Table of Contents

[PRIJEVOD DEEPLIZARD 1](#_Toc64462399)

[ŠTO JE DUBOKO UČENJE 1](#_Toc64462400)

[DUBOKO U DUBOKOM UČENJU 1](#_Toc64462401)

[ŠTO JE UMJETNA NEURALNA MREŽA 2](#_Toc64462402)

[VIZUALIZACIJA UMJETNE NEURALNE MREŽE 3](#_Toc64462403)

[SLOJEVI NEURALNE MREŽE 4](#_Toc64462404)

[TEŽINE SLOJEVA 5](#_Toc64462405)

[UNAPRIJEDNO PROSLJEĐIVANJE U NEURALNOJ MREŽI 5](#_Toc64462406)

[PRONALAŽENJE OPTIMALNIH TEŽINA 6](#_Toc64462407)

[AKTIVACIJSKE FUNKCIJE U NEURALNOJ MREŽI 6](#_Toc64462408)

[ŠTO JE AKTIVACIJSKA FUNKCIJA 6](#_Toc64462409)

[ŠTO ČINE AKTIVACIJSKE FUNKCIJE 6](#_Toc64462410)

[INTUICIJA AKTIVACIJSKE FUNKCIJE 7](#_Toc64462411)

[ZAŠTO SE KORISTE AKTIVACIJSKE FUNKCIJE 8](#_Toc64462412)

[TRENIRANJE UMJETNE NEURALNE MREŽE 9](#_Toc64462413)

[ŠTO ZNAČI TRENIRATI UMJETNU NEURALNU MREŽU 9](#_Toc64462414)

[OPTIMIZACIJSKI ALGORITAM 9](#_Toc64462415)

[FUNKCIJA GUBITKA 9](#_Toc64462416)

[KAKO NEURALNA MREŽA UČI 10](#_Toc64462417)

[ŠTO TO ZNAČI UČITI 10](#_Toc64462418)

[GRADIJENT FUNKCIJE GUBITKA 11](#_Toc64462419)

[STOPA UČENJA 11](#_Toc64462420)

[AŽURIRANJE TEŽINA VEZA 11](#_Toc64462421)

[MODEL UČI 11](#_Toc64462422)

[GUBITAK U NEURALNOJ MREŽI 12](#_Toc64462423)

[SREDNJA KVADRATNA POGREŠKA 12](#_Toc64462424)

[STOPA UČENJA U NEURALNOJ MREŽI 13](#_Toc64462425)

[AŽURIRANJE TEŽINA VEZA U MREŽI 13](#_Toc64462426)

[SKUPOVI ZA TRENIRANJE, TESTIRANJE I VALIDACIJU 14](#_Toc64462427)

[SKUPOVI PODATAKA ZA DUBOKO UČENJE 14](#_Toc64462428)

[SKUP PODATAKA ZA TRENIRANJE 14](#_Toc64462429)

[SKUP PODATAKA ZA VALIDACIJU 15](#_Toc64462430)

[SKUP PODATAKA ZA TESTIRANJE 15](#_Toc64462431)

[PREDVIĐANJE NEURALNE MREŽE 16](#_Toc64462432)

[PROSLJEĐIVANJE UZORAKA BEZ OZNAKA 16](#_Toc64462433)

[*OVERFITTING* U NEURALNOJ MREŽI 17](#_Toc64462434)

[KAKO UOČITI PRETJERANO POKLAPANJE 17](#_Toc64462435)

[SMANJENJE PRETJERANOG POKLAPANJA 17](#_Toc64462436)

[NEDOVOLJNO POKLAPANJE U NEURALNOJ MREŽI 18](#_Toc64462437)

[REDUCIRANJE NEDOVOLJNOG POKLAPANJA 18](#_Toc64462438)

[NADZIRANO UČENJE KOD STROJNOG UČENJA 19](#_Toc64462439)

[OZNAČENI PODACI 19](#_Toc64462440)

[OZNAKE SU NUMERIČKE 20](#_Toc64462441)

[POVEĆANJE PODATAKA ZA STROJNO UČENJE 20](#_Toc64462442)

[ZAŠTO POVEĆATI PODATKE 20](#_Toc64462443)

[ONE-HOT ENKODIRANJE ZA STROJNO UČENJE 20](#_Toc64462444)

[OZNAKE 21](#_Toc64462445)

[TOPLE I HLADNE VRIJEDNOSTI 21](#_Toc64462446)

[VEKTORI S NULAMA I JEDINICAMA 21](#_Toc64462447)

[ONE-HOT ENKODIRANJE ZA VIŠE KATEGORIJA 21](#_Toc64462448)

[JEDAN VEKTOR ZA SVAKU KATEGORIJU 22](#_Toc64462449)

[DUBOKO UČENJE KOD KONVOLUCIJSKIH NEURALNIH MREŽA 22](#_Toc64462450)

[FILTERI I KONVOLUCIJSKE OPERACIJE 23](#_Toc64462451)

[PATTERNS 23](#_Toc64462452)

[FILTERI (DETEKTORI OBRASCA) 23](#_Toc64462453)

[KONVOLUCIJSKI SLOJ 24](#_Toc64462454)

[KONVOLUCIJSKA OPERACIJA 25](#_Toc64462455)

[NAPOMENA O KORIŠTENJU TOČKASTOG PRODUKTA 25](#_Toc64462456)

[ULAZNI I IZLAZNI KANALI 26](#_Toc64462457)

[POPUNJAVANJE NULAMA 29](#_Toc64462458)

[KONVOLUCIJE SMANJUJU DIMENZIJE KANALA 29](#_Toc64462459)

[PROBLEMI S REDUCIRANJEM DIMENZIJA 30](#_Toc64462460)

[POPUNJAVANJE NULAMA 30](#_Toc64462461)

[*VALID* POPUNJAVANJE *I SAME* POPUNJAVANJE 31](#_Toc64462462)

[*MAX POOLING* U KONVOLUCIJSKIM NEURALNIM MREŽAMA 31](#_Toc64462463)

[PRIMJER *MAX-POOLING-A* NA JEDNOM UZROKU IZ MNIST SKUPA PODATAKA 31](#_Toc64462464)

[UMANJENI PRIMJER 33](#_Toc64462465)

[ZAŠTO KORISTITI *MAX POOLING* 34](#_Toc64462466)

[REDUCIRANJE RAČUNSKOG OPTEREĆENJA 34](#_Toc64462467)

[REDUCIRANJE PRETJERANOG POKLAPANJA 34](#_Toc64462468)

[*AVERAGE POOLING* 35](#_Toc64462469)

[POVRATNA PROPAGACIJA U NEURALNIM MREŽAMA 35](#_Toc64462470)

[STOGASTIČKI GRADIJENTNI SPUST 35](#_Toc64462471)

[PROPAGACIJA PREMA NAPRIJED 35](#_Toc64462472)

[RAČUNANJE GUBITKA 36](#_Toc64462473)

[INTUICIJA POVRATNE PROPAGACIJE 37](#_Toc64462474)

[NESTAJUĆI I EKSPLODIRAJUĆI GRADIJENT 39](#_Toc64462475)

[MALI GRADIJENTI 40](#_Toc64462476)

[ZAGLAVLJENE TEŽINE 40](#_Toc64462477)

[EKSPLODIRAJUĆI GRADIJENT 41](#_Toc64462478)

[INICIJALIZACIJA TEŽINA 41](#_Toc64462479)

[KAKO SE TEŽINE INICIJALIZIRAJU 41](#_Toc64462480)

[PRIMJER NASUMIČNE INICIJALIZACIJE 41](#_Toc64462481)

[PROBLEMI S NASUMIČNOM INICIJALIZACIJOM 42](#_Toc64462482)

[XAIVER INICIJALIZACIJA 43](#_Toc64462483)

[PRISTRANOST U UMJETNOJ NEURALNOJ MREŽI 43](#_Toc64462484)

[GDJE SE NALAZI PRISTRANOST 44](#_Toc64462485)

[PRIMJER GDJE SE KORISTI PRISTRANOST 44](#_Toc64462486)

[PARAMETRI U NEURALNOJ MREŽI KOJI SE MOGU NAUČITI 45](#_Toc64462487)

[IZRAČUN PARAMETARA KOJI SE MOGU NAUČITI 45](#_Toc64462488)

[PRIMJER IZRAČUNA BROJA PARAMETARA KOJI SE MOGU NAUČITI 45](#_Toc64462489)

[PARAMETRI KOJI SE MOGU NAUČITI U KONVOLUCIJSKOJ NEURALNOJ MREŽI 46](#_Toc64462490)

[KAKO SE RAČUNA BROJ PARAMETARA U KONVOLUCIJSKOJ NEURALNOJ MREŽI 47](#_Toc64462491)

[PRIMJER RAČUNANJA BROJA PARAMETARA U KONVOLUCIJSKOJ NEURALNOJ MREŽI 47](#_Toc64462492)

[REGULARIZACIJA U KONVOLUCIJSKOJ NEURALNOJ MREŽI 48](#_Toc64462493)

[L2 REGULARIZACIJA 49](#_Toc64462494)

[UTJECAJ REGULARIZACIJE 50](#_Toc64462495)

[VELIČINA PODSKUPOVA U UMJETNOJ NEURALNOJ MREŽI 50](#_Toc64462496)

[PODSKUPOVI U EPOHI 50](#_Toc64462497)

[RAZLOG KORIŠTENJA PODSKUPOVA 50](#_Toc64462498)

[MINI-BATCH GRADIJENTNI SPUST 51](#_Toc64462499)

[FINO PODEŠAVANJE NEURALNE MREŽE – NIJE POTREBNO? 51](#_Toc64462500)

[BATCH NORMALIZACIJA 52](#_Toc64462501)

[TEHNIKE NORMALIZACIJE 52](#_Toc64462502)

[KORIŠTENJE TEHNIKA NORMALIZACIJE 52](#_Toc64462503)

[PRIMJENA BATCH NORMALIZACIJE NA SLOJ 53](#_Toc64462504)

# PRIJEVOD DEEPLIZARD

## ŠTO JE DUBOKO UČENJE

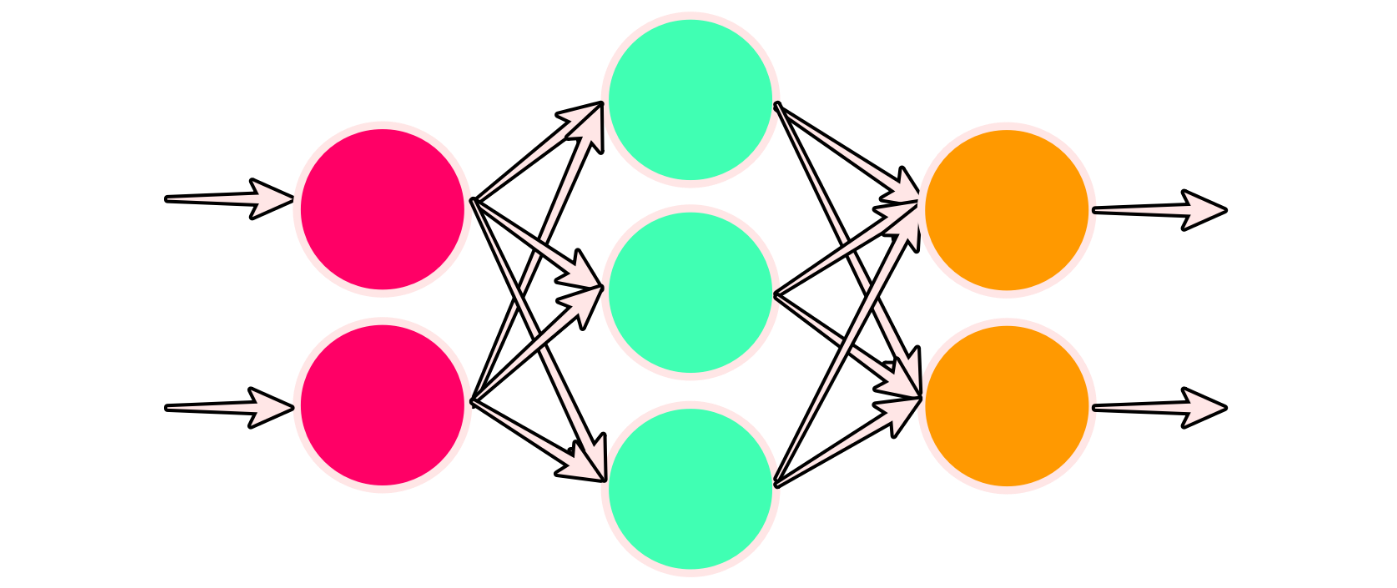
Duboko učenje je polje strojnog učenja koje koristi algoritme koji su inspirirani strukturom i funkcijama neuralnih mreža.

Kod dubokog učenja, još se uvijek misli na algoritme koji uče iz podataka i na algoritme ili modele koji uče na temelju slabe poveznice sa strukturom i funkcijama neuralnih mreža.

Neuralne mreže koje se koriste u dubokom učenju nisu prave biološke neuralne mreže. Kako neuralne mreže u dubokom učenju dijele tek nekoliko karakteristika s biološkim neuralnim mrežama, nazivaju se *umjetne neuralne mreže* (eng. *artificial neural networks, ANNs*).

Popularna engleska kratica za izraz 'umjetna neuralna mreža' jest ANN. Međutim, često se u stranim literaturama može naići i na pojmove na engleskom jeziku koji se također koriste za pojam neuralne mreže. Ti pojmovi su:

* *Net*
* *Neural net*
* *Model*



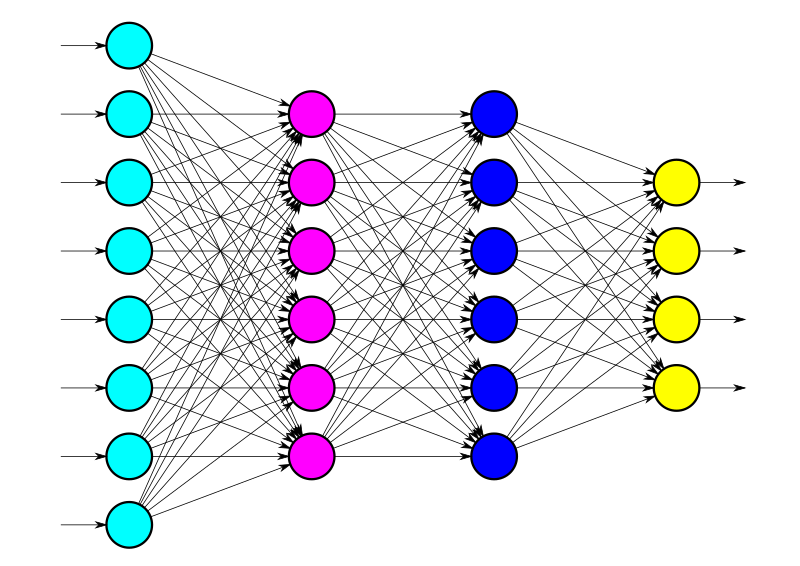
Slika : shematski prikaz jednostavne neuralne mreže

## DUBOKO U DUBOKOM UČENJU

Duboko učenje koristi specifični tip neuralnih mreža koji se naziva 'duboka mreža' ili 'duboka umjetna neuralna mreža' (eng. *deep net* ili *deep artificial neural network*).

Ukratko, struktura neuralne mreže je sljedeća:

1. Umjetne neuralne mreže grade takozvani neuroni .
2. Neuroni su u umjetnim neuralnim mrežama organizirani u takozvane slojeve.
3. Slojevi unutar umjetne neuralne mreže (slojevi bez ulaznih (eng. *input*) i izlaznih (eng. *output*) slojeva) nazivaju se 'skriveni slojevi' (eng. *hidden layers*).
4. Ako se umjetna neuralna mreža sastoji od više skrivenih slojeva, ona se tada naziva duboka neuralna mreža.



Slika : shematski prikaz duboke umjetne neuralne mreže s dva skrivena sloja

Ukratko, duboko učenje koristi umjetne neuralne mreže koje imaju više skrivenih slojeva.

## ŠTO JE UMJETNA NEURALNA MREŽA

U jednom od prethodnih poglavlja smo definirali duboko učenje kao područje strojnog učenja koje koristi algoritme koji su inspirirani strukturom i funkcijama bioloških neuralnih mreža. Iz tog razloga, modeli, koji se koriste u dubokom učenju, nazivaju se 'umjetne neuralne mreže'.

Umjetna neuralna mreža je računalni sustav koji se sastoji od kolekcije povezanih jedinica koji se nazivaju neuronima, a neuroni su organizirani u takozvane slojeve. Nadalje, povezane neuralne jedinice tvore takozvanu mrežu.

Svaka veza između dva neurona transmitira signal od jednog neurona prema drugom. Neuron, koji prima dolazni signal, obrađuje signal te ga šalje neuronima koji se nalaze u sljedećem sloju i s kojima je povezan unutar neuralne mreže.

Neuroni se još često nazivaju čvorovima (eng. *nodes*).

Kao što je već prije navedeno, čvorovi su organizirani u takozvane slojeve. Općenito, u svakoj umjetnoj neuralnoj mreži, slojevi se mogu podijeliti u tri glavne skupine:

1. Ulazni sloj (eng. *input layer*)
2. Skriveni sloj (eng. *hidden layer*)
3. Izlazni sloj (eng. *output layer*)

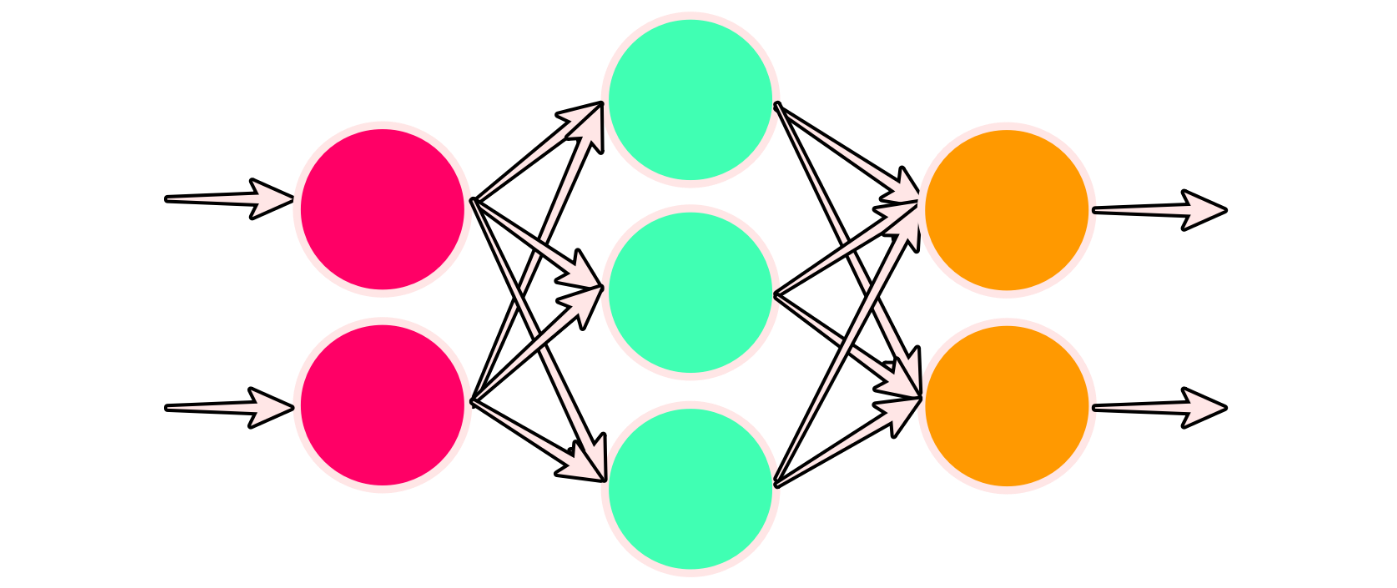
Različiti slojevi izvode različite vrste transformacija na vlastitim ulazima.

