno okruženje neuronskih mreža i mašinskog učenja. Sadrži klase za kreiranje različitih vrsta mreža, takođe podržava i klase za normiranje i procesovanje podataka za odgovarajuću vrstu mreža. Propagira višenitno programiranje. Postoji i radno okruženje u kome mogu da se modeluju i treniraju neuronske mreže. Radi u mnogim integrisanim razvojnim okruženjima kao što su *Eclipse* ili *Netbeans*. Distribuira se kao JAR fajl. Projekti su rađeni u Netbeans programskom okruženju.

*Encog3* se koristi za pravljenje rekurentnih i *feedforeward* neuronskih mreža. Za kreiranje se koriste *BasicNetwork* i *BasicLayer* klase. Pored ove dve klase, koriste se i aktivacione funkcije. Podrazumevana aktivaciona funkcija je hiperbolički tangens. Pored ove aktivacione funkcije *Encog3* podržava i binarnu, linearnu, sigmoidnu, logaritamsku i još neke aktivacione funkcije, gde sama priroda problema nalaže korišćenje specifične funkcije. Veoma efektivna forma treniranja *feedforeward* i jednostavnih rekurentnih neuronskih mreža je *propagation training*. Neke *Encog3* forme *propagation* treninga su: *Backpropagation*, *quick propagation*, *Manhattan Update Rule*, *Resilient propagation*.

## 5.3 Implementacija i metodologija predloženog rešenja

Osnovna ideja ovog projekta dolazi od suštinske ideje analogije slika, gde se na osnovu originalne slike *A* i filtrirane slike *A'*, za datu sliku *B* nalazi slika *B'* takva da je:

$$A : A' = B : B'$$




Slika 8: (a) slika *A* (b) slika *A'* (c) slika *B* (d) slika *B'*

Na Slici 8 prikazan je primer problema. U prozoru *model* (a) odabrana je originalna slika *A*, u prozoru *master* (b) odabrana je slika *A'*, a u prozoru *model* (c) prikazana je novoodabrana slika *B*. Potrebno je pronaći sliku *B'* (d) tako da je ona u istom odnosu sa svojom originalnom slikom *B,* kao i *A* sa *A'*. Učenje preslikavanja slike *A* u sliku *A'* (*model-master* preslikavanje) u programu je prikazano u prozoru *apprentice*.

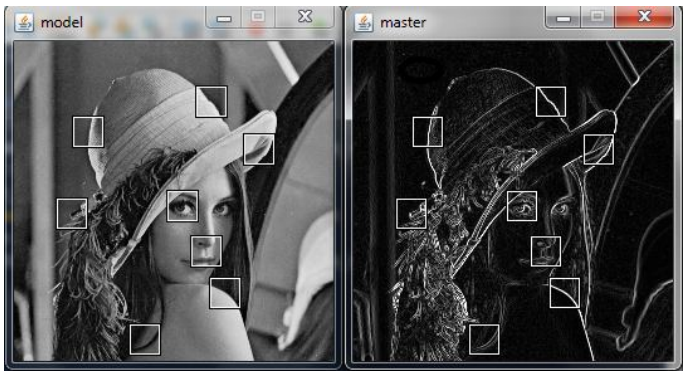
Prva verzija programa, o kojoj ovde neće biti reči, a koja je u principu bila od suštinske važnosti za sam rad, koristi jednostavne rekurzivne neuronske mreže i evoluciju kao način učenja odabranog filtera. Radi sa slikama koje nisu u boji (*grayscale*). Ova verzija je bila više eksperimentalnog karaktera, u smislu da je dala smernice da u tom pravcu može da se postigne očekivani rezultat. Uspešno se pokazala u učenju mnogih jednostavnih filtera kao što su negativ, kontrast, osvetljenje. Druga ideja, da korisnik sam evoluira svoje filtere i tako postane nezavisan od proizvođača istih, proistekla je iz ove verzije programa, tako da je iz tog razloga jako važno pomenuti je. Još jedan razlog zbog koga je prva verzija važna je čisto istraživačke prirode, da ukaže na put kojim je ovo istraživanje išlo.

Druga verzija programa koja pokriva i temu ovog rada koristi *Encog3* neuronske mreže. Razlog korišćenja *Encog3* neuronskih mreža je to što su jako efikasne, testirane su na mnogim problemima za koje je rešenje već poznato. U drugoj verziji je dodata i opcija izbora slike u boji.

### 5.3.1 Izbor tehnike učenja i trening podataka

Kao ulaz u program uzimaju se dve slike, originalna slika *A* (*model*) i filtrirana slika *A'* (*master*). Neuronske mreže uče *model–master* preslikavanje i nakon zadatog broja generacija, dostignuti rezultat se iscrtava u prozoru *apprentice*. Iscrtavanje generacija je dato kako bi proces učenja mogao da se prati vizuelno. Za tehniku učenja koristi se metoda nadgledanog učenja obzirom da su nam poznati i ulazni podaci i očekivani rezultati.

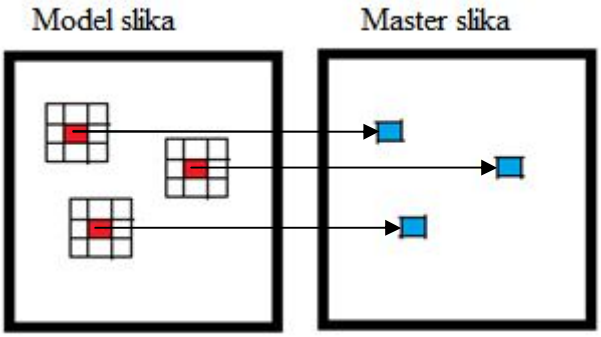
U cilju rešavanja ovog problema u programu je ponuđena opcija izbora trening skup podataka. Omogućena su dva načina: izbor broja random skupa piksela na ulaznim slikama (*model* i *master*) ili ručni odabir površina slike koje će se koristiti kao trening podaci.



Slika 9: primer ručnog odabira trening skupa podataka na originalnoj (*model*) i njoj odgovarajućoj filtriranoj slici (*master*)

Mogućnost ručnog odabira trening skupa je jako važna onda kada znamo koji su trening podaci najbolji da se predaju algoritmu učenja, u cilju dobijanja što preciznijih rezultata. Ako uzmemo na primer, učenje filtera detekcije ivica, najbolje je odabrati

delove slike koji sadrže ivice kao trening podatke umesto izbora random skupa piksela, na Slici 9, su označeni trening podaci za učenje Sobel filtera za detekciju ivica, pomenutog u poglavlju 2.




Slika 10: jednostavan primer trening skupa podataka od 3 piksela, crvenom bojom su označeni ulazni podaci, dok su plavom bojom označeni očekivani rezultati

Boje na i oko bilo kog datog piksela *p* slike *model* odgovaraju bojama na i oko tog istog piksela *p* slike *master*, filter koji treba da se nauči (Slika 10). Crvenom bojom, na slici *model* su označeni pikseli slike koji su odabrani kao ulazni podaci, dok su plavom bojom na slici *master* označeni očekivani rezultati. Na Slici 10 su označena 3 piksela kao trening podaci.

