SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3948

Raspoznavanje objekata konvolucijskim neuronskim mrežama

Dario Smolčić

Zagreb, lipanj 2015.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod

Računalni vid je područje koje uključuje metode za dohvaćanje, obrađivanje i shvaćanje slika i općenito podataka velikih dimenzija te je zanimljivo područje računalne znanosti zbog mogućnosti široke primjene u današnjem svijetu. Jedna od podgrana ovog područja je raspoznavanje objekata.

Ljudi su sposobni prepoznati mnoštvo različitih objekata sa jako malo truda no za računala je to složen proces koji ima brojna ograničenja koja ljudi nemaju. Uzmimo u obzir da se slika u računalu reprezentira kao višedimenzionalni niz jačina svjetlosti. Promjene u prikazu objekta poput različite orijentacije, skaliranja, i osvijetljenja objekta su u digitalnim slikama prestavljene sa različitim podatcima. Objekt također može biti i zaklonjen. Dobar model raspoznavanja mora biti otporan na ove varijacije te je zato problem raspoznavanja objekata još uvijek neriješen i u zadnjih nekoliko desetljeća su razvijene brojne metode kojima se pokušava riješiti ovaj problem. Za razliku od pisanja klasičnih algoritama poput sortiranja brojeva za problem klasifikacije objekata nije očito kako bi se mogao napisati takav algoritam gdje su sve varijacije ulaza posebno obrađene u kodu. Zato se za klasifikaciju objekata koristi pristup usmjeren na podatke (engl. *data-driven approach*). Programu se da veliki broj ulaza sa velikom količinom primjera za svaku klasu te se razvije algoritam učenja koji učitava date primjere te uči o vizualnom prikazu svake klase. Takve programe nazivamo klasifikatorima.

U zadnjih nekoliko desetljeća su razvijeni različiti klasifikatori za što točnije prepoznavanje objekata. Među tim klasifikatorima su i umjetne neuronske mreže. Ispostavilo se da se sa dubokim neuronskim mrežama trenutno dobivaju najbolji rezultati za problem klasifikacije. Najkorišteniji oblik dubokih neuronskih mreža u račnunalnom vidu su konvolucijske neuronske mreže.

Cilj ovog rada je razviti implementaciju konvolucijske neuronske mreže za primjenu na osobnim računalima, optimirati hiperparametre mreže te vrednovati učinak naučene mreže. Razvijena mreža će se testirati na skupu MNIST rukom pisanih znamenki.

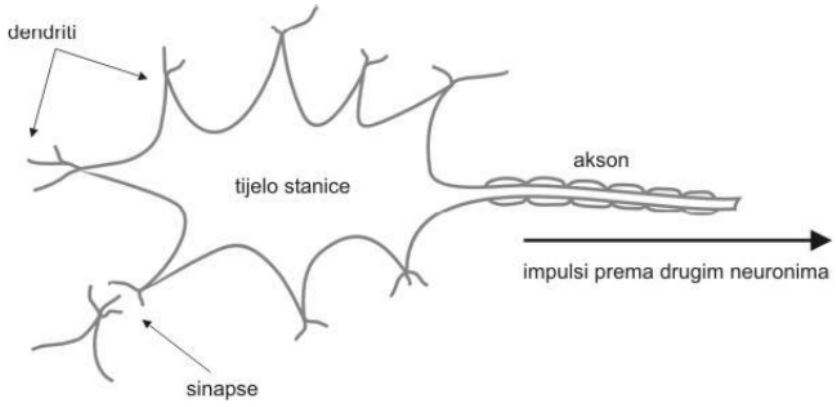
2. Neuronske mreže

Područje umjetnih neuronskih mreža (engl. *Artificial Neural Networks - ANN*) je prvotno bilo inspirirano sa modeliranjem biološkog živčanog sustava, a tek kasnije se počelo koristiti u sklopu strojnog učenja.