Podatak prolazi kroz mrežu tako da ulazi kroz ulazni sloj i kreće se kroz skrivene slojeve sve dok ne stigne do izlaznog sloja. Ovaj tip prolaska podataka kroz mrežu se naziva 'unaprijedno prosljeđivanje kroz mrežu' (eng. *forward pass through the network*). Slojevi, koji su smješteni između ulaznog i izlaznog sloja, nazivaju se 'skriveni slojevi' (eng. *hidden layers*).

Što se tiče broja čvorova u svakom pojedinom sloju, pravila su sljedeća:

1. Ulazni sloj sadrži po jedan čvor za svaku pojedinu komponentu ulaznog podatka.
2. Skriveni slojevi sadrže proizvoljno odabran broj čvorova za svaki pojedini skriveni sloj.
3. Izlazni sloj sadrži po jedan čvor za svaki mogući željeni izlaz.

## VIZUALIZACIJA UMJETNE NEURALNE MREŽE



Slika : shematski prikaz jednostavne neuralne mreže

Umjetna neuralna mreža prema gore prikazanoj slici (Slika 1) sastoji se od sveukupno tri sloja. Lijevi sloj je ulazni sloj, desni sloj je izlazni sloj, dok je sloj u sredini skriveni sloj.

Kao što je već prije spomenuto, svaki sloj u umjetnoj neuralnoj mreži sastoji se od neurona ili čvorova. Neuroni su u gornjoj slici prikazani krugovima te je broj neurona u svakom pojedinom sloju sljedeći:

1. Ulazni sloj (lijevo): 2 neurona
2. Skriveni sloj (sredina): 3 neurona
3. Izlazni sloj (desno): 2 neurona

Kako ova jednostavna neuralna mreža sadrži dva neurona u ulaznom sloju, to znači da svaki ulaz u onu umjetnu neuralnu mrežu mora imati dvije dimenzije. To mogu biti, na primjer, visina i težina.

Nadalje, s obzirom da ova neuralna mreža sadrži dva neurona u izlaznom sloju, to znači da postoje dva moguća izlaza za svaki ulaz koji je unaprijedno proslijeđen kroz mrežu (s lijeva na desno).

Na primjer, izlazi za ovakvu mrežu mogu biti osobine *težak* ili *mršav*. Tu je dobro napomenuti da su izlazne klase poznate i kao 'klase predviđanja' (eng. *prediction classes*).

## SLOJEVI NEURALNE MREŽE

U jednom od prethodnih poglavlja navedeno jer kako su neuroni unutar umjetne neuralne mreže organizirani u slojeve. Primjeri su prikazali takozvane 'guste' slojeve (eng. *dense layers*) koji su još poznati i kao 'potpuno povezani' slojevi (eng. *fully connected layers*).

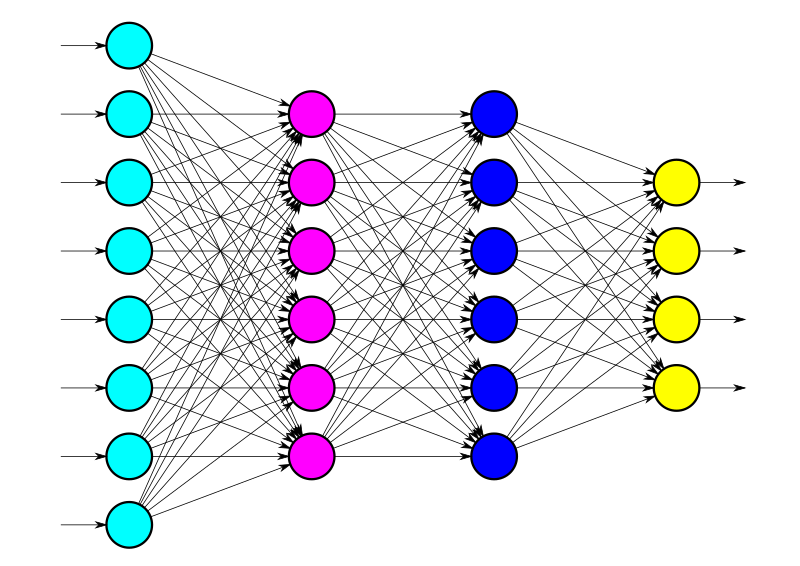
Međutim, postoje različiti tipovi slojeva u umjetnim neuralnim mrežama. Neki od tipova su:

1. Gusti (ili potpuno povezani) slojevi
2. Konvolucijski slojevi
3. Slojevi udruživanja (eng. *pooling layers*)
4. Povratni slojevi (eng. *recurrent layers*)
5. Normalizacijski slojevi (eng. *normalization layers*)

Razlog za postojanje različitih tipova slojeva je taj da različiti slojevi izvode različite transformacije na svojim ulazima te su stoga neki slojevi prikladniji za pojedini zadatak u odnosu na druge slojeve.

Na primjer, konvolucijski sloj je obično korišten u modelima koji rade na obradi slika. Povratni slojevi su korišteni u modelima koji rade s podacima u vremenskim serijama. Također, potpuno povezani slojevi, kao što i samo ime tog sloja sugerira, potpuno povezuje svaki ulaz sa svakim izlazom unutar svog sloja.

Promotrimo sljedeći primjer umjetne neuralne mreže:



Kao što se može primijetiti, prvi, odnosno ulazni, sloj se sastoji od osam čvorova. Svaki od tih osam čvorova u ulaznom sloju predstavlja individualnu značajku odabranog uzorka iz određenog skupa podataka.

To nam govori da se jedan uzorak iz određenog skupa podataka sastoji od osam dimenzija. Nadalje, kada se taj odabrani uzorak iz određenog skupa podataka proslijedi modelu, svaka će od tih osam vrijednosti, odnosno dimenzija, od koji se sastoji odabrani uzorak, biti pridružena odgovarajućem čvoru u ulaznom sloju.

Kao što se može primijetiti, svaki od osam ulaznih čvorova je povezan sa svakim čvorom u sljedećem sloju. Svaka veza između prvog i drugog sloja prenosi izlaz iz čvora, koji se nalazi u prethodnom sloju, na ulaz prijemnog čvora (s lijeva na desno). Svaki od dva sloja, koji se nalaze u sredini neuralne mreže, sadrže po šest čvorova. Ta dva sloja se nazivaju skriveni slojevi jednostavno zbog toga jer se nalaze između ulaznog i izlaznog sloja.

### TEŽINE SLOJEVA

Svaka veza između dva čvora u mreži ima pridruženu težinu. Ta težina je samo neka brojčana vrijednost.

Svaka težina predstavlja snagu veze između dva čvora. Kada mreža na ulazu dobije neku određenu ulaznu vrijednost, ona se pridružuje određenom čvoru u ulaznom sloju. Kada se ta ulazna vrijednost prosljeđuje određenom čvoru u sljedećem sloju, ona će biti pomnožena s težinom koja je pridružena vezi koja povezuje čvor u ulaznom sloju i čvor u skrivenom sloju.

Tada se za svaki čvor u drugom, skrivenom, sloju, izračunava ponderirana (vagana) suma sa svakom od dolaznih veza. Ponderirana suma je tada prosljeđena aktivacijskoj funkciji koja transformira danu sumu. Na primjer, aktivacijska funkcija može transformirati sumu tako da suma poprimi vrijednost između nula i jedan. Naravno, stvarna transformacija će varirati ovisno o tome koja se aktivacijska funkcija koristi.

Izlaz iz čvora = aktivacijska funkcija(ponderirana suma ulaza)

### UNAPRIJEDNO PROSLJEĐIVANJE U NEURALNOJ MREŽI

Jednom kada se dobije izlaz za dani čvor, taj dobiveni izlaz je vrijednost koja se prenosi kao ulazna vrijednost čvorovima u sljedećem sloju.

Isti se proces, koji je opisan u prethodnom poglavlju, ponavlja sve dok se ne dostigne izlazni sloj. Broj čvorova u izlaznom čvoru ovisi o broju mogućih izlaza, odnosno o broju klasa predviđanja. U primjeru gore prikazane neuralne mreže, broj mogućih klasa predviđanja je četiri.

Neka se, na primjer, pretpostavi da je zadatak gore prikazanom modelu neuralne mreže da klasificira četiri vrste životinja. To mogu biti mačka, pas, ljama i gušter. Kategorije, ili klase, ovise o tome koliko klasa postoji u danom skupu podataka.

Za dani uzorak iz skupa podataka, cijeli se proces od ulaznog sloja do izlaznog sloja naziva 'unaprijedno prosljeđivanje kroz mrežu'.

### PRONALAŽENJE OPTIMALNIH TEŽINA

Kako model uči, težine veza se ažuriraju i optimiziraju kako bi ulazni podaci bili preslikani u ispravnu klasu predviđanja na izlazu.

## AKTIVACIJSKE FUNKCIJE U NEURALNOJ MREŽI

U ovom poglavlju će se opisati što je to aktivacijska funkcija te kako se ona koristi u umjetnim neuralnim mrežama.

### ŠTO JE AKTIVACIJSKA FUNKCIJA

Aktivacijska funkcija u umjetnoj neuralnoj mreži jest funkcija koja preslikava ulaznu vrijednost čvora u odgovarajuću izlaznu vrijednost.

Kao što je opisano u jednom od prethodnih poglavlja, ponderirana suma svih dolaznih veza na danom čvoru je prosljeđena aktivacijskoj funkciji.

Izlaz iz čvora = aktivacijska funkcija(ponderirana suma ulaza)

Aktivacijska funkcija na neki način transformira sumu u broj koji se nalazi između unaprijed određene donje i gornje granice. Aktivacijska funkcija u umjetnim neuralnim mrežama obično izvodi neku vrstu ne-linearne transformacije.

### ŠTO ČINE AKTIVACIJSKE FUNKCIJE

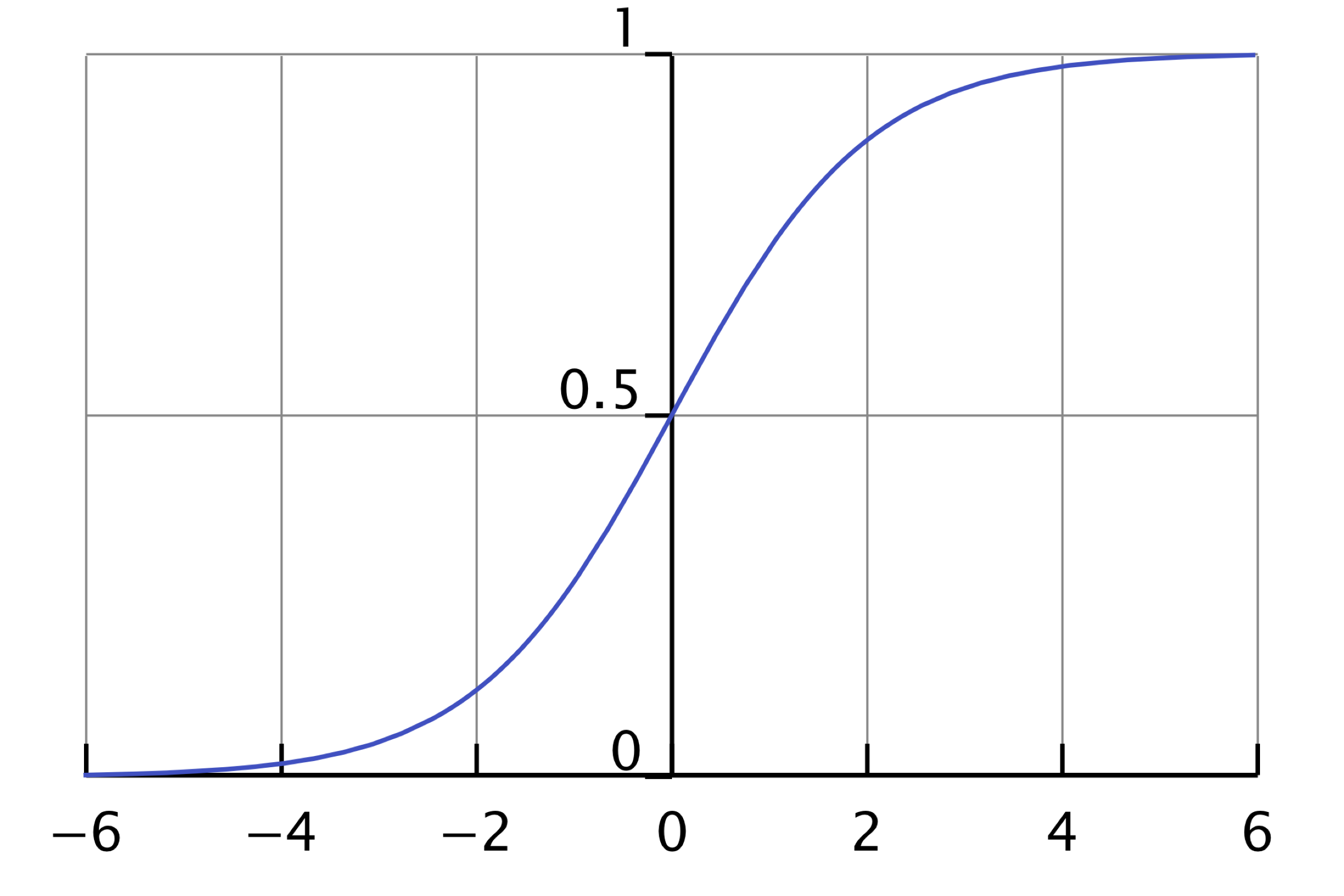
U ovom poglavlju je opisano nekoliko aktivacijskih funkcija.

**SIGMOIDNA AKTIVACIJSKA FUNKCIJA**

Sigmoidna funkcija uzima ulaznu vrijednost te čini sljedeće:

1. Za jako negativne ulazne vrijednosti, sigmoidna funkcija će tu ulaznu vrijednost pretvoriti u vrijednost koja je jako blizu nuli.
2. Za jako pozitivne ulazne vrijednosti, sigmoidna funkcija će tu ulaznu vrijednost pretvoriti u broj jako blizu broju jedan.
3. Za ulaznu vrijednost koja se nalazi relativno blizu broju nula, sigmoidna funkcija će tu ulaznu vrijednost pretvoriti u broj koji se nalazi između nula i jedan.

Matematička formula sigmoidne funkcije jest:



Slika : grafički prikaz sigmoidne funkcije

Tako, za sigmoidnu funkciju, donja granica je nula, a gornja granica je jedan.

### INTUICIJA AKTIVACIJSKE FUNKCIJE

Aktivacijska funkcija je inspirirana aktivnošću u biološkoj neuralnoj mreži u kojoj su različiti neuroni aktivirani različitim podražajima.

U biološkoj neuralnoj mreži, neki su neuroni ili aktivirani ili ne. U aktivacijskoj funkciji, vrijednost neurona može biti bilo koji broj između nula i jedan. Što je ta vrijednost bliža nuli, to je neuron manje aktiviran, a što je ta vrijednost bliže broju jedan, to je neuron više aktiviran.

**ReLU AKTIVACIJSKA FUNKCIJA**

Međutim, aktivacijska funkcija neće uvijek transformirati ulaznu vrijednost tako da transformirana vrijednost bude broj između nula i jedan.

Dapače, jedna od najčešće korištenih aktivacijskih funkcija, ReLU, skraćeno od *Rectified Linear Unit*, transformira ulaznu vrijednost tako da transformirana vrijednost poprima maksimum ili nula ili tu samu ulaznu vrijednost.

Tako, ako je ulazna vrijednost manja ili jednaka broju jedan, ReLU će na ulaznu dati broj nula. Ako je ulazna vrijednost veća od nule, ReLu će na ulazu dati tu istu ulaznu vrijednost.

Ideja kod ReLU aktivacijske funkcije je ta da, što je vrijednost neurona pozitivnija, to će taj neuron biti više aktiviran. Uz ReLU i sigmoidnu aktivacijsku funkciju, postoji još vrsta aktivacijskih funkcija.

### ZAŠTO SE KORISTE AKTIVACIJSKE FUNKCIJE

Kako bi se razumjelo zašto se koriste aktivacijske funkcije, 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 je kompozicija dviju linearnih funkcija također linearna funkcija. To znači da će čak i u dubokim umjetnim neuralnim mrežama, ako se koriste samo linearne funkcije na ulaznim vrijednostima kod prosljeđivanja prema naprijed (eng. *forward propagation*), preslikavanje ulaza na izlaz također biti linearno.

Obično su vrste preslikavanja kod dubokih neuralnih mreža složenije od običnih linearnih preslikavanja.

Većina aktivacijskih funkcija je ne-linearno s razlogom. Aktivacijske ne-linearne funkcije omogućuju neuralnim mrežama računanje proizvoljno složenih funkcija.

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

Kako bi se dokazalo da je ReLU aktivacijska funkcija ne-linearna funkcija, pokazat će se da ReLU ne uspijeva biti linearna funkcija.

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

Neka se pretpostavi da je *a* realan broj za koji vrijedi *a* < 0.

Koristeći pretpostavku da vrijedi *a* < 0, može se vidjeti da je:

i da je:

Radi ovoga dolazimo do zaključka:

Tako je pokazano da funkcija *f*, odnosno ReLU funkcija, ne uspijeva biti linearna funkcija.

## TRENIRANJE UMJETNE NEURALNE MREŽE

U ovom poglavlju će se opisati što to znači trenirati umjetnu neuralnu mrežu. U jednom od prethodnih poglavlja opisana je osnovna arhitektura opće umjetne neuralne mreže. Nakon što je konfigurirana arhitektura modela, sljedeći korak je treniranje tog modela.