Problem sa neuronskim mrežama u učenju datog filtera može da bude premali ili pogrešno odabrani trening skup podataka. To mogu da budu trening podaci koji ne sadrže specifične informacije za dobijanje željenog rezultata, pa je potrebno voditi računa pri odabiru trening skupa.

Procedura učenja uzima dobijene i očekivane rezultate i poredi ih, nakon čega se koriguju težinski koeficijenti mreže. Tokom treninga mreže, isti set podataka se procesuje više puta sve dok se ne usavrše težinski koeficijenti. Postupak se ponavlja dok se ne zadovolji uslov zahtevane sličnosti – *fitness*. Program može da bude prekinut i onog momenta kada je korisnik zadovoljan vizuelnom sličnošću koja je prikazana u prozoru *apprentice*. Nakon završetka, program pamti neuronsku mrežu, kao .eg fajl, koja može da bude primenjena na novoodabranu sliku *B* kako bi se na novoj slici primenio naučeni efekat i proizvela slika *B'*.

### 5.3.2 Izbor arhitekture mreže

Predloženi softver koristi *Encog3* neuronske mreže. Prilikom implementacije softvera bilo je potrebno da se naprave određene odluke, u smislu odabira arhitekture, aktivacione funkcije i algoritma za treniranje neuronskih mreža.

Odabrane su *feedforeward* neuronske mreže, koje su detaljno opisane u poglavlju 3. Ove mreže se sastoje iz tri sloja: ulaznog, skrivenog i izlaznog sloja neurona. Viši slojevi ne vraćaju informaciju nižim slojevima, već je prostiranje informacija samo u jednom smeru od ulaza ka izlazu (primer je prikazan na Slici 3). Grupisanje neurona u slojeve, konekcije između slojeva, funkcije sumiranja i aktivacione funkcije definišu

funkcionisanje neuronskih mreža. Mreža je struktuirana tako da svaki neuron skrivenog sloja prima signale svih neurona prethodnog sloja.

Pored odabiranih piksela koji se koriste kao trening podaci u programu se kao parametar predaje i veličina retine, parametar koji utiče na broj neurona ulaznog i skrivenog sloja. Na Slici 10 (pogledaj *model*) ovaj parametar ima vrednost 3, pa je retina veličine 3x3. Za retinu 3x3 broj neurona ulaznog sloja za sliku u boji je 3*9, jer slika u boji ima 3 banda, jedna komponenta za crvenu, druga za zelenu, treća za plavu boju. Broj izlaznih neurona je u tom slučaju 3. Za *grayscale* sliku i retinu 3x3 broj ulaznih neurona bi bio 1*9, dok bi broj izlaznih neurona bio 1. Korišćen je RGB kolor model.

Na efikasnost neuronske mreže utiče kako broj elemenata svakog sloja tako i broj slojeva. U programu je ponuđena opcija izbora broja skrivenih slojeva čija vrednost ne mora da bude jedan kao što je prikazano na Slici 3, kao i parametar vrednosti svakog skrivenog sloja koji pokazuje koliko je puta broj neurona skrivenog sloja veći od broja neurona ulaznog sloja.

Svaki neuron se sastoji od četiri osnovne funkcije: prihvatanje ulaza, procesovanje ulaza, pretvaranje procesovanog ulaza u izlaz i sinapsi veza sa ostalim neuronima (prikazano na Slici 2(b)). Ulazi u mrežu predstavljeni su matematičkim simbolom *x(n)*, svaki od ovih ulaznih podataka je pomnožen svojom težinom *w(n)*. Dobijeni proizvodi se sabiraju:

$$\text{Sum} = \Sigma w_i x_i \quad \text{(funkcija sumiranja)}$$

i prosleđuju funkciji transfera (aktivacionoj funkciji) koja generiše rezultat, a zatim to prosleđuje kao izlaz.

### 5.3.3 Izbor aktivacione funkcije i algoritma učenja

Aktivaciona funkcija se koristi da bi se skalirali izlazni podaci iz slojeva. Da bi neuronska mreža mogla da nauči nelinearne funkcije, neophodno je korišćenje aktivacione funkcije. Kada ove funkcije ne bi bilo neuroni skrivenih slojeva ne bi imali veće mogućnosti od obične mreže koja se sastoji samo od ulaza i izlaza – perceptronska mreža. Iz tog razloga se na izlazu neurona koristi aktivaciona funkcija koja je nelinearna.

Predloženi softver za aktivacionu funkciju koristi sigmoidnu funkciju. Razlog korišćenja sigmoidne funkcije je očekivana pozitivna vrednost kao izlaz. Izlaz iz sigmoidne funkcije je vrednost od 0 do 1. Ova vrednost se na kraju iteracije konvertuje u vrednosti od 0 do 255 kako bi greška u programu bila predstavljena u nivoima sive. Na taj način pomenuti parametar je intuitivniji korisniku softvera.

Za algoritam učenja odabran je *Resilient propagation* predložen od strane autora *Encog3* neuronskih mreža. Prednost ovog algoritma je što se ne zahteva postavljanje parametara, kao što su stopa učenja ili vrednost momentuma. To je dobro iz razloga što je ponekad jako teško odrediti ove parametre u potrazi za optimalnim rešenjem.

Algoritam može na jednostavan način da se zameni u programu, ali primeri koji su ovde navedeni i testirani koriste algoritam *Resilient propagation.*

Trening je način na koji neuronske mreže uče kako da se poboljšaju težine sinaptičkih veza kako bi se postigao željeni rezultat. Algoritmu za treniranje se predaje trening set ulaznih podataka i idealni izlaz za svaki ulaz. *Resilient propagation* trening algoritam prolazi kroz serije iteracija i nakon svake pokušava da umanji stopu greške za određeni stepen. Stopa greške predstavlja procenat razlike stvarnog izlaza i izlaza proizvedenog trening algoritmom. Računa se gradijent greške za svaku konekciju u neuronskoj mreži. Ne postoji globalni parametar ažuriranja, već za svaku vrednost matrice težina postoji posebna delta-vrednost. Ove vrednosti su u početku proizvoljno inicijalizovane na jako male vrednosti. Posle svake iteracije ove vrednosti se ažuriraju u skladu sa delta-vrednostima. Magnituda gradijenta se koristi da bi se utvrdilo kako delta-vrednosti treba da se menjaju. Na ovaj način svaka matrica težina može individualno da bude trenirana (algoritam je objašnjen u poglavlju 3).

## 5.4 Glavne funkcije programa i interfejs

Interfejs je veoma jednostavan. Komunikacija korisnika sa programom je preko interaktivnog prozora "Pričaj sa mnom" u kome može da se piše *Ruby* kod. Kroz primer učenja Photoshop-ovog filtera *emboss* detaljno će biti opisan način upotrebe softvera kao i glavne funkcije. Prvo će biti opisano, kako se koristi softver da bi se naučio odabrani filter, zatim će biti opisan proces stvaranja sopstvenog filtera i na kraju primena naučenog filtera na novoizabranu sliku. Jednom naučeni filter se čuva kao neuronska mreža u .eg formatu.