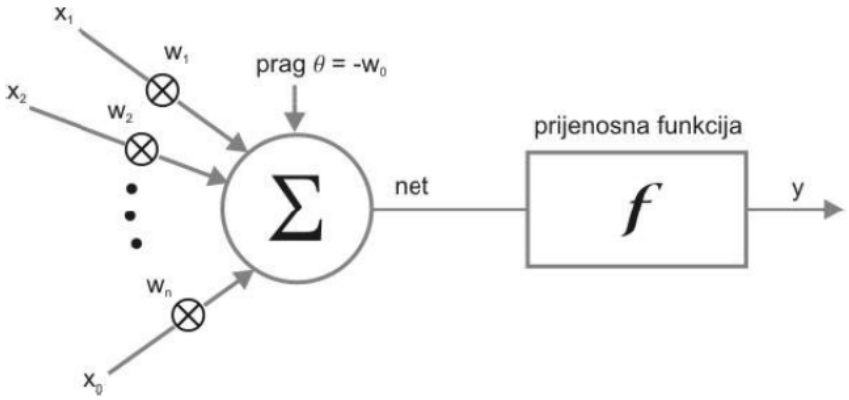
2.1. Neuron

Radi razumijevanja neuronske mreže potrebno je prvo razumijeti funkcioniranje jednog neurona. Ljudski živčani sustav se sastoji od otprilike 86 bilijona neurona koji su povezani sa 10^{14} do 10^{15} sinapsi. Svaki neuron dobiva svoje ulazne signale kroz dendrite i šalje izlazni signal kroz akson. Akson je sa sinapsama spojen sa dendritima drugih neurona. Na slici 2.1 možemo vidjeti izgled biološkog neurona.

U modelu umjetnog neurona signali koji putuju aksonom (npr. x_0) se množe sa sinaptičkim snagama dendrita(težinama) drugih neurona (npr. w_0). Ideja je da se sinaptičke snage mogu mijenjati sa učenjem te određuju utjecaj jednog neurona na drugi. Svaki neuron ima aktivacijsku funkciju koja uzima sumu umnoška ulaza neurona sa pripadnim težinama i praga (θ) te ih preslikava na izlaz neurona koji modelira signal



Slika 2.1: Biološki neuron



Slika 2.2: Umjetni neuron

na aksonu(y). Na slici 2.2 možemo vidjeti model umjetnog neurona.

Označimo ulaze sa $x_1, x_2, ..., x_n$ te njihove pripadne težine sa $w_1, w_2, ..., w_n$, i prag sa θ. Onda možemo izlaz neurona zapisati kao:

$$y = f(\sum_{i=1}^{n}(x_i * w_i) + \theta) \tag{2.1}$$

Radi pojednostavljenja se često uzima oznaka w_0 umjesto θ te se dodaje jedan ulaz x_0 koji je stalno jednak 1. Sa ovom modifikacijom izlaz se može izraziti kao:

$$y = f(\sum_{i=0}^{n}(x_i * w_i)) \tag{2.2}$$

2.1.1. Aktivacijske funkcije

Postoje veliki izbor aktivacijskih funkcija no u praksi se koriste samo neke koje su se pokazale korisnima. Spomenuti ćemo četiri različite aktivacijske funkcije (slika **??**) te njihove karakteristike.

Najobičnija aktivacijska funkcija je linearna aktivacijska funkcija koja je preslikava svoj ulaz na izlaz. Ovakav tip aktivacijske funkcije ne koristimo u dubokim neuronskim mrežama zato što onemogućava učenje mreže.

Step funkcije u neuronima funkcioniraju kao prekidači. Izlaz funkcije može poprimiti samo dvije različite vrijednosti ovisno o tome da li je ulaz manji ili veći od nekog praga. Primjer jedne ovakve funkcije je:

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{2.3}$$

Ovakva funkcija je korisna za binarne klasifikatore ali se ne koristi u dubokim neuronskim mrežama. Jedan od razloga je to što je za algoritam unazadne propagacije (kasnije ojašnjen) potrebna derivabilna ili po dijelovima derivabilna funkcija. Također zbog same definicije funkcije mala promjena ulaza može dovesti do potpuno suprotne aktivacije neurona čak iako su ulazi jako slični što je nepoželjno svojstvo za našu primjenu.

