# STRUKTURA JEDNOSTAVNE KONVOLUCIJSKE NEURONSKE MREŽE

Konvolucijska neuronska mreža je specijalizirana vrsta neuronskih mreža koja je dizajnirana za rad s 2D podacima (slike i fotografije), iako ih se može koristiti i kod 1D ili 3D podataka. Središte konvolucijskih neuronskih mreža čine konvolucijski slojevi.[2]

## ULAZNI SLOJ

Općenito, ulazni sloj prima neovisne ulazne primjere. Ulazi su podaci koji se prosljeđuju neuronskoj mreži i na temelju kojih ona donosi određenu odluku ili predviđanje. Svaka ulazna vrijednost (koja sačinjava ulazni primjer) prosljeđuje se jednom neuronu u ulaznom sloju neuronske mreže. Za razliku od ostalih slojeva u bilo kojoj neuronskoj mreži, ulazni sloj ne izvodi nikakve operacije nad ulaznim primjerima već on služi samo zato da primi ulazne podatke takvima kakvi oni jesu i da ih proslijedi mreži. [13]

Broj neurona u ulaznom sloju određen je samom strukturom podataka koji se prosljeđuju neuronskoj mreži. On odgovara broju piksela, ili broju jedinstvenih značajki, ulaznih podataka.

Jedna od glavnih razlika između tradicionalne neuronske mreže i konvolucijske neuronske mreže je ta da su neuroni u konvolucijskoj mreži organizirani u tri dimenzije (visina, širina i dubina). U konvolucijskom neuronskim mrežama, neuroni ulaznog sloja sadrže vrijednosti piksela ulazne slike.[4]

Zato je potrebno, prije nego se bilo koja slika proslijedi konvolucijskoj neuronskoj mreži, ulazne podatke prvo procesirati kako bi se osiguralo da će mreža uvijek na svoj ulaz primati uniformirane podatke.

Kao što je već spomenuto, ulazni sloj konvolucijske neuronske mreže najčešće prima slikovne podatke. Slikovni podaci su, kao što je maloprije spomenuto, predstavljeni korištenjem 3D matrice. Tu matricu je, prije nego se proslijedi mreži, potrebno transformirati u vektor, odnosno u samo jedan stupac. Neka ulazna slika ima dimenzije 28x28=784 i neka je crno-bijela (dubina joj je 1). Tu sliku je potrebno transformirati u vektor 784x1. To znači da će ulazni sloj u konvolucijsku neuronsku mrežu imati onoliko piksela koliko ih ima ulazna slika u sve svoje tri dimenzije.[14][15]

## KONVOLUCIJSKI SLOJ

Računalo čita slike kao matrice s vrijednostima piksela koje imaju dimenzije NxNx3 (visina, širina, dubina). Slike se najčešće sastoje od 3 kanala (RGB). U kontekstu konvolucijskih neuronskih mreža, konvolucija je linearna operacija koja uključuje množenje skupa težina (filtera) sa skupom odgovarajućih ulaza (matrica s vrijednostima piksela).[5][10]

Ulaz u konvolucijski sloj se još naziva ulaznim kanalom (eng. *input channel*). Izlaz iz konvolucijskog sloja se naziva izlaznim kanalom (eng. *output channel*), matricom značajki (eng. *feature map*) ili matricom aktivacija (eng. *activation map*).[1][2]

### FILTERI

Svaki konvolucijski sloj u konvolucijskoj neuronskoj mreži posjeduje definirani broj filtera. Ti filteri se sustavno primjenjuju na ulazni kanal i tako konvolucijski sloj na svom izlazu kreira matricu značajki.[1][2][5]

Primjena više filtera na ulazni kanal rezultira jednim izlaznim kanalom (matricom značajki). Ulazni kanal je obično 3D slika (broj redaka, stupaca i kanala). Tako i filteri moraju imati 3 dimenzije s jednakim brojem kanala, ali manjim brojem stupaca i redaka u odnosu na ulaznu sliku. Filter sadrži težine (u tradicionalnim neuronskim mrežama bi to bile uobičajene težine veza) čije optimalne vrijednosti mreža treba naučiti tijekom procesa učenja. Težine filtera definiraju strukturu ili obrazac (eng. *feature*) koji će filter detektirati na ulaznom kanalu.[5]

Slika se matematički može izraziti kao matrica sljedećih dimenzija:

Gdje je:

* : visina slike
* : širina slike
* : broj kanala

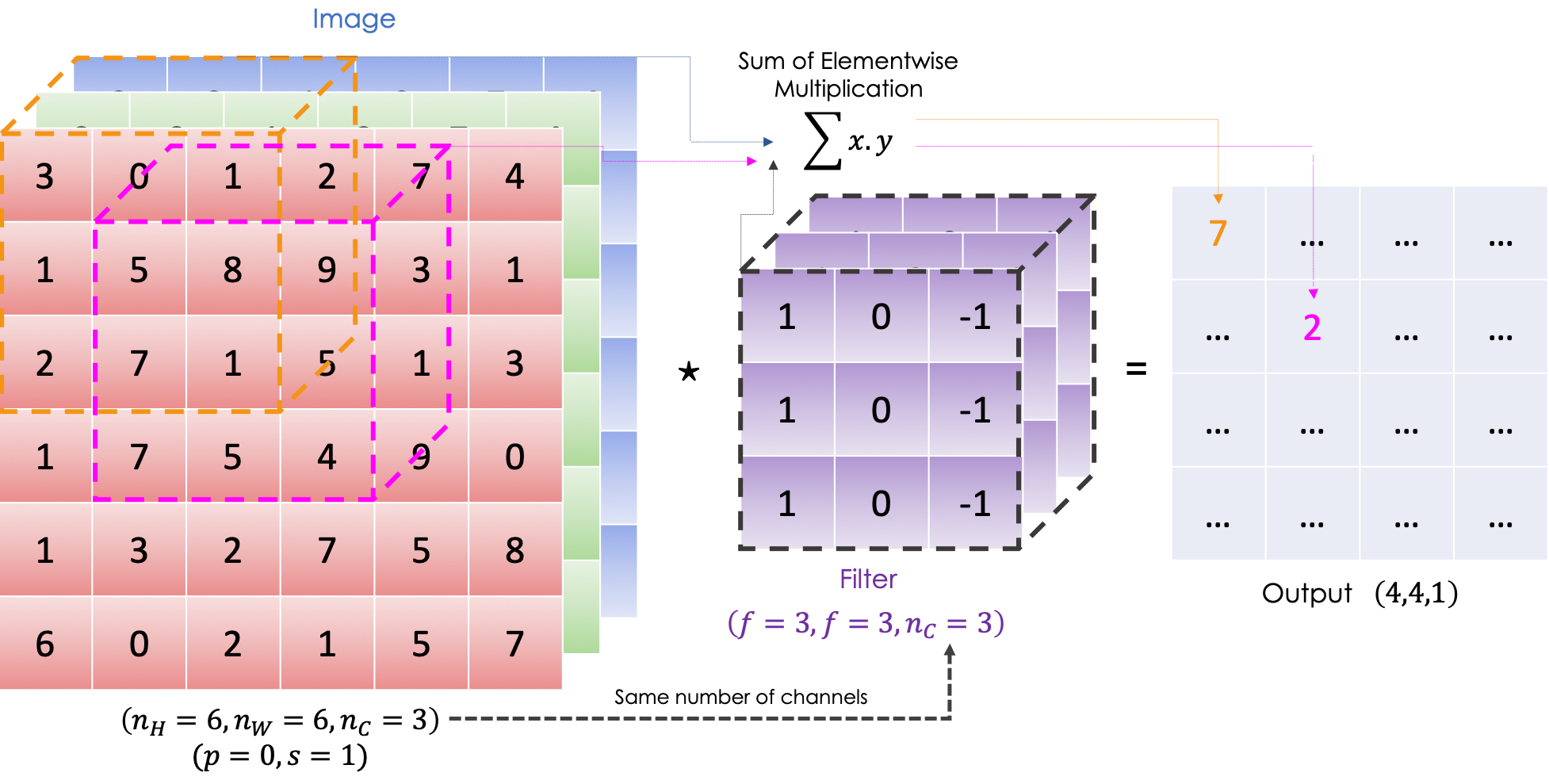
Za RGB sliku, broj kanala je jednak 3 s obzirom da RGB slika posjeduje crveni, zeleni i plavi kanal.[7]

