SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4685

Semantička segmentacija slika dubokim konvolucijskim mrežama

Ivan Grubišić

Zagreb, srpanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 17. ožujka 2016.

ZAVRŠNI ZADATAK br. 4685

Pristupnik: **Ivan Grubišić (0036478093)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Semantička segmentacija slika dubokim konvolucijskim mrežama**

Opis zadatka:

Semantička segmentacija prirodnih scena je neriješen problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme najbolji rezultati u tom području postižu se pristupima utemeljenima na dubokim neuronskim mrežama. Za ovaj rad posebno su zanimljivi strogo nadzirani pristupi gdje u svakoj piknji skupa za učenje na raspolaganju imamo informaciju o pripadnom semantičkom razredu. U okviru rada, potrebno je proučiti dokumentaciju programskog okvira Tensorflow te biblioteke programskog jezika Python za rukovanje matricama i slikama. Izraditi izvedbu programskog sustava za učenje i primjenu segmentacijske mreže korištenjem dostupnih postupaka otvorenog koda. Ostvariti vlastito rješenje u programskom okviru Tensorflow. Prikazati i ocijeniti ostvarene rezultate. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:
Izv. prof. dr. sc. Siniša Šegvić

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

Dielovođa:
Doc. dr. sc. Tomislav Hrkać

Sadržaj

1. Uvod

Prepoznavanje i razumijevanje sadržaja slike je glavni i još uvijek otvoreni problem računalnog vida. U to spada klasifikacija, tj. pridjeljivanje odgovarajuće kategorije ili razreda slici. Sljedeći korak u tome su određivanje položaja više različitih predmeta koji pripadaju različitim razredima i klasifikacija svakog dijela slike. Ovo zadnje naziva se semantička segmentacija. Zadatak semantičke segmentacije je svakom dijelu (pikselu) slike pridijeliti oznaku koja predstavlja njegovo značenje ili pripadnost nekoj kategoriji ili razredu. Primjeri razreda mogu biti *nebo*, *stablo*, *trava*, *čovjek*, *kuća*, *cesta*, i *vozilo*.

Neuronske mreže su matematički modeli nadahnuti biološkim modelima mozga i neurona. Koriste se za rješavanje problema koje je teško ili nemoguće riješiti analitičkim metodama. Jedna od njihovih glavnih karakteristika je da, ovisno o postavljenim parametrima, mogu ostvarivati složene funkcije. Druga glavna karakteristika je da se ti parametri mogu algoritamski prilagođavati, što se naziva učenje ili treniranje.

U posljednje vrijeme najbolji rezultati u zadacima opisanim u prvom ulomku postižu se pristupima temeljenima na dubokim konvolucijskim neuronskim mrežama. Duboke konvolucijske mreže uz manji broj parametara mogu dobro modelirati složenije nelinearne zavisnosti i imaju dobra svojstva prilagođena rješavanju problema vezanih uz sliku i zvuk.

Ovaj rad opisuje sustav za semantičku segmentaciju koji za učenje parametara koristi skup primjera slika s pripadajućim oznakama razreda. Arhitektura tog sustava temelji se na dubokoj konvolucijskoj neuronskoj mreži.

U poglavlju 2 objašnjeni su osnovni pojmovi vezani uz neuronske mreže. Na kraju poglavlja opisane su konvolucijske neuronske mreže. U poglavlju 3 opisani su osnovni algoritmi i metode koje se koriste za učenje neuronskih mreža i

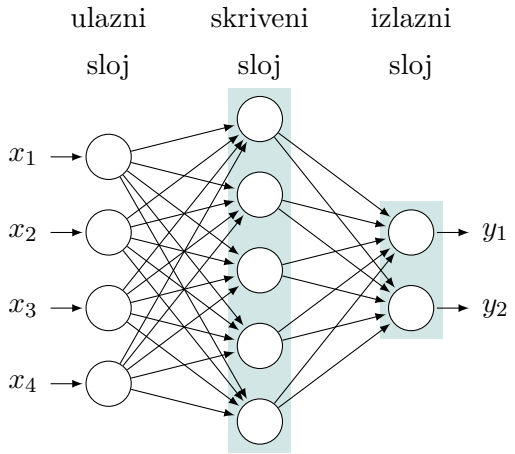
rješavanje problema koji se pri tome javljaju. U poglavlju 4 opisuje se semantička segmentacija i daje se primjer skupa podataka koji se koristi za učenje. U poglavlju 5 su opisani detalji programske izvedbe sustava za semantičku segmentaciju. U poglavlju 6 su opisani rezultati koje je sustav ostvario. U poglavlju 7 je zaključak.

2. Umjetna neuronska mreža

Umjetne neuronske mreže (engl. *artificial neural networks*) su računalni modeli nastali po uzoru na neuronske mreže kod životinja (središnji živčani sustav) koji se koriste za aproksimiranje složenih funkcija koje ovise o velikom broju parametara. Neuronska mreža se sastoji od skupa neurona, koji se još mogu nazivati čvorovima ili jedinicama, i usmjerenih veza među njima koje predstavljaju sinaptičke težine.

Neuronske mreže kojima se ovaj rad bavi spadaju u unaprijedne neuronske mreže (engl. *feed-forward neural network*). Kod takvih mreža veze između neurona ne čine cikluse. Unaprijedne neuronske mreže mogu biti jednoslojne ili višeslojne. Ako se unaprijedna mreža sastoji od dva ili više slojeva, svaki sloj je povezan samo sa susjednim slojevima tako da izlazi jednog sloja predstavljaju ulaze sljedećem sloju.

Primjer neuronske mreže s topologijom potpuno povezanih slojeva prikazan je na slici 2.1. Izlaz unaprijedne neuronske mreže računa se tako da se redom računaju izlazi slojeva od ulaznog sloja prema izlaznom sloju.

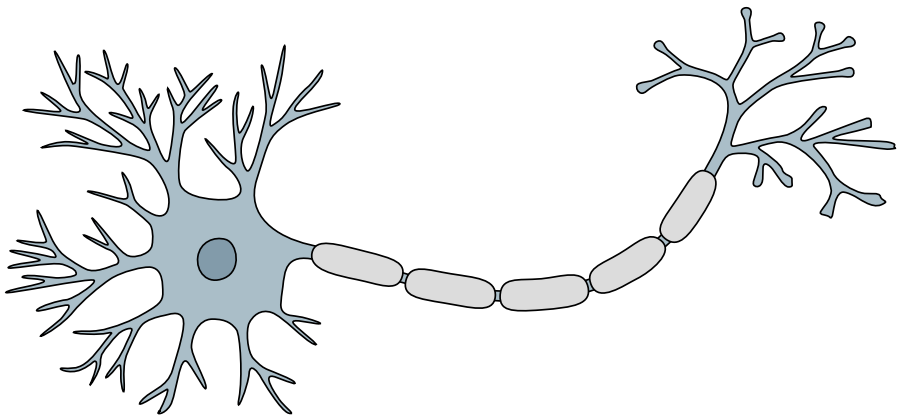


Slika 2.1: Umjetna neuronska mreža

Mreža prikazana na slici 2.1 je primjer dvoslojne unaprijedne mreže. Sastoji se od ulaznog sloja i dva sloja neurona koji obavljaju operacije nad svojim ulazima. Čvorovi ulaznog sloja redom odgovaraju ulazima (x_1, x_2, x_3, x_4). Ulazni čvorovi povezani su s neuronima skrivenog sloja. Izlazi neurona skrivenog sloja čine ulaze neuronima izlaznog sloja, čiji izlazi redom odgovaraju izlazima neuronske mreže (y_1, y_2).

2.1. Umjetni neuron

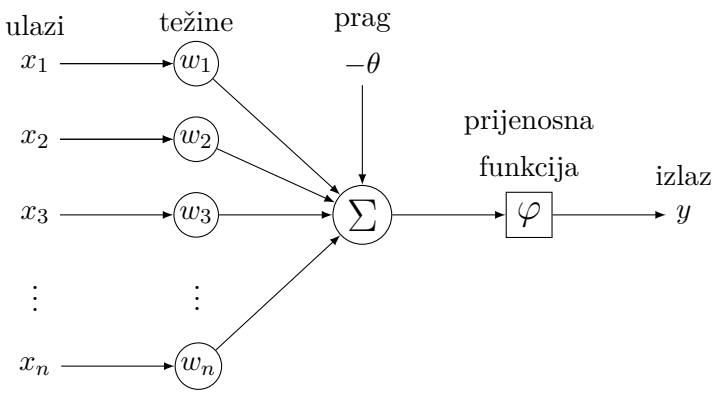
Osnovni element umjetne neuronske mreže je umjetni neuron. Umjetni neuron (slika 2.3) je pojednostavljeni matematički model biološkog neurona (slika 2.2). Analogno biološkom neuronu koji ulazne signale prima preko dendrita koji su sinapsama povezanih s krajevima aksona drugih neurona, umjetni neuron ulaze x_i koji odgovaraju izlazima drugih neurona množi s odgovarajućim sinaptičkim težinama (engl. *weights*) w_i. Težinama pomnoženi ulazi $x_i w_i$ i prag (engl. *bias*) θ se zatim zbrajaju i na zbroj se primjenjuje prijenosna funkcija (aktivacijska funkcija, engl. *activation function*) φ koja daje izlaz y koji je analogan signalu koji se kod biološkog neurona odašilje preko aksona. Težine w_i i prag θ mogu se prilagođavati učenjem, što će biti opisano kasnije u poglavlju 3.



Slika 2.2: Životinjski neuron

Preslikavanje koje odgovara neuronu može se ovako matematički izraziti:

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i - \theta\right), \tag{2.1}$$



Slika 2.3: Model neurona

gdje je n broj ulaza, $x_1, ..., x_n$ ulazi, $w_1, ..., w_n$ sinaptičke težine, θ prag, φ prijenosna funkcija, a y izlaz neurona. Često se za oznaku praga umjesto θ koristi w_0 te se dodaje jedan konstantni ulaz x_0 koji uvijek iznosi -1. Koristeći te oznake, izlaz neurona može se jednostavnije izraziti:

$$y = \varphi\left(\sum_{i=0}^{n} w_i x_i\right). \tag{2.2}$$

Prikladno je težine i druge vrijednosti koje se javljaju unutar mreže izražavati vektorima ili, ovisno o arhitekturi mreže, matricama ili višedimenzionalnim tenzorima. Jednadžba (2.2) se može i ovako zapisati:

$$y = \varphi(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}). \tag{2.3}$$

2.1.1. Prijenosna funkcija

Za prijenosnu funkciju mogu se koristiti različite vrste funkcija ovisno o primijenjenoj arhitekturi mreže. Općenito, prijenosne funkcije su monotono-rastuće i nelinearne. Korištenje linearne prijenosne funkcije jako ograničava mogućnosti mreže. Budući da za svaki zbroj i svaku kompoziciju linearnih funkcija vrijedi da je linearna funkcija, za svaku neuronsku mrežu koja koristi linearnu prijenosnu funkciju i ima n slojeva i k izlaza postoji ekvivalentna mreža s jednim slojem od k neurona. Linearna prijenosna funkcija nalazi primjenu samo u jednostavnijim modelima.