Sigmoidalne aktivacijske funkcije se najčešće koriste u praksi kod dubokih neuronskih mreža. Ovakve funkcije su derivabilne na cijeloj domeni i ograničene su što su dobra svojstva za algoritam unazadne propagacije i učenje mreže. Dvije najčešće korištene sigmoidalne funkcije su logistička funkcija i funkcija hiperbolnog tangensa. Primjer logističke funkcije dan je u izrazu 2.4.

$$f(x) = \frac{1}{1 - e^{-kx}} \tag{2.4}$$

U ovom radu nećemo koristiti logističku funkcije već funkciju hiperbolnog tangensa. Po uzoru na DODATI REFERENCU NA ČLANAK koristiti ćemo skaliranu funkciju hiperbolnog tangensa prema izrazu 2.7 čija je derivacija dana sa 2.6

$$f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right) \tag{2.5}$$

$$\frac{f(x)}{dx} = 1.444\left(1 - \tanh^2\left(\frac{2}{3}x\right)\right) \tag{2.6}$$

Još jedna aktivacijska funkcija koja je u zadnje vrijeme davala jako dobre rezultate je linearna rektifikacijska funkcija definirana kao:

$$f(x) = max(0, x) \tag{2.7}$$

Prema REFERENCA NA ČLANAK korištenjem ove aktivacijske funkcije je postignuta čak 6 puta brža konvergencija mreže. Funkcija je po dijelovima derivabilna i nije linearna(objašnjeno u poglavlju REFERENCA NA POGLAVLJE) te je također vrlo jeftina za izračunat (dovoljan je jedan if uvjet u kodu).

2.2. Arhitektura neuronske mreže

Povezivanjem velikog broja neurona nastaju neuronske mreže. Neuronske mreže su modelirane kao kolekcije neurona koje su povezane acikličkim grafom. Neuroni u neuronskim mrežama su najčešće organizirani po slojevima (slika **??**). Razlikujemo ulazni, izlazni i skriveni sloj. U ulaznom sloju na ulaze neurona dovodimo podatke

koje je potrebno klasificirati. Na primjer, za rukom pisane znamenke bi pojedini ulaz bio pisana znamenka. Izlazi neurona ulaznog sloja su spojeni sa ulazima neurona skrivenog sloja. Skrivenih slojeva može biti više pa su zato izlazi neurona skrivenih slojeva povezani sa ulazima neurona idućih skrivenih slojeva ili sa ulazima neurona izlaznog sloja. Izlaz iz neurona izlaznog sloja se interpretira kao klasa koju je mreža klasificirala. Na primjer, ako klasificiramo rukom pisane znamenke onda postoji 10 različitih klasa i deset neurona u izlaznom sloju. Na temelju tog izlaza (najčešće u obliku brojeva od 0 do 1) inerpretiramo rezultat klasifikacije neuronske mreže. Slojevi mogu ali i nemoraju biti u potpunosti povezani poput primjera na slici **??**. To znači da su svi neuroni trenutnog sloja povezani sa svim neuronima sljedećeg sloja.