### ŠTO ZNAČI TRENIRATI UMJETNU NEURALNU MREŽU

Treniranjem modela, pokušava se riješiti problem optimizacije, odnosno pokušavaju se optimizirati težine veza u danom modelu. Zadatak optimizacije jest pronaći težine koje najispravnije preslikavaju ulazne podatke u klase predviđanja. Preslikavanje je ono što model, odnosno umjetna neuralna mreža, treba naučiti.

U jednom od prethodnih poglavlja je prikazano kako je svakoj vezi između dva čvora pridružena neka težina. Tijekom treniranja, težine se iterativno ažuriraju prema svojim optimalnim vrijednostima.

### OPTIMIZACIJSKI ALGORITAM

Težine veza se optimiziraju takozvanim optimizacijskim algoritmom. Optimizacijski proces ovisi o odabranom optimizacijskom algoritmu. Optimizacijski algoritam se u literaturi još naziva i 'optimizatorom' (eng. *optimizer*). Najpoznatiji optimizacijski algoritam se naziva 'Stohastički gradijentni spust' (eng. *stochastic gradient descent*), ili skraćeno SGD.

Kod bilo kojeg problema optimizacije, važno je imati cilj optimizacije. Ovdje će se opisati cilj optimizacije SGD-a prilikom optimizacije težine veza u modelu.

Cilj SGD-a jest minimizirati funkciju gubitka (eng. *loss function*). SGD ažurira težine veza u modelu na način da funkciju gubitka približi što je više moguće svojoj minimalnoj vrijednosti.

### FUNKCIJA GUBITKA

Funkcija gubitka, koja se često koristi u neuralnim mrežama, jest funkcija srednje kvadratne pogreške (eng. *mean squared error*), ili skraćeno MSE. Uz MSE postoji još nekoliko često korištenih funkcija gubitaka.

Što je to točno gubitak će se objasniti preko sljedećeg primjera.

Tijekom treniranja modela, model se opskrbljuje podacima i odgovarajućim oznakama (eng. *labels*) za te podatke.

Na primjer, neka se model trenira s ciljem da uspješno klasificira dolazne fotografije na fotografije mački i fotografije pasa. Model se opskrbljuje fotografijama mački i pasa zajedno s oznakama za fotografije koje govore je li ulazna fotografija mačke ili psa.

Neka se pretpostavi da je modelu dana fotografija mačke. Kad unaprijedno prosljeđivanje dođe kraju i fotografija mačke je prošla kroz mrežu, model će na izlazu dati neku vrijednost. Ta vrijednost će pokazati misli li model da je dobio fotografiju mačke ili psa.

U doslovnom smislu, izlaz će se sastojati od vjerojatnosti za mačku ili psa. Na primjer, model na izlazu može dodijeliti vjerojatnost od 75% da je ulazna fotografija ona s mačkom i vjerojatnost od 25% da je ulazna fotografija ona sa psom.

* 75% vjerojatnosti da je na fotografiji mačka
* 25% vjerojatnosti da je na fotografiji pas

Gubitak je pogreška ili razlika između što mreža predviđa za sliku i onoga što je prava oznaka slike. SGD će pokušati minimizirati ovu pogrešku tako da model postane što točniji kod predviđanja. Nakon što su svi podaci prošli kroz model, svi podaci se ponovno propuštaju kroz model. Proces ponovnog propuštanja istih podataka kroz mrežu se smatra treniranjem. Tako, kroz ovaj proces koji se iterativno ponavlja u kombinaciji s SGD-om, model je u stanju učiti na danim podacima.

## KAKO NEURALNA MREŽA UČI

U prethodnom je poglavlju objašnjeno što to znači kad umjetna neuralna mreža uči te kako izgleda proces treniranja i kako se svaki podatak, koji se koristi u procesu treniranja, prosljeđuje kroz mrežu. Prosljeđivanje podataka od ulaza do izlaza se naziva *unaprijedno prosljeđivanje*. Rezultat izlaza ovisi o težini svake pojedine veze unutar mreže.

Kada su svi podaci za treniranje iz skupa podataka za treniranje prosljeđeni kroz mrežu, kaže se da je završena jedna epoha (eng. *epoch*). Epoha se odnosi na jedan prolazak cijelog skupa podataka kroz mrežu tijekom procesa treniranja. Ovdje je dobro spomenuti da se, tijekom procesa treniranja, kako model uči, 'događa' više epoha.

### ŠTO TO ZNAČI UČITI

Kako bi se objasnio pojam učenja, dobro je prisjetiti se da, kada se model inicijalizira, težine veza su postavljene na neke proizvoljne vrijednosti te da model, na izlazu iz mreže, daje određeni izlaz za dani ulaz.

Kada se dobije izlaz, računa se gubitak (ili pogreška) za dani izlaz tako da se usporedi vrijednost koju je model predvidio s pravom vrijednošću oznake. Računanje gubitka ovisi o odabranoj funkciji gubitka.

### GRADIJENT FUNKCIJE GUBITKA

Kada je izračunat gubitak, gradijent odabrane funkcije gubitka se računa u odnosu na svaku težinu veze unutar mreže. Ovdje se pod pojmom 'gradijent' misli na derivaciju funkcije koja ima više varijabli.

Kako bi se što jednostavnije objasnio gradijent funkcije gubitka, opisat će se gubitak u odnosu na samo jedan izlaz. Kada se izračuna gubitak tog jednog izlaza, računa se gradijent tog gubitka u odnosu na jednu odabranu težinu. Gradijent se računa tehnikom koja se zove 'povratna propagacija' (eng. *backpropagation*).

Na temelju izračunate vrijednosti gradijenta funkcije gubitka, ažurira se težina veze u modelu. Gradijent govori koji smjer pomiče gubitak prema svojoj najmanjoj vrijednosti te je cilj mijenjati vrijednost težine veze u tom smjeru u kojem će gubitak približavati svojoj minimalnoj vrijednosti.

### STOPA UČENJA

Kada se izračuna vrijednost gradijenta funkcije gubitka, ona se tada množi s vrijednošću koja se zove 'stopa učenja' (eng. *learing rate*). Stopa učenja je broj koji se obično nalazi u rasponu između 0.01 i 0.0001. Naravno, prava vrijednost stope učenja može varirati.

Stopa učenja govori koliki se 'korak' mora napraviti u smjeru minimalne vrijednosti gubitka.

### AŽURIRANJE TEŽINA VEZA

Kako bi se izračunala nova vrijednost težine veze, kao što je napisano u prethodnom poglavlju, gradijent funkcije gubitka se množi sa stopom učenja. Dobivena vrijednost se tada oduzima od vrijednosti težine veze te se dobiva nova vrijednost težine veze.

nova težina = stara težina - (stopa učenja \* gradijent)

Ovaj primjer se fokusirao na samo jednu težinu veze kako bi se objasnio koncept, ali se isti ovaj proces odnosi na svaku težinu veze u modelu svaki putu kada kroz model prolaze podaci iz skupa podataka za treniranje.

Jedina razlika je ta što će vrijednost gradijenta biti različita za svaku pojedinu težinu jer se gradijent funkcije gubitka računa u odnosu na svaku pojedinu težinu veze u neuralnoj mreži.

Ako se pretpostavi da se veze ažuriraju na kraju svake epohe, one će se inkrementalno približavati svojim optimalnim vrijednostima tako što SGD nastoji minimizirati funkciju gubitka.

### MODEL UČI

Pod pojmom da model uči se misli na to da se ažuriraju težine veza. Učiti koje se vrijednosti trebaju pripisati svakoj pojedinoj vezi na temelju toga kako te inkrementalne promjene (vrijednosti težine veza) utječu na funkciju gubitka. Kako se težine mijenjaju, model postaje sve 'pametniji' u smislu da sve ispravnije preslikava ulaze u odgovarajuće izlaze.

## GUBITAK U NEURALNOJ MREŽI

U ovom poglavlju će se objasniti što je to funkcija gubitka i kako se ona koristi u umjetnoj neuralnoj mreži.

Funkcija gubitka je ono što stohastički gradijentni spust nastoji minimizirati tako što iterativno ažurira težine veza u neuralnoj mreži. Na kraju svake epohe tijekom procesa treniranja se izračunava gubitak korištenjem razlike izlaza, odnosno onoga što je model predvidio, i pravih oznaka za pojedini ulaz.

Kao što je dan primjer u jednom od prethodnih poglavlja, neka se pretpostavi da model klasificira fotografije mački i pasa te neka je oznaka za mačku 0, a za psa 1.

* Mačka: 0
* Pas: 1

Nadalje, neka se modelu proslijedi fotografija mačke i neka model na izlazu da vrijednost 0.25. U ovom slučaju, vrijednost između toga što je model predvidio i prave vrijednosti oznake jest 0.25 – 0.00 = 0.25. Ova razlika se naziva 'pogreška' (eng. *error*).

pogreška = 0.25 – 0.00 = 0.25

Ovaj proces računanja pogreške se ponavlja za svaki izlaz. Na kraju svake epohe, pogreška akumulira svoju vrijednost za svaki pojedini izlaz.

U sljedećem poglavlju će se opisati funkcija gubitka koja se obično koristi za računanje pogreške i koja se zove 'srednja kvadratna pogreška' (eng. *mean squared error*), skraćeno MSE.

### SREDNJA KVADRATNA POGREŠKA

Kod samo jednog uzorka, za računanje srednje kvadratne pogreške, prvo se izračuna razlika (pogreška) između onoga što je model predvidio za dani ulaz i prave vrijednosti oznake za dani ulaz. Ta razlika se potom kvadrira kako bi se dobila vrijednosti srednje kvadratne pogreške za samo jednu ulaznu vrijednost.

MSE(ulaz) = (izlaz – oznaka)(izlaz – oznaka)

Kada bi se modelu proslijedilo više uzoraka odjednom, serija uzoraka (eng. *batch of samples*), tada bi se uzele srednje kvadratne pogreške na svim uzorcima.

Ovaj primjer je ilustrirao matematiku jedne funkcije pogreške, MSE. Uz ovu funkciju pogreške, još ih se nekoliko koristi u praksi. Međutim, ova ideja računanja pogreške se na pojedinom uzorku se koristi i kod ostalih funkcija gubitaka. Implementacija toga što se zapravo čini sa svakom pogreškom će ovisiti o algoritmu dane funkcije gubitka. Na primjer, u ovom poglavlju se izračunao prosjek pogreške kvadrata kako bi se izračunao MSE, dok će druge funkcije gubitka koristiti neke druge algoritme kako bi se izračunala vrijednosti gubitka.

Kada bi se modelu proslijedio cijeli skup podataka za treniranje odjednom, tada bi se proces računanja gubitka događao na kraju svake epohe tijekom treniranja. Ako se skup podataka podijeli na manje skupove, serije podataka (eng. *batches*) i kada bi se ti skupovi jedan po jedan prosljeđivali modelu, gubitak bi se računao na kraju svake serije podataka.

S bilo kojom metodom, s obzirom da gubitak ovisi o težinama veza u modelu, očekuje se neka promjena vrijednosti gubitka svaki put nakon što se težine ažuriraju. Kako je cilj stohastičkog gradijentnog spusta minimizirati gubitak, očekuje se da se gubitak smanjuje sa svakom epohom.

## STOPA UČENJA U NEURALNOJ MREŽI

U ovom poglavlju će se objasniti što je to stopa učenja te će pokazati na koji se način ona koristi za treniranje neuralne mreže.

U jednom od prethodnih poglavlja je spomenuto što to znači kada neuralna mreža uči te se spomenulo da je stopa učenja broj s kojim pomnožimo gradijent funkcije gubitka.

Cilj treniranja modela jest pomoću SGD-a minimizirati gubitak između stvarne vrijednosti oznake i onoga što je model predvidio za dani uzorak za treniranje na ulazu. Put prema minimizaciji gubitka se odvija u nekoliko koraka.

Kao što je već prije spomenuto, proces treniranja započinje tako da se težinama veza dodjele neke proizvoljne vrijednosti koje se onda inkrementalno ažuriraju kako se gubitak približava svojoj minimalnoj vrijednosti.

Veličina koraka, kojom se gubitak približava svojoj minimalnoj vrijednosti, će ovisiti o stopi učenja. Konceptualno, o stopi učenja danog modela se može misliti kao o veličini koraka.

Zna se da se tijekom treniranja, nakon što se gubitak računa za svaki ulaz, gradijent gubitka izračunava u odnosu na svaku pojedinu težinu veze u danom modelu. Kada se izračuna svaki pojedini gradijent, oni će se pomnožiti sa stopom učenja.

gradijenti \* stopa učenja

Stopa učenja je, kao što je prije navedeno, broj koji se obično nalazi u rasponu od 0.01 i 0.0001. Naravno, stvarna vrijednost stope učenja može varirati i svaka će vrijednost gradijenta, nakon što se pomnoži sa stopom učenja, postati jako mala.

### AŽURIRANJE TEŽINA VEZA U MREŽI

Kada se dobije vrijednost množenja gradijenta i stope učenja, svaka pojedina vrijednost se koristi kako bi se ažurirala određena težina veze tako da se vrijednost umnoška oduzme od stare vrijednosti težine veze.

nova težina veze = stara težina veze – (gradijent \* stopa učenja)

Stara težina veze se odbacuje i umjesto nje se na vezu stavlja nova određena težina.

Vrijednost koja se dodjeljuje stopi učenja zahtjeva neko testiranje. Stopa učenja jest jedan od *hiperparametara* (eng. *hyperparameters*) koji se moraju testirati i ugoditi na svakom modelu posebno kako bi se odredilo s kojom će stopom učenja model najbolje učiti. Ali, kao što je prije spomenuto, obično se vrijednost stope učenja nalazi negdje između 0.01 i 0.0001.

Kada bi se stopa učenja postavila na neku vrijednost koja je bliže vrijednosti 0.01, postoji opasnost od mogućnosti dobivanja pretjeranog rezultata (eng. *overshooting*). To se događa kada se uzme stopa, u smjeru minimuma, koja je prevelika te ona preskače minimalnu vrijednost gubitka.

Kako bi se izbjeglo prebacivanje, vrijednost stope učenja se može postaviti na broj koji je bliže vrijednosti 0.0001. Na ovaj način, kako su koraci jako mali, postizanje minimalne vrijednosti gubitka će zahtijevati puno više vremena.

Čin izbora između više i niže vrijednosti stope učenja postaje idejom kompromisa.

## SKUPOVI ZA TRENIRANJE, TESTIRANJE I VALIDACIJU

### SKUPOVI PODATAKA ZA DUBOKO UČENJE

U ovom poglavlju će se objasniti razlika između različitih skupova podataka koji se koriste kod treniranja i testiranja neuralne mreže.

Za potrebe treniranja i testiranja odabranog modela, skup podataka se dijeli na tri različita dijela. Tako su skupovi podataka sljedeći:

* skup podataka za treniranje
* skup podataka za validaciju
* skup podataka za testiranje.

### SKUP PODATAKA ZA TRENIRANJE

Skup podataka za treniranje čine podaci na kojima se model trenira. Tijekom svake epohe, model će se iznova i iznova trenirati na tom istom skupu podataka za treniranje te će nastavljati učiti o značajkama tih podataka.

Cilj je na taj način naučiti model koji će poslije moći točno predvidjeti značajke podataka koje prije nije vidio. Model će svoje pretpostavke temeljiti na onome što je naučio na skupu podataka za treniranje.

### SKUP PODATAKA ZA VALIDACIJU

Skup podataka za validaciju je skup podataka koji je odvojen od skupa podataka za treniranje, ali se koristi tijekom treniranja kako bi se validirao model. Validacijski proces pomaže kod dobivanja informacija koje poslije mogu pomoći kod podešavanja hiperparametara. Tijekom svake epohe, model će učiti na skupu podataka za treniranje, ali će u isto vrijeme biti validiran na skupu podataka za validaciju.

Tijekom procesa treniranja, model će svaki ulaz iz skupa za treniranje preslikavati u određeni izlaz, odnosno klasu predviđanja. Kada model završi klasifikaciju svakog ulaza, izračunava se gubitak preko kojeg se podešavaju težine veza u modelu. Tijekom sljedeće epohe se isti ulazi ponovno klasificiraju u izlaze.

Također tijekom treniranja, model će istovremeno klasificirati svaki ulaz iz skupa za validaciju. Model će svoju validaciju temeljiti samo na onome što je naučio na temelju skupa podataka za treniranje. Težine veza se neće ažurirati na temelju gubitka iz skupa podataka za validaciju.

Kao što je već spomenuto, skup podataka za validaciju je odvojen od skupa podataka za treniranje. Tako, kada se model validira na temelju skupa podataka za validaciju, ti podaci se ne sastoje od uzoraka koje model već poznaje iz skupa podataka za treniranje.

Jedan od glavnih razloga zašto postoji skup podataka za validaciju jest taj da se model previše ne prilagodi skupu podataka za treniranje. Ta pojava se zove *ovefitting*. Ideja *overfitting*-a je ta da model postane jako dobar u klasificiranju podataka iz skupa za treniranje, ali nije sposoban generalizirati i ispravno klasificirati podatke na kojima nije bio treniran.

Ako se tijekom treniranja u isto vrijeme provjerava model pomoću skupa za validaciju i, ako su rezultati skupa za validaciju jednako dobri kao rezultati skupa za treniranje, velika je vjerojatnost da neće doći do pojave *overfitting*-a. Ako su model vrlo dobro klasificira podatke iz skupa za treniranje, a loše klasificira podatke iz skupa za validaciju, došlo je do pojave *overfitting*-a.

Pomoću skupa podataka za validaciju se provjerava koliko dobro model može generalizirati tijekom treniranja.

### SKUP PODATAKA ZA TESTIRANJE

Skup podataka za testiranje se koristi na modelu koji je istreniran. Skup podataka za testiranje je odvojen i od skupa za treniranje i od skupa za validaciju.

Nakon što je model treniran i validiran na skupovima za treniranje i validaciju, proslijedit će mu se skup podataka za testiranje, koji nije označen, za koji će trebati predvidjeti odgovarajući izlaz.

Glavna razlika između skupa za testiranje i druga dva skupa jest da skup za testiranje ne smije imati oznake. Skupovi za treniranje i validaciju moraju biti označeni kako bi se izračunao gubitak i točnost preslikavanja svake epohe. Ovako model prolazi kroz isti proces kao i kada bi model koristio podatke iz stvarnog svijeta.

Skup za testiranje pruža konačnu provjeru je li model dobro generalizira prije nego ga se stavi u produkciju.