Za učenje *emboss* filtera iskorišćena je slika Lene. Izabrana slika je veličine 512x512 piksela.

### 5.4.1 Proces učenja odabranog filtera

Pokretanjem programa, učitava se prvo slika *model* - originalna slika, nakon toga se učitava slka *master* - odabrana filtrirana slika. Svaka slika se pojavljuje u posebnom prozoru. U prozoru *apprentice* posle svakih 500 epoha biće iscrtana nova slika, odnosno pokušaji neuronske mreže da nauči zadati master na osnovu zadatog modela. U slučaju da se program prekine ili u slučaju da je zadovoljena određena sličnost u prozoru *apprentice* se iscrtava rezultat.

Prozor "Pričaj sa mnom" je interaktivni prozor u kome može da se piše Ruby kod. Otvaranjem *model* i *master* slike u ovom prozoru se ispisuju svi parametri čije vrednosti mogu da se promene, a inicijalno su postavljene na podrazumevane vrednosti.

*$tolerrable_err* je globalna promenljiva čija je vrednost inicijalno postavljena na vrednost 0.1. Ova promenljiva predstavlja dozvoljenu grešku i uslov je zaustavljanja programa. Ova vrednost može da se promeni, npr. na vrednost 0.01 komandom: *$tolerrable_err = 0.01*. Vrednost greške je u intervalu od 0 do 255 u nivoima sive.

Program može da se zaustavi i pre nego što greška bude manja ili jednaka dozvoljenoj pritiskom na komandu *Ctrl+Alt+Q*.

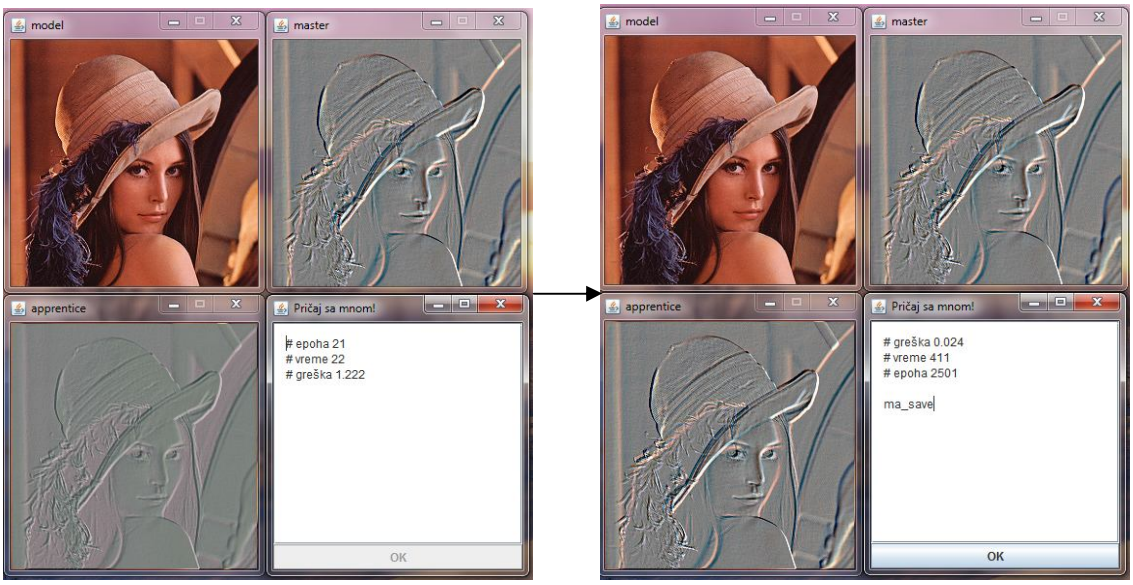
*$training_set_size* je globalna promenljiva koja predstavlja broj trening podataka. Inicijalno je postavljena na 5000, što je broj proizvoljno odabranih piksela na slici. Ovo je jedan način za izbor podataka za trening.

Drugi način za izbor podataka za trening nije slučajan. Klikom na *model* ili *master* sliku pojavljuje se kvadrat veličine zadate preko globalne promenljive *$domain_width*. Ova vrednost je inicijalno postavljena na 38, što predstavlja kvadrat veličine 38x38 piksela. U slučaju izbora ovog načina uzimanja trening podataka zanemaruje se veličina promenljive *$training_set_size*.

Globalna promenljiva *$hidden_layers* predstavlja niz koeficijenata skrivenog sloja. Broj koeficijenata određuje broj skrivenih slojeva, dok je broj neurona unutrašnjeg sloja jednak broju neurona ulaznog sloja pomnoženog sa samim koeficijentom. Inicijalno je postavljen jedan skriveni sloj na vrednost 1.5, što znači da je broj neurona tog skrivenog sloja 1.5 puta veći od broja neurona ulaznog sloja.

Pozivom funkcije *ma_retina* možemo da promenimo veličinu retine. Npr. *ma_retina 3*, menjamo inicijalnu vrednost retine koja je 5x5, na vrednost 3x3. Ukoliko je veličina retine 5x5, broj neurona ulaznog sloja za sliku u boji je 3*25, jer slika u boji ima 3 banda, jedna komponenta za crvenu, druga za zelenu, dok je treća za plavu. Broj izlaznih neurona je u tom slučaju 3. Za *grayscale* sliku i retinu 5x5 broj ulaznih neurona bi bio 25, dok bi broj izlaznih neurona bio 1.

Ukoliko su slika i filter u boji pozivom funkcije *ma_grey* slika se konvertuje u *grayscale* sliku. U slučaju da je slika *grayscale*, pozivom funkcije *ma_col* slika se tretira kao da je u boji. Funkcija *ma_run* pokreće učenje klikom na dugme *OK*.




Slika 11: (a) posle 21 generacije (b) posle 2501 generacije

Rezultat učenja odabranog filtera emboss (*master*), za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Na Slici 11 (a) je prikazan rezultat (*apprentice*) posle 21. epohe, gde je greška 1.222, a vreme izvršavanja 22 sekunde. Na Slici 11 (b) je prikazan rezultat (*apprentice*) posle 2501. epohe, greška je 0.024 dok je vreme izvršavanja 411 sekundi. Ovde se učenje zaustavilo zato što je postignuta zahtevana sličnost. Za ovaj primer, učenje *emboss* filtera, promenjena je veličina retine na 3 i greška na 0.025, dok ostalim parametrima inicijalne vrednosti nisu promenjene.

Nakon dostignute sličnosti, rezultat može da se sačuva pozivom funkcije *ma_save*. Čuva se neuronska mreža naučenog filtera, koja može proizvoljan broj puta da se primeni na bilo koju drugu odabranu sliku.