Filter, koji se označava s *K*, najčešće ima jednake dimenzije visine i širine (3x3, 5x5 ili 7x7) i koje se označavaju s *f*. Takav filter, s neparnim brojem za dimenzije, omogućava da se svaki piksel na slici u jednom trenutku nalazi u sredini filtera. Filter mora imati jednak broj kanala kao i ulazna slika.[7]

Dimenzije filtera se tako označavaju na sljedeći način:

Konvolucijski produkt slike i filtera je 2D matrica značajki. Toj matrici je svaki element rezultat sume množenja po odgovarajućim elementima (eng. *elementwise multiplication*). Filteri sadrže težine koje su inicijalizirane nasumičnim brojčanim vrijednostima. Filter u jednom trenutku računa točkasti produkt (eng. *dot product*) na kocki (jer posjeduje visinu, širinu i dubinu) ulazne matrice. Kocka ima jednake dimenzije kao i filter. Ta 'kocka' koju filter promatra u danom trenutku se naziva '*patch'* ili 'receptivno polje' (eng. *receptive field*).[7][8]

Primjer jednog konvolucijskog produkta prikazan je na slici:



Konvolucijska operacija se matematički može opisati kao:

Gdje je:

* *I*: ulazna slika
* *K*: filter
* *x*: broj retka u matrici značajki
* *y*: broj stupca u matrici značajki
* *i*: redni broj retka ulazne matrice
* *j*: redni broj stupca ulazne matrice
* *k*: redni broj kanala ulazne matrice

Dimenzije matrice značajki se dobiju na sljedeći način:

Gdje je pod (eng. *floor*) funkcija od *x*.

Postoje neke specijalne vrste konvolucije:

1. *Valid* konvolucija:
2. *Same* konvolucija: veličina izlaza je jednaka veličini ulaza:
3. 1x1 konvolucija: ; ovo je korisno u nekim slučajevima kako bi se smanjio broj kanala bez da se smanje dimenzije ().

Konvolucijski sloj izvodi konvolucijske operacije na svojim ulazima tako što istovremeno primjenjuje više filtera. Izlazi iz konvolucijskog sloja se prosljeđuju aktivacijskoj funkciji koja se u ovom primjeru označava s .

Ovdje je potrebno napomenuti da literatura nije usuglašena pripadaju li aktivacijske funkcije konvolucijskom sloju ili čine zasebni, aktivacijski sloj. U ovom poglavlju će se opisati na koji se način aktivacijske funkcije primjenjuju na rezultate konvolucije dok će aktivacijske funkcije biti obrađene u zasebnom poglavlju.

U *l*-tom sloju neuronske mreže vrijede sljedeće anotacije:

* Ulaz: dimenzija gdje je slika koja ulazi u neuronsku mrežu.
* padding: i pomak:
* broj filtera: (broj kanala u tom sloju) gdje svaki (oznaka za filter) ima dimenzije
* prag n-te konvolucije:
* aktivacijska funkcija:
* izlaz: s dimenzijama

Sada vrijedi:

:

I nadalje:

S tim da:

Parametri u l-tom sloju koji se mogu naučiti su:

* filteri s parametara
* prag s parametara

### ZERO PADDING

Svaki konvolucijski sloj posjeduje definirani broj filtera određenih dimenzija. Sustavnom primjenom filtera, dimenzije matrice značajki su reducirane u odnosu na dimenzije ulaznog kanala.[2]

Kao što je matematički opisano u prethodnom poglavlju, filter se počinje primjenjivati na ulazni kanal tako što se gornji lijevi kut filtera poklapa s gornjim lijevim kutom ulaznog kanala. Nakon što izračuna točkasti produkt, filter se pomiče za jedno mjesto (ili u slučaju slike, piksel) u desno (jer se uzima u obzir da vrijedni osnovna situacija kada je pomak postavljen na 1) te ponovno računa točkasti produkt. Cijeli proces se ponavlja sve dok se gornji desni kut filtera ne poklopi s gornji desnim kutom slike. Znači, filter se primjenjuje samo unutar slike, ne izvan nje i zbog toga dolazi do redukcije dimenzija.

Redukcija dimenzija matrice značajki u odnosu na ulazni kanal se naziva 'efekt ruba' (eng. *border effect*). Taj efekt ne predstavlja veliki problem kod velikih slika i malih filtera, iako se i dalje gube potencijalno važne informacije koje se nalaze na rubovima slike. Problem je puno veći kada se filter primjenjuje na slike malih dimenzija, kada je više konvolucijskih slojeva povezano u seriji ili kod dubokih mreža s desecima ili stotinama skrivenih slojeva. U jednom trenutku mreža može ostati bez podataka u matrici značajki nad kojom bi mogla izvoditi operacije ili predviđanja.[2][7][11]

Pikseli, koji se nalaze na rubovima ulaznog kanala (ili slike), bit će izloženi samo rubovima filtera i rjeđe se koriste od piksela koji se nalaze po sredini slike. Kada bi se filter počeo primjenjivati tako da se njegov rub nalazi izvan okvira slike, pikseli, koji se nalaze na rubovima ulaznog kanala, imaju veću priliku utjecati na rezultat konvolucije. Drugim riječima, dobiva se više prilika za detekciju određenih struktura. Još jedna posljedica je da matrica značajki ima iste dimenzije kao i ulazni kanal.[2][7]

Kako bi se riješio taj problem 'odbacivanja' piksela na rubovima ulaznog kanala, oko rubova ulaznog kanala se dodaje neka vrsta popune (eng. *padding*). Dodaju se dodatni pikseli oko cijelog okvira ulaznog kanala. Prema konvenciji se taj *padding* puni nulama. Zato se naziva *zero padding*. *Padding* se označava s *p* i označava broj elemenata koji su dodani na sve četiri strane ulaznog kanala.[7][11]

S obzirom na to koristi li se *padding* ili ne, postoje dvije vrste konvolucije: *same* konvolucija i *valid* konvolucija. *Valid* konvolucija znači da se konvoluira originalni ulazni kanal bez dodanog *padding*-a. *Same* konvolucija znači da se oko ruba slike stavlja dodatni *padding* s nulama tako da matrica značajki ima iste dimenzije kao i ulazni kanal. U drugom slučaju, 'debljina' *padding*-a se računa na sljedeći način:

Gdje su:

* *p*: 'debljina' obruba
* *f*: visina ili širina filtera.[1][11]

### POMAK

Kao što je već opisano, filter se po ulaznom kanalu pomiče s lijeva na desno, od gore prema dolje po jedan piksel horizontalno pa vertikalno. Broj piksela koliko se filter pomiče može varirati, ali je uvijek simetričan u horizontalnom i vertikalnom smjeru. Taj broj se naziva 'pomak' (eng. *stride*).[2][8]