Cilj treniranja, validiranja i testiranja modela jest taj da model dobro klasificira podatke koje prije nije vidio. Cilj dubokog učenja jest razviti modele koji mogu dobro generalizirati.

| Skupovi podataka u dubokom učenju | | |
| --- | --- | --- |
| **Skup podataka** | **Ažuriranje veza** | **Opis** |
| Skup za treniranje | Da | Koristi se za treniranje modela. Cilj treniranja jest da model dobro  klasificira podatke, ali u isto vrijeme dobro generalizira. |
| Skup za validaciju | Ne | Koristi se tijekom treniranja kako bi se provjerilo koliko dobro  model generalizira. |
| Skup za testiranje | Ne | Koristi se kako bi se konačno provjerila modelova sposobnost  generalizacije prije nego ga se stavi u produkciju |

Glavni razlog za imati tri različita skupa podataka jest osigurati da će model moći dobro generalizirati te dobro klasificirati podatke koje prije nije vidio. Ako model ne može dobro generalizirati, najčešće je došlo do pojave *overfitting*-a ili *underfitting*-a.

## PREDVIĐANJE NEURALNE MREŽE

U prethodnom poglavlju je objašnjeno što to znači trenirati mrežu. Kada je proces treniranja završen i ako je model postigao zadovoljavajuće rezultate na skupu za treniranje i validaciju, model se provjerava na skupu podataka za testiranje.

Za razliku od skupa podataka za treniranje i validaciju koji se modelu prosljeđuju zajedno s odgovarajućim oznakama, skup za treniranje se prosljeđuje modelu bez odgovarajućih oznaka.

### PROSLJEĐIVANJE UZORAKA BEZ OZNAKA

Kod predviđanja, modelu se prosljeđuje skup podataka za testiranje bez odgovarajućih oznaka te model mora sam predvidjeti izlaz za svaki uzorak iz skupa za testiranje. Predviđanje se temelji na onome što je model naučio tijekom treniranja.

Na primjer, neka je model treniran kako bi mogao klasificirati različite pasmine na temelju fotografija. Za svaki ulazni uzorak, donosno fotografiju, model predviđa vjerojatnost za odgovarajuću pasminu.

Neka se modelu proslijedi skup za testiranje u kojem se nalaze fotografije sa psima na kojima model nije bio treniran. Kao što je već spomenuto, model nema pristup oznakama za fotografije iz skupa za testiranje. Ovaj proces testiranja će pokazati koliko dobro model radi s podacima koje prije nije vidio na temelju toga koliko se dobro predviđanja modela preklapaju s pravim oznaka za ulazne podatke.

Također, ovaj će proces dati dobar uvid u to što model nije naučio. Na primjer, neka je model treniran samo na fotografijama velikih pasmina, dok skup za testiranje sadrži fotografije s malim pasminama. Kada se modelu proslijedi jedna fotografija s malim psom, on najvjerojatnije neće dobro predvidjeti odgovarajuću pasminu jer nije bio dobro treniran na malim pasminama.

To znači da se mora osigurati da skupovi za treniranje i validaciju dobro predstavljaju prave podatke na kojima će model trebati donositi predviđanja.

## *OVERFITTING* U NEURALNOJ MREŽI

Do *overfitting*-a dolazi kada model postane jako dobar u klasificiranju ili predviđanju podataka koji su bili uključeni u skup za treniranje, ali nije jednako dobar u klasificiranju podataka na kojima nije bio treniran. Tada se kaže da se model pretjerano poklapa s podacima iz skupa za treniranje.

### KAKO UOČITI PRETJERANO POKLAPANJE

Je li došlo do pretjeranog poklapanja se može uočiti na temelju metrike za dani skup za treniranje i skup za validaciju. Tijekom treniranja, dobiju se rezultati točnosti i gubitka za skup za validaciju kao i za skup za treniranje.

Ako su rezultati skupa za validaciju primjetno gori od skupa za treniranje, to je indikacija da se model pretjerano poklapa sa skupom za treniranje. Također se može primijetiti je li došlo do pretjeranog poklapanja ako su rezultati treniranja jako dobri, dok model neispravno klasificira podatke iz skupa za testiranje.

Koncept pretjeranog poklapanja se svodi na činjenicu da model ne može dobro generalizirati podatke. Model je jako dobro naučio značajke podataka iz skupa za treniranje, ali, ako mu se daju podaci koji se neznatno razlikuju od podataka iz skupa za treniranje, model ne može dobro generalizirati i predvidjeti odgovarajući izlaz.

### SMANJENJE PRETJERANOG POKLAPANJA

#### UBACIVANJE PODATAKA U SKUP ZA TRENIRANJE

Najjednostavnija stvar koja se može napraviti, ako je moguće, jest ubaciti još podataka u skup za treniranje. Na što se više podataka trenira model, on će moći više naučiti. Također, s više podataka se dodaje veća raznolikost skupu za treniranje.

#### POVEĆANJE PODATAKA

Još jedan način pomoću kojeg se može reducirati pretjerano poklapanje jest povećanje podataka. To je proces kojim se stvaraju dodatni, izmijenjeni podaci tako što se razumno modificiraju podaci iz skupa za treniranje. Kod slikovnih podataka, podaci se mogu modificirati na sljedeće načine:

* obrezivanjem
* rotiranjem
* okretanjem
* zumiranjem

Glavna ideja povećanja podataka jest to da se u skup za treniranje ubacuju podaci koji su slični već postojećim podacima, samo što su oni razumno izmijenjeni do neke mjere tako da nisu potpuno isti.

#### REDUCIRANJE SLOŽENOSTI MODELA

Nešto što se može napraviti kako bi se smanjilo pretjerano poklapanje jest smanjiti složenost modela. Složenost se može smanjiti tako što se može ukloniti pojedini sloj iz modela ili se može smanjiti broj neurona u sloju. Na taj način model može bolje generalizirati podatke koje prije nije vidio.

#### ISPUŠTANJE

Ideja iza ispuštanja (eng. *dropout*) jest ta da se nasumično ignoriraju podskupovi neurona u sloju. Ta radnja će spriječiti ispuštene neurone da sudjeluju u predviđanju podataka.

Ova tehnika također može pomoći modelu bolje generalizirati podatke koje do tada nije vidio.

## NEDOVOLJNO POKLAPANJE U NEURALNOJ MREŽI

U ovom poglavlju će se objasniti što to znači kada se model nedovoljno poklapa (eng. *underfitting*). Također će se navesti tehnike pomoću kojih se može reducirati nedovoljno poklapanje kada do njega dođe.

Kaže se da se model nedovoljno poklapa kada nije sposoban niti klasificirati podatke na kojima je treniran, a pogotovo podatke koje do tada nije vidio. Model tada ima loše rezultate na skupu podataka za treniranje, ispravnost preklapanja je niska, a gubitak je velik. Ako model nije sposoban klasificirati podatke na kojima je bio treniran, najvjerojatnije neće dobro klasificirati podatke koje do tada nije vidio.

### REDUCIRANJE NEDOVOLJNOG POKLAPANJA

#### POVEĆANJE SLOŽENOSTI MODELA

Način na koji se može reducirati nedovoljno poklapanje jest povećati složenost modela. To je tehnika suprotna od one kojom se reducira pretjerano poklapanje. Ako su podaci u skupu na kojem treniramo model složeni, a model je relativno jednostavan, model najvjerojatnije neće biti dovoljno sofisticiran kako bi mogao ispravno klasificirati složene podatke.

Složenost modela se može povećati na sljedeće načine:

* povećanjem broja slojeva u modelu
* povećanjem broja neurona u svakom sloju
* promjenom tipa i mjesta sloja koji se koristi u modelu.

#### DODAVANJE OZNAKA ULAZNIM UZORCIMA

Jedna od tehnika kojom možemo reducirati nedovoljno poklapanje jest dodavanjem značajki ulaznim uzorcima, ako je to moguće, iz skupa za treniranje. Te dodatne značajke mogu pomoći modelu kako bi ispravnije klasificirao ulazne podatke.

Na primjer, neka model pokušava predvidjeti cijene dionice na temelju njene cijene prilikom zatvaranja u posljednja tri dana. Ulazni podaci bi se sastojali od sljedećih značajki:

* cijena na kraju prvog dana
* cijena na kraju drugog dana
* cijena na kraju trećeg dana.

Kada bi se dodale dodatne značajke ovim podacima, kao na primjer cijene dionice prilikom otvaranja burze, možda bi to pomoglo modelu da ispravnije klasificira podatke.

#### REDUCIRANJE ISPUŠTANJA

Kod korištenja ispuštanja (eng. *dropout*), može se odrediti koliki će se postotak neurona ili čvorova ispustiti iz mreže. Na primjer, ako se koristi stopa ispuštanja od 50%, i model se nedovoljno poklapa, potrebno je smanjiti postotak ispuštenih neurona tako da taj postotak bude manji od 50%.

Neuroni, koji su ispušteni, ispušteni su samo za potrebe treniranja, ali nisu ispušteni tijekom validacije modela. Tako, ako model bolje preslikava podatke iz skupa za validaciju od skupa za treniranje, tada je dobro smanjiti postotak ispuštenih neurona.

## NADZIRANO UČENJE KOD STROJNOG UČENJA

### OZNAČENI PODACI

Kod nadziranog učenja, podaci su u skupu za treniranje označeni. Oznake se koriste kako bi se nadzirao i usmjeravao proces učenja.

Kako je spomenuto u jednom od prethodnih poglavlja, podaci u skupu za treniranje i u skupu za validaciju imaju svoje oznake te su, zajedno, prosljeđeni modelu. U ovom slučaju se radi o nadziranom učenju,

Kod nadziranog učenja, svaki podatak, koji se prosljeđuje modelu tijekom treniranja, je par koji se sastoji od ulaznog objekta, ili uzorka, i odgovarajuće oznake (eng. *label*), ili izlazne vrijednosti. Ono što je bitno, kod nadziranog učenja, jest to da model uči kako preslikavati dani ulaz u odgovarajuće izlaze na temelju toga što je naučio iz označenih podataka za treniranje.

Na temelju onoga što je spomenuto u poglavlju o treniranju modela, model će preslikati dani ulaz u određeni izlaz te će onda procijeniti pogrešku za taj ulaz tako što će izračunati razliku između vrijednosti koju je previdio i prave oznake za taj ulaz.

### OZNAKE SU NUMERIČKE

Oznake, koje se pridružuju ulaznim uzorcima, enkodiraju se u nešto numeričko (0, 1, 2…)

Nakon toga se prolazi kroz proces utvrđivanja pogreške ili gubitka za sve podatke iz skupa za treniranje za svaku definiranu epohu. Cilj treniranja jest minimizirati gubitak tako da model može ispravno predviđati na podacima na kojima nije bio treniran. Model će svoje pretpostavke temeljiti na označenim podacima koje je vidio tijekom treniranja.

## POVEĆANJE PODATAKA ZA STROJNO UČENJE

U ovom poglavlju će se obraditi proces povećanja podataka te će se spomenuti u kojim slučajevima je to dobro napraviti.

Povećanje podataka se obavlja kada se žele stvoriti novi podaci tako što se izmijene postojeći podaci. Stvaraju se novi podaci tako da se razumno izmijene postojeći podaci iz skupa za treniranje.

Na primjer, fotografija se može izmijeniti tako da se:

* okrene horizontalno
* okrene vertikalno
* rotira
* poveća
* smanji
* obreže
* promijeni boja

### ZAŠTO POVEĆATI PODATKE

Jedan od razloga zašto povećati količinu podataka jest taj da se dodaje više podataka u skup za treniranje.

Još jedan razlog je taj da se smanji pretjerano poklapanje. Ako se model pretjerano poklapa s podacima za treniranje, u skup za treniranje se na taj način doda više podataka.

## ONE-HOT ENKODIRANJE ZA STROJNO UČENJE

U ovom poglavlju će se opisati enkodiranje u strojnom učenju koje se zove *one-hot encoding*.

### OZNAKE

Kada se neki model trenira pomoću nadziranog učenja, modelu se proslijedi neki označeni ulaz te model na to daje neki predviđeni izlaz.

Na primjer, neka model klasificira ulazne fotografije. Modelu se kao ulaz prosljeđuju fotografije životinja s odgovarajućim oznakama. Model te oznake ne interpretira kao riječi, na primjer 'mačka' ili 'pas'. Isto tako, model na izlazu ne daje predviđanja u obliku određenih riječi kao što su 'mačka' ili 'pas'. Naime, većinu puta su oznake enkodirane tako da mogu poprimiti oblik cijelog broja ili vektora cijelih brojeva.

### TOPLE I HLADNE VRIJEDNOSTI

Jedan tip enkodiranja, koji se često koristi u kategoričkim podacima s numeričkim vrijednostima, jest *one-hot encoding*.

*One-hot* enkodiranje transformira kategoričke oznake u vektore s nulama i jedinicama. Duljina vektora jest broj klasa ili kategorija koje se očekuju da ih model navede.

|  |  |
| --- | --- |
| **Vrijednost** | **Interpretacija** |
| 0 | Hladno (*cold*) |
| 1 | Toplo (*hot*) |

### VEKTORI S NULAMA I JEDINICAMA

Kada bi model trebao predvidjeti dvije moguće klase predviđanja, tada bi duljina *one-hot* enkodiranog vektora odgovarala tom broju klasa. Odnosno, svaka od klasa predviđanja bi bila vektor duljine dva za svaku odgovarajuću klasu.

Na primjer, neka model treba klasificirati slike psa, mačke i guštera. Tada bi odgovarajući *one-hot* enkodirani vektori svaki bili duljine tri jer model treba ulaze klasificirati u tri kategorije.

### ONE-HOT ENKODIRANJE ZA VIŠE KATEGORIJA

Na temelju primjera iz prošlog poglavlja, neka model treba klasificirati ulazne fotografije na one mačke, psa i guštera. Svaki odgovarajući vektor za svaki pojedini izlaz ima duljinu tri. Svaki indeks, odnosno svaki element u vektoru odgovara jednoj od tri moguće kategorije.

Neka, na primjer, oznaka za mačku odgovara prvom elementu, oznaka za psa drugom i oznaka za guštera trećem elementu u vektoru. Svaka od kategorija ima svoje mjesto u odgovarajućem vektoru.

U svakom *one-hot* enkodiranom vektoru, svaki će element imati vrijednost nula osim onog elementa koji odgovara stvarnoj kategoriji za dani ulaz. Ovaj element će biti 'vruća jedinica' (eng. *hot one*).

Kao što je već spomenuto, oznaka za mačku odgovara prvom elementu u vektoru, oznaka za psa drugom i oznaka za guštera trećem elementu u vektoru. Svaka od ovih kategorija bi izgledala na sljedeći način:

|  |  |  |  |
| --- | --- | --- | --- |
| **Oznaka** | **Indeks-0** | **Indeks-1** | **Indeks-2** |
| Mačka | 1 | 0 | 0 |
| Pas | 0 | 1 | 0 |
| Gušter | 0 | 0 | 1 |

Kod oznake za mačku, prvi element u vektoru će imati vrijednost jedan, dok će ostali imati vrijednost nula. To je zato što je svaki element u danom vektoru nula osim onog koji odgovara stvarnoj kategoriji, a kao što je već spomenuto, kategorija mačke odgovara prvom elementu.

### JEDAN VEKTOR ZA SVAKU KATEGORIJU

Kao što se može vidjeti, svaki put kada model primi ulaz koji odgovara mački, model tu vrijednost ne interpretira kao riječ 'mačka'. Model taj ulaz interpretira tu oznaku kao vektor [1, 0, 0].

|  |  |
| --- | --- |
| **Oznaka** | **Vektor** |
| Mačka | [1,0,0] |
| Pas | [0,1,0] |
| Gušter | [0,0,1] |

## DUBOKO UČENJE KOD KONVOLUCIJSKIH NEURALNIH MREŽA

Konvolucijska neuralna mreža, još poznata i kao CNN ili *ConvNet*, jest umjetna neuralna mreža koja se najčešće koristi za analiziranje fotografija za potrebe računalnog vida.

Konvolucijska neuralna mreža je specijalizirana za otkrivanje obrazaca koje je jako korisno u analizi fotografija. Ta mreža je, između ostalog, građena od skrivenih slojeva koji se nazivaju 'konvolucijski slojevi' (eng. *convolution layers*).

Kao i bilo koji drugi sloj, konvolucijski sloj prima ulaz koji potom transformira na neki određeni način te ga šalje sljedećem sloju. Ulazi u konvolucijski sloj se zovu 'ulazni kanali' (eng. *input channels*), a izlazi se zovu 'izlazni kanali' (eng. *output channels*).

Operacija kojom konvolucijski sloj transformira ulaz se zove 'konvolucijska operacija' (eng. *convolution operation*). Konvolucijske operacije koje izvodi konvolucijski sloj se u matematici zovu 'unakrsnim korelacijama' (eng. *cross-correlations*).

### FILTERI I KONVOLUCIJSKE OPERACIJE

Kao što je već spomenuto, konvolucijske neuralne mreže mogu otkrivati obrasce na fotografijama. U svakom konvolucijskom sloju se mora odrediti broj filtera koji svaki sloj mora imati. Filteri su ti koji otkrivaju obrasce.

### PATTERNS

Obrasce u slici ili fotografiji čine razne stvari kao što su:

* rubovi
* oblici
* teksture
* krivine
* objekti
* boje

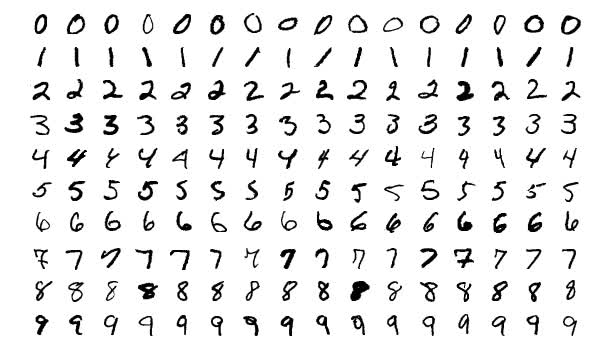
Jedan tip obrasca koji filter može otkriti na fotografiji su rubovi. Tako bi se filter za otkrivanje rubova zvao 'detektor rubova' (eng. *edge detector*).

Osim rubova, neki filteri mogu otkrivati kutove. Neki filteri mogu otkrivati krugove ili pravokutnike. Ti jednostavni filteri se pojavljuju na početku konvolucijske neuralne mreže.

Što se putuje dulje u mrežu, filteri u mreži postaju sve sofisticiraniji. U kasnijim slojevima, umjesto rubova i jednostavnih oblika, filteri mogu raspoznavati specifične objekte kao što su oči, uši, usta… U još dubljim slojevima filteri mogu raspoznavati još sofisticiranije objekte kao što s lica, ptice, automobile…

### FILTERI (DETEKTORI OBRASCA)

Neka konvolucijska mreža prima fotografije ručno pisanih znamenki (MNIST skup podataka) te da ih mreža klasificira u odgovarajuće kategorije fotografija s brojevima 1, 2, 3 itd.