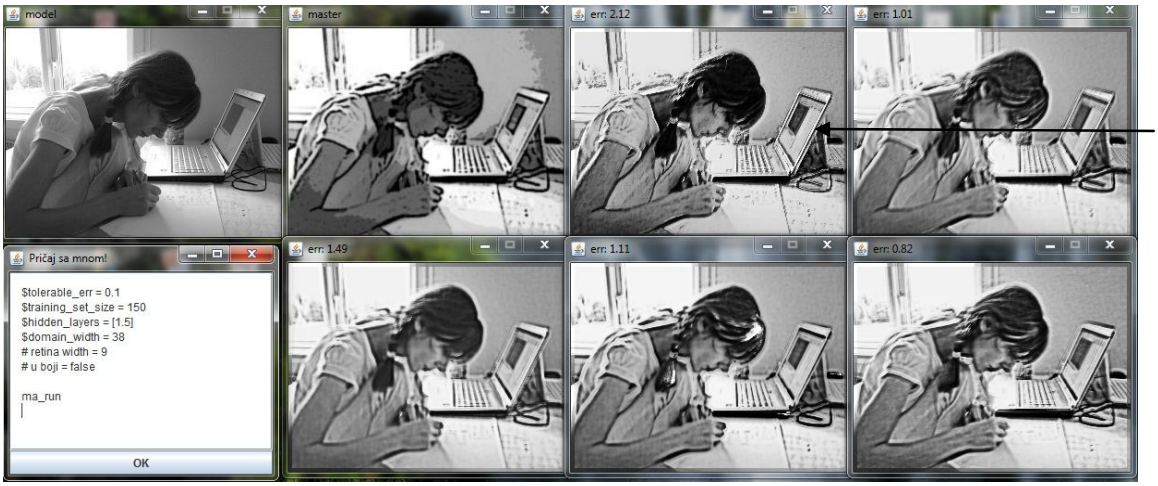
Pozivom funkcije *ma_init* se ponovo otvaraju *model* i *master* slike za novo učenje filtera ili se otvara slika na koju želimo da primenimo već naučeni i zapamćeni filter. Funkcija *ma_info* služi za dobijanje informacija o parametrima čije vrednosti mogu da se menjaju.

### 5.4.2 Proces stvaranja sopstvenog filtera

Važno je imati na umu da grafički dizajneri, umetnici i drugi ljudi čiji posao zavisi od slika, koriste iste programe za manipulaciju. Predloženi softver daje interesantne rezultate vezane za proizvodnju filtera koje korisnik sam stvara. S jedne strane korisnik bi morao da provede neko vreme stvarajući svoj filter, što deluje obeshrabrujuće ako se ne uzme u obzir da jednom naučeni fiter, na drugim slikama može da se proizvede u realnom vremenu neograničen broj puta. Na taj način korisnik bi mogao da pravi svoju ličnu paletu filtera i takođe sa istom da radi, a samim tim da se razlikuje od svoje konkurencije.

Ovde je prikazan način upotrebe softvera koji se koristi kako bi se stvorio filter vođen ličnom estetikom. Pokretanjem programa bira se slika *model* i slika *master*. Razlika u odnosu na opciju učenja odabranog filtera je u tome što je broj *apprentice* prozora veći, odnosno broj neuronskih mreža koje se treniraju u cilju učenja odabranog filtera. Način na koji se ovde simulira evolucija je odabrani mali broj trening podataka. Na taj način simuliramo varijacije, odnosno raznolikost dobijenih slika posle svake iscrtane generacije. Nakon svake završene generacije korisnik može da zameni trenutno odabranu master sliku, sa nekom od dobijenih slika nakon protekle generacije i na taj način kreira filter vođen ličnom estetikom.

Inicijalno broj *apprentice* prozora je šest, ali taj broj može biti i smanjen zatvaranjem nekog od *apprentice* prozora. Pozivom funkcije *ma_make*, otvara se još jedan *apprentice* prozor. Drugi način povećanja broja apprentice prozora je dupli klik bilo na *master*, bilo na *model* sliku. Broj *apprentice* prozora predstavlja veličinu generacije, dok su varijacije simulirane uzimanjem malog broja trening podataka. Inicijalno *$training_set_size* je 150.




Slika 12: proces stvaranja filtera u pet paralelnih prozora, odabrana je originalna slika – *model,* filtrirana slika– *master,* nakon 100 generacija označena slika je odabrana kao nova master slika

Za ovu svrhu, kao polazna tačka u stvaranju lično vođenog filtera uzet je *Photoshop*-ov filter *poster edges*. Na Slici 12 je prikazan rezultat posle 100 generacija. Nakon svakih 100 generacija iscrtava se slika u prozoru *apprentice*, program se zaustavlja pozivanjem funkcije *ma_stop*. Program zaustavljamo onog momenta kada nam se neka od slika u *apprentice* prozorima više dopadne od trenutne master slike. Klikom na odabranu sliku, ta slika postaje master slika i na taj način proces može da se nastavi pozivom funkcije *ma_run*, nakon čega dobijamo novu generaciju slika. Ovaj proces se ponavlja dokle god ne dobijemo sliku koju bi voleli da sačuvamo. Svakim odabirom nove slike može da se sačuva neuronska mreža naučenog filtera. Na Slici 12 je strelicom označena slika koja je odabrana kao nova *master* slika.


Slika 13: odabrane slike u procesu stvaranja filtera vođenog ličnim ukusom

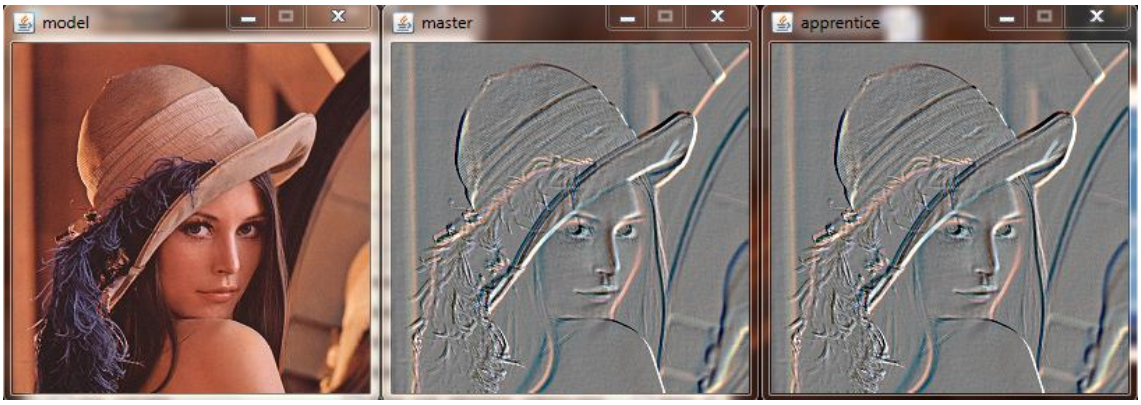
Na Slici 13 su prikazane samo slike koje su odabrane kao *master* slike nakon zaustavljene generacije. Poslednja slika je dobijena nakon desete zaustavljene

generacije. Proces može da se nastavi. Neuronska mreža svake od prikazanih slika je sačuvana kao poseban .eg fajl, tako da svaki od ovih naučenih filtera može da se primeni na proizvoljan broj novoizabranih slika.