U sljedećim ulomcima bit će navedene neke češće korištene prijenosne funkcije.

Funkcija praga ili Heavisideova funkcija skoka može poprimiti samo

vrijednosti iz skupa $\{0, 1\}$ i ovako se izražava:

$$\mathrm{H}(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0. \end{cases} \tag{2.4}$$

Neuron s takvom prijenosnom funkcijom bit će aktiviran, tj. imat će na izlazu vrijednost 1 samo ako je razlika zbroja težinama pomnoženih ulaza i praga veća od 0. Funkcija praga nije diferencijabilna u točki $x = 0$, a u ostalim točkama je njen gradijent $\mathbf{0}$. Zbog toga učenje efikasnim algoritmima temeljenima na gradijentu u tom slučaju nije moguće ostvariti.

Neuron s funkcijom praga kao prijenosnom funkcijom u literaturi se naziva McCulloch-Pitts-ov model kao priznanje za pionirski rad McCulloch-a i Pitts-a (1943.).

Zbog dobrih svojstava često se primjenjuju sigmoidalne funkcije. To su najčešće logistička funkcija σ s kodomenom $[0, 1]$ i tangens hiperbolni s kodomenom $[-1, 1]$:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2\sigma(-2x) - 1. \tag{2.6}$$

Pokazalo se da je tangens hiperbolni prikladniji za učenje mreže nego logistička funkcija [10]. Za derivacije navedenih funkcija vrijedi:

$$\frac{d}{dx}\sigma(x) = \frac{1}{1 + e^{-x}} \frac{e^{-x}}{1 + e^{-x}} = \sigma(x)(1 - \sigma(x)) \tag{2.7}$$

$$\frac{d}{dx}\tanh(x) = 1 - \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right)^2 = 1 - \tanh(x)^2. \tag{2.8}$$

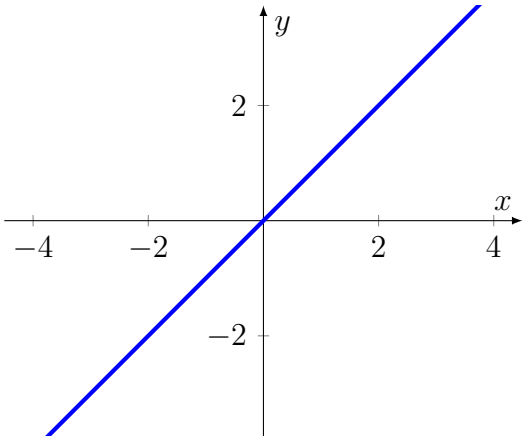
U zadnjih nekoliko godina kao prijenosna funkcija često se koristi zglobnica (poluvalno ispravljena linearna funkcija, kosina (engl. *ramp*, *rectifier*, *ReLU - rectified linear unit*)) R za čija svojstva se pokazalo da omogućuju efikasnije računanje i bolje rješavanje nekih problema koji se javljaju pri učenju neuronske mreže, a prisutni su kod sigmoidnih funkcija [6]. Izrazi za zglobnicu i njenu glatku aproksimaciju koja se naziva *softplus* su:

$$\mathrm{R}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0, \end{cases} \tag{2.9}$$

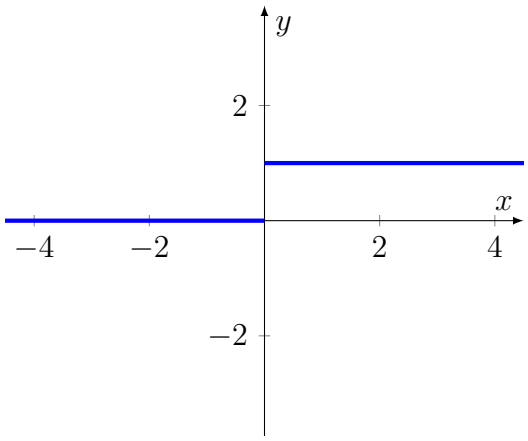
$$\mathrm{softplus}(x) = \ln(1 + e^{x}). \tag{2.10}$$

Derivacija zglobnice je skok H, a derivacija funkcije *softplus* je logistička funkcija σ.

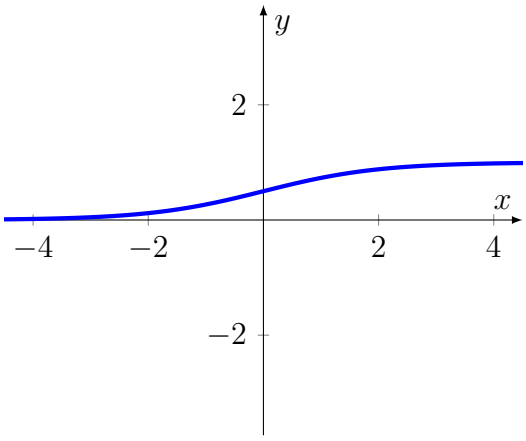
Navedene funkcije prikazane su na slici 2.4.



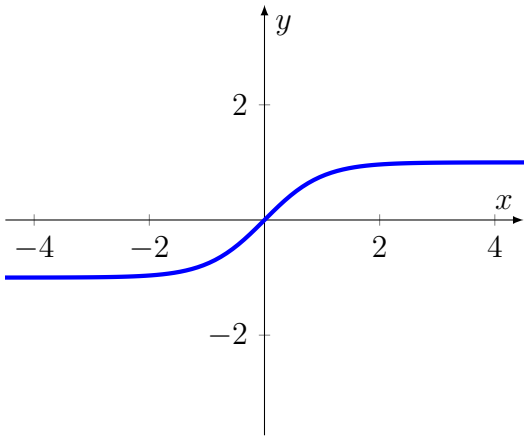
(a) Linearna funkcija



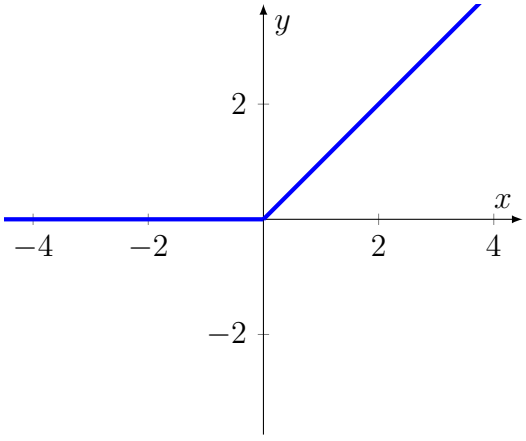
(b) Funkcija skoka



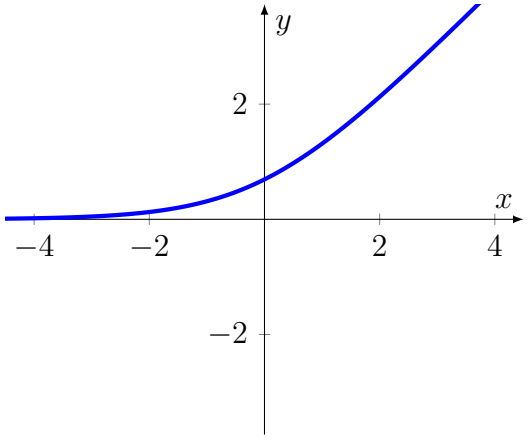
(c) Logistička funkcija



(d) Tangens hiberbolni



(e) Zglobnica



(f) Softplus

Slika 2.4: Prijenosne funkcije

Kod klasifikacije se u izlaznom sloju mreže najčešće nalazi vektor čija svaka komponenta predstavlja izglednost za pripadnost pojedinom razredu. Zbog zahtjeva da zbroj izglednosti pripadnosti razredima bude 1, za izlaze se koristi posebna funkcija koja se naziva *softmax*. Ona je definirana ovako:

$$\text{softmax}(\boldsymbol{z}, j) = \frac{e^{z_j}}{\sum_{i=0}^{k} e^{z_i}}, \quad j = 1, \ldots, k, \tag{2.11}$$

gdje je k broj razreda, a $\boldsymbol{z}$ k-dimenzionalni vektor koji odgovara izlazima prethodnog sloja.

2.2. Neuronske mreže kao univerzalni aproksimatori

Jedno od najzanimljivijih svojstava neuronskih mreža je da su univerzalni aproksimatori. Neuronska mreža s jednim skrivenim slojem i konačnim brojem neurona može aproksimirati kontinuirane funkcije nad kompaktnim podskupovima $\mathbb{R}^n$ uz neka blaga ograničenja. Teorem 2.2.1 to izražava za slučaj jednog izlaza. Općenit slučaj se može lako izvesti.

Teorem 2.2.1 (Univerzalni aproksimacijski teorem [2]) *Neka je φ nekonstantna, ograničena i monotono-rastuća kontinuirana funkcija. Neka I_m označava m-dimenzionalnu jediničnu hiperkocku $[0,1]^m$, a $C(I_m)$ prostor kontinuiranih funkcija definiranih nad I_m. Tada za svaku funkciju $f \in C(I_m)$ i realnu konstantu $\varepsilon > 0$ postoje prirodni broj N, realne konstante $v_i, b_i \in \mathbb{R}$ i realni vektori $\boldsymbol{w_i} \in \mathbb{R}^m$ uz $i = 1, \ldots, N$, tako da možemo definirati*

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} v_i \varphi\left(\boldsymbol{w_i}^{\mathsf{T}} \boldsymbol{x} + b_i\right) \tag{2.12}$$

kao aproksimaciju funkcije f, gdje je f neovisna o φ, tj. da vrijedi

$$|F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon \tag{2.13}$$

za svaki $\boldsymbol{x} \in I_m$.

To vrijedi i ako se I_m zamijeni bilo kojim kompaktnim podskupom $\mathbb{R}^m$.

Postoje i jači aproksimacijski teoremi te je pokazano da i prijenosne funkcije koje nisu ograničene (npr. zglobnica) mogu biti univerzalni aproksimatori. [6]

2.3. Regresija i klasifikacija

Dvije najčešće vrste problema za čije se rješavanje koriste neuronske mreže su regresija i klasifikacija. Regresija je preslikavanje ulaza na vrijednost iz kontinuiranog skupa vrijednosti. Klasifikacija je preslikavanje ulaza u diskretnu vrijednost koja predstavlja razred (kategoriju). Klasifikacijske mreže češće mapiraju podatak u skup vrijednosti koje definiraju razdiobu izglednosti klasifikacije u pojedine razrede.

Postoje još i različita poopćenja klasifikacije. Jedno od njih je klasifikacija kod koje se ulaz preslikava u više razdioba izglednosti pripadnosti razredima. Primjer takve klasifikacije je semantička segmentacija (opisana u poglavlju 4) kod koje se nad svakim užim dijelom slike posebno vrši takva klasifikacija.