Osnovni pomak je (1,1) za pomicanje filtera po širini i visini ulaznog kanala i u većini slučajeva ima zadovoljavajuće performanse. Ako se ne koristi pomak (1,1), onda se najčešće koristi pomak (2,2). Na taj način se filter pomiče dva stupca u desno i dva reda dolje. Posljedica većeg pomaka je da će matrica značajki imati puno manje dimenzije od ulaznog kanala. Na primjer, ako se 3x3 filter primjenjuje na 6x6 sliku, matrica značajki će imati dimenzije 3x3.[2]

## AKTIVACIJSKE FUNKCIJE U NEURALNOJ MREŽI

Aktivnost u biološkoj neuronskoj mreži, gdje su različiti neuroni aktivirani različitim podražajima, bila je inspiracija za korištenje aktivacijskih funkcija u umjetnim neuronskim mrežama. U biološkoj neuronskoj mreži, neki neuroni su u danom trenutku ili aktivirani ili ne. Primjenom aktivacijske funkcije u umjetnoj neuronskoj mreži, aktivnost neurona može biti prikazana bilo kojim brojem između nula (ili minus jedan) i jedan. Što je vrijednost neurona bliža nuli, to je neuron manje aktiviran. S druge strane, što je vrijednost neurona bliža jedinici, to je neuron više aktiviran.[1][2][4][6]

Aktivacijska funkcija u umjetnoj neuronskoj mreži jest funkcija koja preslikava ulaznu vrijednost, ponderiranu sumu svih ulaznih veza određenog čvora, u odgovarajuću izlaznu vrijednost.

Aktivacijska funkcija na neki unaprijed određeni način transformira ulaznu vrijednost u broj koji se nalazi unutar unaprijed određene donje i gornje granice. Obično, umjetne neuronske mreže koriste ne-linearne aktivacijske funkcije jer su vrste preslikavanja kod dubokih neuronskih mreža složenije od običnih linearnih preslikavanja i zbog toga što ne-linearne aktivacijske funkcije omogućuju računanje proizvoljno složenih funkcija. Još jedna prednost aktivacijskih funkcija jest ta što aktivacijske funkcije dopuštaju povratnu propagaciju koja je objašnjena u drugom poglavlju.[3][4][6]

Obično su aktivacijske funkcije, koje se koriste u skrivenim slojevima, iste za sve skrivene slojeve u određenoj neuronskoj mreži. Nije uobičajeno vidjeti ReLU aktivacijsku funkciju u jednom skrivenom sloju i sigmoidnu aktivacijsku funkciju u drugom skrivenom sloju. Koristi se ili samo ReLU aktivacijska funkcija ili samo sigmoidna aktivacijska funkcija u svim skrivenim slojevima određene mreže.

Kako bi se razumjelo zašto se koriste ne-linearne aktivacijske funkcije, i kako bi se poslije dokazalo da je jedna od aktivacijskih funkcija (ReLU) ne-linearna, prvo je potrebno razumjeti linearne funkcije.

Neka je *f* funkcija na skupu *X*.

Neka su *a* i  *b* elementi skupa *X*.

Neka je *x* neki realan broj.

Funkcija *f* je linearna funkcija ako i samo ako vrijedi:

i ako vrijedi:

.

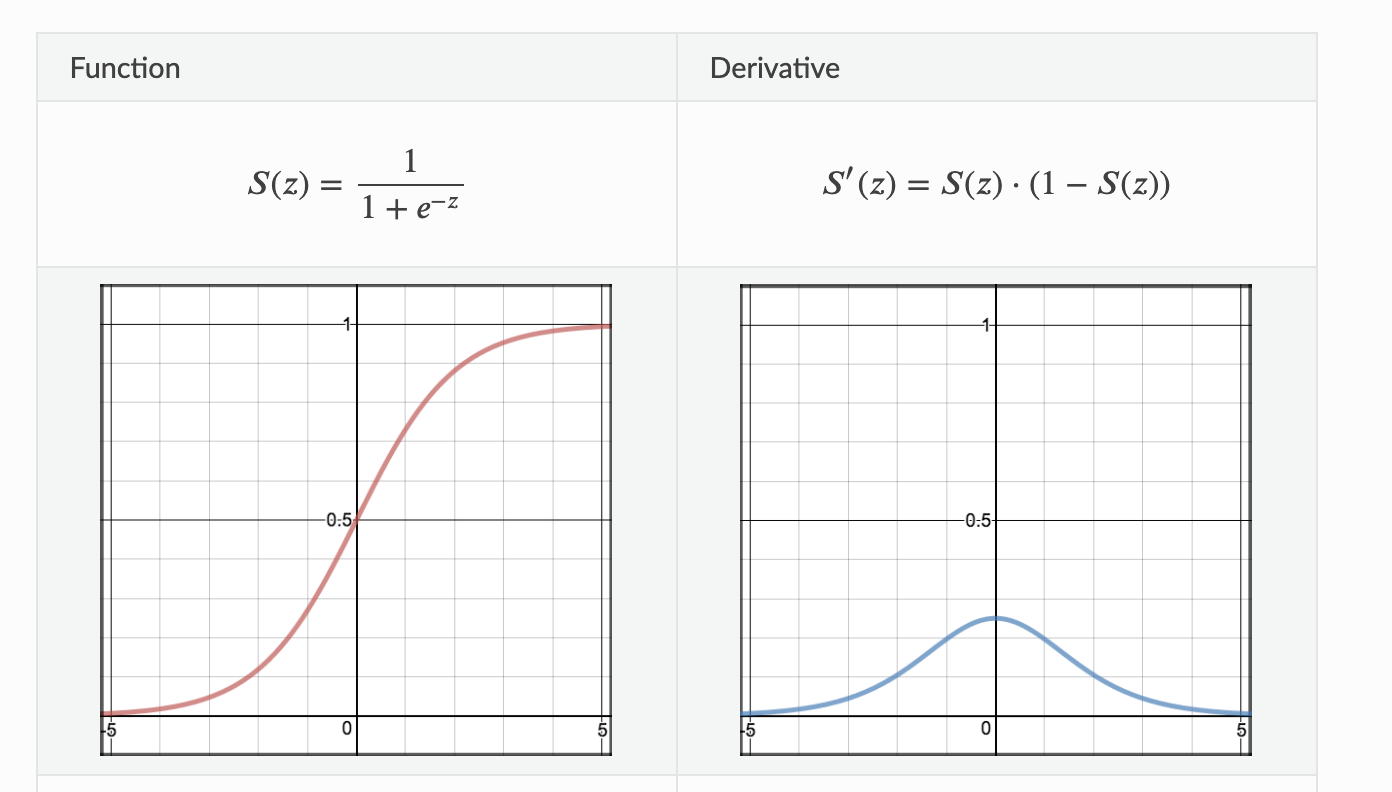
Važna značajka linearnih funkcija jest da kompozicija dviju linearnih funkcija čini novu linearnu funkciju. To znači da će, čak i u dubokim umjetnim neuronskim mrežama, ako se primjenjuju samo linearne funkcije nad ulaznim vrijednostima kod prosljeđivanja prema naprijed (eng. *forward propagation*), preslikavanje ulaza na izlaz također biti linearno. Nadalje, kada bi svaki sloj u neuronskoj mreži koristio samo pragove (eng. *bias*) i težine (eng. *weights*) bez aktivacijskih funkcija, cijela mreža bi bila jednaka linearnoj kombinaciji težina i pristranosti. Drugim riječima, formula neuronske mreže bi se mogla faktorizirati i svesti na jednostavan linearni regresijski model. Takav model bi mogao rješavati jednostavne linearne ovisnosti, ali ne bi mogao obavljati konkretne zadatke neuronskih mreža kao što su obrada slika i zvukova.[1]