Promena parametara, dobijanje informacija o vrednostima parametara i primena naučenog filtera na novoizabranu sliku se vrši na isti način kao i kod učenja odabranog filtera.

### 5.4.3 Proces primene naučenog filtera na novoizabranu sliku

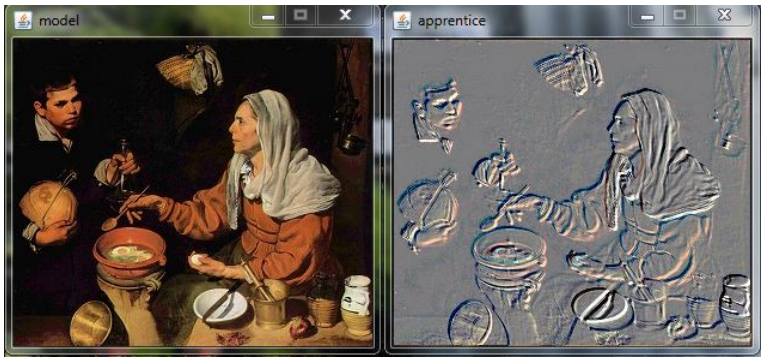
Jednom naučeni filter može da se primeni proizvoljan broj puta na bilo koje druge slike. Navedeni naučeni filter *emboss* (u fazi učenja) je sačuvan kao .eg fajl Slika 14.




Slika 14: rezultat učenja *emboss* filtera (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Na Slici 15 i 16 prikazana je primena naučenog filtera sa Slike 14 na dve novoodabrane slike. Otvaranje ovog fajla vrši se pozivom funkcije *ma_open* za otvaranje *.eg* fajlova, dok se pozivom funkcije *ma_use* neuronska mreža primenjuje na novoizabranu sliku.


Slika 15: primena naučenog *emboss* filtera na novoodabranu sliku (Šarl Žalaber)



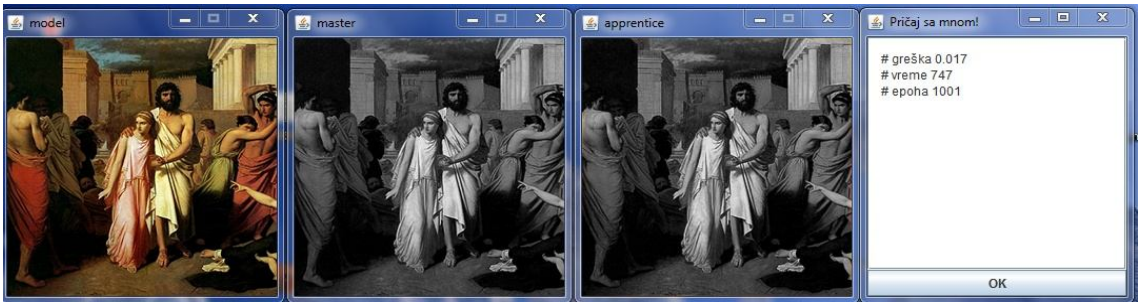
Slika 16: primena naučenog *emboss* filtera na novoodabranu sliku (Dijego Velaskez)

## 5.5 Rezultati

Ovde će biti priloženi rezultati dobijeni korišćenjem predloženog softvera. Za svaki odabrani filter, prvo će biti prikazan rezultat učenja odabranog filtera. Nakon čega će slediti rezultati primene naučenog filtera na novoizabrane slike. Za primere filtera koje predajemo neuronskoj mreži da nauči, uzeti su filteri dobijeni standardnim metodama, objašnjenim u prvom poglavlju: procesi u jednoj tački, detekcija, ublažavanje ivica i izoštravanje ivica. Pored ovih filtera biće razmotren problem kolorizacije i super rezolucije. Obzirom da je svrha korišćenja analogija slika, da se nauče filteri koji su kombinacija primene više filtera na odabranu sliku, biće dat i primer učenja takve vrste transformacija.

Najjednostavniji filteri su operacije na jednoj tački. Definisani su kao funkcija koja se obavlja na svakom pikselu slike, nezavisno od ostalih piksela na toj slici. Neki od procesa na jednoj tački su: pretvaranje slike koja je u boji u *grayscale* sliku, povećanje ili smanjenje kontrasta ili osvetljenja, negativ slike i *threshold.*

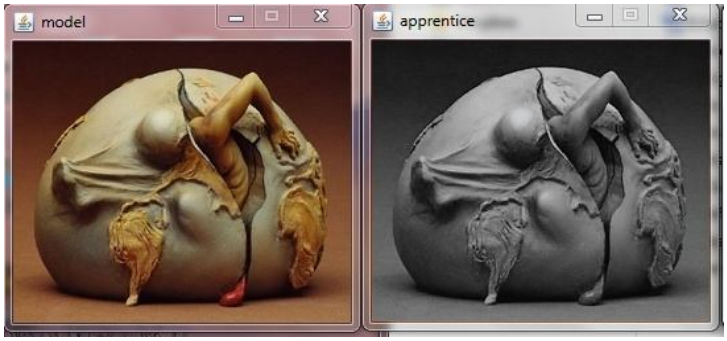
*Grayscale* slike su slike koje sadrže samo informacije o intenzitetu osvetljenja svakog piksela. Slike ove vrste sadrže isključivo nijanse sive boje, koje variraju od crne najslabijeg intenziteta do bele najjačeg intenziteta. Na Slici 17 prikazan je rezultat učenja filtera koji konvertuje sliku u boji u *grayscale* sliku.



Slika 17: rezultat učenja filtera koji konvertuje sliku u boji (model) u grayscale sliku (master) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 3, za ostale parametre zadržane su predefinisane vrednosti. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.017, vreme izvršavanja je 747 sekundi, a broj epoha je 1001.

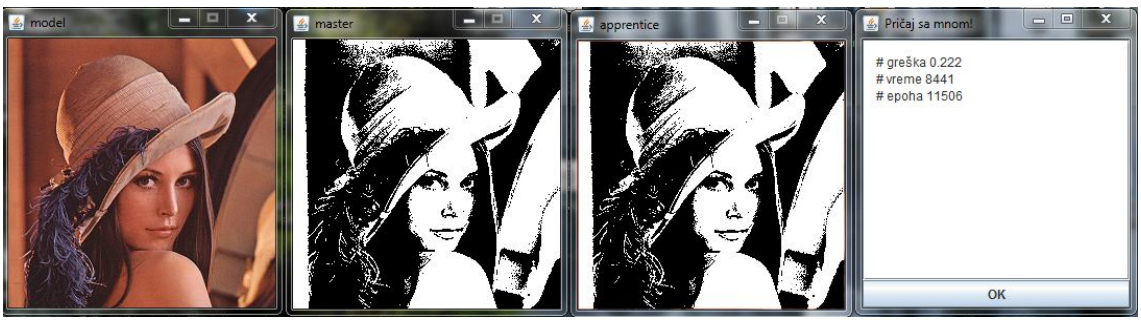
Na Slici 18 i 19 prikazana je primena naučenog filtera sa Slike 17 na dve novoodabrane slike.