Dubokim neuronskim mrežama nazivamo mreže koje imaju dva ili više skrivenih slojeva. Ispostavilo se da su duboke neuronske mreže pogodnije za kompleksnije probleme klasifikacije i da ostvaruju dobre rezultate. Tu činjenicu možemo objasniti sa intuitivnim objašnjenjem svrhe skrivenih slojeva u mreži. Možemo reći da svaki sloj mreže obrađuje podatke na drugoj razini apstrakcije i na temelju tih podataka donosi neku odluku, odnosno daje neki izlaz. Uzmimo na primjer neuronsku mrežu napravljenu za svrhu raspoznavanja ljudskih lica na slici. Izlaz iz te mreže bi nam govorio da li se na slici nalazi ljudsko lice ili ne. Razmatrajmo prvo višu razinu apstrakcije. Sa ljudskog aspekta problem postojanja ljudskog lica na slici bismo mogli rastaviti na više manjih problema poput: da li su na slici dva oka? Da li se na slici nalazi nos, usta? Jesu li usta ispod nosa? U kontekstu neuronske mreže bismo mogli reći da odgovore na ta pitanja dobivamo od zadnjeg skrivenog sloja. U izlaznom sloju bismo onda odlučili ovisno o odgovorima da li je na slici ljudsko lice. Sljedeća razina bi bila dekompozicija trenutnih podproblema. Na primjer za pitanje da li se na slici nalazi oko se možemo pitati da li se vidi obrva, trepavice i zjenica. Za ovu razinu apstrakcije bi bio zadužen predzadnji skriveni sloj mreže. Daljnjom dekompozicijom problema bismo došli do podmreža koje daju odgovore na toliko jednostavna pitanja koja se mogu odgovoriti na temelju nekoliko piksela. Takva pitanja bi mogli odgovoriti neuroni u ulaznom sloju direktno povezani sa pikselima ulazne slike. Dekompozicija problema na manje i jednostavnije probleme se upravo odvija u slojevima neuronske mreže. Kretanjem od ulaznog sloja prema izlaznom razina apstrakcije se povećava te se grade kompleksniji i apstraktiniji koncepti odlučivanja. Ulazni sloj obrađuje podatke na razinama piksela dok izlazni sloj radi na najapstraktnijoj razini i daje nam rezultat klasifikacije. Sada je jasno da mreža sa više slojeva može donositi kompleksnije i sofisticiranije odluke od mreže sa manje slojeva.

2.3. Učenje neuronskih mreža

Pošto neuronske mreže imaju milijune parametara (težina) koje je potrebno odrediti kako bi mreža radila dobru klasifikaciju trebamo znati kako odrediti te parametre. Dva algoritma su ključna za rad neuronske mreže i za njezino učenje a to su: algoritam *feedforward* i algoritam sa širenjem pogreške unatrag(engl. *backpropagation*).

2.3.1. Algoritam feedforward

Algoritam feedforward omogućava rad neuronske mreže. Algoritam je vrlo jednostavan. Za svaki sloj računamo njegov izlaz krećući od ulaznog. Ulaz ulaznog sloja su podatci za klasifikaciju dok je njegov izlaz ulaz sljedećeg sloja. Jedino na što treba paziti je povezanost slojeva koja za svaki neuron određuje koji dio ulaza utječe na njegov izlaz. Primjer pseudokoda za poptpuno povezane slojeve je:

2.3.2. Algoritam backpropagation

Neuronsku mrežu možemo shvatiti kao funkciju više varijabli. Varijable su težine na prijelazima neurona a izlaz iz funkcije je pogreška mreže. U našem promatranju smatramo da je ulaz konstantan i da nije varijabla. Naš cilj je minimizirati pogrešku što se svodi na pretraživanje *n*-dimenzionalnog prostora gdje je *n* ukupan broj težina u mreži. Pogreška u takvom prostoru se može vizualizirati kao hiper-površina sa više lokalnih minimuma.

Ideja algoritma backpropagation je određivanje greške i gradijenata u svakom sloju te ažuriranje težina na temelju gradijenata tako smanjujući grešku neuronske mreže (gradijentni spust). Prvo se pomoću algoritma feedforward dobije odziv mreže za neki ulaz. Zatim se izračunaju greške izlaznog sloja (greške se računaju na svakom neuronu). Zatim se za prethodni sloj određuje utjecaj neurona na greške u idućem sloju te se izračuna greška prethodnog sloja. Zatim se izračuna gradijent greške po težinama koje povezuju te slojeve te se težine ažuriraju. Ovaj postupak se ponavlja za svaki ulaz i određen broj puta.