### SIGMOIDNA AKTIVACIJSKA FUNKCIJA

Sve sigmoidne funkcije imaju sposobnost preslikavanja niza brojeva u brojeve koji se nalaze u malom opsegu između 0 i 1 ili -1 i 1. Tako se sigmoidna funkcija može koristiti za pretvorbu realnog broja u vrijednost koja se potom tumači kao vjerojatnost. Zbog toga se sigmoidna funkcija, osim u skrivenim slojevima, može koristiti i u izlaznom sloju gdje pretvara izlazni rezultat cjelokupne mreže u prikaze vjerojatnosti s kojima se onda može lakše raditi i koje se mogu lakše protumačiti. Zbog svog 'S' oblika na grafu nosi naziv sigmoidna funkcija. Taj naziv se često odnosi na naziv 'logistička sigmoidna funkcija'.[5][7]

Matematička formula sigmoidne funkcije glasi:

Grafički prikaz sigmoidne funkcije i njene derivacije prikazan je na slici.



Slika : sigmoidna aktivacijska funkcija i njena derivacija

Sigmoidna funkcija je ne-linearna, kontinuirano diferencirana, monotona funkcija. Glavna prednost joj je to što je jednostavna i dovoljno dobra kao funkcija za klasifikaciju. S druge strane, veliki nedostatak joj je što uzrokuje problem 'nestajućeg gradijenta' (eng. *vanishing gradient problem*) jer joj vrijednost nije centrirana oko nule. Zbog toga ažurirane vrijednosti mogu ići predaleko u različitim smjerovima. Nadalje, korištenjem sigmoidne funkcije, teže je provesti optimizaciju te računanje u skrivenim slojevima oduzima puno vremena.[5][7]

#### KORIŠTENJE SIGMOIDNE AKTIVACIJSKE FUNKCIJE

Glavno područje strojnog učenja, gdje je bitno korištenje sigmoidne funkcije, je područje korištenja logističkog regresijskog modela. Model logističke regresije se koristi kako bi se procijenila vjerojatnost na binarnom nivou, kao što je, na primjer, 'živo' ili 'neživo', 'lažno' ili 'istinito' i slično. Model vraća vrijednost koja se nalazi između 0 i 1. [7]

Razlog zašto se baš (logistička) sigmoidna funkcija koristi u logističkog regresiji je zbog činjenice da funkcija uvijek vraća vrijednost između 0 i 1, a logistička regresija je izvedena iz pretpostavke da su podaci iz obje klase normalno distribuirani.[7]

#### USPOREDBA SIGMOIDNE I ReLU FUNKCIJE

U modernim umjetnim neuronskim mrežama običaj je vidjeti, umjesto sigmoidne funkcije, ReLU aktivacijsku funkciju.

ReLU aktivacijska funkcija donosi nekoliko glavnih prednosti u odnosu na sigmoidnu funkciju. Glavna prednost jest da se ReLU funkcija puno brže izračunava. Nadalje, aktivacijski potencijal se u biološkim neuronskim mrežama ne mijenja za negativne ulaze. Zbog toga se čini da ReLU funkcija bolje oponaša biološke neuronske funkcije. [7]

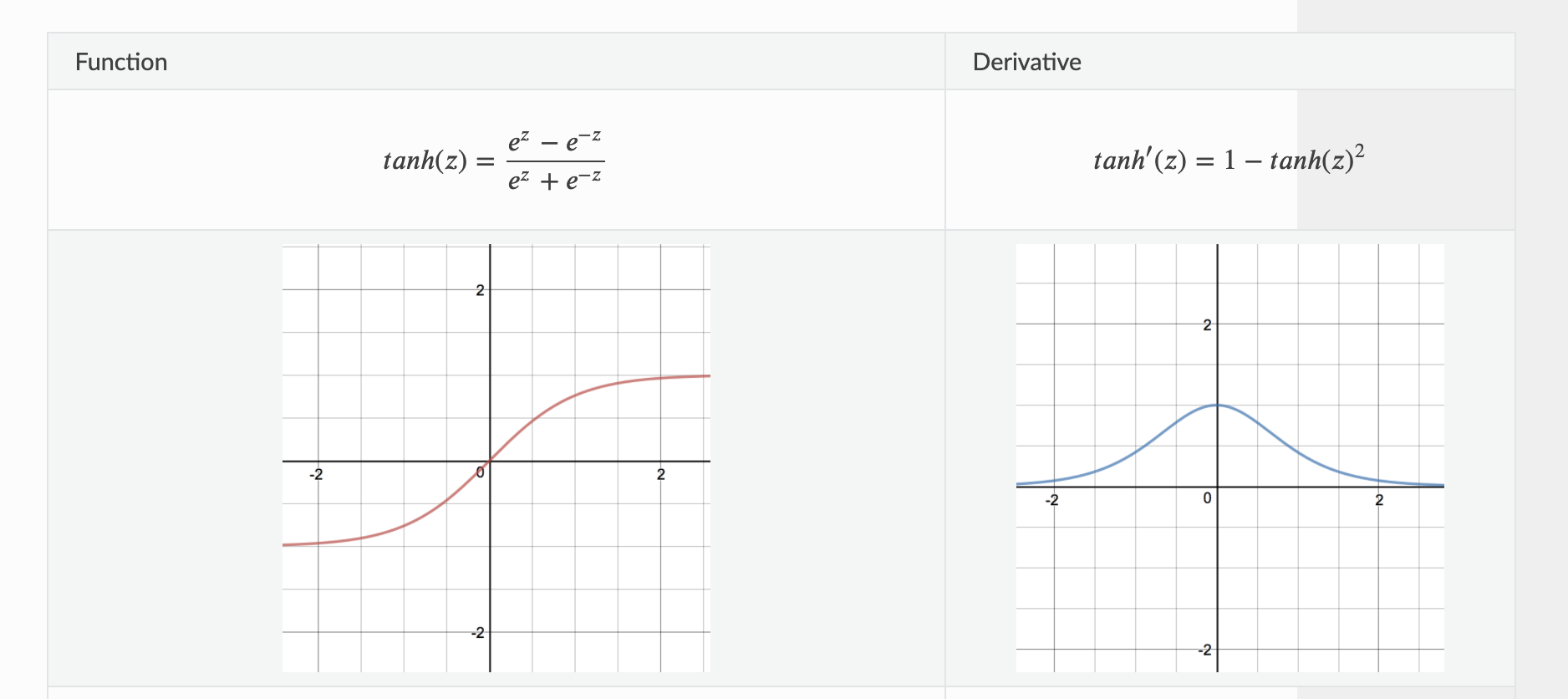
Još jedna u nizu prednosti se odnosi na pozitivne vrijednosti . ReLU funkcija posjeduje konstantni gradijent 1, dok sigmoidna funkcija posjeduje gradijent koji vrlo brzo konvergira prema 0. Ovo svojstvo neuronsku mrežu, koja sadrži sigmoidnu aktivacijsku funkciju, čini sporom za učenje. [7]

### TANH AKTIVACIJSKA FUNKCIJA

Tanh je poboljšana sigmoidna funkcija. Opseg tanh funkcije je od -1 do 1 te, kao i sigmoidna funkcija, ima 'S' oblik.[8][5]