2.4. Konvolucijske neuronske mreže

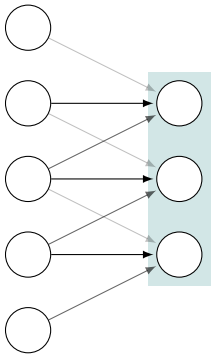
Konvolucijske neuronske mreže su unaprijedne neuronske mreže koje se sastoje od većeg broja slojeva sa svojstvima koja omogućuju efikasnije učenje i ostvarivanje boljih rezultata kod zadataka iz područja računalnog vida.

Kod općenite višeslojne unaprijedne neuronske mreže svaki skriveni sloj sastoji se od neurona koji su potpuno povezani s neuronima prethodnog sloja i od kojih svaki ima težine neovisne o težinama drugih neurona. Takva mreža bi za podatke većih dimenzija (slike) imala prevelik broj parametara, teško bi učila i bila bi sklona prenaučenosti.

Konvolucijske neuronske mreže iskorištavaju lokalne ovisnosti koje se pojavljuju na slikama. Slojevi su rijetko i lokalno povezani (slika 2.5). Svaki neuron povezan je s malim brojem bliskih neurona prethodnog sloja. Osnovna vrsta sloja je konvolucijski sloj. Konvolucijski sloj čini skup naučenih filtera od kojih svaki stvara jednu izlaznu matricu. Unutar konvolucijskog sloja neuroni koji stvaraju jednu od izlaznih matrica, tj. istu komponentu izlaza dijele iste težine, čime se iste značajke prepoznaju neovisno o položaju na ulaznom sloju. Za klasifikaciju se na kraju mreže obično koristi jedan ili više potpuno povezani slojeva (klasifikator), a unutar mreže se obično nalaze slojevi sažimanja koji smanjuju dimenzije izlaza konvolucijskih slojeva koji im prethode.

Ulaz konvolucijske mreže obično je dvodimenzionalan (monokromatska slika)

ili trodimenzionalan (višekomponentna slika, npr. *RGB*-slika). Komponente izlaza slojeva konvolucijske mreže su matrice koje se nazivaju mapama značajki. U slučaju konvolucijskog sloja, svaka mapa značajki dobije se filtriranjem svih (ili dijela) komponenata (mapa značajki) ulaza ovisno o sinaptičkim težinama pojedine komponenete sloja.



Slika 2.5: Primjer jednostavnog lokalno povezanog sloja s dijeljenim težinama

2.4.1. Konvolucijski sloj

Svojstva konvolucijskog sloja temelje se na pretpostavci da, ako je korisno prepoznavanje neke značajke na jednom mjestu, trebalo bi biti korisno takvu značajku prepoznati i na drugim mjestima na slici. To se ostvaruje dijeljenjm parametara za neurone unutar jedne komponente sloja. Izlaz jednog takvog sloja može se izračuanti konvolucijom dijeljenih težina s ulazom.

Težine koje određuju pojedinu komponentu konvolucijskog sloja nazivaju se filterom ili jezgrom (engl. *kernel*). Dvodimenzionalnom konvolucijom ulaza s jezgrom i naknadnom primjenom prijenosne funkcije dobiva se pojedina komponenta izlaza konvolucijskog sloja koja se naziva mapa značajki.

Jezgre su malih dimenzija, često 5×5 ili 7×7, ali imaju dubinu koja odgovara dubini (broju komponenata) ulaza, tj sastoje se od većeg broja dvodimenzionalnih jezgri koje se zasebno uče za svaku komponentu ulaza. Konvolucija se provodi pomicanjem jezgre po prvim dimenzijama (visina i širina) prethodnog sloja i zbrajanjem praga i umnožaka elemenata jezgre s odgovarajućim vrijednosti prethodnog sloja. Na dobivene rezultate konvolucije za dobivanje izlaza konvolucijskog sloja na njih se još primjenjuje prijenosna funkcija. Na taj način stvaraju se mape značajki od kojih svaka predstavlja ulaz

filtriran na drugačiji način. To se može formalno izraziti ovako:

$$\boldsymbol{H}_{lp} = \varphi\left(-\theta_{lp} + \sum_{q} \boldsymbol{w}_{lpq} * \boldsymbol{H}_{(l-1)q}\right). \tag{2.14}$$

Pri tome $\boldsymbol{H}_{lp}$ označava mapu značajki s redim brojem p sloja l, θ_{lp} odgovarajući prag, q redni broj mape značajki prethodnog sloja, $\boldsymbol{w}_{lpq}$ odgovarajuću komponentu konvolucijske jezgre, a oznaka $*$ operator konvolucije koji može biti definiran ovako:

$$(\boldsymbol{h} * \boldsymbol{x})_{mn} = \sum_{i} \sum_{j} h_{(m-i)(n-j)} x_{mn}, \tag{2.15}$$

gdje su $\boldsymbol{h}$ i $\boldsymbol{x}$ matrice (s pomaknutim koordinatama) ili funkcije $\mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$.

Na slici 2.6 prikazan je primjer konvolucije ulazne matrice i konvolucijske jezgre koje su dubine 1. Izlazna matrica je mapa značajki koja označava lijeve (pozitivne vrijednosti) i desne (negativne vrijednosti) rubove. U slučaju da se ulaz sastoji od više komponenata, izlazna matrica jednaka je zbroju rezulata konvolucija svih komponenata s odgovarajućim komponentama konvolucijske jezgre.

$$\begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 2 & 2 & 1 \\ 1 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 0 & 0 \\ 3 & 0 & -3 \\ 1 & 1 & -3 \end{bmatrix}$$

Slika 2.6: Primjer jednostavne dvodimenzionalne diskretne konvolucije (jedna jezgra dimenzija $2 \times 2 \times 1$) kojom se stvara jedna mapa značajki na temelju jedne ulazne mape značajki

Kako bi izlazne mape značajki bile jednake širine i visine kao ulaz, potrebno je nekako nadopuniti ulaz na mjestima na kojima konvolucijska jezgra prelazi njegove rubove. Rubovi ulaza se najčešće nadopunjuju nulama. Ponekad se jezgra umjesto jedan po jedan piksel pomiče po više piksela odjednom, čime se smanjuju dimenzije izlaza. To se naziva korak (engl. *stride*).

2.4.2. Sloj sažimanja

Često se iza konvolucijskih slojeva koriste slojevi sažimanja, čija je uloga smanjiti broj parametara mreže, čime se ubrzava računanje, a također se

smanjuje sklonost mreže prenaučenosti. Postoji više načina sažimanja, ali se pokazalo da se najbolji rezultati ostvaruju sažimanjem maksimalnom vrijednošću (engl. *max-pooling*). Pri takvom sažimanju se ulazna slika (ili mapa značajki) dijeli na grupe susjednih vrijednosti u obliku nepreklapajućih četverokuta i iz svake grupe se odabire najveća vrijednost za izlaz. Jednostavan primjer prikazan je na slici 2.7. Dimenzije tih četverokuta su najčešće 2×2, čime se širina i visina ulaza dvostruko smanjuju. Sažimanje se provodi na svakoj mapi značajki nezavisno i dubina izlaznog volumena ostaje jednaka.

$$\begin{bmatrix} 1 & 6 & 1 & 2 \\ 0 & 3 & 1 & 0 \\ 2 & 3 & 2 & 0 \\ 2 & 2 & 4 & 1 \end{bmatrix} \longrightarrow \left[\begin{array}{cc|cc} 1 & 6 & 1 & 2 \\ 0 & 3 & 1 & 0 \\ \hline 2 & 3 & 2 & 0 \\ 2 & 2 & 4 & 1 \end{array}\right] \longrightarrow \begin{bmatrix} 6 & 2 \\ 3 & 4 \end{bmatrix}$$

Slika 2.7: Sažimanje maksimalnom vrijednošću

3. Učenje neuronske mreže

Najzanimljivija karakteristika neuronskih mreža je sposobnost učenja. Parametri (težine i pragovi) neuronske mreže se učenjem nad dostupnim podacima mogu iterativno prilagođavati tako da mreža postane sposobna ostvarivati funkciju kojom dobro rješava i zadatke s kojima se nije "srela", tj. da ima sposobnost generalizacije.

Dva najčešća pristupa učenju su nadzirano i nenadzirano učenje. Kod nenadziranog učenja mreži se daju samo ulazni podaci i ona pokušava pronaći karakteristične značajke kojima se mogu dobro predstavljati ili kategorizirati podaci. Kod nadziranog učenja skup podataka nad kojim mreža uči čine parovi ulaznih i odgovarajućih izlaznih podataka. Cilj je postići da izlazi mreže budu što bliži ispravnim izlazima pazeći pri tome da se mreža dobro generalizira. Za izbjegavanje prenaučenosti se pri učenju često koristi poseban skup podataka za provjeru. U potpoglavlju 3.3 su opisane još neke tehnike koje se koriste za rješavanje problema koji se javljaju kod učenja. U nastavku poglavlja opisane su osnovne metode koje se primjenjuju kod nadziranog učenja.

Kako bi se moglo ostvariti učenje, potrebno je definirati mjeru pogreške (gubitka) kao funkciju koja ovisi o težinama $\boldsymbol{w}$ i pragovima $\boldsymbol{\theta}$. Kod regresijskih problema često se koristi srednja kvadratna pogreška:

$$E = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{3.1}$$

pri čemu je n broj izlaza pomnožen brojem uzoraka, $\hat{y}_i$ procjena izlaza, y_i ispravna vrijednost izlaza.

Kod klasifikacijskih problema se kao mjera pogreške češće koriste unakrsna entropija ili negativni logaritam izglednosti. Ispravni izlaz neuronske mreže koja se koristi za rješavanje klasifikacijskih problema najčešće se predstavlja kao vektor čija dimenzija odgovara broju razreda i kojemu samo jedna komponenta ima vrijednost 1, a ostale 0 (engl. *one-hot vector*), a moguće je i da predstavlja

razdiobu razreda. Kod regresije komponente izlaza mogu imati realne vrijednosti. Binarna unakrsna entropija je prikladna mjera pogreške ako je aktivacijska funkcija izlaza logistička funkcija [11]. Logistička funkcija se obično koristi ako neuronska mreža ima jedan ili više neovisnih izlaza za binarnu klasifikaciju. Uz iste oznake, binarna unakrsna entropija definirana je ovako:

$$E = -\frac{1}{n}\sum_{i=1}^{n}(y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)), \tag{3.2}$$

gdje $\hat{y}_i$ predstavlja procjenu komponente izlaza s rednim brojem i, a y_i stvarni izlaz koji odgovara ulazu. Ako se za prijenosnu funkciju izlaznog sloja primjenjuje softmax, pri čemu izlazi najčešće predstavljaju razdiobu vjerojatnosti razreda, prikladna funkcija pogreške je unakrsna entropija:

$$E = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic} \ln \hat{y}_{ic}, \tag{3.3}$$

gdje $\hat{y}_{ic}$ označava procjenu vjerojatnosti pripadanja uzorka razredu kojemu odgovara broj c. Ako stvarni izlazi predstavljaju točno određena razred, onda stvarna razdioba ima vrijednost 1 za jedan razred, dok za ostale razrede ima vrijednost 0. Tada se unakrsna entropija svodi na negativni logaritam izglednosti [11, 1]:

$$E = -\frac{1}{n}\sum_{i=1}^{n} \ln \hat{y}_{ic}, \quad y_{ic} = 1, \tag{3.4}$$

tj. y_{ic} je komponenta vektora $\boldsymbol{y_i}$ koja jedina ima vrijednost 1, tj. odgovara rednom broju razreda u koji je potrebno klasificirati ulaz.