Slika : skup ručno pisanih znamenki

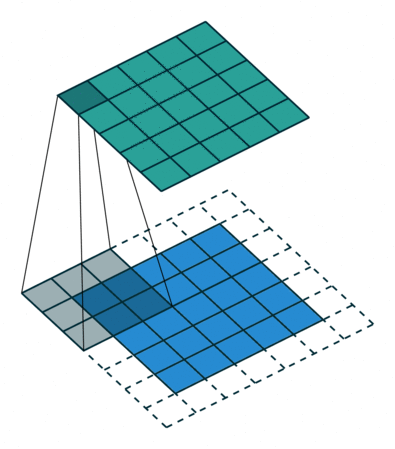
Neka se pretpostavi da je prvi skriveni sloj u danom modelu konvolucijski sloj. Kao što je prije spomenuto, s dodavanjem konvolucijskog sloja, potrebno je specificirati koliko će taj sloj imati filtera.

Broj filtera određuje broj izlaznih kanala.

Filter može biti relativno mala matrica, još poznata i kao tenzor (eng. *tensor*), kojoj se određuje broj redova i stupova. Svakoj ćeliji u matrici se pridjeljuje neka nasumična brojčana vrijednost.

Neka matrica u prvom kovolucijskom sloju ima dimenzije 3x3.

### KONVOLUCIJSKI SLOJ



Slika : prelazak filtera preko ulazne matrice

Slika 6 prikazuje konvolucijski proces bez brojeva. Ulazni kanal u konvolucijski sloj čini plava matrica. Sjena na ulaznom kanalu označava filter u konvolucijskom sloju koji klizi preko ulaznog kanala te zelena matrica označava izlazni kanal.

* Plava matrica (dno): ulazni kanal
* Sjena na plavoj matrici: 3x3 konvolucijski filter
* Zelena matrica (gore): izlazni kanal

Za svaku poziciju na plavom ulaznom kanalu, konvolucijski filter dimenzija 3x3 vrši proračun koji se preslikava na odgovarajuću poziciju na zelenom kanalu.

Ovaj konvolucijski sloj prima ulazni kanal, filter konvolucijskog sloja klizi preko svakog skupa ulaznih piksela dimenzija 3x3 dok ne prijeđe preko svakog 3x3 bloka piksela cijele ulazne fotografije.

### KONVOLUCIJSKA OPERACIJA

Klizanje filtera preko ulaznog kanala se zove konvolucija (eng. *convolving*) te se kaže da filter klizi preko svakog 3x3 bloka piksela na ulaznoj fotografiji.

Neka je plavi ulazni kanal matrična reprezentacija fotografije iz MNIST skupa podataka. Vrijednosti u ovoj matrici su individualni pikseli na slici. Fotografije iz ovog skupa podataka su slike u sivim tonovima te stoga imaju samo jedan ulazni kanal.

* Fotografije u sivim tonovima imaju jedan ulazni kanal
* RGB fotografije imaju tri ulazna kanala (jedan za svaku boju)

Ovaj ulaz se prosljeđuje konvolucijskom sloju.

Kao što je već prije navedeno, prvi konvolucijski sloj posjeduje samo jedan filter koji klizi preko svakog 3x3 bloka piksela ulazne matrice. Kada se filter spusti na prvi 3x3 blok piksela, izračunava se točkasti produkt (eng. *dot product*) filtera i tog bloka te se dobivena vrijednost sprema na prvo mjesto u matrici izlaznih vrijednosti. Isti ovaj proces se ponavlja za svaki 3x3 blok na koji se filter spusti. Na primjer, filter, nakon što izračuna vrijednost prvog točkastog produkta, klizi do sljedećeg 3x3 bloka piksela za koji se opet računa točkasti produkt koji se sprema na drugo mjesto u matrici ulaznih vrijednosti i tako za sve 3x3 blokove piksela.

Kada filter prođe preko cijelog ulaza, dobit će se nova reprezentacija ulaza koja je spremljena u izlazni kanal. Taj izlaz se zove 'matrica značajki' (eng. *feature map*).

Zelena matrica na slici 6, koja označava izlaz iz prvog konvolucijskog sloja, postaje ulazni kanal za sljedeći sloj u mreži. Ako je taj sljedeći sloj također konvolucijski sloj, gore opisani proces se ponavlja.

### NAPOMENA O KORIŠTENJU TOČKASTOG PRODUKTA

Termin 'točkasti produkt' se koristi sa zadrškom. Taj termin se koristi kako bi se opisala gore navedena operacija. Ono što se zapravo radi jest zbrajanje produkata po elementima svakog para iz dvije matrice.

Na primjer, neka postoje dvije matrice A i B:

Zbrajanje po elementima se čini na sljedeći način:

Tako da je, tehnički, gore opisana operacija, operacija zbrajanja produkata po elementima. U literaturi se ipak češće nailazi na termin 'točkasti produkt'. Razlog je taj što je ovdje operacija prikazana kao unutarnji produkt koji je generalizacija točkastog produkta. Ova se operacija u literaturi još može naći pod nazivom 'Frobeniusov unutarnji produkt' (eng. *Frobenius inner product*) ili 'sumacija Hadamardovog produkta' (eng. *the summation* *of the Hadamard product*).

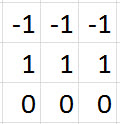
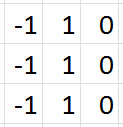
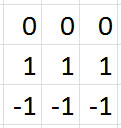
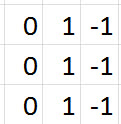
### ULAZNI I IZLAZNI KANALI

Neka je dolje prikazana slika (u sivim tonovima) broja sedam, iz MNIST skupa podataka, ulaz u zadani model:



Slika : broj sedam iz MNIST skupa podataka

Neka model u svom prvom konvolucijskom sloju raspolaže sa četiri filtera dimenzija 3x3 i neka su popunjeni na sljedeći način.

Slika : prvi filter Slika : drugi filter Slika : treći filter Slika : četvrti filter

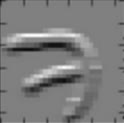
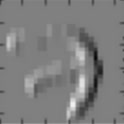
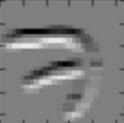
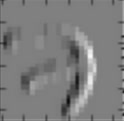
Vrijednosti filtera su postavljene tako da broj -1 odgovara crnoj boji, broj 1 odgovara bijeloj boji, a broj 0 odgovara sivoj boji.

Tako bi filteri odgovarali sljedećim slikama u sivim tonovima:

Slika : slike odgovaraju gore navedenim filterima po redu

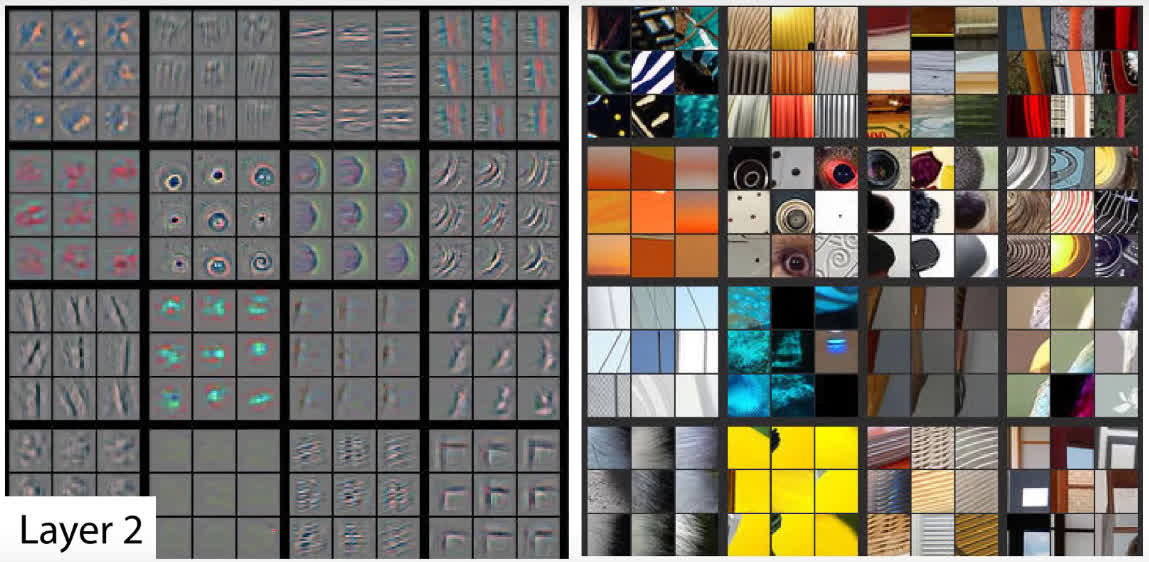
Kada bi se sa svakim gore prikazanim filterom pojedinačno prešlo preko ulazne slike, izlaz iz svakog pojedinog filtera bi izgledao na sljedeći način:

Slika : rezultat svakog filtera po redu

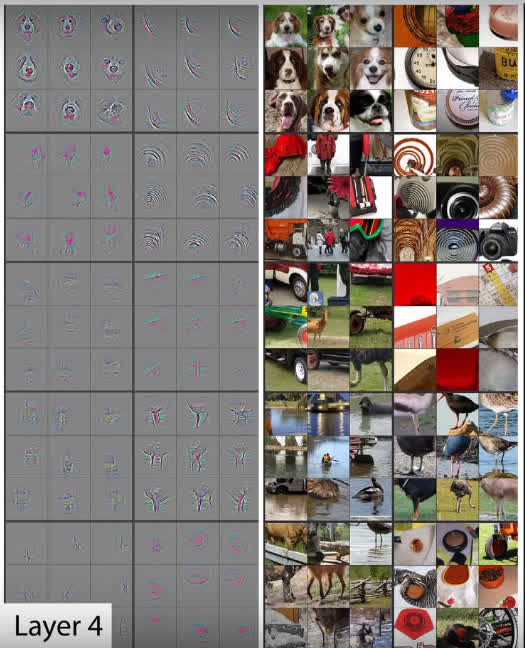
Kao što se može primijetiti, svaki od ova četiri filtera detektira rubove. U izlaznim kanalima, najosvjetljeniji pikseli će biti protumačeni kao ono što je filter detektirao. Kao što je prikazano na slikama, prvi filter detektira gornje horizontalne rubove. Drugi filter detektira lijeve vertikalne rubove, treći filter detektira donje horizontalne rubove, a četvrti filter detektira desne vertikalne rubove.

Ovi su filteri, kao što je već spomenuto, osnovni filteri koji samo detektiraju rubove. Takvi su filteri koji se mogu vidjeti u prvim konvolucijskim slojevima neuralnih mreža. Složeniji filteri se nalaze dublje u mreži te postepeno mogu detektirati sofisticiranije uzorke kao što je prikazano na donjoj slici:



Slika : filteri koji se nalaze dublje u mreži mogu detektirati sofisticiranije uzorke

Kao što se može primijetiti na slici, filteri, koji se nalaze na lijevom dijelu slike, su detektirali krugove, iskrivljene linije i uglove s desnog dijela slike. Filteri, koji se nalaze još dublje u mreži, mogu detektirati još složenije uzorke kao što su lica psa, oči, noge ptica što je prikazano na donjoj slici.



Slika : filteri koji se nalaze jako duboko u mreži mogu detektirati jako složene uzorke

Ono što je dobro spomenuti jest to da su filteri koji detektiraju uzorke automatski izvedeni od strane mreže. Vrijednosti filtera su inicirani s nasumičnim vrijednostima koje se mijenjaju kako mreža uči tijekom procesa treniranja. U prošlosti su eksperti za računalni vid morali ručno razvijati filtere za detekciju uzoraka. Primjer jednog takvog filtera jest Sobel filter.

## POPUNJAVANJE NULAMA

Ovo poglavlje će započeti objašnjenjem motivacije iza popunjavanja nulama (eng. *zero padding*). Nakon toga će se ući u detalje što je to zapravo popunjavanje nulama. Na kraju će se nabrojati problemi do kojih se može doći ako se ne koristi popunjavanje nulama.

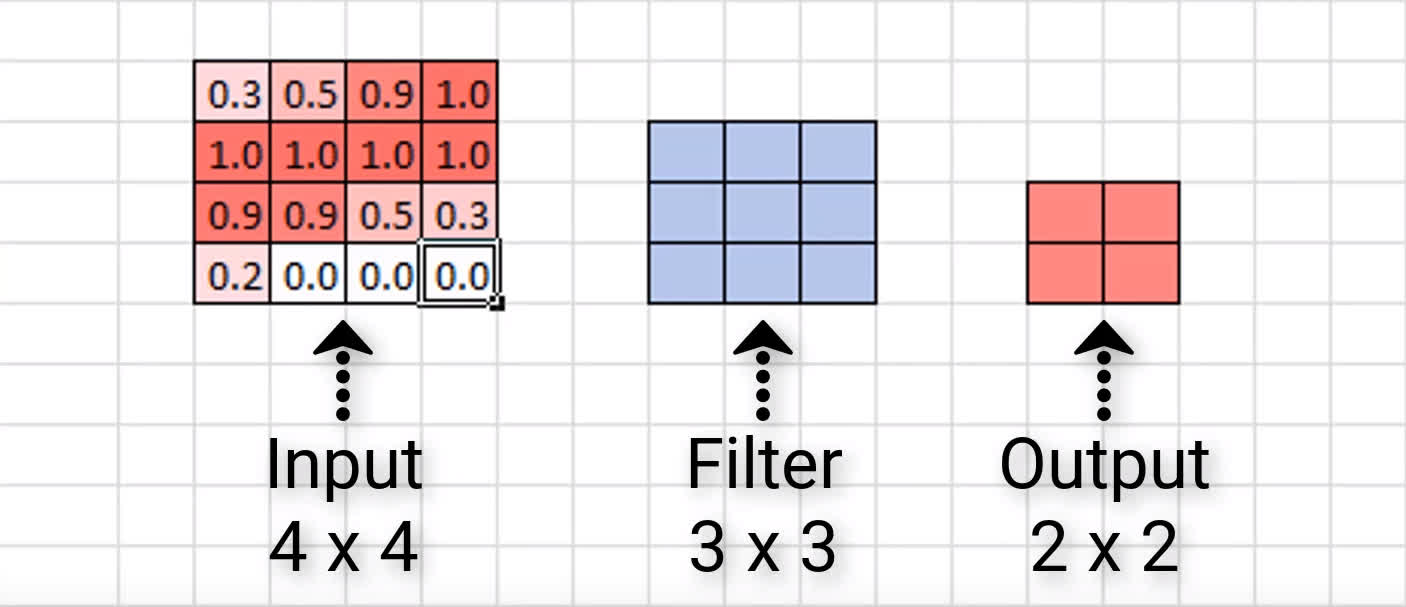
### KONVOLUCIJE SMANJUJU DIMENZIJE KANALA

Kao što je rečeno u prethodnom poglavlju, svaki konvolucijski sloj u konvolucijskoj neuralnoj mreži posjeduje određen broj definiranih filtera s definiranim dimenzijama. Također je prikazano na koji način filteri klize po slici (ulazu u konvolucijski sloj) kako bi stvorili izlaz.

Kada filter klizi po danom ulaznom kanalu, vraća izlazni kanal. Izlazni kanal jest matrica piksela s vrijednostima koje su izračunate tijekom konvolucijskog procesa na danom ulaznom kanalu.

Kada se ovo dogodi, dimenzije ulaznog kanala se reduciraju.

Kako bi se to bolje objasnilo, dan je jednostavan primjer s manjim dimenzijama. Na dolje prikazanoj slici ulazni kanal čini matrica dimenzija 4x4. Konvolucijski sloj posjeduje jedan filter dimenzija 3x3. Izlazni kanal čini matrica dimenzija 2x2.



Slika : prikaz redukcije dimenzija ulaznog kanala

To znači da, ako je ulazni kanal dimenzija nxn i, ako se na njemu izvodi konvolucijska operacija s filterom dimenzija fxf, dimenzije rezultirajućeg izlaznog kanala će odgovarati rezultatu matematičke operacije: .

### PROBLEMI S REDUCIRANJEM DIMENZIJA

Moglo bi se pomisliti da se nije izgubilo mnogo podataka jer su najvažniji podaci smješteni po sredini ulaznog kanala. Međutim, problem bi nastao kada bi se važni podaci nalazi po rubovima ulaznog kanala.

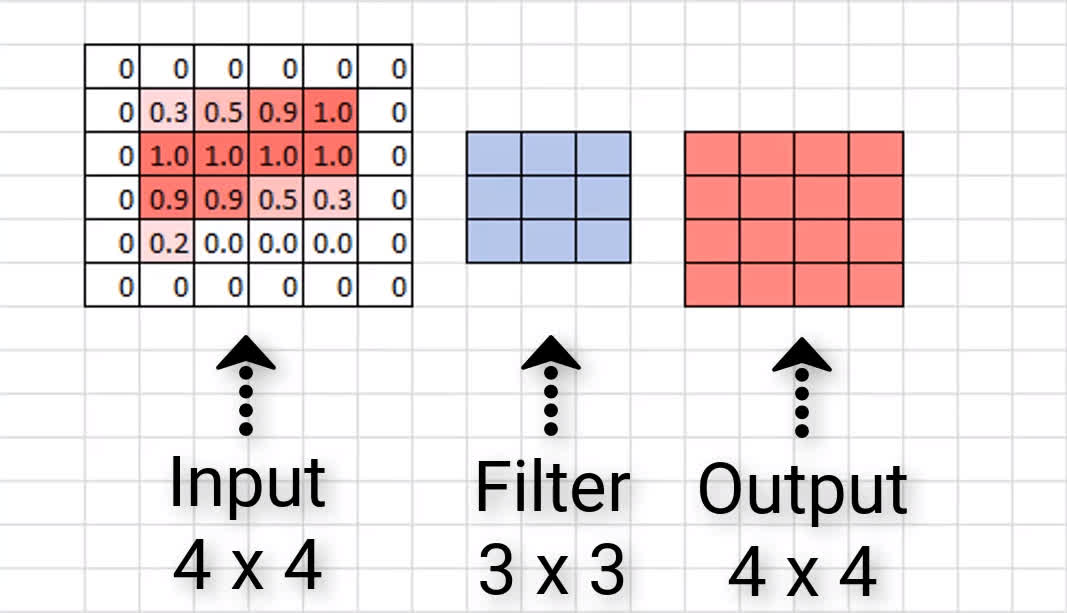
Nadalje, ulazni kanal je konvoluiran samo jednim filterom. Ali, kada bi ulazni kanal prolazio mrežom s više konvolucijskih slojeva i bio konvoluiran od strane više filtera, izlazni kanal iz svakog konvolucijskom sloja bi bivao sve manjim i to je problem.