Slika 18: primena naučenog filtera koji konvertuje sliku u boji u *grayscale* sliku na novoodabranu sliku (Salvador Dali)


Slika 19: primena naučenog filtera koji konvertuje sliku u boji u *grayscale* sliku na novoodabranu sliku

Segmentacija slike se odnosi na postupak podele slike na regione sa sličnim atributima. U procesu segmentacije mogu da se koriste različite karakteristike kao što su mere teksture i ivice. Segmentacija je u analizi slika jedan od prvih i najvažnijih koraka. Teorijski ne postoji parametar za kvantitativnu procenu koliko je neki postupak segmentacije dobar. Jedan od metoda za segmentaciju je određivanje praga – *threshold* osvetljenosti. Na Slici 20 prikazan je rezultat učenja filtera *threshold*.

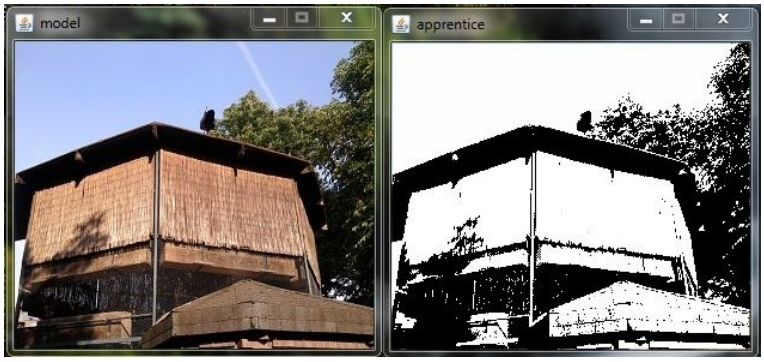



Slika 20: rezultat učenja *threshold* filtera (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 3, za ostale parametre zadržane su predefinisane vrednosti. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.222, vreme izvršavanja je 8441 sekunda, a broj epoha je 11506.

Na Slici 21 i 22 prikazana je primena naučenog filtera sa Slike 20 na dve novoodabrane slike.


Slika 21: primena naučenog *threshold* filtera na novoodabranu sliku (Dijego Velaskez)




Slika 22: primena naučenog *threshold* filtera na novoodabranu sliku

Negativ filter je specijalno pogodan za poboljšanje belih i sivih detalja okruženih tamnim regionima slike. Naročito je koristan kada su crne površine dominantne. Na Slici 23 prikazan je rezultat učenja filtera negativ.


Slika23: rezultat učenja filtera negativ (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 3, za ostale parametre zadržane su predefinisane vrednosti. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.04, vreme izvršavanja je 688 sekundi, a broj epoha je 965.

Na Slici 24 i 25 prikazana je primena naučenog filtera sa Slike 23 na dve novoodabrane slike.


Slika 24: primena naučenog *negativ* filtera na novoodabranu sliku (Vinsent van Gog)


Slika 25: primena naučenog *negativ* filtera na novoodabranu sliku (Salvador Dali)

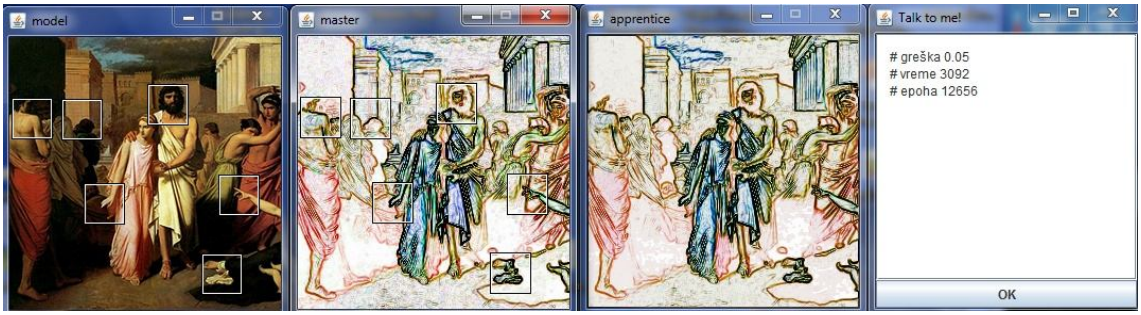
Kao primer procene koliko je naučeni negativ filter dobar, odabrana je negativ slika na koju je primenjen naučeni negativ filter. Ono što očekujemo kao rezultat jeste originalna slika, obzirom da je $(f^{-1})^{-1} = f$. Očekivani rezultat je dobijen (Slika 26).


Slika 26: primena naučenog *negativ* filtera na odabranu negativ sliku (Salvador Dali)

Osnovni cilj detektovanja ivica je izvlačenje bitnih informacija iz slike pomoću kojih može da se izvrši računarska interpretacija kao i analiza slike. Ivice odgovaraju značajnim lokalnim promenama inteziteta na slici. Intuitivno, ivica je skup povezanih

piksela koji leže na granici između dva regiona. Postoji mnogo filtera za detekciju ivica neki od njih su: Roberts cross, Sobel, Prewitt, Kirsch, Canny i Laplacian filter.

Na Slici 27 prikazan je rezultat učenja *Photoshop*-ovog filtera detekcije ivica.




Slika 27: rezultat učenja *Photoshop*-ovog filtera detekcije ivica (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

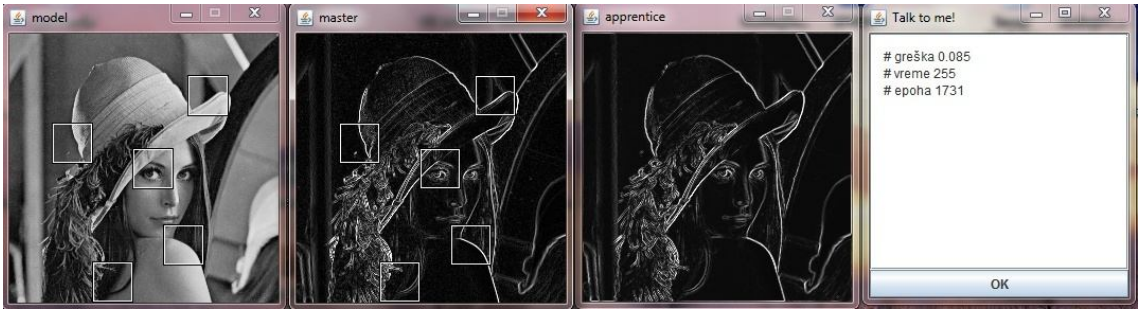
Za parametar retine odabrana je veličina 3, za ostale parametre zadržane su predefinisane vrednosti. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.05, vreme izvršavanja je 3092 sekunde, a broj epoha je 12656.