3.1. Gradijentni spust

Optimizacija neuronske mreže se ostvaruje pronalaženjem težina i pragova kojima se minimizira funkcija pogreške. Za to se koristi gradijent funkcije pogreške, odnosno njene parcijalne derivacije po pojedinim težinama i pragovima. Uz poznavanja gradijenta, moguće je iterativno se približavati nekom minimumu funkcije pogreške. Osnovni algoritam kojim se to ostvaruje naziva se gradijentni spust.

Pretpostavimo da je funkcija pogreške E funkcija m parametara, $w_1, \ldots, w_m$. Tada za razliku ΔE uzrokovana malom promjenom $\Delta \boldsymbol{w} = (\Delta w_1, \ldots, \Delta w_m)$ vrijedi:

$$\Delta E \approx \nabla E^{\mathsf{T}} \cdot \Delta \boldsymbol{w}, \tag{3.5}$$

gdje je gradijent:

$$\nabla E = \left(\frac{\partial C}{\partial w_1}, \ldots, \frac{\partial C}{\partial w_m} \right). \tag{3.6}$$

Možemo odabrati pomak

$$\Delta \boldsymbol{w} = -\eta \nabla E, \tag{3.7}$$

pri čemu je η mali pozitivni broj koji se naziva stopa učenja. Uz dovoljno malu stopu učenja jednadžbe (3.5) i (3.7) osiguravaju da razlika pogreške ili pomak ΔE ne bude negativan broj, tj. da će se smanjiti pogreška E. To omogućuje postupnu minimizaciju funkcije ponavljanjem sljedećeg koraka pridruživanja nove vrijednosti parametrima:

$$\boldsymbol{w}' := \boldsymbol{w} - \eta \nabla E. \tag{3.8}$$

Kod gradijentnog spusta nije garantirano da će se pronaći globalni minimum funkcije. Međutim, u praksi se problemi uspješno rješavaju korištenjem gradijentnog spusta i sličnih algoritama temeljenih na gradijentnom spustu.

3.1.1. Stohastički gradijentni spust

Stohastički gradijentni spust je optimizacijski algoritam koji za računanje pogreške u pojedinom koraku ne koristi cijeli skup podataka, već neki njegov manji dio. Korak stohastičkog gradijentnog spusta možemo izraziti ovako:

$$\boldsymbol{w}' := \boldsymbol{w} - \eta \nabla \hat{E}_i, \tag{3.9}$$

gdje i označava da se procjena pogreške $\hat{E}_i$ u pojedinom koraku izračunava na temelju jednog primjerka ili nekog dijela skupa podataka koji se razlikuje između koraka. Time se povećava brzina izvođenja, a također se i omogućuje bolje izbjegavanje lokalnih minimuma. Pokazalo se da je u koracima gradijentnog spusta najčešće dobro koristiti manje dijelove (grupe) skupa podataka (engl. *mini-batches*). Prije svakog novog prolaza kroz skup podataka za učenje dobro je nasumično promijeniti redoslijed primjeraka.

3.1.2. Metode ubrzavanja učenja

Metoda zaleta

Metoda zaleta se koristi za ublažavanje oscilacija promjena smijera koje se pojavljuju kod stohastičkog gradijentnog spusta i ubrzavanje konvergencije

analogno lopti koja se kotrlja nizbrdo i zbog dobivene količine gibanja prelazi preko manjih povišenja na koja nailazi. Metoda zaleta promjeni parametara $\Delta \boldsymbol{w}$ u nekom koraku pridružuje promjenu parametara iz prethodnog koraka pomnoženu faktorom $\alpha \in [0, 1]$, što se može izraziti ovako:

$$\Delta \boldsymbol{w}' := \alpha \Delta \boldsymbol{w} - \eta \nabla E. \tag{3.10}$$

Uloga faktora α je prigušivanje koje omogućuje zaustavljanja u minimumu. Često se za njega uzima vrijednost oko 0.5, a dobro ga je povećati kod nižih i stabilnijih gradijenata.

Druge metode i algoritmi

Za ubrzavanje učenja pronađene su još razne druge metode. Neke prilagođavaju stopu učenja posebno za svaku težinu ovisno o stabilnosti pogreške između koraka algoritma. Kod izračuna pomaka kod algoritma *Rprop* koriste se predznaci parcijalnih derivacija funkcije pogreške u trenutnom i prethodnom koraku i iznos pomaka prethodnog koraka algoritma. Algoritam *Adagrad* postupno smanjuje stopu učenja dijeljenjem korijenom zbroja kvadrata iznosa pomaka svih prethodnih koraka za svaki parametar. Algoritam *RMSprop* [8] je bolje prilagođen učenju s korištenjem manjih grupa. Kod njega se pomak dijeli drugim korijenom pokretnog prosjeka kvadrata iznosa gradijenta. To sprječava iščezavanje gradijenta na način da iznos pomaka u prosjeku bude neovisan o iznosu gradijenta. Postoje još neke uspješne metode koje se temelje na sličnim načelima i često se kombiniraju s metodom zaleta [14].

Algoritam *Adam*

Adam (*Adaptive Moment Estimation*) [9] kao i *RMSprop* koristi pokretni prosjek kvadrata iznosa prethodnih gradijenata $\boldsymbol{m}_t$. Osim toga koristi i pokretni prosjek gradijenata $\boldsymbol{v}_t$. Uz $\boldsymbol{g}_t := (\nabla E)_t$ u koraku t, to se može izraziti ovako:

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t \tag{3.11}$$

$$\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2. \tag{3.12}$$

$\boldsymbol{m}_t$ je procjena prvog momenta (srednje vrijednosti), $\boldsymbol{m}_t$ je procjena drugog momenta s obzirom na $\mathbf{0}$ (umjesto na srednju vrijednost) i u početku se inicijaliziraju nulama. Dobre vrijednosti za faktore β_1 i β_2 su blizu 1. Kako bi se

spriječile premale vrijednosti momenata u početnim koracima, oni se na sljedeći način moduliraju:

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \tag{3.13}$$

$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1 - \beta_2^t}. \tag{3.14}$$

Sljedeći izraz opisuje ažuriranje težina (sve operacije su po elementima vektora):

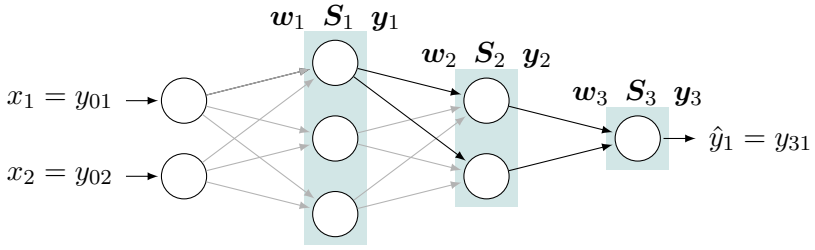
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\eta}{\sqrt{\boldsymbol{v}_t} + \epsilon}\boldsymbol{m}_t \tag{3.15}$$

Pokazalo se da su dobre vrijednosti za β_1 0.9, za β_2 0.999 i za ϵ 10^{-8}.

3.2. Propagacija pogreške unatrag

Propagacija pogreške unatrag (engl. *backpropagation*) je algoritam koji se koristi za efikasno izračunavanje gradijenta, tj. lokalne ovisnosti izlaza o pojedinim parametrima mreže.

Na slici 3.1 prikazan je primjer neuronske mreže na kojemu tamni bridovi povezuju čvorove čije je pogreške potrebno izračunati kako bi se dobila ovisnost funkcije pogreške o težini prvog sloja kojoj odgovara prvi tamni brid. Može se primijetiti da povećanjem dubine broj čvorova za koje je potrebno izračunati pogreške kako bi se izračunala pogreška jednog parametra raste eksponencijalno. Međutim, budući da se te iste pogreške koriste za računanje pogreške drugih parametara, nije ih potrebno sve svaki put ponovo računati.



Slika 3.1: Primjer neuronske mreže na kojemu tamni bridovi povezuju čvorove čije je pogreške potrebno izračunati kako bi se dobila pogreška težine prvog sloja

Izlaz pojedinog neurona nekog sloja, uz poznavanje parametara mreže, jednoznačno je određen izlazima neurona prethodnog sloja s kojima je taj

neuron povezan. Jednadžba (2.2) se za opći slučaj mreže s potpuno povezanim slojevima može napisati ovako:

$$y_{lj} = \begin{cases} -1, & j = 0 \\ x_j, & l = 0 \wedge j > 0 \\ \varphi(S_{lj}), & l > 0 \wedge j > 0, \end{cases} \tag{3.16}$$

$$S_{lj} = \sum_k y_{(l-1)k} w_{ljk}. \tag{3.17}$$

U indekse uvodimo oznake sloja i neurona. Prva oznaka u indeksu označava redni broj sloja, a druga redni broj neurona u pojedinom sloju. Radi jednostavnosti zapisa, za ulaze mreže uvodimo ekvivalentne oznake $y_{0i} = x_i$. y_{l0} označava konstantnu vrijednost jednaku -1 koju množe pragovi. Kod težina treća oznaka u indeksu označava neuron u prethodnom sloju na kojeg se odnosi težina ili prag u slučaju kad je ta oznaka 0.