U svim oznakama koje slijede vrijedi konvencija označavanja trenutnog sloja sa *j* te prethodnog sloja sa *i*, izlaza neurona sa y te ukupan ulaz neurona sa z. Stoga y_i označava izlaz *i*-tog neurona prethodnog sloja a y_j izlaz *j*-tog neurona trenutnog sloja, z_j ulaz *j*-tog neurona trenutnog sloja, b_j prag j-tog neurona trenutnog sloja te w_{ij} težinu koja spaja i-ti neuron prethodnog sloja sa j-tim neuronom trenutnog sloja.

Da bismo odredili grešku izlaznog sloja moramo prvo odrediti funkciju pogreške. Najčešće se koristi srednja kvadratna pogreška:

$$E = \frac{1}{2}\sum_j (t_j - y_j)^2 \tag{2.8}$$

Parametar t_j predstavlja očekivani izlaz j-tog neurona. Grešku trenutnog sloja definiramo kao:

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j} \tag{2.9}$$

Parcijalnu derivaciju pogreške po izlazu neurona y_j za srednju kvadratnu pogrešku možemo raspisati kao:

$$\frac{\partial E}{\partial y_j} = \frac{1}{2}\frac{\partial}{\partial y_j}(t_j - y_j)^2 = -(t_j - y_j) \tag{2.10}$$

Druga parcijalna derivacija u izrazu 2.9 je jednaka derivaciji aktivacijske funkcije. Derivacija aktivacijske funkcije skaliranog hiperbolnog tangensa je već dana u izrazu 2.6.

Nakon računanja greške trenutnog sloja računa se greška prethodnog sloja koja je dana sa izrazom:

$$\frac{\partial E}{\partial z_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial z_i} \tag{2.11}$$

Druga parcijalna derivacije je ponovno jednaka derivaciji aktivacijske funkcije a parcijalna derivaciju pogreške po izlazu neurona prethodnog sloja dobijemo sumiranjem utjecaja neurona na sve neurone trenutnog sloja:

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_i} \tag{2.12}$$

Raspišimo z_j kao:

$$z_j = \sum_i w_{ij} y_i + b_j \tag{2.13}$$

Uvrštavanjem 2.13 u 2.12 dobivamo:

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_i} = \sum_j w_{ij}\frac{\partial E}{\partial z_j} \tag{2.14}$$

Na kraju se određuju parcijalne derivacije po težinama i pragovima:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} = \frac{\partial E}{\partial z_j} y_i \tag{2.15}$$

$$\frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial b_j} = \frac{\partial E}{\partial z_j} * 1 \tag{2.16}$$

Nakon čega se težine i pragovi ažuriraju u ovisnosti o stopi učenja η:

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} = w_{ij} - \eta * y_i \frac{\partial E}{\partial z_j} \tag{2.17}$$

$$b_j \leftarrow b_j - \eta \frac{\partial E}{\partial b_j} = b_j - \eta \frac{\partial E}{\partial z_j} \tag{2.18}$$

Stopa učenja η je mali pozitivni broj koji nam govori koliko brzo ćemo se kretati u smijeru negativnog gradijenta. Gradijent pokazuje u smijeru rasta funkcije pa je zato kod ažuriranja težina i pragova potrebno dodati negativan predznak jer pokušavamo minimizirati funkciju.

Pseudokod algoritma backpropagation dan je u nastavku:

Algorithm 1 Backpropagation

Ulaz: D(skup za učenje), η stopa učenja
Inicijaliziraj težine na male slučajno generirane vrijednosti
Dok nije ispunjen uvjet zaustavljanja:
Za svaki (x, t) iz D:
Izračunaj izlaz svakog sloja mreže za ulaz x
Izračunaj pogrešku izlaznog sloja prema formulama 2.9 i 2.10
Izračunaj pogrešku prethodnog sloja prema formulama 2.11 i 2.14
Izračunaj parcijalne derivacije pogreške po težinama i pragovima prema formulama 2.15 i 2.16
Ažuriraj težine i pragove prema formulama 2.17 i 2.18

Uvjet zaustavljanja algoritma je najčešće unaprijed zadan broj iteracija. Svaku iteraciju algoritma nazivamo epohom. Uvjet nemora nužno biti zadan brojem epoha, također je moguće da se kao uvjet postavi minimalna pogreška izlaza tj, da algoritam staje kad je pogreška dovoljno mala.