Kada bi, na primjer, ulazni kanal činila slika dimenzija 4x4, tada bi nakon samo jednog ili dva konvolucijska sloj rezultirajući izlaz postao skoro pa besmislen zbog svoje male dimenzije. Još jedan problem jest taj što bi se tada izgubili vrijedni podaci jer bi , na ovaj način, podaci, koji se nalaze po rubovima ulaznog kanala, izbacili prilikom konvolucije.

### POPUNJAVANJE NULAMA

Popunjavanje nulama (eng. *zero padding*) je tehnikama kojom se drži očuvanom orginalna veličina ulaznog kanala. Ta tehnika se specificira kod svakoj konvolucijskog sloja posebno. Sa svakim konvolucijskim slojem definira se broj filtera, njihova veličina te se definira hoće li se koristiti popunjavanje nulama ili ne.

Popunjavanje nulama se odvija tako da se na rubovima ulaznog kanala (u ovom slučaju ulaznoj slici) dodaje sloj piksela koji imaju vrijednost nula. Ova tehnika daje neku vrstu popune nulama vanjskoj strani slike. Zbog toga se ova tehnika naziva 'popunjavanje nulama'. Vraćajući se na posljednji primjer, kada bi se vanjska strana ulazne matrice popunila pikselima koji imaju vrijednost nula, rezultirajući izlaz će imati iste dimenzije kao i ulazna matrica.



Slika : rezultat primjene tehnike 'popunjavanje nulama'

Kao što se vidi na gore prikazanoj slici, izlaz kanal doista ima dimenzije 4x4 čime ostaje sačuvana dimenzija ulaznog kanala. Naravno, nekada će biti dovoljno dodati jedan sloj piksela vrijednosti nula, dok će u nekim slučajevima (kada se radi s filterima većih dimenzija) biti potrebno dodati više slojeva piksela.

### *VALID* POPUNJAVANJE *I SAME* POPUNJAVANJE

Postoje dvije kategorije kod popunjavanja nulama, *valid* popunjavanje i *same* popunjavanje. *Valid* popunjavanje samo znači da nema popunjavanja. Kada se koristi *valid* popunjavanje, to znači da se ulazni kanal neće popuniti nulama te da se veličina ulaznog kanala neće uspjeti održati.

*Same* popunjavanje se odnosi na to da se želi sačuvati veličina originalnog ulaznog kanala. U tom slučaju će se vanjski rubovi ulaznog kanala, prije konvoluiranja, popuniti slojem (ili slojevima) piksela kako bi se sačuvala originalna veličina ulaznog kanala.

|  |  |  |
| --- | --- | --- |
| **Način popunjavanja** | **Opis** | **Utjecaj** |
| *Valid* | Nema popunjavanja | Reduciranje dimenzija ulaznog kanala |
| *Same* | Popunjavanje nulama oko  rubova ulaznog kanala | Očuvanje dimenzija ulaznog kanala |

## *MAX POOLING* U KONVOLUCIJSKIM NEURALNIM MREŽAMA

*Max pooling* je vrsta operacije koja se obično dodaje konvolucijskoj neuralnoj mreži iza pojedinog konvolucijskog sloja. Kada se *max pooling* doda modelu, on reducira dimenzije ulaznog kanala (na primjer slike) na način da reducira broj piksela izlaznog kanala iz prethodnog konvolucijskog sloja.

### PRIMJER *MAX-POOLING-A* NA JEDNOM UZROKU IZ MNIST SKUPA PODATAKA

Kao što je opisano u jednom od prethodnih poglavlja, svaki konvolucijski sloj raspolaže nekim brojem definiranih filtera s definiranim dimenzijama i ti filteri konvoluiraju po ulaznom kanalu (slici).

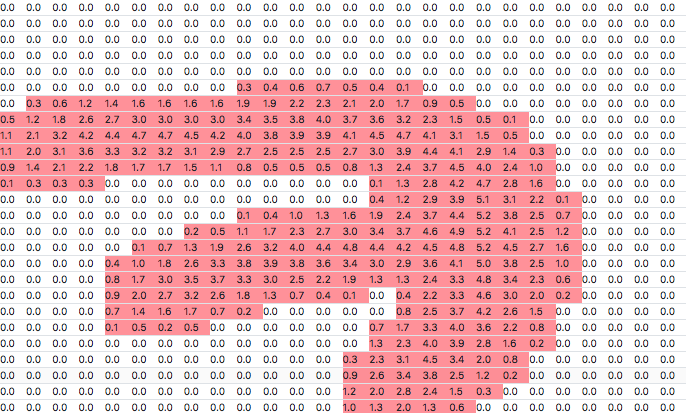
Kada filter konvoluira po ulaznom kanalu, on na izlazu daje izlazni kanal. Taj izlazni kanal je matrica piksela s vrijednostima koje su izračunate tijekom konvolucijskog procesa.

U ovom primjeru će se već prije prikazana slika broja 7 iz MNIST skupa podataka prikazati kao matrica vrijednosti piksela i s dimenzijama 28x28.



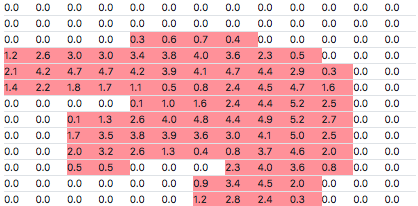
Slika : broj sedam iz MNIST skupa podataka u dimenziji 28x28 kao ulazni kanal

Po ovom ulaznom kanalu konvoluira filter konvolucijskog sloja s dimenzijama 3x3 te stvara sljedeći izlazni kanal s dimenzijama 26x26:



Slika : izlazni kanal dimenzija 26x26

Kao što je već spomenuto, *max-pooling* je implementiran nakon konvolucijskog sloja. Izlazni kanal iz konvolucijskog sloja je ulaz *max pooling* operacije. Nakon primjene *max pooling* operacije, izlazni kanal izgleda na ovaj način:



Slika : izlazni kanal nakon max pooling operacije ima dimenzije 13x13

*Max pooling* operacija se izvodi na sljedeći način: definira se neka regija dimenzije nxn kao odgovarajući filter za *max pooling* operaciju. U ovom primjeru će taj filter biti dimenzija 2x2.

Uz dimenzije filtera, definira se i korak (eng. *stride*) koji određuje za koliko će se piksela pomaknuti filter kada prelazi preko slike. Neka to u ovom primjeru bude broj dva.

Na izlazu iz konvolucijskog sloja se uzima prva regija veličine 2x2 te se računa njena maksimalna vrijednost. Maksimalna vrijednost jednaka je maksimalnoj vrijednosti iz odabrane 2x2 regije (izlaza iz konvolucijskog sloja). Ta vrijednost se sprema u izlazni kanal što čini puni izlaz iz *max pooling* operacije.

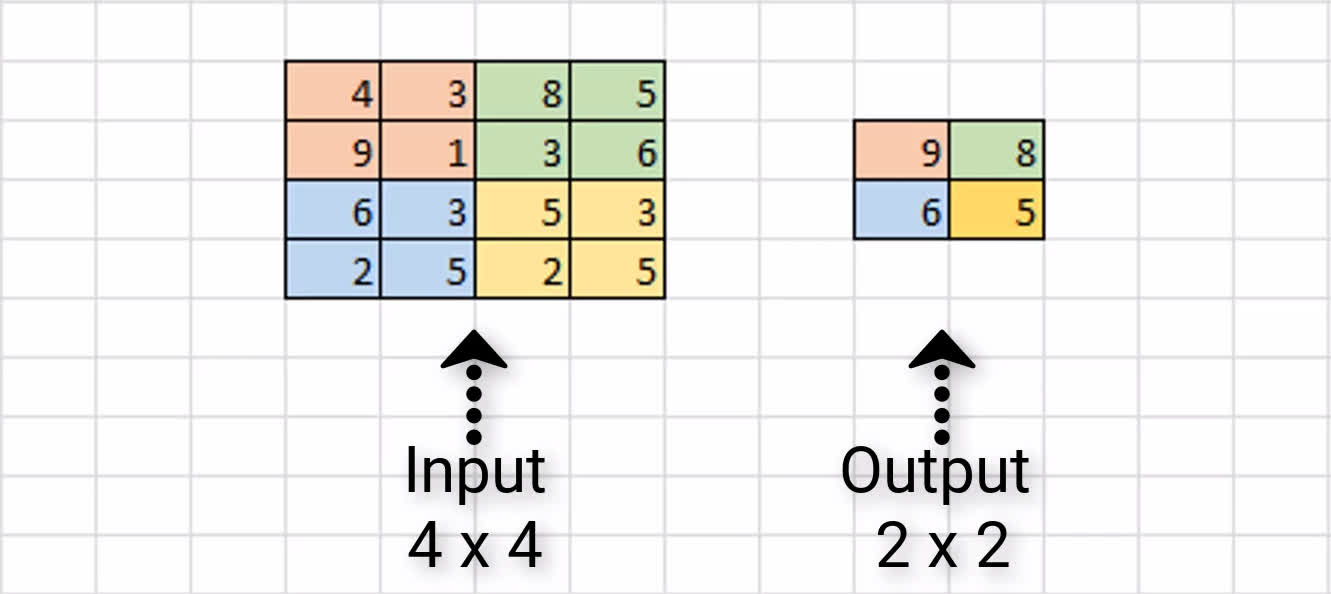
Nakon što se izračuna prva vrijednost, filter *max pooling* operacije se pomiče za već prije definirani broj koraka. Kako je to već prije odlučeno, ovdje će se filter pomaknuti za dva koraka udesno. Na toj novoj regiji se ponovno računa maksimalna vrijednost koja se potom sprema na odgovarajuće mjesto na izlaznom kanalu. Proces se ponavlja do kraja reda ulaznog kanala. Kada se dođe do kraja reda ulaznog kanala, filter se vraća na početak ulaza i spušta se za isti broj koraka koji je već definiran. Cijeli proces računanja operacije *max pooling*, pomicanja filtera i spremanja vrijednosti se ponavlja sve dok filter ne dođe do donjeg desnog kuta ulaznog kanala. Proces *max pooling*-a je tada završen i dobivena je nova prezentacija izlaznog kanala iz konvolucijskog sloja.

Ovaj blok, odnosno filter, dimenzije 2x2 se može usporediti s 'bazenom' (eng. *pool*) brojeva iz kojeg se izvlači najviše vrijednost. Iz ove slikovne reprezentacije operacije je došao naziv *max pooling*.

U gore navedenom primjeru izlaz iz konvolucijskog sloja ima dimenzije 26x26. Nakon operacije *max pooling*, dimenzije slike su se reducirale za faktor 2 te sada nova dimenzija iznosi 13x13.

### UMANJENI PRIMJER

Neka vrijedi sljedeće:



Slika : umanjeni primjer max pooling operacije

Na slici je prikazan uzorak ulaznog kanala veličine 4x4. Pretpostavljeni su filter veličine 2x2 i korak veličine 2 kao parametri *max-*pooling operacije.

Prva regija 2x2 obojena je u narančasto. Maksimalna vrijednost te regije iznosi 9 i ta je vrijednost spremljena na prvo mjesto izlaznog kanala.

Kako je korak jednak 2, filter je pomaknut za dva mjesta u desno te je druga regija obojana u zeleno. Maksimalna vrijednost te regije iznosi 8 i ona je spremljena na drugo mjesto izlaznog kanala.

Kako je filter došao do ruba matrice, pomiče se na krajnji lijevi rub i spušta se za dva mjesta dolje. Tako je iduća regija obojana u plavo. Maksimalna vrijednost te regije iznosi 6 i ta je vrijednost spremljena na sljedeće mjesto izlaznog kanala.

Konačno, posljednja regija je obojana u žuto, njena maksimalna vrijednost iznosi 5 i ta je vrijednost spremljena na posljednje mjesto izlaznog kanala.

Ovim je završena *max pooling* operacija na uzorku veličine 4x4 ulaznog kanala. Rezultirajući izlaz ima veličinu 2x2 čime se vidi da je ulazna dimenzija smanjena za faktor dva.

### ZAŠTO KORISTITI *MAX POOLING*

Postoji nekoliko razloga zašto se dodavanje *max pooling* operacije u mrežu pokazalo korisnim.

### REDUCIRANJE RAČUNSKOG OPTEREĆENJA

S obzirom da *max pooling* operacija reducira dimenzije danog ulaza iz konovolucijskog sloja, mreža će s vremenom gledati veća područja slike. Korištenjem *max pooling* operacije, smanjuje se broj parametara u mreži i posljedično se smanjuje računsko opterećenje mreže.

### REDUCIRANJE PRETJERANOG POKLAPANJA

Nadalje, *max pooling* operacija pomaže kod reduciranja pretjeranog poklapanja (eng. *overfitting*). Intuicija iza toga jest da će odabrana mreža težiti tome da ekstrahira određene značajke iz slike.

Neka mreža pokušava identificirati brojeve iz MNIST skupa podataka. Ona tada traži rubove, iskrivljene linije, krugove i slično. Izlaz iz konvolucijskog sloja se može tumačiti na način da će više vrijednosti piksela uzrokovati veću aktivaciju tog piksela.

Korištenjem *max pooling* operacije odabrat će se najaktivniji pikseli iz ulaza koji će se sačuvati u mreži. S druge strane, odbacit će se pikseli s niskim vrijednostima jer će biti protumačeni kao nedovoljno aktivni.

### *AVERAGE POOLING*

Kod *average pooling* operacije uzima se prosječna vrijednost regije nad kojom se u tom trenutku nalazi filter.

## POVRATNA PROPAGACIJA U NEURALNIM MREŽAMA

### STOGASTIČKI GRADIJENTNI SPUST

U ovom poglavlju će se prvo obraditi točke u vezi SGD-a koje su iznesene u prethodnim poglavljima. Nakon toga će se opisati gdje se točno koristi SGD te će se ostatak poglavlja baviti intuicijom iz toga što stohastički gradijentni spust zapravo radi.

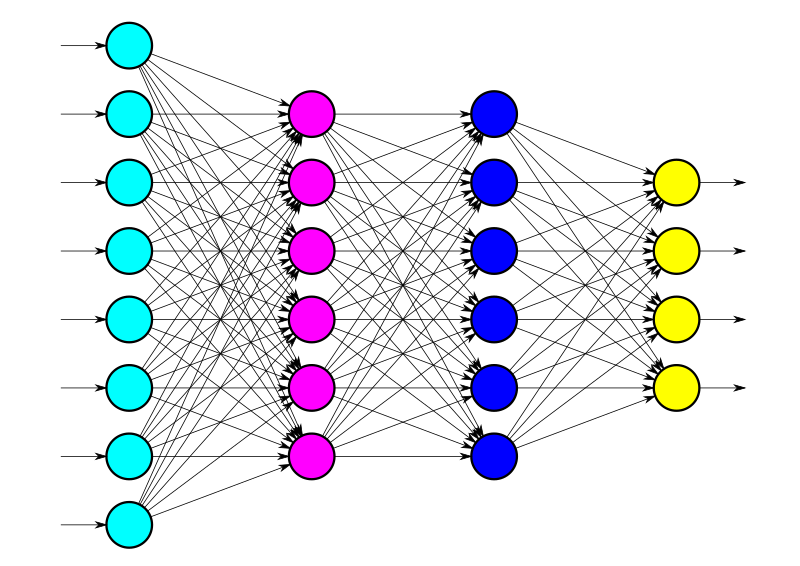
U jednom od prethodnih poglavlja je spomenuto kako, tijekom treniranja modela, stohastički gradijentni spust radi na tome da minimizira funkciju gubitka tako što ažurira težine na kraju svake epohe.

Također je spomenuto kako se ažuriranje težina veza obavlja tako što se izračuna gradijent (ili derivacija) funkcije gubitka u ovisnosti o svakoj pojedinoj težini veze u modelu. Međutim, ova zadnja točka nije bila u potpunosti objašnjena.

To računanje gradijenta s ciljem ažuriranja težina veza u modelu se odvije u procesu koji se zove 'povratna propagacija' (eng. *backpropagation*).

### PROPAGACIJA PREMA NAPRIJED

U dolje prikazanoj slici dana je proizvoljna mreža s dva skrivena sloja. Kako bi se intuicija povratne propagacije što jasnije objasnila, ubuduće će se primjer odnositi na jedan jedini ulaz koji se prosljeđuje danom modelu.



Slika : neuralna mreža s dva skrivena sloja

Kao što je već spomenuto, tijekom procesa treniranja, svaki put kada se podatak proslijedi modelu, taj podatak se prosljeđuje kroz mrežu prema naprijed sve dok podatak ne stigne do izlaznog sloja.