Za računanje parcijalnih derivacija funkcije pogreške po težinama (u koje uključujemo i pragove) $\partial E/\partial w_{ljk}$ uvodimo još oznaku δ_{lj} koja predstavlja pogrešku težinskog zbroja unutar pojedinog neurona:

$$\delta_{lj} = \frac{\partial E}{\partial S_{lj}}. \tag{3.18}$$

Za pogrešku neurona δ_{lj}, tj. ovisnost pogreške E o zbroju S_{lj} vrijedi:

$$\begin{aligned} \delta_{lj} &= \frac{\partial E}{\partial y_{lj}} \frac{\partial y_{lj}}{\partial S_{lj}} \\ &= \frac{\partial E}{\partial y_{lj}} \varphi'(S_{lj}). \end{aligned} \tag{3.19}$$

Izravna primjena ove jednadžbe omogućuje izračunavanje ovisnosti pogreške samo za neurone izlaznog sloja. Poznavanjem pogrešaka nekog sloja moguće je izračunati pogreške sloja koji mu prethodi. Za skrivene slojeve vrijedi:

$$\begin{aligned} \frac{\partial E}{\partial y_{lj}} &= \sum_k \frac{\partial E}{\partial y_{(l+1)k}} \frac{\partial y_{(l+1)k}}{\partial S_{(l+1)k}} \frac{\partial S_{(l+1)k}}{\partial y_{lj}} \\ &= \sum_k \delta_{(l+1)k} w_{(l+1)kj} \end{aligned} \tag{3.20}$$

Za pogreške neurona nekog sloja u ovisnosti o pogreškama u sloju iza primjenom dobivene jednadžbe i jednadžbe (3.19) slijedi:

$$\delta_{lj} = \left(\sum_k \delta_{(l+1)k} w_{(l+1)kj} \right) \varphi'(S_{lj}). \tag{3.21}$$

Za ovisnost pogreške o težini (ili pragu) w_{ljk} vrijedi:

$$\begin{aligned}\frac{\partial E}{\partial w_{ljk}} &= \frac{\partial E}{\partial y_{lj}} \frac{\partial y_{lj}}{\partial S_{lj}} \frac{\partial S_{lj}}{\partial w_{ljk}} \\ &= \delta_{lj} y_{(l-1)k}.\end{aligned} \tag{3.22}$$

Ovom jednadžbom izračunava se ovisnost pogreške o pojedinim parametrima, što i je cilj ovog algoritma. Jednadžbe (3.19), (3.21) se mogu zapisati u matričnom obliku i s jednadžbom (3.22) čine skup jednadžbi koje definiraju algoritam propagacije pogreške unatrag:

$$\boldsymbol{\delta}_L = \nabla_y E \odot \varphi'(\boldsymbol{S}_L), \tag{3.23}$$

$$\boldsymbol{\delta}_l = \nabla_{y_l} E \odot \varphi'(\boldsymbol{S}_l), \tag{3.24}$$

$$\frac{\partial E}{\partial w_{ljk}} = \delta_{lj} y_{(l-1)k}, \tag{3.25}$$

pri čemu je L redni broj zadnjeg sloja, a oznaka $\odot$ označava hadamardov produkt[1].

Algoritam propagacije unatrag se može ovako iskazati:

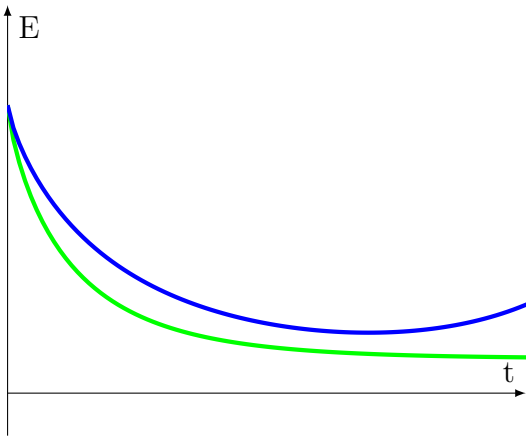
1. Postavi ulaz $\boldsymbol{y}_0 = \boldsymbol{x}$.
2. Iterativno prema naprijed izračunaj izlaze slojeva uz $l = 1, \ldots, L$: $\boldsymbol{S}_l = \boldsymbol{w}_l \boldsymbol{y}_{l-1}, \quad \boldsymbol{y}_l = \varphi(\boldsymbol{S}_l)$.
3. Izračunaj pogrešku E iz izlaza $\hat{\boldsymbol{y}} = \boldsymbol{y_L}$ i ispravnog izlaza $\boldsymbol{y}$.
4. Izračunaj pogreške zadnjeg sloja: $\boldsymbol{\delta}_L = \nabla_y E \odot \varphi'(\boldsymbol{S}_L)$.
5. Iterativno prema nazad izračunaj pogreške slojeva uz $l = L-1, \ldots, 1$: $\boldsymbol{\delta}_l = (\boldsymbol{w}_{l+1}^{\mathsf{T}} \boldsymbol{\delta}_{l+1}) \odot \varphi'(\boldsymbol{S}_L)$.
6. Izračunaj ovisnost pogreške o svakom parametru mreže: $\partial E / \partial w_{ljk} = \delta_{lj} y_{(l-1)k}$.

3.3. Poboljšavanje učenja i generalizacije

Učenjem se mreža sve više prilagođava skupu podataka za učenje. Ako je dovoljno složena, može se dogoditi da se prenauči, što znači da se previše

[1]Hadamardov produkt je za matrice $\boldsymbol{A}$ i $\boldsymbol{B}$ koje moraju biti jednakih dimenzija definiran izrazom $(\boldsymbol{A} \odot \boldsymbol{B})_{ij} = A_{ij} B_{ij}$.

prilagodi skupu za učenje i ne generalizira dobro, što daje lošije rezultate na podacima koji nisu bili korišteni za učenje. Na slici 3.2 prikazana je ovisnost pogreške na skupu za učenje i na skupu za ispitivanje o vremenu učenja. Vidi se da nakon nekog trenutka mreža počinje slabije generalizirati i pogreška na skupu za ispitivanje počinje rasti dok na skupu za učenje još uvijek opada.



Slika 3.2: Ovisnost pogreške E na skupu za učenje (zeleno) i na skupu za ispitivanje (plavo) o vremenu učenja t

3.3.1. Podjela skupa podataka

Kako bi se imalo što bolji uvid u stvarne sposobnosti mreže, skup podataka se često dijeli na tri dijela: dio za učenje, dio za validaciju i dio za testiranje. Tijekom učenja se za prilagođavanje parametra koristi samo skup za učenje. Skup za validaciju koristi se za provjeru mreže tijekom učenja, obično nakon svake epohe. Skup za testiranje koristi se za konačnu provjeru dobivenog modela.

Često je količina podataka za učenje ograničena i nije praktično odvojiti velik dio podataka na skup za validaciju. U takvim slučajevima se može koristiti unakrsna validacija. Kod K-struke unakrsne validacije (engl. *K-fold cross-validation*) podaci za učenje se nasumično podijele u K podskupova i mreža se uči na $K-1$ podskupova dok se preostali podskup koristi za validaciju. To se ponavlja za svaki od K podskupova i prosječni rezultat validacije se uzima za procjenu generalizacije.

Uz dobro odabran način validacije, moguće je zaustaviti učenje kada se prepozna točka u kojoj se počinje smanjivati njena sposobnost generalizacije.

Kada postupak učenja završi, konačnu kvalitetu rada mreže provjerava se na na skupu za testiranje. Ta se provjera radi samo jednom, kada je učenje gotovo, i mjeri konačnu kvalitetu mreže.

3.3.2. Pronalaženje dobre arhitekture mreže

Određivanje arhitekture mreže uključuje određivanje vrsta, redoslijeda i povezanosti slojeva koji se koriste i određivanje aktivacijske funkcije. Važan je i način inicijalizacije težina. Ako se sve težine jednako inicijaliziraju na nulu, ovisno o sloju i povezanosti, izlaz će na jednak način ovisiti o svakoj težini pojedinog sloja i neuroni se neće moći specijalizirati.

Kako bi mreža mogla ostvarivati složene funkcije, mora biti dovoljno velika, ali prevelika mreža je sporija i podložna je prenaučenosti. Najjednostavniji način za sprječavanje prenaučenosti je smanjivanje broja parametara mreže. To se ostvaruje smanjivanjem broja skrivenih neurona, prorjeđivanjem težina ili dijeljenjem težina, što je opisano u odjeljku 2.4. Smanjivanjem broja parametara smanjuje se ekspresivnost mreže, tj. sposobna je modelirati uži razred funkcija.

3.3.3. Regularizacija

Regularizacija obuhvaća tehnike koje se koriste za ograničavanje složenosti modela. To se često ostvaruje dodavanjem dodatnih članova funkciji pogreške kojima se potiču jednostavnija rješenja i sprječava prenaučenost.

L^1-regularizacija i L^2-regularizacija

Jedan od načina ostvarivanja regularizacije je ograničavanje težina dodavanjem člana koji je jednak L^1-normi ili L^2-normi u funkciju pogreške. Time se postiže zaglađivanje funkcije koju ostvaruje neuronska mreža [4]. Općenito, L^p-norma vektora $\boldsymbol{w}$ je definirana kao:

$$\|\boldsymbol{w}\|_p = \left(\sum_i w_i^p \right)^{\frac{1}{p}}. \tag{3.26}$$

Uz L^p regularizaciju, funkcija pogreške se može ovako napisati:

$$E_r = E + \lambda \|\boldsymbol{w}\|_p^p, \tag{3.27}$$

gdje je E pogreška, a λ faktor koji određuje utjecaj L^p regularizacije. Za regularizacijski član funkcije pogreške najčešće se koristi L^1-norma ili kvadrat L^2-norme parametara mreže. Ovisno o problemu, u praksi se najbolji rezultati postižu uz L^2-regularizaciju koja se još naziva i propadanje težina (engl. *weight decay*).

Isključivanje neurona

Točnost modela čije učenje počinje sa slučajno inicijaliziranim parametrima se povećava ako se koristi prosjek rezultata većeg broja naučenih modela, što se naziva uprosječavanje modela. Kod takvog pristupa se javljaju problemi s brzinom izvršavanja kod učenja i kod korištenja.

Ideja isključivanja neurona (engl. *dropout*) je nasumično isključiti pojedine neurone (s njihovim vezama) iz neuronske mreže tijekom učenja. Time se sprječava da utjecaj nekih neurona postane prevelik i time uzrokuje prenaučenost. Tijekom učenja se u svakom koraku uči jedna od malo manje od 2^n mogućih mreža koje se mogu dobiti isključivanjem neurona (prorjeđivanjem), gdje je n broj neurona skrivenih slojeva nad kojima isključivanje neurona ima učinak. Kao rezultat se dobije jedna neprorijeđena neuronska mreža s uravnoteženijim ulogama neurona koja bolje generalizira.

Odabir neurona koji će se isključiti je nasumičan. Svaki neuron pojedinog sloja ima vjerojatnost p da će biti zadržan. p može biti odabran na temelju rezultata provjere ili može biti fiksan. Pokazalo se da se najbolji rezultati postižu uz vrijednost vjerojatnosti zadržavanja oko 0.5, a za ulazni sloj uz neku vrijednost bližu 1. [15]

Umjetno proširivanje skupa za učenje

Za ostvarivanje dobrih modela potrebni su veliki skupovi za učenje. Što je veći skup za učenje, to model koji je učen na njemu može ostvarivati bolje rezultate kod provjere generalizacije. No, veličina skupa za učenje je ograničena jer uglavnom nije jednostavno i vremenski je zahtjevno pripremiti velike količine podataka za nadzirano učenje. Zato je dobro na odgovarajući način proširiti skup za učenje.

Kod računalnog vida to se može ostvariti transformacijama sa slučajno

odabranim parametrima. To mogu biti geometrijske transformacije kao što su rotacija, zrcaljenje i skaliranje i transformacije boje, svjetline i kontrasta. Dobar način proširivanja skupa za učenje je provođenje nasumičnih transformacija na svakom uzorku prije korištenja u koraku učenja.