Matematička formula tanh funkcije glasi:

Grafički prikaz tanh funkcije i njene prve derivacije prikazan je na donjoj slici:



Slika : grafički prikaz tnh funkcije i njene prve derivacije

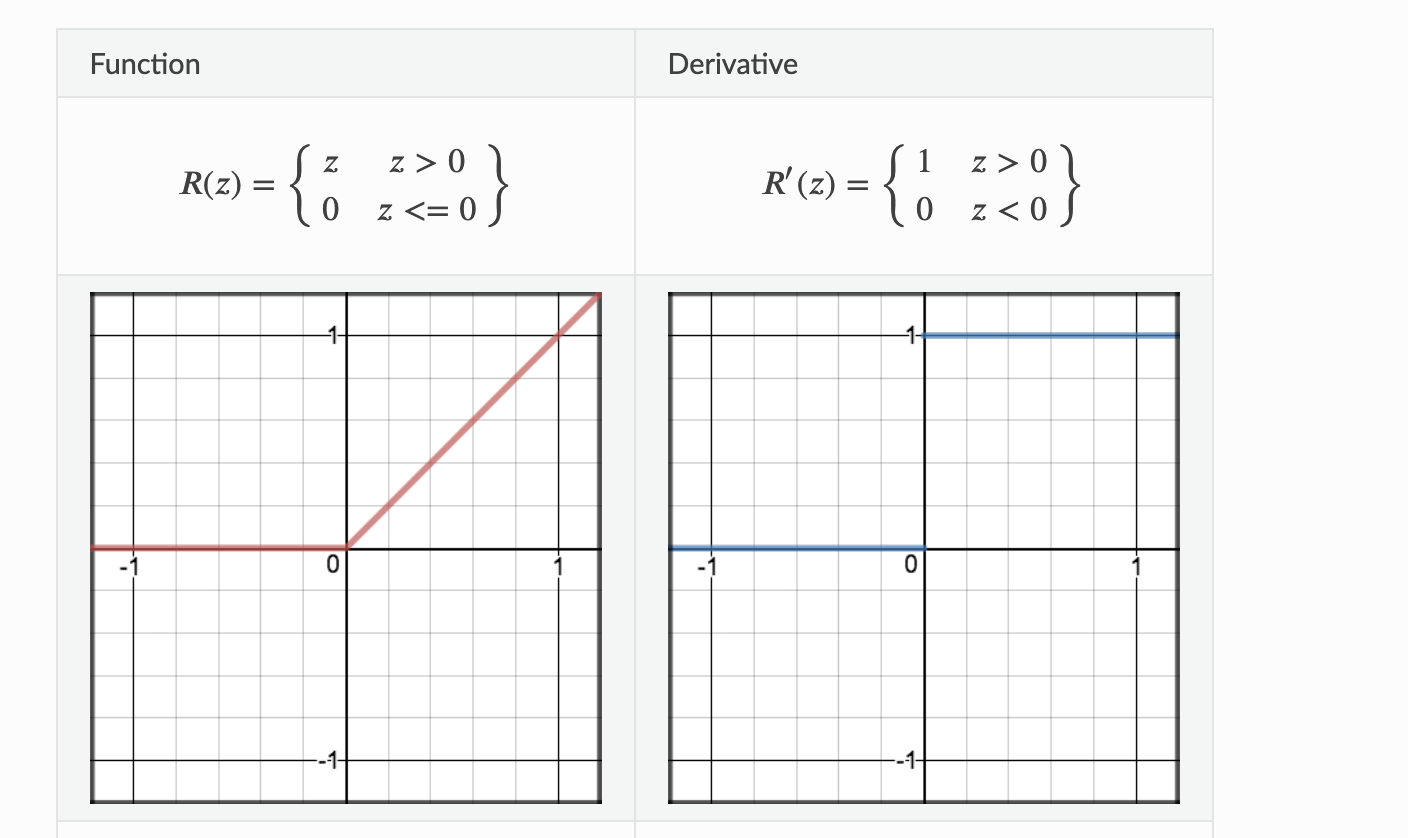
Prednost tanh nad sigmoidnom funkcijom je što će negativne vrijednosti biti preslikane u izrazito negativne vrijednosti, a vrijednosti, koje imaju približnu vrijednost 0, će biti preslikane u vrijednosti koje su blizu 0 i tako rješava problem sigmoidne funkcije. Nadalje, pokazala se da daje bolje rezultate kod više-slojnih neuronskih mreža. Međutim, kao i sigmoidna funkcija, uzrokuje problem nestajućeg gradijenta, zbog čega se danas preferira korištenje ReLU funkcije.[8][9]

Ova funkcija se najčešće koristi za klasifikaciju podataka u dvije kategorije.

### ReLU AKTIVACIJSKA FUNKCIJA

ReLU aktivacijska funkcija je kratica od *Rectified Linear Unit* i koja se izražava matematičkom formulom: .

To je jedna od najboljih i najčešće korištenih aktivacijskih funkcija koje se danas koriste (pogotovo u konvolucijskim neuronskim mrežama) u odnosu na sigmoidnu i tanh funkciju jer se uspješno suočava s problemom nestajućeg gradijenta (eng. *vanishing gradient problem*). Grafički prikaz ReLU aktivacijske funkcije i njene derivacije nalazi se na donjoj slici.[10]



Slika : ReLU aktivacijska funkcija i njena derivacija

Kao što se može vidjeti na slici, ReLU aktivacijska funkcija se ponaša kao linearna funkcija za sve pozitivne vrijednosti i vraća nulu za sve negativne vrijednosti. To znači nekoliko stvari:

1. ReLU funkcija se jednostavno računa. Zato će modelu trebati manje vremena za učenje i kasnije za donošenje odluka.
2. Brže konvergira. Linearnost ReLU funkcije znači da njen nagib nema 'visoravan' kada *x* poprimi veliku vrijednost.
3. Slabo je aktivirana. S obzirom da ReLU poprima vrijednost nula za sve negativne ulazne vrijednosti, postoji vjerojatnost da se neki neuroni u mreži nikada neće aktivirati, što je u većini slučajeva dobro.[10]

U biološkim neuronskim mrežama, gdje postoji na milijarde neurona, nisu svi neuroni aktivni u isto vrijeme jer imaju različite uloge i aktiviraju ih različiti podražaji. Slaba aktiviranost u modelima rezultira bržim i boljim predviđanjima i manjoj prenaučenosti.[10]

#### DOKAZ DA JE ReLU NE-LINEARNA FUNKCIJA

Kako bi se dokazalo da je ReLU aktivacijska funkcija ne-linearna funkcija, dokazat će se da ReLU ne uspijeva biti linearna funkcija. Kao što je već spomenuto, aktivacijske funkcije su ne-linearne funkcije.

Za svaki realan broj *x*, funkcija *f* je definirana tako da je:

Neka se pretpostavi da je *a* realan broj za koji vrijedi .