Na Slici 28 i 29 prikazana je primena naučenog filtera sa Slike 27 na dve novoodabrane slike.


Slika 28: primena naučenog *Photoshop*-ovog filtera detekcije ivica na novoodabranu sliku


Slika 29: primena naučenog *Photoshop*-ovog filtera detekcije ivica na novoodabranu sliku

Na Slici 30 prikazan je rezultat učenja *Sobel* filtera za detekciju ivica objašnjenog u drugom poglavlju.




Slika 30: rezultat učenja *Sobel* filtera detekcije ivica (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 3, za ostale parametre zadržane su predefinisane vrednosti. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.085, vreme izvršavanja je 255 sekundi, a broj epoha je 1731. Umesto proizvoljno izabranog trening skupa podataka, trening podaci su označeni kvadratima 38x38 piksela.

Na Slici 31 i 32 prikazana je primena naučenog filtera sa Slike 30 na dve novoodabrane slike.


Slika 31: primena naučenog *Sobel* filtera detekcije ivica na novoodabranu sliku

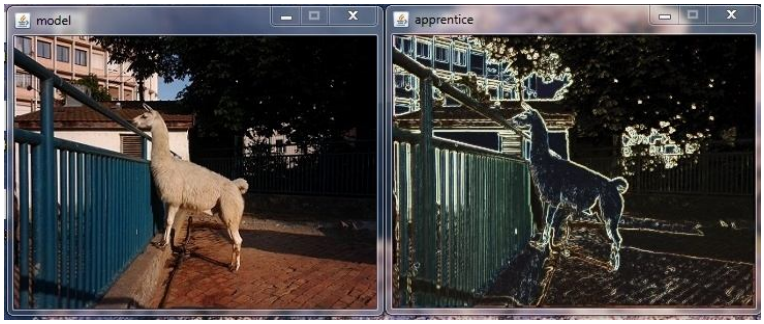

Slika 32: primena naučenog *Sobel* filtera detekcije ivica na novoodabranu sliku

Na Slici 33 prikazan je rezultat učenja Photoshop-ovog filtera *glowing edges*. Za parametar retine odabrana je veličina 5. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.09, vreme izvršavanja je 5740 sekundi, a broj epoha je 795.


Slika 33: rezultat učenja *Photoshop*-ovog filtera *glowing edges* (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Na Slici 34 i 35 prikazana je primena naučenog filtera sa Slike 33 na dve novoodabrane slike.


Slika 34: primena naučenog filtera *glowing edges* na novoodabranu sliku




Slika 35: primena naučenog filtera *glowing edges* na novoodabranu sliku

Na Slici 36 prikazan je rezultat učenja Photoshop-ovog filtera *blur*. Za parametar retine odabrana je veličina 5. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.012, vreme izvršavanja je 3671 sekunda, a broj epoha je 26501.


Slika 36: rezultat učenja *Photoshop*-ovog filtera *blur* (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

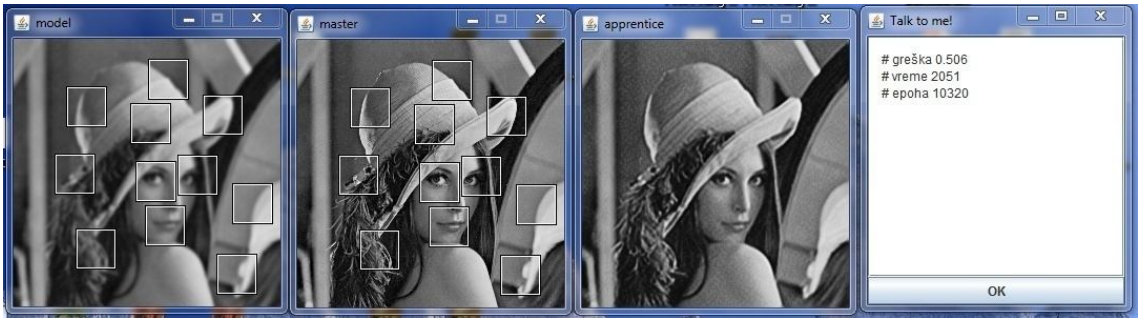
Na Slici 37 i 38 prikazana je primena naučenog filtera sa Slike 36 na dve novoodabrane slike.


Slika 37: primena naučenog filtera *blur* na novoodabranu sliku


Slika 38: primena naučenog filtera *blur* na novoodabranu sliku

Interpretacija slike zavisi od njene oštrine, od mogućnosti izdvajanja informacija koje slika sadrži. Oštrije slike pokazuju više detalja. Međutim, nije lako definisati šta je oštra slika. Obično se za definiciju zamućene slike uzima definicija slike koja je izgubila visokofrekventne informacije. Slike koje dobijamo korišćenjem fotoaparata su često zamućene. Umesto pojedinačne obrade svake fotografije bilo bi mnogo lakše korišćenje filtera koji popravlja kvalitet loše dobijene fotografije.

Na Slici 39 je prikazano učenje *deblur* filtera.




Slika 39: rezultat učenja filtera *deblur* (*master*) za odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 5. Trening podaci nisu proizvoljno odabrani već selekcijom površina slike 38x38. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.506, vreme izvršavanja je 2051 sekunda, a broj epoha je 10320. Na Slici 40 i 41 prikazana je primena naučenog filtera sa Slike 39 na dve novoodabrane slike.


Slika 40: primena naučenog filtera *deblur* na novoodabranu sliku


Slika 41: primena naučenog filtera *deblur* na novoodabranu sliku

Na navedenim slikama bio je prikazano učenje jednog specifično odabranog filtera. Prednost korišćenja predloženog softvera je mogućnost učenja filtera koji je kombinacija primene ne samo jednog već većeg broja filtera na slici. Kao primer uzeta je slika na kojoj je u Photoshop-u primenjeno više filtera: *solarize*, *brightness*, *contrast* i *blur* filter (Slika 42).

Slika 42: rezultat učenja filtera koji je primena većeg broja transformacija (*master*) na odabranu originalnu sliku (*model*) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 5. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.1, vreme izvršavanja je 998 sekundi, a broj epoha je 9000.

Na Slici 43 i 44 prikazana je primena naučenog filtera sa Slike 42 na dve novoodabrane slike.

Slika 43: primena naučenog filtera (kombinacija primene većeg broja transformacija: *solarize*, *brightness*, *contrast* i *blur*) na novoodabranu sliku (Salvador Dali)

Slika 44: primena naučenog filtera (kombinacija primene većeg broja transformacija: *solarize*, *brightness*, *contrast* i *blur*) na novoodabranu sliku

Bojenje *grayscale* slika je težak problem. Jako je teško da se oboji slika ako prethodno nemamo neke informacije o toj slici. Isti predmeti mogu da budu različito obojeni. Problem je i sa bojenjem prirodnih objekata. Mnoga rešenja za ovaj problem se sastoje od uzimanja informacija o boji od samog korisnika softvera. Na Slici 45 prikazan je primer učenja ovog filtera.