4. Semantička segmentacija

Semantička segmentacija je postupak pridjelivanja semantičke oznake (engl. *label*) svakom dijelu slike. Oznake koje se pridjeljuju predstavljaju kategorije ili razrede kojima pripadaju označeni predmeti ili dijelovi slike. Rezultat semantičke segmentacije je obično matrica kod koje je za svaki piksel slike vrijednost odgovarajućeg elementa jednak cjelobrojnoj vrijednosti koja označava semantički razred.

Za prepoznavanje nekog predmeta na slici prvo je potrebno prepoznati značajke niže razine kao što mogu biti oblik nekih njegovih dijelova, obrisi i drugi uzorci koje ga karakteriziraju. Za to su prikladne konvolucijske neuronske mreže. Konvolucijski slojevi služe kao izlučivači značajki (engl. *feature extractors*). Slojevi bliže ulazu prepoznaju jednostavnije značajke kao što su rubovi i neki drugi jednostavniji uzorci. Izlazi slojeva zato se nazivaju se mape značajki (engl. *feature maps*). Svaki sljedeći sloj koristi mape značajki prethodnog sloja za prepoznavanje značajki više razine. Kod klasifikacije se na kraju mreže koristi jedan ili više po pikselima potpuno povezanih slojeva[1] koji služe za konačnu klasifikaciju na temelju značajki prepoznatih konvolucijskim dijelom[2] mreže. Izlaz mreže koja se koristi za klasifikaciju je obično vektor dimenzije jednake broju razreda, pri čemu redni broj komponente s najvećom vrijednošću odgovara rednom broju razreda u koji je slika klasificirana. Kod semantičke segmentacije se svaki piksel posebno klasificira i određuje se samo na temelju piksela na odgovarajućim položajima u mapama značajki na izlazu konvolucijskog dijela mreže.

[1] Po pikselima potpuno povezani slojevi odgovaraju konvolucijskim slojevima s jezgrom dimenzija $1 \times 1 \times n$.

[2] Pod izrazom *konvolucijski dio* se misli na dio mreže koji se sastoji od konvolucijskih slojeva i slojeva sažimanja, tj. lokalno povezanih slojeva. Taj dio konvolucijske mreže još se naziva *izlučivač značajki* (engl.*feature extractor*).

4.1. Skupovi podataka za semantičku segmentaciju

Skupovi podataka za semantičku segmentaciju se sastoje od slika i odgovarajućih datoteka sa semantičkim oznakama. Slike, tj. ulazi za sustav za semantičku segmentaciju obično je *RGB* ili *RGBD*[3] slika. Prije ulaza u neuronsku mrežu, takve slike se često predobrađuju. Mogu se prebaciti u prostor boja i normalizirati na vrijednosti iz intervala $[-1, 1]$ kako bi ulazi prvog konvolucijskog sloja u prosjeku bili blizu nuli. Oznake mogu biti na različite načine predstavljene u tekstualnim ili binarnim datotekama. U nastavku je opisan skup podataka korišten u ovom radu.

Primjer skupa podataka: *Stanford Background Dataset*

Stanford Background Dataset [7] je skup podataka za vrednovanje geometrijskog i semantičkog razumijevanja scene. Sastoji se od 715 slika odabranih iz skupova podataka *LabelMe*, *MSRC*, *PASCAL VOC* i *Geometric Context*. Slike prikazuju vanjske scene i dimenzija su oko 320×240 piksela i uglavnom sadrže barem jedan predmet prednjeg plana. Semantičke oznake su izrađene ručno.

Za svaku sliku postoje 3 vrste oznaka od kojih su za ovaj rad korištene oznake s razredima *nebo*, *stablo*, *cesta*, *trava*, *voda*, *građevina*, *planina* i *predmet prednjeg plana*. Semantičke oznake su spremljene u tekstualnim datotekama u obliku brojeva od 0 do 7 odvojenih razmacima i znakovima novog reda. Na slici 4.1 prikazan je primjer slike iz skupa stopljene s odgovarajućim semantičkim oznakama koje su predstavljene različitim bojama.

[3] D označava prostornu dubinu.

Slika 4.1: Primjer slike iz skupa podataka *Stanford Background Dataset* obojane prema semantičkim oznakama. Ljubičasta boja označava prednji plan, narančasta stabla, zelena cestu, plava građevine, a crvena nebo.

5. Programska izvedba

Programska izvedba sustava za semantičku segmentaciju ostvarena je po uzoru na sustav opisan u radu [5] uz neke izmjene. Također su korištene neke ideje iz rada [3].

5.1. Korišteni alati i tehnologije

Za programsku izvedbu korišten je programski jezik *Python 3.5.1*[1] i biblioteke *TensorFlow 0.8*[2], *NumPy*[3], *OpenCV*[4] i *scikit-learn*[5]. *TensorFlow* je biblioteka za numeričko računanje korištenjem grafova toka podataka. Korištena je za definiranje neuronske mreže i efikasno izvršavanje definiranih operacija i učenja. Biblioteka *TensorFlow* omogućuje parelelno korištenje više CPU-ova i GPU-ova (trenutno uz ograničenje da GPU-ovi podržavaju tehnologiju *CUDA*[6]). Sustavi se za *TensorFlow* simbolički definiraju varijablama (*Variable*) i operacijama (*Op*) i *Tensorflow* na temelju njih gradi optimirani graf toka podataka. Biblioteka *Numpy* korištena je za efikasno izvršavanje različitih operacija nad matricama i spremanje i učitavanje semantičkih oznaka. Biblioteka *OpenCV* korištena je za spremanje i učitavanje slika i obavljanje različitih transformacija nad njima. Biblioteka *scikit-learn* korištena je za segmentaciju slika.

[1] `https://www.python.org`
[2] `https://www.tensorflow.org`
[3] `http://www.numpy.org`
[4] `http://opencv.org`
[5] `http://scikit-learn.org`
[6] `https://en.wikipedia.org/wiki/CUDA`

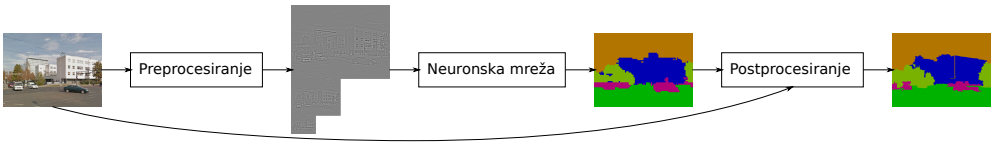
5.2. Struktura programske izvedbe

Osnovna struktura programske izvedbe se može podijeliti na dio za pripremu podataka, dio za pristupanje podacima, dio za obradu izvan neuronske mreže (preprocesiranje i postprocesiranje), neuronsku mrežu i dio za vrednovanje. Središnji dio je konvolucijska neuronska mreža definirana u razredu `SemSegNet`. Datotečna struktura programske izvedbe prikazana je na slici 5.1. Imena datoteka koje sadrže razrede odgovaraju imenima razreda.

```
semsegnet/
├── data/
│   ├── preparers/
│   │   ├── abstract_preparer.py
│   │   └── iccv09_preparer.py
│   ├── abstract_dataset.py
│   ├── dataset.py
│   └── dataset_dir.py
├── processing/
│   ├── format.py
│   ├── labels.py
│   ├── segmentation.py
│   ├── shape.py
│   └── transform.py
├── util/
│   ├── directory.py
│   ├── file.py
│   ├── path.py
│   └── display_window.py
├── scripts/
│   └── *.py
├── processing.py
├── evaluation.py
├── semsegnet.py
└── tf_utils.py
```

Slika 5.1: Osnovna datotečna struktura programske izvedbe

Logičku struktura rada sustava čine 3 dijela: preprocesiranje, konvolucijska neuronska mreža i postprocesiranje. To je ilustrirano na slici 5.2. Ulazna slika se prvo priprema za neuronsku mrežu, u njoj se obradi i dobije se grublja segmentacija. U postprocesiranju se ulazna slika segmentira superpikselima, i segmentacija se bolje prilagođava obrisima.



Slika 5.2: Osnovna arhitektura sustava

5.3. Priprema i pristup podacima

Svaki skup podataka koji se koristi mora imati svoj razred koji služi za preoblikovanje skupa podataka u *standardni* oblik koji je definiran na sljedeći način (slika 5.3): proizvoljno nazvan direktorij sadrži direktorij `images`, direktorij `labels` i tekstualnu datoteku `dataset.info` u koju je upisan broj razreda. Direktorij `images` sadrži *RGB*-slike u formatu *PNG*. Direktorij `labels` sadrži dvodimenzionalne matrice okteta u formatu biblioteke *NumPy* koje imaju vrijednosti koje odgovaraju razredima odgovarajućih piksela.

```
<skup podataka>/
├── images/
│   └── *.png
├── labels/
│   └── *.lab.npy
└── dataset.info
```

Slika 5.3: Datotečna struktura skupa podataka u standardnom obliku

Razred za preoblikovanje skupa podataka mora sadržavati funkciju `prepare` koja kao ulaznu vrijednost prima putanju izvornog skupa podataka, a vraća putanju preoblikovanog skupa podataka, tj skupa podataka u *standardnom* obliku. To je definirano u apstraktnom razredu `AbstractPreparer` u modulu `abstract_preparer.py` i za *Stanford Background Dataset* ostvareno u razredu `Iccv09Preparer` u modulu `iccv09_preparer.py`.

U modulu `dataset_dir.py` definirane su funkcije za spremanje i učitavanje slika i labela za standardni oblik skupa podataka.

Razred `Dataset` se koristi za učitavanje podataka, podjelu u podskupove stvaranjem novih primjeraka istog razreda i slijedno dohvaćanje manjih grupa parova slika i oznaka. Također omogućuje nasumično miješanje podataka kako bi se pri učenju u novoj epohi bili poredani izmijenjenim redoslijedom.

5.4. Preprocesiranje

Prije postavljanje na ulaz neuronske mreže, slika se predobrađuje. Prvo se iz prostora boja RGB prebacuje u prostor boja YUV[7], gdje Y predstavlja komponentu svjetline koja će u neuronskoj mreži imati više veza s prvim slojem neuronske mreže od ostalih komponenata. Zatim se prebacuje u zapis realnim brojevima i vrijednosti se iz raspona $[0, 255]$ normaliziraju na raspon $[0, 1]$. Na kraju se iz slike konstruira Laplaceova piramida[8] (bez najniže razine) koja se može postaviti na ulaz neuronske mreže. Primjer piramide Y-komponente je na slici 5.4.

Slika 5.4: Prve tri razine Laplaceove piramide. Slika je dobivena skaliranjem vrijednosti iz raspona $[-1, 1]$ na raspon $[0, 255]$.