Koristeći pretpostavku da vrijedi , može se vidjeti da je:

i da je:

Radi ovoga dolazimo do zaključka:

Tako je dokazano da funkcija *f*, odnosno ReLU funkcija, ne uspijeva biti linearna funkcija.[1]

#### UMIRUĆI ReLU

Kao što je već spomenuto, slaba aktiviranost ReLU funkcije je u većini slučajeva poželjna jer to rezultira bržim i boljim rezultatima te manjoj prenaučenosti. Međutim, negativna strana toga što ReLU poprima vrijednost nula za sve negativne vrijednosti je ta što se može pojaviti problem koji se naziva 'umirući ReLU' (eng. *dying ReLU*).[10][5]

ReLU neuron je 'mrtav' ako je ostao zaglavljen na negativnoj strani vrijednosti zbog čega uvijek ima vrijednost nula. Kako je nagib ReLU funkcije za negativne vrijednosti nula, jednom kada neuron poprimi negativnu vrijednost, malo je vjerojatno da će se oporaviti od te vrijednosti. Takvi neuroni ne pridonose u preslikavanju ulaza na izlaz te postaju praktički beskorisni. Tijekom vremena se može dogoditi da veliki dio mreže ne radi ništa (u nekim slučajevima čak oko 40%).[10][5][11]

Zbog toga se, osim ReLU funkcije, znaju koristiti njene izvedenice koje nastoje zaobići taj problem. Neke od njih će biti opisane u zasebnim poglavljima. Neke od izvedenica su:

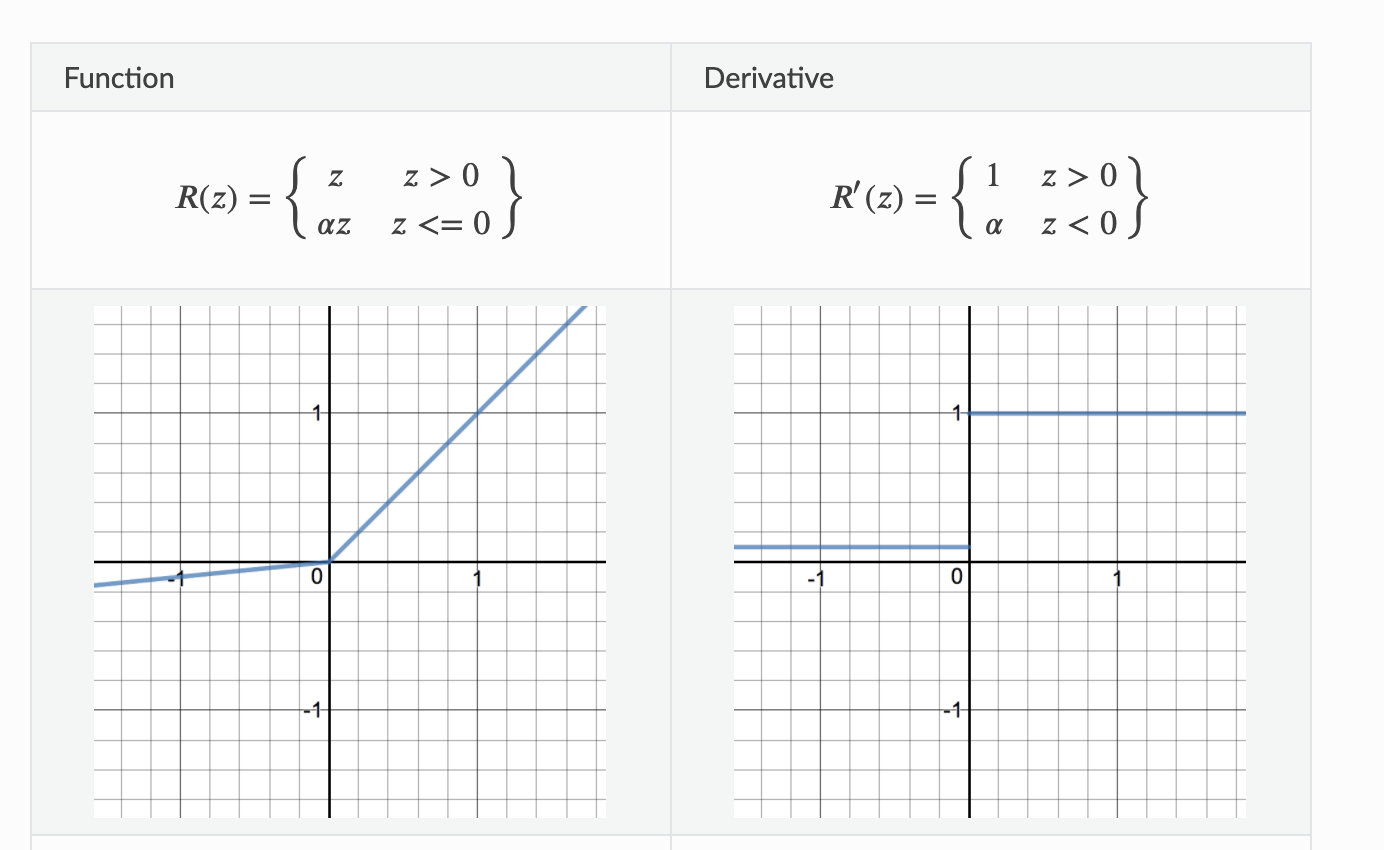
1. Leaky ReLU
2. Parametric ReLU (PReLU)
3. ReLU-6
4. Concatenated ReLU (CReLU)[10]

### LEAKY ReLU AKTIVACIJSKA FUNKCIJA

*Leaky* ReLU aktivacijska funkcija je izvedenica ReLU funkcije. Izvedena je s ciljem da riješi problem 'umirućeg ReLU-a'. Ona, za razliku od ReLU funkcije, ima blagi pozitivni nagib u negativnom dijelu vrijednosti i tako omogućuje izvedbu povratne propagacije i za negativne vrijednosti. To znači da je opseg *leaky* ReLu funkcije .[10][6][5]

Matematička formula *Leaky* ReLU funkcije glasi:

Grafički prikaz *leaky* ReLU funkcije kao i njene prve derivacije prikazan je na donjoj slici:



Slika : grafički prikaz leaky ReLU funkcije i njene prve derivacije

Međutim, *leaky* ReLU ne nudi konzistentna predviđanja za negativne vrijednosti. Tijekom propagacije prema naprijed, ako je stopa učenja postavljena na preveliki broj, uzrokovat će *overshooting* i tako 'ubiti' neuron.[10][6]

Zbog toga se ideja *leaky* ReLU-a može dalje proširiti. Umjesto da se *x* pomnoži nekim konstantnim koeficijentom, može se pomnožiti s hiper-parametrom kojeg mreža može naučiti tijekom procesa učenja. Ova izvedenica *leaky* ReLU-a je poznata kao *parametric* ReLU.

### SWISH AKTIVACIJSKA FUNKCIJA

Aktivacijske funkcije imaju dugu povijest. Prvo se koristila sigmoidna funkcija zbog svoje jednostavne derivacije, opsega između 0 i 1 te zbog svog probabilističkog oblika. Tanh funkcija se smatrala zamjenom za 'običnu' sigmoidnu funkciju jer je ulazne vrijednosti preslikavala u vrijednosti koje se nalaze između -1 i 1. Međutim, te dvije funkcije su u većini slučajeva zamijenjene ReLU aktivacijskom funkcijom. [12]

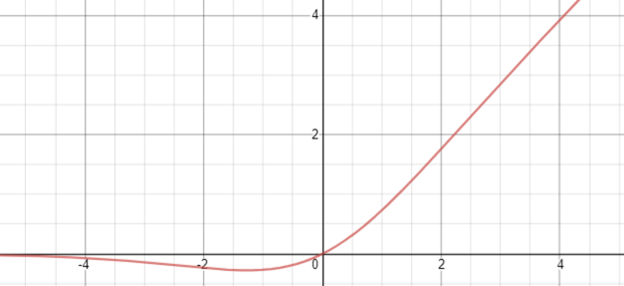
Kao i ReLU, Swish, aktivacijska funkcija razvijena od strane Google razvojnog tima, je omeđena s donje strane. To znači da, kako se vrijednost približava negativnoj vrijednosti, se približava nekoj konstantnoj vrijednosti. Isto tako je, kao i ReLU, neograničena s gornje strane (kako se vrijednost približava pozitivnoj vrijednosti, se približava nekoj beskonačnoj vrijednosti). Ali, za razliku od ReLU funkcije, Swish je 'glatka', odnosno nema iznenadne promjene gibanja.[12]