Prethodno opisani algoritam koristi stohastički gradijentni spust što znači da se težine ažuriraju nakon svakog ulaza. To znači da nije nužno da se uvijek krećemo u smijeru negativnog gradijenta na razini cijelog skupa za učenje. Ovakva varijanta gradijentnog spusta više oscilira te je upravo zbog tog svojstva otpornija na zapinjanje u lokalnim minimumima. Standardna varijanta gradijentnog spusta ažurira težine ili nakon nekog određenog broja ulaza (engl. batch) ili nakon svake epohe. U obzir se uzima prosjek gradijenata na svim obrađenim ulazima te je zato ova varijanta stabilnija i ima manje oscilacije ali zato ima veće šanse zapinjanja u lokalnim minimumima. Mi ćemo koristiti navedeni stohastički gradijentni spust pošto se u praksi pokazao veoma efikasnim a ujedno je računski puno manje zahtjevan od standardnog.

3. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže možemo smatrati proširenjima standardnih neuronskih mreža koje su se pokazale učinkovitijima prilikom klasifikacija slika. Neuroni u konvolucijskim neuronskim mrežama su dvodimenzionalni i nazivamo ih mapama značajki (engl. *feature maps*). Ulaz je također dvodimenzionalan a umjesto težina se koriste jezgre (engl. *kernels*).

3.1. Struktura mreže

Konvolucijske neuronske mreže su građene od tri različite vrste slojeva: konvolucijski slojevi, slojevi sažimanja i potpuno povezani slojevi. Na ulazu mreže se nalazi jedna monokromatska ili višekanalna slika u boji. Zatim slijede naizmjence konvolucijski slojevi i slojevi sažimanja. Mape značajki u tim slojevima u svakom sloju postaju sve manjih dimenzija krećući se od ulaznog sloja. Zadnji takav sloj je dimenzija 1 $\times$ 1. Na takav sloj se vežu potpuno povezani slojevi koji su jednodiemnzionalni te se ponašaju kao obične neuronske mreže opisane u prethodnom poglavlju. Primjer ovakve strukture vidimo na slici **??**.

3.1.1. Konvolucijski slojevi

Konvolucijski slojevi uzimaju mape na ulazu sloja te rade 2D kovoluciju sa jezgrama. Označimo sa M^j mape j-tog sloja, sa M dimenzije tih mapa te sa K^j jezgre koje povezuju mape prethnodnog sloja sa mapama trenutnog sloja i sa K dimenzije tih jezgri. Radi jednostavnosti ćemo koristiti kvadratne mape značajki i kvadratne jezgre te kad govorimo o dimenziji M misli se na $M \times M$ (ekvivalentno i sa dimenzijama jezgri). Također označimo sa S korak pomaka jezgre po širini i visini prilikom konvolucije. Veličine mapi značajki u nekom sloju dana je sa izrazom:

$$M = \frac{M - K}{S} + 1 \tag{3.1}$$

Konvolucija se tvori prolazom kroz ulaznu mapu sa prozorom jednake veličine kao i jezgra te se množe vrijednosti ulazne mape unutar prozora sa korespodentnim vrijednostima jezgre(možemo zamisliti koda preklopimo jezgru preko dijela ulazne mape i množimo vrijednosti koje su jedna na drugoj). Sumiramo te umnožke za sve ulazne mape značajki i dodamo prag te izračunamo izlaz aktivacijske funkcije koji zapisujemo u odgovarajući neuron izlazne mape značajki. Pod pojmom neuron u ovom kontekstu se misli na jednu jedinicu mape značajki. Dakle jedna mapa značajki dimenzije M ima $M \times M$ neurona. Nakon toga pomičemo okvir za S vodoravno, ili okomito ako smo došli do kraja reda te radimo proces isponova za idući neuron.