Pri učenju se prije opisane pripreme za neuronsku mrežu umjetno proširuje skup podataka na način da se slika i odgovarajuće oznake geometrijski transformiraju. Prvo se rotira za slučajno odabran broj stupnjeva iz intervala $[-8, 8]$, zatim se skalira slučajnim faktorom iz intervala $[0.9, 1.1]$ i zrcali oko

[7] https://en.wikipedia.org/wiki/YUV

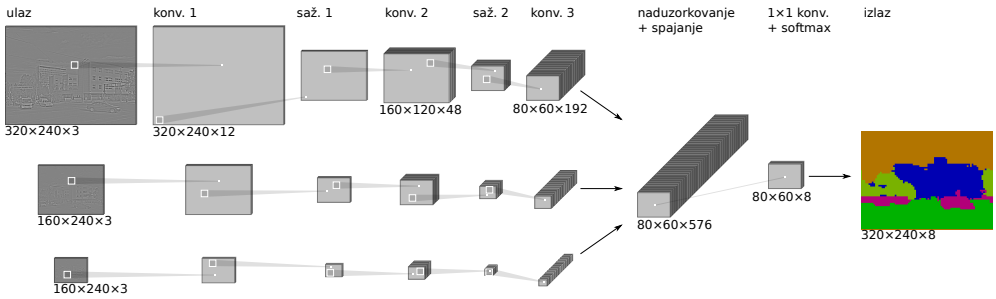
[8] https://en.wikipedia.org/wiki/Pyramid_(image_processing)

vertikalne osi s vjerojatnošću 0.5.

Paket `processing` sadrži module s funkcijama koje služe za obradu i prilagođavanje slika i semantičkih oznaka različitim dijelovima programske izvedbe. Većina tih funkcija služi predobradi. U modulu `preprocessing.py` nalaze se funkcije koje apstrahiraju postupke navedene u dva prethodna odlomka.

5.5. Arhitektura mreže

Neuronska mreža se može podijeliti u dvije logičke cjeline: izlučivač značajki i klasifikator. Detektor značajki se sastoji od nekoliko konvolucijskih slojeva i slojeva sažimanja u više razina. Klasifikator se sastoji od po pikselima potpuno povezanog sloja, na što se primjenjuje *softmax* za dobivanje razdioba izglednosti razreda. Arhitektura mreže je prikazana na slici 5.5.



Slika 5.5: Arhitektura mreže

Detektor značajki sastoji se od tri paralelne razine od kojih svaka za ulaz ima jednu razinu Laplaceove piramide. Svaka razina se sastoji od konvolucijskog sloja s konvolucijskom jezgrom dimenzija 7×7 i prijenosnom funkcijom $ReLU$ i sloja sažimanja koji izlaz konvolucijskog sloja sažima maksimalnom vrijednošću. Jedino se iza zadnjeg (trećeg) konvolucijskog sloja ne nalazi sloj sažimanja. Konvolucijske jezgre su zajedničke odgovarajućim slojevima različitih razina budući da je cilj pronaći jednake značajke neovisno o njihovoj veličini. Osim za prvi sloj, svaka mapa značajki nekog sloja računa se na temelju svih mapa značajki prethodnog sloja. Ovisno o dubini prvog sloja, otprilike 5/8 mapa značajki prvog sloja računa se samo na temelju Y-komponente ulaza, dok se ostale računaju samo na temelju kromatskih komponenata (U i V) ulaza. Dimenzije prvog sloja svake razine su $2^{-(S-1)}w \times 2^{-(S-1)}h \times D$, gdje su w i h

širina i visina originalne slike (ili prve razine Laplaceove piramide), a S redni broj razine, a D dubina prvog sloja. Ako mreža ima L konvolucijskih slojeva, izlazne mape značajki konvolucijskog dijela mreže su 2^{L-1} puta manje visine i širine i 4^{L-1} puta veće dubine od mapa značajki prvog konvolucijskog sloja.

Dobivene mape značajki se za ulaz klasifikatora interpolacijom *najbližeg susjeda* (engl. *nearest neighbour*) povećavaju na veličinu najveće od njih, što je u slučaju mreže s L konvolucijskih slojeva i $L-1$ slojeva sažimanja između njih $2^{-(L-1)}w \times 2^{-(L-1)}h \times D$. Spajaju se u jednu mapu značajki dubine $3 \cdot 4^{L-1}D$ koja čini ulaz klasifikatora koji svaki ulazni piksel klasificira neovisno o položaju te je ekvivalentan konvolucijskom sloju s jezgrom dimenzija 1×1. Dobiva se izlaz širine i visine kao njegov ulazna mapa značajki i dubine jednake broju razreda. Na njega se primjenjuje *softmax* za svaki piksel.

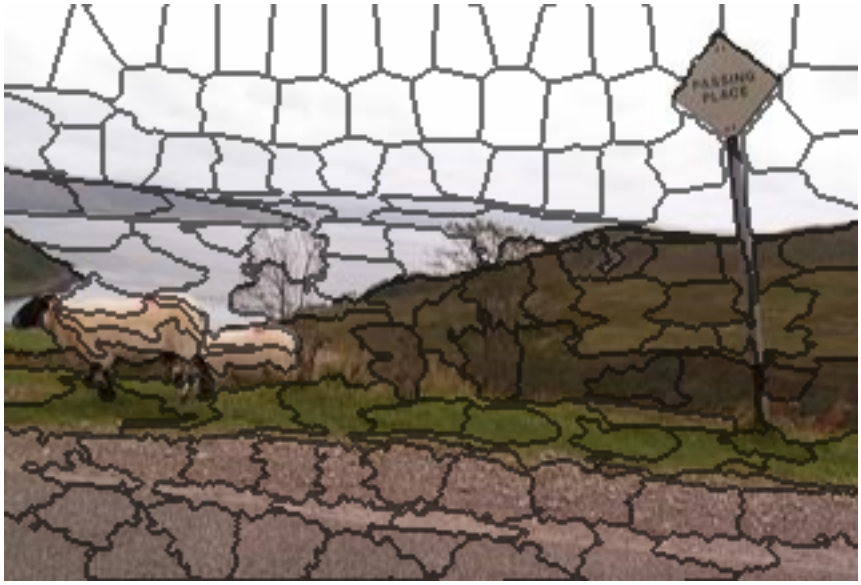
Neuronska mreža je ostvarena u razredu `SemSegNet`. Moguća je konfiguracija hiperparametara mreže kao što su broj slojeva, broj razina, dubina, algoritam učenja pri instanciranju, učenje, spremanje i pokretanje modela. Unaprijed zadane postavke dimenzija mreže su: $L = 3$, $S = 3$ i $D = 12$.

5.6. Postprocesiranje

Semantičke oznake koje se dobivaju kao izlaz mreže su šumovite i ne prate dobro obrise. Zato se često koriste metode kojima se prepoznaju obrisi i grupiraju područja slike koja imaju određene karakteristike koje ih odvajaju od drugih područja slike. U ovom radu korištena je segmentacija slike superpikselima algoritmom *SLIC* [13] implementiranim u biblioteci *scikit-learn*. Na slici 5.6 prikazan je rezultat segmentacije jedne slike na 200 segmenata.

Konačne semantičke oznake koje sustav pridjeljuje ulaznoj slici dobivaju se tako da se za svakoj skupini piksela dobivenoj segmentacijom superpikselima pridijeli semantička oznaka izlaza mreže koja se unutar te skupine najčešće pojavljuje.

Funkcije koje to obavljaju nalaze se u modulu `segmentation.py`.



Slika 5.6: Primjer slike iz skupa podataka *Stanford Background Dataset* s označenim segmentima dobivenim algoritmom *SLIC*

6. Rezultati

Sustav je evaluiran na skupu *Stanford Background Dataset* i rezultati su uspoređeni s rezultatima drugih autora. Također je uspoređena točnost uz i bez korištenja superpiksela.

6.1. Mjere

Najjednostavnija mjera kod klasifikacije je točnost (engl. *accuracy*). Ona predstavlja omjer broja točno klasificiranih primjeraka i ukupnog broja primjeraka. Kod semantičke segmentacije tome odgovara točnost piksela (engl. *pixel accuracy*), što predstavlja omjer točno klasificiranih piksela i ukupnog broj piksela.

Kod semantičke segmentacije udio različitih razreda često je neravnomjeran i događa se da model kod kojega se kod učenja to ne uzima u obzir zanemaruje rijetko zastupljeni razredi. Mjera (srednje) točnosti razreda (odziv, engl. *(mean) class accuracy*, *recall*) je definirana kao srednja točnost piksela po svakom razredu, tj. zbroj omjera ispravno pogođenih piksela svih slika koji pripadaju nekom razredu i ukupnog broj piksela koji pripadaju tom razredu za svaki razred podijeljen brojem razreda. Može se izraziti na sljedeći način:

$$\frac{1}{C}\sum_{c=1}^{C}\frac{|\hat{Y}_c \cap Y_c|}{|Y_c|}, \tag{6.1}$$

gdje je C broj razreda, $\hat{Y}_c$ skup piksela za koje je procijenjeno da pripadaju razredu s rednim brojem c, a Y_c skup piksela koji stvarno pripadaju tom razredu.

Još jedna česta mjera koju valja spomenuti je omjer presjeka i unije (engl. *intersection over union*, *IOU*) koja se uz iste oznake izražava ovako:

$$\frac{1}{C}\sum_{c=1}^{C}\frac{|\hat{Y}_c \cap Y_c|}{|\hat{Y}_c \cup Y_c|}. \tag{6.2}$$

6.2. Rezultati na skupu *Stanford Background Dataset*

Za skup *Stanford Background Dataset* korištena je arhitektura i postupci opisani u odjeljcima 5.3 - 5.6. Zapis slika prilagođenih na veličinu 320×240 pretvara se iz *RGB* u *YUV*. Raspon vrijednosti piksela normalizira se na interval $[0, 1]$ i konstruira se trorazinska Laplaceova piramida od koje svaka razine predstavlja ulaz jedne razine konvolucijske neuronske mreže. Prvi konvolucijski sloj pojedine razine stvara 12 mapa značajki od kojih se 7 računa samo na temelju Y-komponente, a preostalih 5 na temelju preostalih komponenata ulaza. Na temelju mapa značajki prvog sloja sažetih maksimalnom vrijednošću drugi konvolucijski sloj stvara 48 mapa značajki. Još jednom se provodi sažimanje i posljednji konvolucijski sloj stvara 192 mape značajki. Izlazne mape značajki druge i treće razine se povećavaju na veličinu izlazne mape značajki prvog sloja i sve se spajaju u jednu mapu značajki dimenzija $80 \times 60 \times 576$. Dobivena mapa značajki se koristi kao ulaz klasifikatora koji svaki piksel neovisno klasificira uz primjenu funkcije *softmax* za dobivanje izlaznih oznaka koje se mogu uspoređivati s oznakama za učenje.