Matematička formula Swish aktivacijske funkcije glasi:

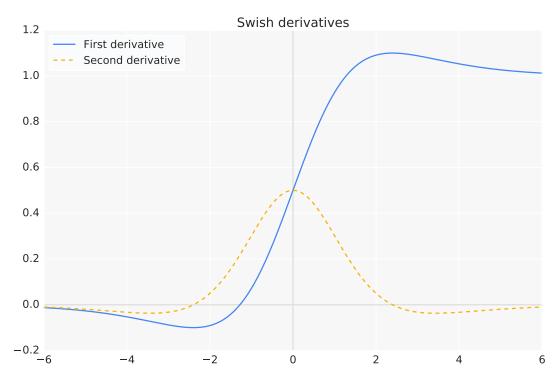
odnosno:

Derivacija Swish funkcije je onda:

Grafički prikaz Swish funkcije kao i grafički prikaz njene derivacije nalaze se na slikama.



Slika : grafički prikaz Swish aktivacijske funkcije

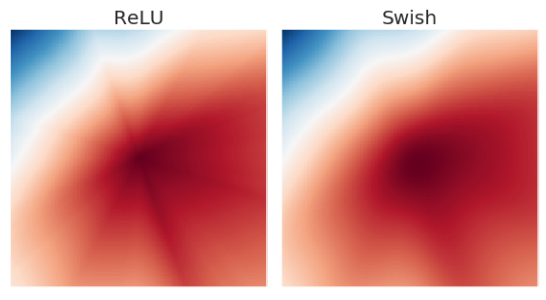


Slika : grafički prikaz prve (plavo) i druge (žuto) derivacije Swish funkcije

#### SVOJSTVA SWISH FUNKCIJE

Neomeđenost funkcije je poželjna kod svake aktivacijske funkcije jer se time izbjegava dugo vrijeme učenja s vrijednostima gradijenta koje su blizu 0, kao što su sigmoidna i tanh funkcije koje su ograničene s gornje i donje strane i zbog čega se mreže moraju oprezno inicijalizirati.[12]

Imati funkciju koja je omeđena s donje strane, kao što je ReLU funkcija, daje prednost tijekom procesa učenja jer se odbacuju velike negativne ulazne vrijednosti. Nadalje, uglađenost funkcije pomaže kod optimiziranja i generalizacije neuronske mreže. Na dolje prikazanoj slici uspoređene su ReLU i Swish funkcija gdje se vidi kako Swish funkcija ima bolju uglađenost. [12]



Slika : prikaz uglađenosti ReLU i Swish aktivacijske funkcije

Pejzažna glatkoća direktno korelira s pejzažem greški, što je on glađi, to je lakše pronaći minimum.[12]

Eksperimenti, koje je proveo Googleov razvojni tim, pokazali su da Swish funkcija daje bolje rezultate od ReLU funkcije kod dubljih modela. Na primjer, samo zamjenom ReLU funkcije Swish funkcijom, poboljšana je ispravnost klasifikacije ImageNet baze podataka za 0.9% i NASANetA za 0.6%. Nadalje, Swish je pokazala bolje izvođenje od ReLU kod modela koji imaju između 40 i 50 slojeva, kada optimizacija postaje teška. Što se tiče veličine grupa (eng. *batch*), obje funkcije bilježe pad u kvaliteti izvođenja, ali i tada Swish ipak ima bolju izvedbu od ReLU.[13]

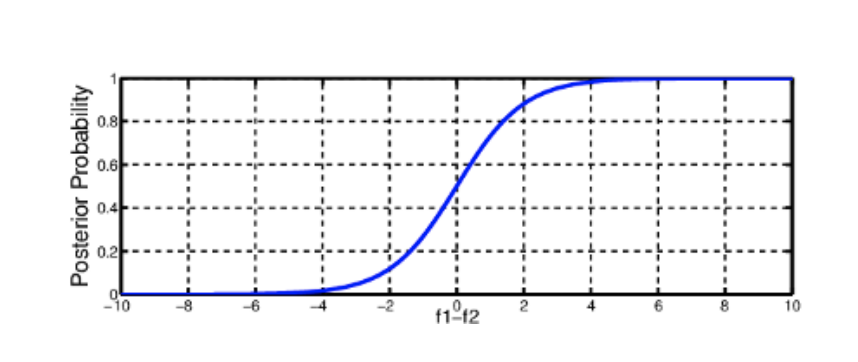
### SOFTMAX AKTIVACIJSKA FUNKCIJA

Još poznata i kao 'softargmax' funkcija ili više-klasna logistička regresija, softmax funkcija je funkcija koja prima vektor K realnih vrijednosti i vraća vektor K realnih vrijednosti čiji zbroj daje vrijednost 1. Ulazne vrijednosti mogu biti pozitivne, negativne, 0, veće od jedan, manje od jedan, ali će ih softmax funkcija uvijek transformirati u vrijednosti koje se nalaze između 0 i 1 tako da mogu biti interpretirane kao vjerojatnosti.[14]

Matematička formula softmax funkcije slična je onoj sigmoidne funkcije i glasi:

Sličnost je u tome što softmax funkcija prima vektor, dok sigmoidna funkcija prima skalar. Dapače, sigmoidna funkcija je posebni slučaj softmax funkcije kada je ulazne vrijednosti potrebno klasificirati u dvije izlazne klase.[5]

Grafički prikaz sigmoidne prikazan je na donjoj slici:



Slika : grafički prikaz softmax funkcije

Softmax funkcija se najčešće koristi na kraju neuralne mreže gdje je potrebno izlazne rezultate prikazati kao vjerojatnosti da ulaz pripada određenom izlazu.[5]

## SLOJ SAŽIMANJA

Sloj sažimanja (eng. *pooling layer*) slijedi nakon jednog ili više konvolucijskih slojeva. Taj sloj, kao što mu i samo ime govori, reducira dimenzije izlaza iz konvolucijskog sloja. Drugim riječima, uzorkuje prostornu dimenzionalnost (visinu i širinu, ali ne i broj kanala) te tako smanjuje broj parametara i kontrolira prenaučenost mreže. Postoji nekoliko razloga zašto konvolucijska neuronska mreža sadrži takav skriveni sloj.[4][5][6][8]

Konvolucijski slojevi sustavno primjenjuju određeni filter (ili više njih) na ulazne kanale kako bi kreirali matrice značajki koje ukazuju na prisutnost odgovarajućih struktura na ulaznom kanalu. Ti slojevi su se pokazali jako efikasnima i slaganje konvolucijskih slojeva u seriju omogućuje da se detektiraju sofisticirane strukture kao što su konkretni oblici i specifični objekti.[1]

Međutim, matrice značajki pohranjuju precizne lokacije detektiranih struktura. To znači da bi mali pomak u lokaciji rezultirao potpuno različitom matricom značajki. Takav mali pomak u lokaciji se događa kod manjih modifikacija na, na primjer, ulaznim slikama.[3]