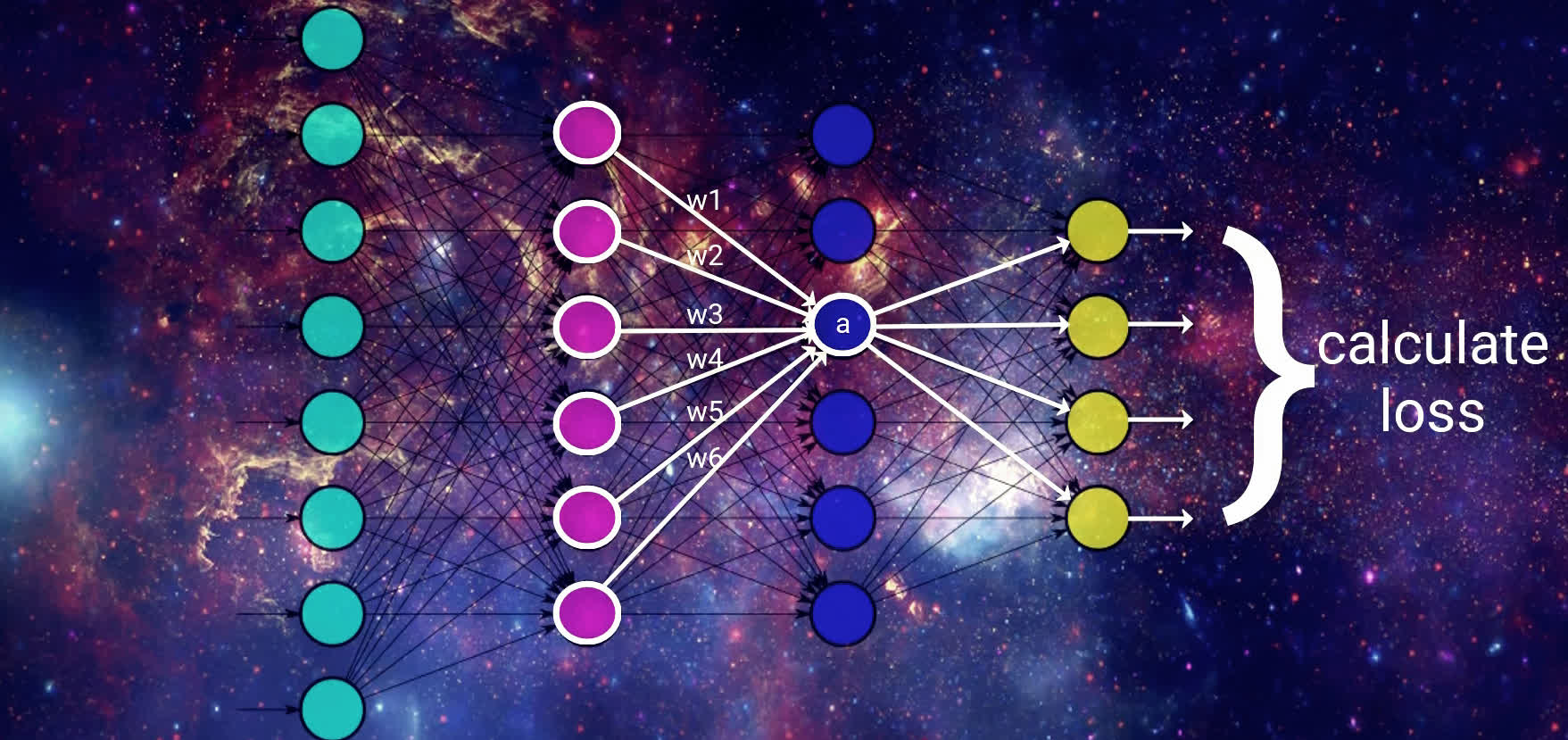
Također, svaki čvor u mreži prima ulaz koji mu se prosljeđuje iz prethodnog sloja. Taj ulaz jest ponderirana suma koju čine težine na svakoj vezi pomnožene s izlaznom vrijednošću prethodnog sloja.

Ponderirana suma se prosljeđuje aktivacijskoj funkciji te je rezultat aktivacijske funkcije izlazna vrijednost iz određenog čvora. Ta vrijednost se zatim prosljeđuje kao ulaz u čvor koji se nalazi u sljedećem sloju. Ovaj proces se odvija za svaki sloj u mreži sve dok se ne dostigne izlazni sloj. Ovaj proces se naziva 'propagacija prema naprijed' (eng. *forward propagation*).

Jednom kada se dostigne izlazni sloj, dobiveni rezultat jest izlazna vrijednost modela za danu ulaznu vrijednost. Ako model, na primjer, treba klasificirati slike životinja, tada će svaki izlazni čvor odgovarati drugoj vrsti životinje. Izlazni čvor, koji ima najvišu izlaznu vrijednost (najvišu aktivnost), predstavlja izlaz za koji je model predvidio da najviše odgovara odgovarajućoj ulaznoj vrijednosti.

### RAČUNANJE GUBITKA

Gubitak se računa na temelju razlike danih izlaznih rezultata i očekivanog rezultata. Način na koji se gubitak računa ovisi od određenoj funkciji gubitka koja se koristi. Zbog jednostavnosti, neka funkcija gubitka bude udaljenost između stvarnog rezultata i predviđenog rezultata.



Slika : računanje gubitka

O gubitku možemo misliti kao o razlici između onoga što je model predvidio za dani uzlaz i prave vrijednosti za dani ulaz.

Kao što je već spomenuto, cilj gradijentnog spusta jest minimizirati funkciju gubitka. To se čini tako što se uzme rezultat derivacije, odnosno gradijenta, funkcije gubitka u ovisnosti o vezama težina u danom modelu.

Ovdje dolazi povratna propagacija.

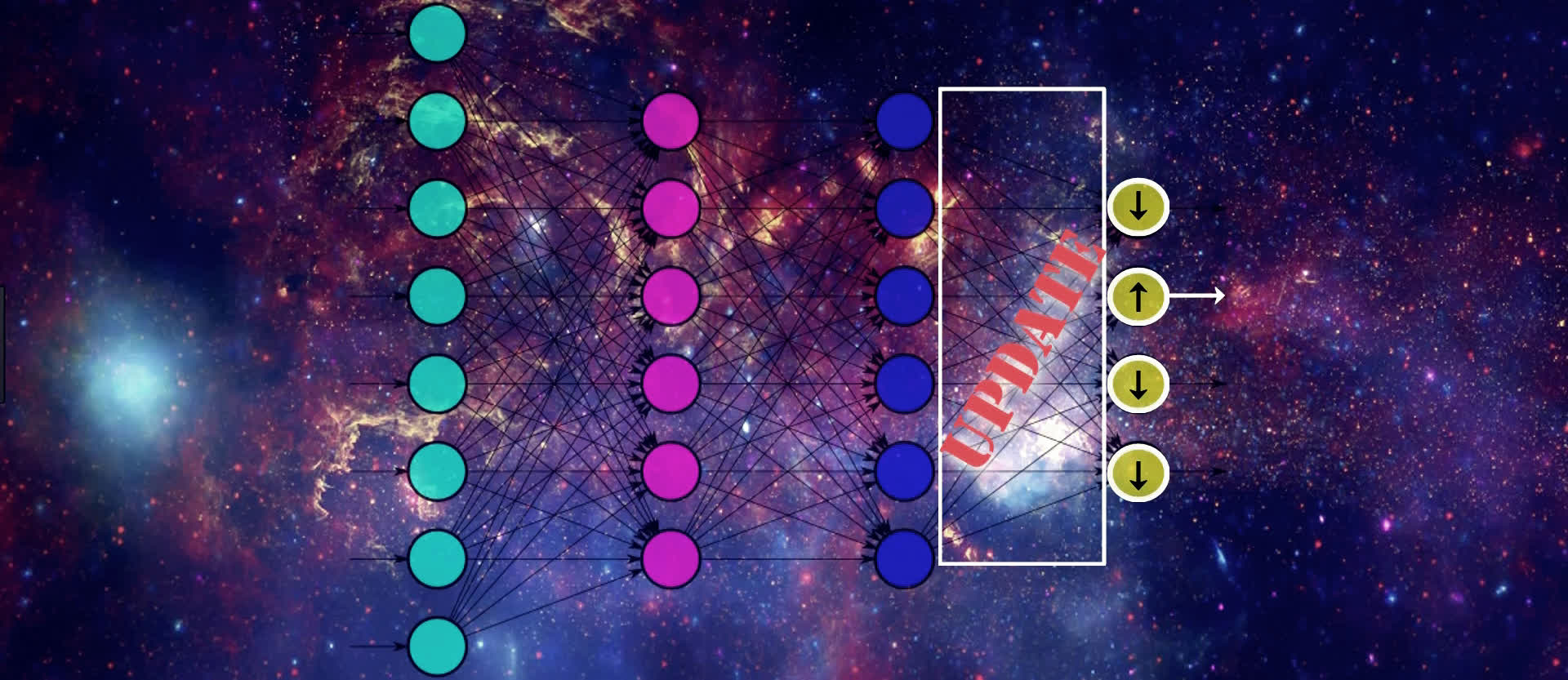
Povratna propagacija jest alat kojeg koristi gradijentni spust kako bi izračunao gradijent funkcije gubitka.

Nakon što je izračunata razlika onoga što je model predvidio i prave vrijednosti, odnosno, nakon što je izračunat gubitak za dani izlaz, gradijentni spust počinje ažurirati težine korištenjem povratne propagacije s ciljem da minimizira funkciju gubitka.

### INTUICIJA POVRATNE PROPAGACIJE

Kako bi gradijentni spust ažurirao težine, on počinje gledati izlaze aktivacijskih funkcija izlaznih čvorova.

Prema dolje prikazanoj slici, neka izlazni čvor, kojem je dodijeljena strelica koja pokazuje prema gore, predstavlja pravi izlaz iz modela za dani ulaz. U tom slučaju, gradijentni spust će raditi na tome da poveća vrijednost tog određenog čvora i da smanji vrijednosti svih ostalih čvorova s ciljem da smanji gubitak za taj dani ulaz.

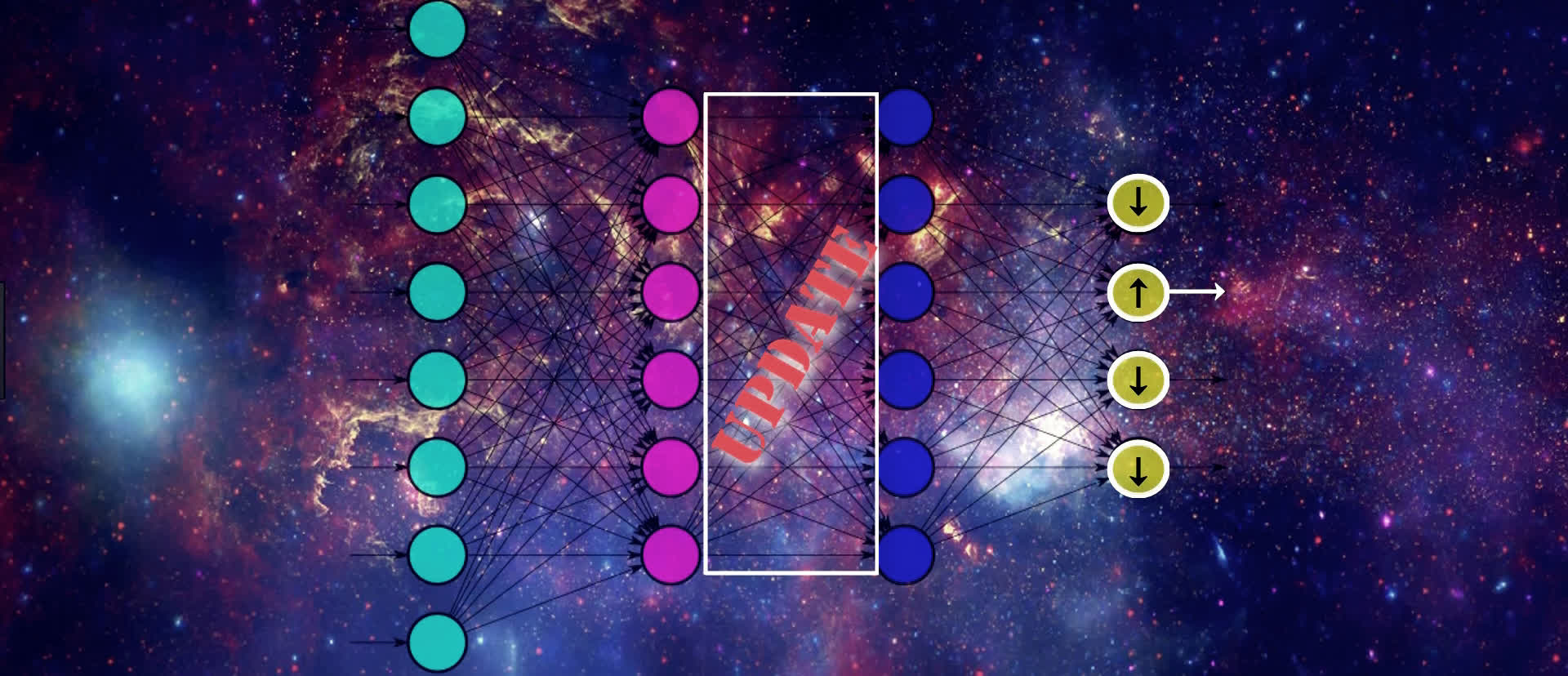


Slika : SGD će nastojati povećati vrijednost čvora kojem je dodijeljena strelica koja pokazuje prema gore

Kao što je već spomenuto, vrijednosti na izlaznim čvorovima su rezultat ponderirane sume koju čine težine veza između izlaznog i predzadnjeg sloja koje su pomnožene izlazima iz predzadnjih slojeva. Ta ponderirana suma je potom dodijeljena aktivacijskoj funkciji izlaznog sloja.

Tako, kako bi se ažurirale vrijednosti izlaznih čvorova, trebaju se ažurirati vrijednosti težina veza koje su povezane s izlaznim slojem. Vrijednosti izlaznih čvorova se mogu ažurirati i tako da se promijeni izlaz iz aktivacijske funkcije prethodnog sloja.

Nije moguće direktno promijeniti izlaz aktivacijske funkcije jer je to izračun temeljen na težinama veza i izlaznih vrijednosti iz njemu prethodnog sloja. Međutim, moguće je indirektno utjecati na promjenu izlaza aktivacijske funkcije (u ovom primjeru predzadnjeg sloja) tako da se cijeli izračun prebaci za još jedan sloj natrag (sloj prije predzadnjeg sloja) i tako da se ažuriraju težine veza na isti način kao što je bilo opisano za izlazni sloj.

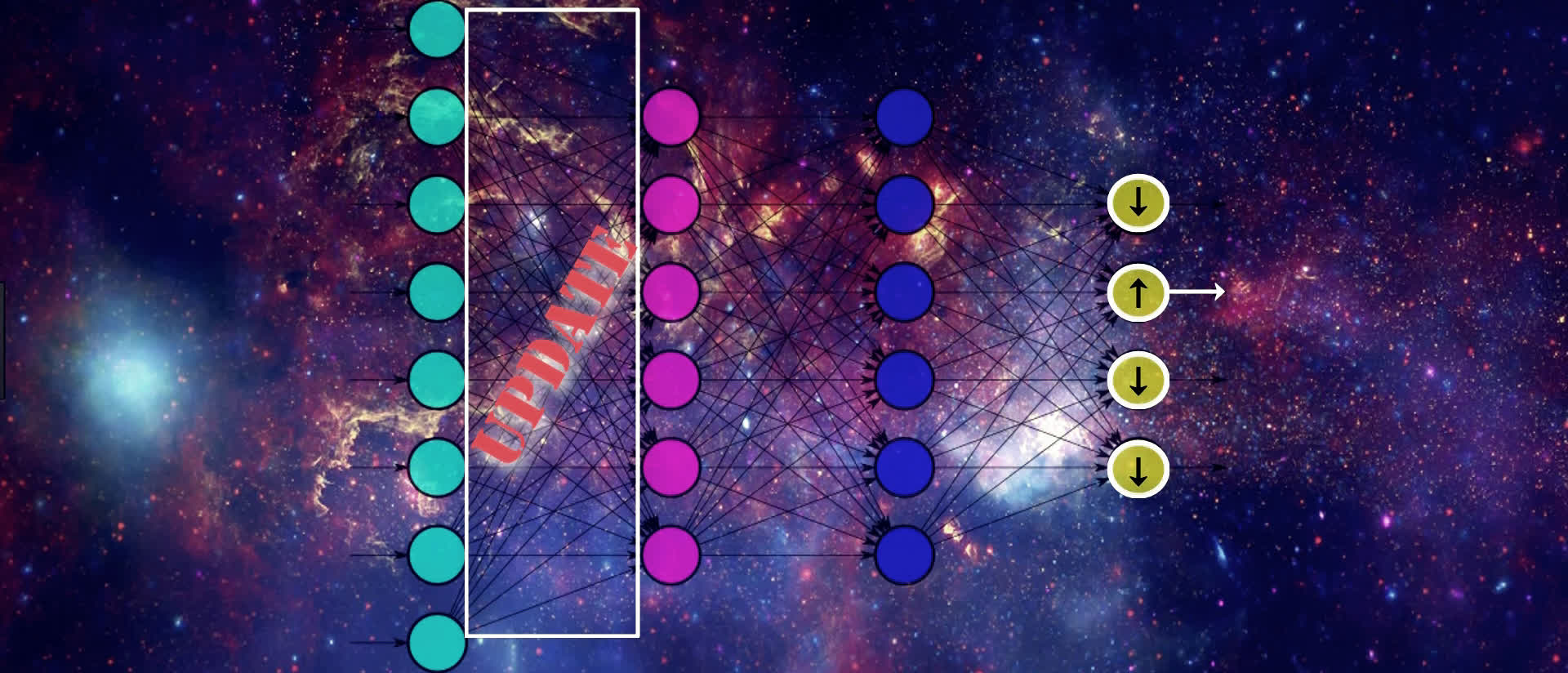


Slika : kako bi se promijenili izlazi aktivacijskih funkcija predzadnjeg sloja, potrebno je ažurirati teže koje vežu taj sloj s njemu prethodnim slojem

Cijeli proces se ponavlja dok se ne dostigne ulazni sloj. Ulaznom sloju se ne mijenjaju vrijednosti jer on sadrži samo stvarne ulazne podatke.

Kao što se moglo primijetiti, cijeli proces se odvija tako da se povratno propagira kroz mrežu i ažurira težine s desna na lijevo s ciljem da se neznatno pomaknu vrijednosti izlaznih čvorova u smjeru u kojem bi se smanjio gubitak.

To znači da, za jedan jedini uzorak ulazne vrijednosti, SGD nastoji povećati vrijednost izlazne vrijednosti za ispravni izlazni čvor i smanjiti izlazne vrijednosti neispravnih izlaznih čvorova. Na taj se način smanjuje gubitak.



Slika : težine veza se povratno ažuriraju sve dok se ne stigne do ulaznog sloja

Važno je spomenuti da povratna propagacija, uz ažuriranje težina u željenom pozitivnom ili negativnom smjeru, nastoji efikasno ažurirati težine na taj način kako bi se najefikasnije smanjila vrijednost funkcije gubitka.

Razmjer u kojem će se težine ažurirati u odnosu prema drugim težinama je relativan jer ovisi o tome koliko će ažuriranje određene težine imati utjecaja na mrežu kao cjelinu kako bi se smanjio gubitak.

Ovaj primjer je pokazao na koji način radi povratna propagacija na jednom ulazu. Međutim, ovaj isti proces se odvija za svaki ulaz iz svake serije (eng. *batch*) koja se prosljeđuje mreži. Rezultirajuće ažurirane težine veza će onda činiti prosječne ažurirane vrijednosti koje su izračunate za svaki pojedini ulaz.

Ti prosječni rezultati za svaku težinu odgovaraju gradijentu funkcije gubitka u odnosu na svaku težinu.

MATEMATIKU POVRATNE PROPAGACIJE

POSEBNO OBRADIT KASNIJE

## NESTAJUĆI I EKSPLODIRAJUĆI GRADIJENT

U ovom poglavlju će se opisati problem koji se zna često pojavljivati tijekom procesa treniranja umjetne neuralne mreže.

To je problem nestabilnih gradijenata koji se popularno naziva 'problem nestajućih gradijenata' (eng. *vanishing gradient problem*).