Vidimo da na jedan neuron izlazne mape značajki utječu samo dijelovi ulaznih mapi značajki koji su unutar okvira koji je potreban za taj neuron. To područje ulazne mape značajki "vidljivo" neuronu nazivamo vizualnim ili receptivnim poljem neurona. Ako se neuron u izlaznoj mapi nalazi na kordinatama (x, y) onda je njegovo vizualno polje definirano sa kvadratom dimenzija jednakih dimenzijama jezgre K, a kordinate gornjeg lijevog kuta vizualnog polja (x', y') u kordinatnom sustavu ulaznih mapi značajki su definirane kao:

$$x' = x * S \tag{3.2}$$

$$y' = y * S \tag{3.3}$$

Označimo sa M_k^j k-tu mapu j-tog sloja te sa w_{ik}^j jezgru koja povezuje k-tu mapu j-tog sloja sa i-tom mapom prethodnog sloja. Svaka mapa značajki ima po jedan prag b_k^j. Pošto su mape značajki i njihove jezgre dvodimenzionalne njihove elemente indeksiramo sa zagradama. Tako će vrijednost mape značajki na lokaciji (x, y) biti jednaka $M_k^j(x, y)$ a jezgre $w_{ik}^j(x, y)$. Uz ovaj sustav oznaka vrijednost mape k u sloju j na lokaciji (x, y) možemo prikazati sa sljedećim izrazom:

$$M_k^j(x, y) = f(\sum_i \sum_{x'=0}^{K} \sum_{y'=0}^{K} M_i^{j-1}(x' + x, y' + y) w_{ik}^j(x', y') + b_k^j) \tag{3.4}$$

Funkcija u jednadžbi je aktivacijska funkcija te je podrazumijevani pomak okvira S jednak 1. Naravno pričati ćemo samo o potpuno povezanim mrežama gdje je svaka mapa značajki trenutnog sloja povezana sa svim mapama značajki prethodnog sloja. Vrijednosti mapa značajki prilikom unaprijedne propagacije se računaju prema formuli 3.4.

3.1.2. Slojevi sažimanja

Slojevi sažimanja (engl. *pooling*) nemaju parametre koji se mogu učiti i služe za smanjenje dimenzija mapi značajki i uklanjanje varijance što znači da će se slični izlazi dobiti za male translacije ulaza. U ovim slojevima također imamo okvire sa kojima prolazimo po ulaznoj mapi značajki. Mapa se sažima na taj način da okvir predstavimo sa jednom vrijednošću. Na primjer okvir veličine 2×2 (najčešća veličina okvira koju ćemo i mi koristiti) se reprezentira sa jednom vrijednoću dobivenom iz 4 vrijednosti unutar okvira čime smanjujemo mapu 4 puta. Okvir se najčešće pomiče na način da se svaka vrijednost iz mape značajki koristi u samo jednom sažimanju. Pomak okvira bi za navedeni primjer bio jednak 2 u horizontalnom i vertikalnom smijeru.

Sažimanje usrednjavanjem

Sažimanje usrednjavanjem (engl. *mean pooling*) se ostvarije uzimanjem aritmetičke sredine vrijednosti unutar okvira sažimanja. Npr ako imamo mapu $M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$ sažimanjem usrednjavanjem sa okvirom veličine 2×2 i pomakom 2 po horizontali i vertikali ćemo dobiti 4 puta manju mapu značajki $M' = \begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix}$

Sažimanje maksimalnom vrijednošću

Sažimanje maksimalnom vrijednošću (engl. *max pooling*) se ostvaruje uzimanjem maksimalne vrijednosti unutar okvira sažimanja. Za istu mapu značajki M iz prethodnog primjera i za iste dimenzije sažimanja bismo dobili mapu značajki $M' = \begin{bmatrix} 6 & 8 \\ 14 & 16 \end{bmatrix}$