Slika 45: rezultat učenja filtera koji grayscale sliku (*model*) konvertuje u sliku u boji (*master*) prikazan je u prozoru *apprentice*

Za parametar retine odabrana je veličina 5. Dobijeni rezultat je prikazan u prozoru *apprentice*. Greška dobijenog rezultata je 0.264, vreme izvršavanja je 1106 sekundi, a broj epoha je 1501.

Na slici 46 (a) prikazana je primena naučenog filtera na novoodabranu sliku. Dok je na slici (b) prikazana originalna slika. Ovo izgleda kao dobro rešenje, međutim problem nastaje kada očekivana slika ne sadrži nijanse boja treniranog para.


Slika 46: (a) rezultat primene filtera koji boji grayscale sliku

(b) originalna slika

# 6 Zaključak

Očigledno je da ne možemo da očekujemo da softveri koji rešavaju problem analogije slika rade savršen posao u smislu učenja svih mogućih filtera, naročito iz razloga što se za trening uzima samo jedan par slika. Iznenađujuće je ipak da mnoge vrste filtera mogu da se nauče koristeći ovaj princip. Predloženo rešenje problema analogije slika korišćenjem neuronskih mreža se pokazalo kao dobar način za učenje različitih vrsta filtera kao što su: *threshold*, prevođenje slike koja je u boji u *grayscale* sliku, invertovanje slike, povećanje i smanjenje kontrasta i osvetljenja, *detecting*, *bluring* i *sharpening edges*. Softver podržava i učenje filtera koji su kombinacija više primenjenih transformacija na sliku. Ovo je veoma važno ako je potrebno da se obradi velika količina slika. Softver rešava i problem dobijanja slike veće rezolucije, što je naročito korisno ako je potrebno da se na slici uveća neki deo ili ukoliko je slika zamućena što se često događa kada se koristi fotoaparat. Softver obezbeđuje prirodne transformacije slika umesto izbora različitih filtera i njihovih podešavanja. Korisnik jednostavno može da izabere odgovarajući efekat i isti proizvede na novoj slici. Uzimajući ovo u obzir filteri ne moraju individualno da budu pronađeni ili posebno programirani, idealno, isti mehanizam bi mogao da se iskoristi da se proizvede širok spektar efekata. Pored učenja odabranog filtera softver omogućava i stvaranje sopstvenih filtera. Predloženi softver ima i svoja ograničenja, filteri koji uključuju distorziju, filteri koji zavise od konteksta same slike ne mogu biti naučeni predloženim rešenjem.

# Reference


[1] Gonzalez, R.C., Woods, R.E, (2002), *Digital Image Processing*, 2nd ed., New Jeresy: Prentice Hall.

[2] Pratt, W.K., (2007), *Digital Image Processing*, 4th ed., Los Altos: PixelSoft.

[3] Jayaraman, S., Esakkirajan, S. & Veerakumar, T., (2009), *Digital Image Processing,* New Delhi: Tata McGraw-Hill Education.

[4] Kriesel, D., (2007), *A Breaf Introduction to Neural Networks*, <http://www.dkriesel.com/en/science/neural_networks>, [accessed 21.08.2011].

[5] Krose, B. & Smagt, P., (1996), 8th ed., *An Introduction to Neural Networks*, Amsterdam: University of Amsterdam.

[6] Heaton, J., (2011), *Getting Started whith Encog3 in Java,* Chesterfield: Heaton Research.

[7] Karlik, B., Olgac, A.V., (2011), Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks, *International Journal of Artificial Intelligence and Expert Systems*, 1(4), pp. 72-122.

[8] Anastasiadis, A.D., Magoulas, G.D., Vrahatis, M.N., (2004), A New Learning Rates Adaptation Strategy for Resilient Propagation Algorithm*, Proceedings of European Symposium on Artificial Neural Networks,* Bruges, Belgium, April 28-30, 2004, pp. 1-6.

[9] Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D. H., (2001), Image Analogies, *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'01).*, New York, August 2001, pp. 327-340.

[10] Hertzmann, A., Oliver, N., Curless B., Seitz, S. M., (2002), Curve Amalogies, *In Proceedings of the 13th Eurographics Workshop on Rendering (EGRW '02).*, Pisa, Italy, June 26-28, 2002, pp. 233- 246.

[11] Freeman, W. T., Jones, T.R., Pasztor, E.C., (2002), Example-based superresolution, *IEEE Computer Graphics and Application.,* 22(2), pp. 56-65.

[12] Joshi, N., Matusik, W., Adelson, E.H., Kriegman, D. J., (2010), Personal photo enhancement using example images, *ACM Transactions Graph*ics., 29(2), pp. 1-15.

[13] Cheng, L., Vistwanathan, S.V.N., Zhang, X., (2008), Consistent Image Analogies using Semi-supervised Learning, IEEE *Computer Vision and Pattern Recognition.*, Anchorage, June 23-28, 2008, pp. 1-8.

[14] Efros, A. A., Freeman, W.T., (2001), Image quilting for texture synthesis and transfer, Proceedings of the 28th annual conference on Computer graphics and interactive techniques SIGGRAPH'01, August 2001, pp. 341–346.

[15] Li, M., Wang, F., Liu, X., Jin, H., (2009), Super-Resolution Rendering for Digital Earth Applications, *Proceedings of Sixth International Symposium on Digital Earth: Models, Algorithms, and Virtual Reality*, Beijing , China, 9-12 September, 2009, 7840-03.

[16] Levin, A., Lischinski, D., Weiss, Y., (2004), Colorization using optimization, ACM *Trasactions On Graphics*, 23(3), pp. 689-694.

[17] Irony R., Cohen, C., Lischinski, D., (2005), Colorization by Example, *In Proceedings Of Eurographics Symposium on Rendering.,* pp. 201-210.

[18] Lackey, J. B., Colagrosso M.D., (2004) Supervised segmentation of visible human data with image analogies, *Proceedings of the International Conference on Artificial Intelligence/International Conference on Machine Learning, Models, Technologies and Applications,* Las Vegas, Jun 21-24, 2004, pp. 843-847.

[19] Bychkovsky, V., Paris, S., Chan, E., Durand, F., (2011), Learning Photographic Global Tonal Adjustment with a Database of Input/Output Image, *IEEE Computer Vision and Pattern Recognition.,* Colorado Springs, June 2011.

[20] Thomas D., Fowler, C., Hunt, A., (2007), *Programming Ruby 1.9,* Pragmatic Bookshelf.

[21] Eckel, B., (2007), *Thinking in Java,* Prentice Hall.

[22] Tasić, J., (2012), Implementation of image analogies using neural networks, *Proceedings of the Second International Students Conference on Informatics,* Sibiu, Romania, April 26-28, 2012, pp. 202-214.