Pri učenju se prije klasifikatora primjenjivalo se isključivanje neurona s vjerojatnošću isključivanja neurona 0.5. Za funkciju pogreške korišten je prosječni negativni logaritam izglednosti po pikselima, a kao optimizacijski algoritam korišten je algoritam *Adam* sa stopom učenja 10^{-3} i preporučenim vrijednostima ostalih parametara. Za računanje gradijenta u pojedinom koraku učenja korištene su manje grupe uzoraka iz skupa za učenje veličine 11.

Mreža je testirana podjelom skupa podataka na 5 dijelova jednake veličine i provođenjem učenja 5 puta, svaki put izdvajajući jedan od tih 5 dijelova za testiranje, a učeći na preostala 4 dijela (engl. *5-fold cross-validation*). Konačni rezultati dobiveni su kao prosjek rezultata svih 5 testiranja i prikazani su tablici 6.1.

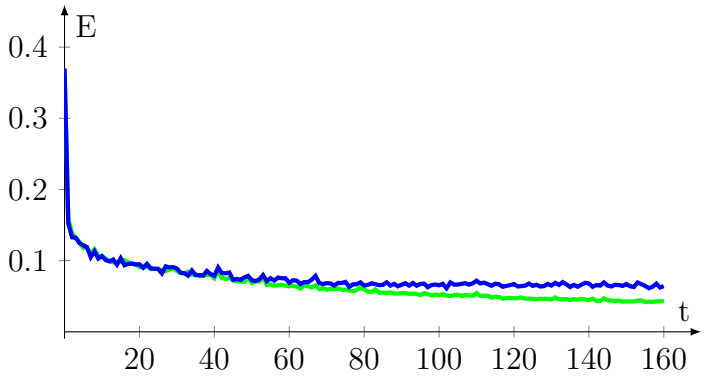
Može se primijetiti da su postignuti lošiji rezultati točnosti razreda. Glavni razlog tomu je što se neki razredi rjeđe pojavljuju, a u funkciji pogreške koja se koristila pri učenju to se nije uzimalo u obzir. Primjer funkcije pogreške koja to uzima u obzir je Bayesov negativni logaritam izglednosti gdje se pogreška pojedine klasifikacije množi faktorom obrnuto proporcionalnim učestalosti

razreda koji je krivo klasificiran.

Sustav	Točnost piksela	Točnost razreda
Farabet et al. 2013.	0.788	0.724
Farabet et al. + superpikseli 2013.	0.804	0.746
Mohan 2014.	0.842	0.784
Borko 2015.	0.742	0.681
Konv. mreža s 3 razine	0.743	0.597
Konv. mreža s 3 razine + *dropout*	0.757	0.603
Konv. mreža s 3 razine + *dropout* + *SLIC*	0.756	0.602

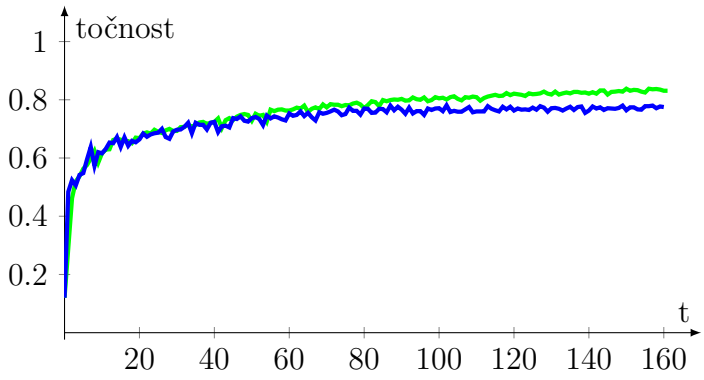
Tablica 6.1: Rezultati na skupu *Stanford Background Dataset*

Na slici 6.1 i slici 6.2 prikazani su rezultati mjerenja pogreške tijekom 160 epoha učenja. Učenje i testiranje je na grafičkoj kartici *NVIDIA GTX 970* trajalo oko 80 minuta.



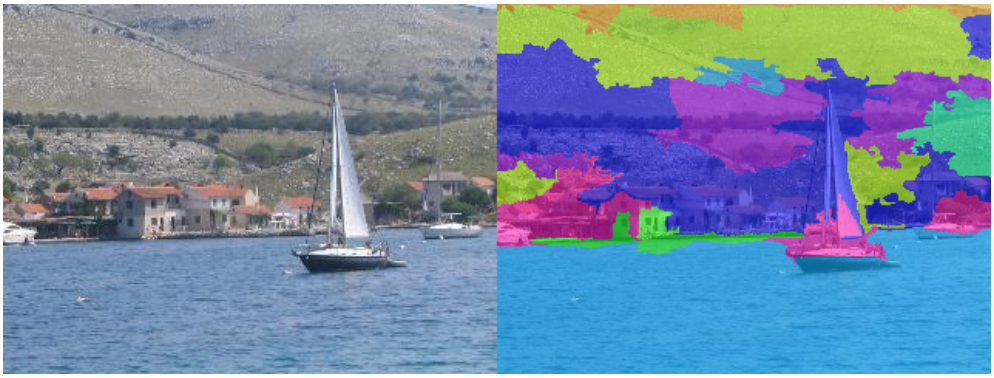
Slika 6.1: Ovisnost funkcije pogreške E na skupu za učenje (zeleno) i na skupu za ispitivanje (plavo) o broju završenih epoha t.

Na slici 6.3 prikazan je rezultat semantičke segmentacijom na primjerku iz skupa za testiranje. Primjer lošijeg rezultata prikazan je na slici 6.4.



Slika 6.2: Ovisnost točnosti piksela na skupu za učenje (zeleno) i na skupu za ispitivanje (plavo) o broju završenih epoha t.

Slika 6.3: Primjer rezultata semantičke segmentacije. Narančasta boja označava nebo, žuto-zelena stabla, plava građevine, zelena cestu, a ljubičasta prednji plan.



Slika 6.4: Primjer lošijeg rezultata semantičke segmentacije. Plava boja označava vodu, plavija ljubičasta planinu, svijetloplava travu. Za ostale boje vrijedi isto kao na slici 6.3.

7. Zaključak

U ovom radu prikazani su način rada, osnovna svojstva neuronskih mreža i pojmovi vezani uz njih. Opisana su svojstva konvolucijskih mreža. Detaljnije su razrađena načela učenja neuronskih mreža gradijentnim spustom i algoritam propagacije pogreške unatrag. Također su prikazani načini ubrzavanja učenja i rješavanja problema prenaučenosti.

Razvijena je programska izvedba sustava za semantičku segmentaciju koji se temelji na dubokoj višerazinskoj konvolucijskoj neuronskoj mreži. Mreža je ostvarena korištenjem biblioteke *TensorFlow* koja omogućuje simboličko definiranje neuronske mreže i efikasno izvršavanje na grafičkim karticama.

Mreža je napravljena po uzoru na konvolucijsku mrežu predstavljenu u radu [5]. Glavna ideja je da se mreža sastoji od tri razine od kojih svaka razina kao ulaz dobiva obrađenu sliku dvostruko manje veličine od više razine i da se dijeljenjem težina između razina ostvaruje prepoznavanje predmeta neovisno o njegovoj veličini na slici. Ostvarena je slična mreža uz glavnu razliku manje dubine slojeva i jednostavnijeg načina povezanosti između konvolucijskih slojeva i njihovih ulaznih mapa značajki nakon prvog sloja.

Sustav je testiran na skupu podataka *Stanford Background Dataset* i postignuti su zadovoljavajući razultati. U budućnosti bi bilo dobro isprobati za funkciju pogreške koristiti neku funkciju koja potiče veću točnost razreda. Također bi bilo dobro isprobati drugačiju povezanost konvolucijskih slojeva i drugačiju organizaciju klasifikacijskog dijela mreže.

Literatura

[1] Why minimize negative log likelihood? 2011. URL https://quantivity.wordpress.com/2011/05/23/why-minimize-negative-log-likelihood.

[2] Universal approximation theorem — Wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/wiki/Universal_approximation_theorem. [10.5.2016.].

[3] Ivan Borko. Semantička segmentacija prirodnih scena dubokim neuronskim mrežama. Magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva. URL https://github.com/iborko/theano-conv-semantic.

[4] John A. Bullinaria. Neural computation: Improving generalization. 2015. URL http://www.cs.bham.ac.uk/~jxb/INC/l10.pdf.

[5] Clement Farabet, Camille Couprie, Laurent Najman, i Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. URL http://yann.lecun.com/exdb/publis/pdf/farabet-pami-13.pdf.

[6] Xavier Glorot, Antoine Bordes, i Yoshua Bengio. Deep sparse rectifier neural networks. U *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, svezak 15, stranice 315–323, 2011. URL http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf.

[7] S. Gould, R. Fulton, i D. Koller. Decomposing a scene into geometric and semantically consistent regions. U *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009. URL http://ai.stanford.edu/~sgould/papers/iccv09-sceneDecomposition.pdf.

[8] Geoffrey Hinton. Neural networks for machine learning, lecture 6a: Overview of mini-batch gradient descent. 2012. URL http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[9] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization. 2014. URL http://arxiv.org/pdf/1412.6980.pdf.

[10] Y. LeCun, L. Bottou, G. Orr, i K. Muller. Efficient backprop. U *Neural Networks: Tricks of the trade*, 1998. URL http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf.

[11] Michael A. Nielsen. Neural networks and deep learning, 2015. URL http://neuralnetworksanddeeplearning.com/. [10.5.2016.].

[12] Christopher Olah. Home — colah's blog, 2016. URL http://colah.github.io/. [10.5.2016.].

[13] Kevin Smith Aurelien Lucchi Pascal Fua Radhakrishna Achanta, Appu Shaji i Sabine Süsstrunk. SLIC superpixels. 2016. URL http://ferko.fer.hr/ui/ann-20160529.pdf.

[14] Sebastian Ruder. Optimizing gradient descent. 2016. URL http://sebastianruder.com/optimizing-gradient-descent/index.html.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, i Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[16] Marko Čupić. Umjetna inteligencija: Umjetne neuronske mreže. 2016. URL http://ferko.fer.hr/ui/ann-20160529.pdf.

Semantička segmentacija slika dubokim konvolucijskim mrežama

Sažetak

U ovom radu opisane su neuronske mreže s posebnim naglaskom na duboke konvolucijske neuronske mreže. Obrađene su načela učenja i metode koje se primjenjuju za učenje neuronskih mreža. Programski je izveden i evaluiran sustav za semantičku segmentaciju koji se temelji na dubokoj konvolucijskoj neuronskoj mreži. Na kraju su prikazani rezultati.

Ključne riječi: računalni vid, neuronske mreže, konvolucijske neuronske mreže, klasifikacija, semantička segmentacija

Semantic Image Segmentation with Deep Convolutional Networks

Abstract

This paper describes neural networks with focus on deep convolution neural networks. Neural network training principles and methods are described. A system for semantic segmantation based on a deep convolutional network was implemented and evaluated. Lastly, the results are presented.

Keywords: computer vision, neural networks, convolutional neural networks, classification, semantic segmentation