Taj problem se rješava uzorkovanjem, odnosno sažimanjem. Izlaz iz sloja sažimanja je matrica značajki manjih dimenzija koja i dalje sadrži važne informacije, ali ne sadrži informacije koje vrlo vjerojatno nisu potrebne. Takve matrice značajki su korisne jer će male promjene u lokaciji rezultirati jako sličnom, ako ne i istom, matricom značajki čime je postignuta generalizacija pretpostavljenih struktura. Ova sposobnost sažimanja se zove 'neosjetljivost mreže na lokalnu translaciju' (eng. *net's invariance to local translation*).[3][6][8][9]

Slojevi sažimanja također sadrže filtere, odnosno receptivno polje. Njihova dimenzija je skoro uvijek 2x2. Ti filteri su često puno manji od filtera konvolucijskog sloja i nikada ne sadrže vlastite brojčane vrijednosti. Moglo bi ih se nazivati okvirima. Također, pomak filtera u sloju sažimanja je postavljen na brojčanu vrijednost koja namjerno izbjegava preklapanje, a to je najčešće 2.[3][8]

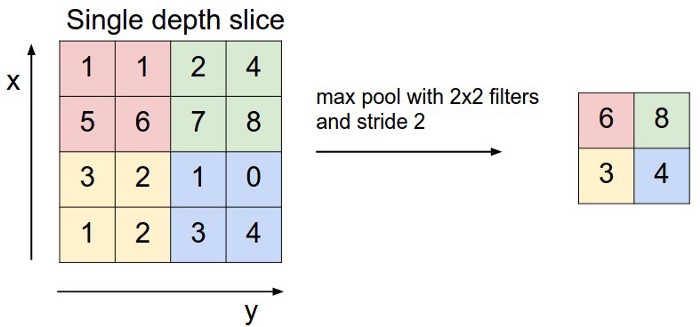
To znači da će sloj sažimanja skoro uvijek reducirati matricu značajki za faktor 2, odnosno svaka će se dimenzija prepoloviti (visina i širina). Broj elemenata ulazne matrice značajki će se smanjiti na četvrtinu.[3]

Postoje dvije moguće operacije u sloju sažimanja: sažimanje maksimalnom vrijednošću (eng. *max pooling*) i sažimanje prosječnom vrijednošću (eng. *average pooling*).[3][5]

Sažimanje maksimalnom vrijednošću uzima maksimalnu vrijednost iz okvira. To se čini tako što filter klizi po matrici značajki i, sa svakim pomakom, uzima najveću vrijednost iz okvira.[5]

Sažimanje prosječnom vrijednošću uzima prosječnu vrijednost iz okvira i pohranjuje ju u matricu značajki reduciranih dimenzija.[3]

Na dolje prikazanoj slici je opisano sažimanje maksimalnom vrijednošću:



Na slici je prikazana matrica značajki dimenzija 4x4. Filter sažimanja ima dimenzije 2x2 i pomak je postavljen na vrijednost 2.

Prva regija matrice značajki je obojena u crveno. Maksimalna vrijednost tog područja je 6 i ona se sprema u prvo polje sažete matrice značajki.

Filter se pomiče za dva mjesta u desno. Maksimalna vrijednost tog područja je 8 i ona se sprema u drugo polje.

Kako je filter došao do ruba matrice, vraća se na početak i spušta se za dva polja dolje. Cijeli proces se ponavlja i određene vrijednosti se pohranjuju na odgovarajuća mjesta u matrici.

Isti ovaj proces se odvija i kod sažimanja prosječnom vrijednošću. Jedina razlika je što se umjesto maksimalne vrijednosti uzima prosječna vrijednost područja koja se potom sprema na odgovarajuće mjesto u matrici.

Matematički se cijeli proces može opisati na sljedeći način:

Neka vrijedi sljedeća anotacija:

* Ulaz: s dimenzijama gdje je slika koja ulazi u neuronsku mrežu.
* *padding*: (rijetko se koristi) i pomak:
* veličina filtera za sažimanje:
* funkcija sažimanja:
* izlaz: s dimenzijama

Vrijedi:

S tim da:

Sloj za sažimanje ne posjeduje parametre koje konvolucijska neuronska mreža može naučiti.[7]

## POTPUNO POVEZANI SLOJEVI I IZLAZNI SLOJ

Potpuno povezani slojevi u konvolucijskoj neuronskoj mreži su standardni skriveni slojevi tradicionalne neuronske mreže. Ovi slojevi također sadrže neke od ne-linearnih aktivacijskih funkcija. Ako je taj sloj izlazni sloj, često će kao aktivacijsku funkciju imati softmax ili sigmoidnu aktivacijsku funkciju.[8]

Potpuno povezani slojevi se koriste pri kraju neuronske mreže. Oni koriste rezultate konvolucijskog sloja ili sloja sažimanja kako bi generirali određene odluke mreže.[8][9]

Postoji nekoliko vrsta potpuno povezanih slojeva u konvolucijskoj neuronskoj mreži. To su:

1. Sloj za izravnavanje (eng. *flatten layer*): uzima izlaz iz prethodnih slojeva (koji su u obliku matrice) te ih 'izravnava', odnosno transformira u vektor vrijednosti.
2. Potpuno povezani sloj: na svoje ulaze prima vrijednosti prethodnog sloja za izravnavanje pomnožene s težinama veza i na njima primjenjuje aktivacijsku funkciju. Ovaj sloj se može smatrati standardnim skrivenim slojem u tradicionalnim umjetnim neuronskim mrežama.
3. Potpuno povezani izlazni sloj: na svojim izlazima daje konačne odluke neuronske mreže za određene ulazne primjere.[9]

Još jedan način na koji se može gledati potpuno povezane slojeve je taj da potpuno povezani slojevi čine tradicionalnu neuronsku mrežu koja je integrirana s konvolucijskom neuronskom mrežom.

[1] [www.deeplizard.com](http://www.deeplizard.com)

[2] <https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/>

[3] <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/>

[4] <https://arxiv.org/pdf/1511.08458.pdf>

[5] <https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17>

[6] <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/>

[7] <https://towardsdatascience.com/convolutional-neural-networks-mathematics-1beb3e6447c0> --> tu su formule

[8] <https://machinelearningmastery.com/crash-course-convolutional-neural-networks/>

[9] <https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/>

[10] <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>

[11] <https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9>

[12] <https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529>

[13] <https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/>

[14] <https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529>

[15] <https://stackoverflow.com/questions/52272592/how-many-neurons-does-the-cnn-input-layer-have/52273707>

## AKTIVACIJSKE FUNKCIJE LITERATURA

[1] www.deeplizard.com

[2] autor nije naveden, <https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/>, 3.3.2021.

[3] autor nije naveden, <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/>, 3.3.2021.

[4] Udeme Udofia, <https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17>, 3.3.2021.

[5] Mukesh Chaudhary, <https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5>, 3.3.2021.

[6] = [8] Sagar Sharma, https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, 3.3.2021.

[7] Thomas Wood, <https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function>, 3.3.2021.

[9] autor nije imenovan, <https://paperswithcode.com/method/tanh-activation>, 3.3.2021.

[10] Danqing Liu,<https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7>, 3.3.2021.

[11] Andre Ye,<https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820>, 3.3.2021.

[12] Andre Ye,<https://towardsdatascience.com/swish-booting-relu-from-the-activation-function-throne-78f87e5ab6eb>, 3.3.2021.

[13] Random Nerd ,<https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820>, 3.3.2021.

[14] Thomas Wood, <https://deepai.org/machine-learning-glossary-and-terms/softmax-layer>, 3.3.2021.