3.1.3. Backpropagation u konvolucijskim mrežama

Potrebno je definirati izmijenjeni algoritam backpropagation za primjenu u konvolucijskim slojevima i slojevima sažimanja. Pošto su zadnji slojevi mreže potpuno povezani slojevi kao u običnim dubokim neuronskim mrežama unazadna propagacija pogreške i ažuriranje težina se u tim slojevima obavlja na već opisani način iz prethodnog poglavlja. Također ćemo radi lakšeg razumijevanja i unazadne propagacije iz konvolucijskog sloja izdvojiti primjenu aktivacijskih funkcija u posebni sloj. Nazovimo ga aktivacijski

sloj. To znači da se u konvolucijskom sloju rade samo sva potrebna sumiranja a u aktivacijskom sloju se na svaku vrijednost prethodne mape značajki (konvolucijskog sloja) primjenjuje aktivacijska funkcija. Također držati ćemo se sljedeće konvencije imenovanja: $\delta_k^j(x,y)$ je pogreška k-te mape značajki u sloju j na lokaciji (x,y), $w_{ik}^j(x,y)$ je vrijednost jezgre j-tog sloja između k-te mape značajki j-tog sloja i i-te mape značajki prethodnog sloja na lokaciji (x,y), b_k^j je prag k-te mape značajki j-tog sloja te $M_k^j(x,y)$ je izlaz neurona k-te mape j-tog sloja koji se nalazi na lokaciji (x,y). Za svaki sloj posebno ćemo objasniti računanje greške prethodnog sloja pod uvjetom da imamo izračunatu grešku trenutnog sloja.

Konvolucijski slojevi

Označimo sa z_k^j:

$$z_k^j(x,y) = \sum_i \sum_{x'=0}^{K} \sum_{y'=0}^{K} M_i^{j-1}(x'+x, y'+y) w_{ik}^j(x',y') + b_k^j \tag{3.5}$$

$$M_k^j(x,y) = z_i^j \tag{3.6}$$

Poznato nam je $\dfrac{\partial E}{\partial M_k^j(x,y)}$ za svaki neuron mape značajki trenutnog sloja (tu informaciju smo dobili od idućeg sloja). Prvo moramo dobiti grešku trenutnog sloja. Greška pojedinog neurona trenutnog sloja je jednaka:

$$\frac{\partial E}{z_k^j(x,y)} = \frac{\partial E}{\partial M_k^j(x,y)} \frac{\partial M_k^j(x,y)}{\partial z_k^j(x,y)} \tag{3.7}$$

Uvrštavanjem 3.6 dobivamo:

$$\frac{\partial E}{z_k^j(x,y)} = \frac{\partial E}{\partial M_k^j(x,y)} \frac{\partial M_k^j(x,y)}{\partial z_k^j(x,y)} = \frac{\partial E}{\partial M_k^j(x,y)} \tag{3.8}$$

Potrebno je izračunati $\dfrac{\partial E}{\partial M_k^{j-1}(x,y)}$ tako da sumiramo utjecaj neurona na sve neurone trenutnog sloja. Sumiranje obavljamo po mapama značajki trenutnog sloja i po lokacijama u tim mapama na koje utječe izlaz neurona $M_k^{j-1}(x,y)$:

$$\frac{\partial E}{\partial M_k^{j-1}(x,y)} = \sum_i \sum_{x'=max(x-K+1,0)}^{x} \sum_{y'=max(y-K+1,0)}^{y} \frac{\partial E}{z_i^j(x',y')} \frac{z_i^j(x',y')}{\partial M_k^{j-1}(x,y)} \tag{3.9}$$

3.1.4. Hiperparametri mreže

4. Zaključak

Zaključak.

LITERATURA

Raspoznavanje objekata konvolucijskim neuronskim mrežama

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.