Općenito, problem nestajućeg gradijenta je problem koji uzrokuje velike poteškoće tijekom treniranja neuralne mreže. To je problem koji je vezan uz težine veza koje se nalaze vežu ranije slojeve u mreži.

Kao što je spomenuto u prethodnom poglavlju, stohastički gradijentni spust (SGD) nastoji izračunati gradijent gubitka u odnosu na težine veza u mreži. Nekada će gradijent u odnosu na težine veza u ranijim slojevima mreže postaje jako mali, odnosno nestajući.

### MALI GRADIJENTI

Kada SGD izračuna vrijednost gradijenta u odnosu na određenu težinu, on koristi tu vrijednost kako bi ažurirao tu težinu na način koji je proporcionalan gradijentu. Ako gradijent ima jako malu, nestajuću, vrijednost, tada će i ažuriranje biti nestajuće malo. Pod malom vrijednosti se misli na vrijednost koja je manja od jedan.

Tako mala promjena neće dovoljno pomoći mreži kako bi smanjila gubitak jer se ona jedva pomakla s mjesta na kojem je bila prije ažuriranja težina. Rezultat je taj da je veza na neki način zapela na mjestu te se nikad neće dovoljno ažurirati na način da se dovoljno približi svojoj optimalnoj vrijednosti. Tako se umanjuje i sama sposobnost mreže da nauči.

Kao što je prije spomenuto, gradijent gubitka u odnosu na danu težinu će biti produkt nekih izvoda koji ovise o komponentama koje se nalaze kasnije u mreži. Tako, što se težina nalazi ranije u mreži, bit će potrebno više pojmova u produktu kako bi se izračunao gradijent gubitka u odnosu na tu težinu. Stoga, ako više pojmova u izračunu ima vrijednost koja je manja od jedan, umnožak tih pojmova će imati još manju vrijednost. Ta mala vrijednost se onda koristi kako bi se ažurirala vrijednost težine. Podsjetnik, ovaj mali broj (gradijent, koji je umnožak brojeva koji su manji od jedan) se množi sa stopom učenja koja je sama po sebi mali broj, najčešće između 0.01 i 0.0001.

Tada će rezultat tog produkta biti još manji broj. Kada se dobije rezultat umnoška, taj broj se oduzima od vrijednosti težine. Konačna, nova, vrijednost težine će biti taj rezultat.

### ZAGLAVLJENE TEŽINE

Neka dobiveni gradijent u odnosu na neku težinu ima jako malu, odnosno nestajuću, vrijednost. Kada se ta mala vrijednost pomnoži sa stopom učenja, dobije se još manja vrijednost. Taj jako mali broj se potom oduzme od vrijednosti težine i dobivena vrijednost se jako malo pomakla od prethodne.

Vrijednost težine na neki način postaje zaglavljena. Ne miče se, ne uči i na taj način ne pomaže u minimiziranju gubitka u vezi.

Težine u ranijim slojevima su sklone ovom problemu jer, kao što je spomenuto, što se težina nalazi ranije u mreži, više je pojmova potrebno uključiti u produkt kako bi se izračunao njen gradijent. A što se više pojmova, koji su manji od jedan, pomnoži, to će gradijent brže 'nestati.

### EKSPLODIRAJUĆI GRADIJENT

Eksplodirajući gradijent se javlja kada su, za razliku od nestajućeg gradijenta, pojmovi u umnošku puno veći od jedan.

Isto pravilo vrijedi i ovdje. Što se težina nalazi ranije u mreži, bit će potrebno više pojmova kako bi se izračunao njen gradijent. Rezultat je taj da će gradijentu 'eksplodirati' vrijednost. S ovakvim gradijentom, prolazi se kroz isti proces ažuriranja težina koji je prije opisan.

Mađutim, umjesto da se vrijednost težine jedva promijeni, ovdje će se vrijednost težine jako promijeniti. Toliko će se promijeniti da se neće dostići optimalna vrijednost težine zbog toga što je proporcija, kojom se ažurira težina, toliko velika da se sve više udaljava od svoje optimalne vrijednosti.

## INICIJALIZACIJA TEŽINA

U umjetnim neuralnim vezama, težine su ono što povezuje čvorove između slojeva. U ovom poglavlju će se objasniti kako se težine inicijaliziraju i kako te inicijalizirane vrijednosti mogu negativno utjecati na proces treniranja.

### KAKO SE TEŽINE INICIJALIZIRAJU

Koncept inicijaliziranja je kratko spomenut u poglavlju o povratnoj propagaciji. Ondje je spomenuto kako se težine inicijaliziraju nasumičnim vrijednostima. Nadalje, svaki put kada se gradi mreža, težine se inicijaliziraju nasumičnim brojevima. Jedan nasumični broj po težini. Obično će ti nasumični brojevi biti normalno distribuirani na taj način da će raspodjela tih brojeva imati srednju vrijednost nula i standardno odstupanje od jedan.

### PRIMJER NASUMIČNE INICIJALIZACIJE

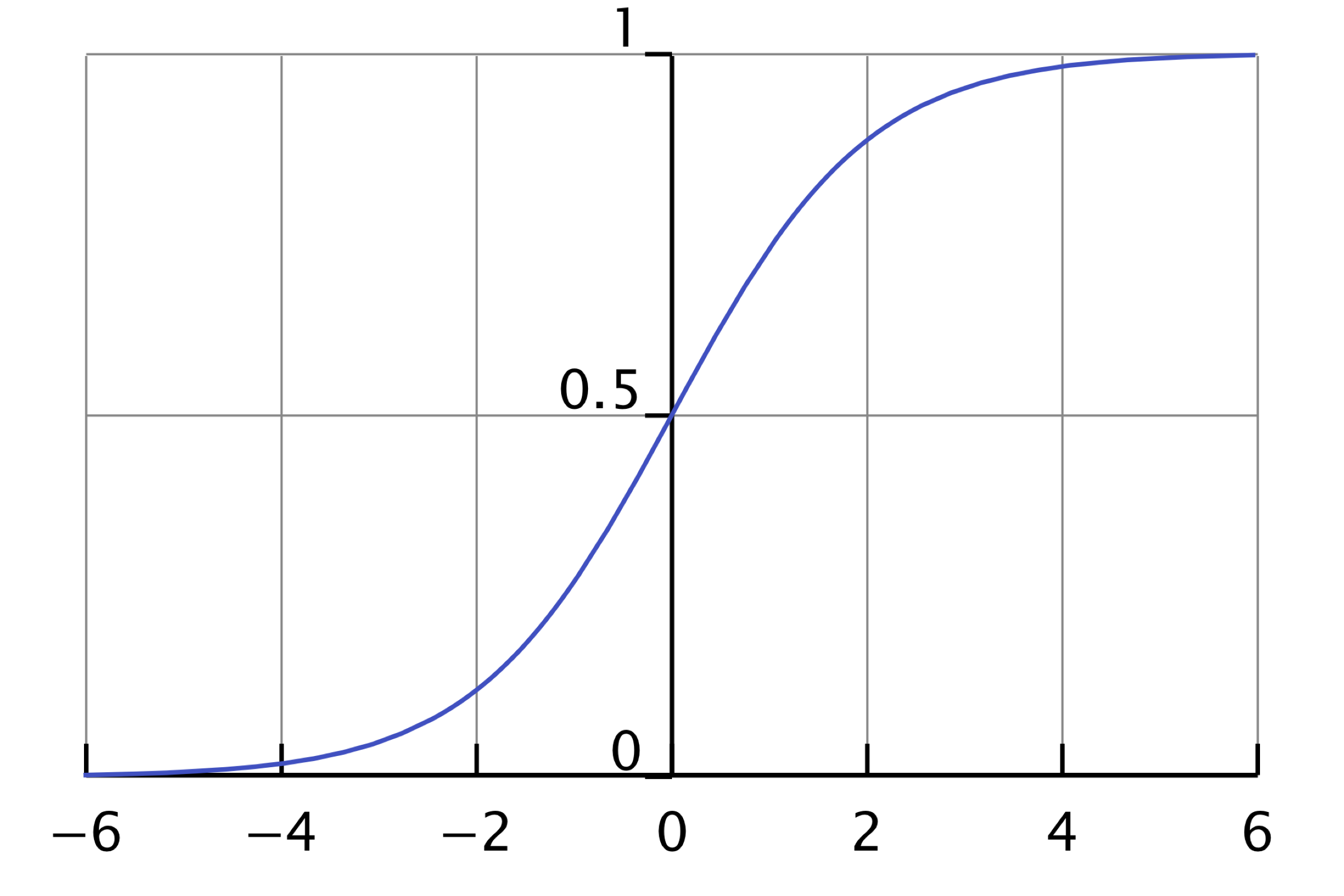
Neka ulazni sloj neuralne mreže ima 250 ulaznih čvorova i, radi jednostavnosti, neka vrijednost svakog od ovih 250 čvorova bude jedan. Isto radi jednostavnosti, primjer će se fokusirati na težine koje povezuju ulazni sloj s jednim čvorom u skrivenom sloju. Ukupno ima 250 težina koje povezuju taj jedan čvor u skrivenom sloju sa svim čvorovima u ulaznom sloju.

Svaka od tih težina je nasumično inicijalizirana i normalno distribuirana oko broja nula sa standardnom devijacijom jedan. Potrebno je izračunati srednju ponderiranu vrijednost, z, koju čvor u skrivenom sloju prihvaća kao ulaz.

Kako su vrijednosti svih ulaznih čvorova jedan, svaka vrijednost težine će biti pomnožena s jedan, tako da z postaje zbroj vrijednosti težina. Kako je vrijednost z u ovom slučaju zbroj normalno distribuiranih brojeva sa srednjom vrijednošću nula, z će također biti normalno ditribuirana oko nule, ali će njena standardna devijacija iznositi puno više od jedan. To je zbog toga što je varijanca zbroja neovisnih nasumičnih brojeva, zbroj varijanci svakog od tih brojeva. Tako, pošto je varijanca svakog nasumičnog broja jedan, varijanca broja z će iznositi 250. Kada se uzme kvadratni korijen ove vrijednosti, dobije se da standardna devijacija broja z iznosi 15.811.

Zbog toga, kada se pogleda normalna distribucija vrijednosti z, vidi se da je ona puno šira od normalne distribucije sa standardnom devijacijom jedan.

Zbog ovako velike standardne devijacije, vrijednost z će najvjerojatnije postići vrijednost koja je značajno veća ili manja od jedan. Kada se ova vrijednost proslijedi aktivacijskoj funkciji, na primjer sigmoidnoj funkciji, zna se da će se jako pozitivni ulazi preslikati u vrijednost jedan, a jako negativni u vrijednost nula.



Slika : grafički prikaz sigmoidne funkcije

### PROBLEMI S NASUMIČNOM INICIJALIZACIJOM

Ako je željeni izlaz aktivacijske funkcije na suprotnoj strani od mjesta zasićenosti težina, tada će se vrijednosti težina, kada SGD ažurira težine s ciljem da utječe na izlaz aktivacijske funkcije, jako malo promijeniti u željenom smjeru. Tako će se smanjiti sposobnost mreže da nauči i samo treniranje će zaglaviti u neefikasnom stanju.

Ovi problemi također doprinose problemi nestajućeg i eksplodirajućeg gradijenta.

### XAIVER INICIJALIZACIJA

Kako je varijanca ulaza u dani čvor određena varijancom težina koje spajaju dani čvor s prethodnim slojem, potrebno je smanjiti varijancu tih težina. Kada se smanji varijanca težina, smanjit će se i varijanca ponderiranog zbroja.

Neka istraživanja su identificirala vrijednost varijance težina koja bi uklonila problem s inicijalizacijom nasumičnih vrijednosti težina. Vrijednost varijance težina koje spajaju dani čvor s prethodnim slojem iznosi gdje je n broj težina koje spajaju dani čvor s prethodnim slojem.

Tako, umjesto da je distribucija težina smještena oko nule s varijancom jedan, distribucija je i dalje smještena oko nule, ali s manjom varijancom, .

Kako bi se dobile težine koje imaju varijancu , nakon što se nasumično generiraju težine smještene oko nule s varijancom jedan, svaka se težina pomnoži s . Ovaj način inicijalizacije se naziva 'Xavier inicijalizacija' (eng. *Xavier initialization*) ili 'Glorot inicijalizacija' (eng. *Glorot initialization*).

Važno je spomenuti da, ako se kao aktivacija funkcija koristi ReLU aktivacijska funkcija, idealna varijanca je . Osim toga, sve što je prije opisano u rješenju problema s inicijalizacijom težina ostaje nepromijenjeno. Također je važno spomenuti da, kako je *n* broj težina koje spajaju ulazni sloj s prethodnim slojem, inicijalizacija težina je proces koji se odvija za svaki sloj posebno.

Dobro je spomenuti i da, kada je Xavier inicijalizacija prvotno bila obznanjena, za određivanje varijance se preporučivalo koristiti formulu gdje je *nin* broj definiran težinama koje ulaze u dani neuron, a *nout* broj definiran težinama koje izlaze iz danog neurona.

## PRISTRANOST U UMJETNOJ NEURALNOJ MREŽI

U literaturi, uz sam naziv *bias*, može se naići i na pojmove *bias* neuroni, *bias* čvorovi ili *bias* jedinice u neuralnim mrežama.

U ovom poglavlju će se objasniti što je to pristranost (ili *bias*) u neuralnoj mreži. Nakon toga će se objasniti kako se ona implementira te će se na kraju opisati jednostavan primjer kako bi se ilustrirao utjecaj *bias-*a u neuralnoj mreži.

Kod pristranosti, misli se na pristranost svakog neurona pojedinačno. Svaki neuron ima svoju pristranost te je cijela mreža sastavljena od više pristranosti. Vrijednosti, koje su pripisane pristranosti, se mogu naučiti, kao i težine veza. Kao što SGD povratnom propagacijom uči i ažurira težine veza, tako isto uči i ažurira vrijednosti pristranosti.

Konceptualno se o pristranosti može misliti kao o nekoj vrsti praga. To je zbog toga što će vrijednost pristranosti odrediti hoće li se izlaz iz neurona propagirati kroz mrežu. Drugim riječima, pristranost će odrediti hoće li se ili neće, i koliko, aktivirati neuron.

### GDJE SE NALAZI PRISTRANOST

Kao što je objašnjeno u jednom od prethodnih poglavlja, svaki neuron prima ponderiranu sumu ulaza od prethodnog sloja te se ta ponderirana suma prosljeđuje aktivacijskoj funkciji.

Vrijednost pristranosti neurona će se zbrojiti s ponderiranom sumom ulaza u neuron te će onda ta nova vrijednost proslijediti aktivacijskoj funkciji.

### PRIMJER GDJE SE KORISTI PRISTRANOST

Neka neuralna mreža ima dva ulazna neurona. Prvi neuron neka ima vrijednost 1, a drugi neka ima vrijednost 2. Primjer će se fokusirati na jedan neuron u prvom skrivenom sloju.

Aktivacijska funkcija koja se koristi u prvom aktivacijskom sloju će biti ReLU. Težinama veza se pridjeljuju nasumične vrijednosti. Neuron na početku neće imati pristranost.

Ponderirana suma koju neuron prima na ulaz izgleda ovako:

Ova vrijednost se zatim prosljeđuje aktivacijskoj funkciji. Kao što je već objašnjeno, izlazna vrijednost iz ReLU će biti nula za sve negativne vrijednosti, a za sve pozitivne vrijednost će ostati nepromijenjena.

U ovom slučaju, izlazna vrijednost će biti 0: ReLU(-0.35) = 0.

S obzirom da je izlazna vrijednost aktivacijske funkcije 0, neuron se smatra neaktiviranim. U ovom slučaju je prag aktivacije neurona vrijednost 0.

Neka se sad prag neurona pomakne na -1. Vrijednost pristranosti će tada biti suprotna od -1, odnosno bit će +1.

Zbroj ponderirane sume i pristranosti neurona iznosit će:

Kada se ova vrijednost proslijedi ReLU, rezultat će iznositi:

ReLU(0.65) = 0.65

Neuron se sada smatra aktiviranim te model sada ima povećanu fleksibilnost što se tiče prilagođavanja podacima jer sada posjeduje veći opseg u vezi toga koje će ga vrijednosti aktivirati, a koje ne.

Ovaj isti proces se može ponoviti i u suprotnom slučaju kako bi se suzio opseg vrijednosti koje će aktivirati neuron. Na primjer, ako se smatra da bi neuron trebao biti aktiviran ako je izlazna vrijednost aktivacijske funkcije veća ili jednaka 5, tada se pristranost neurona postavlja na vrijednost -5.

U ovom primjeru se eksplicitno postavila vrijednost pristranosti neurona, kao i vrijednosti težina. Nakon što su inicijalizirane vrijednosti pristranosti neurona nasumičnim brojevima ili nulama, vrijednosti pristranosti će se ažurirati tijekom procesa učenja dopuštajući modelu da nauči kada će aktivirati i kada neće aktivirati svaki pojedini neuron.

## PARAMETRI U NEURALNOJ MREŽI KOJI SE MOGU NAUČITI

Već se dosta toga spomenulo u vezi parametara koji se mogu naučiti u neuralnoj mreži. Međutim, nije se dao iscrpni popis što to sve neuralna mreža može naučiti.

U ovom poglavlju će se definirati što je parametar koji se može naučiti u neuralnoj mreži. Nakon toga će se opisati kako se broj parametara može izračunati te će se na kraju taj izračun pokazati na jednostavnom primjeru.

Parametar koji se može naučiti u neuralnoj mreži je svaki parametar čiju vrijednost treniranja mreža uči tijekom procesa. Te parametre se može nazivati i parametrima koji se mogu trenirati jer se njihova optimalna vrijednost uči tijekom procesa učenja.

### IZRAČUN PARAMETARA KOJI SE MOGU NAUČITI

Broj parametara, koje mreža može naučiti, računa se za svaki sloj pojedinačno te se svi brojevi parametara zbroje kako bi se dobio ukupan broj parametara.

Kako bi se izračunao ukupan broj parametara u jednom sloju, prvo je potrebno izračunati nekoliko stvari:

1. Broj ulaza u dani sloj
2. Broj izlaza iz danog sloja
3. Sadrži li, ako da, koliko, sloj pristranosti

Ovdje je potrebno napomenuti da se u ovom slučaju misli na potpuno povezanu mrežu koju čine gusti slojevi.

Nakon što se izračunaju brojevi s gore navedene liste, kako bi se izračunao broj parametara, broj ulaza u sloj se pomnoži s brojem izlaza iz čvora . Pod brojem izlaza iz sloja se misli na broj čvorova koji se nalaze u danom sloju kojem se računa broj parametara. Rezultat umnoška jest broj težina veza koje ulaze u taj sloj.

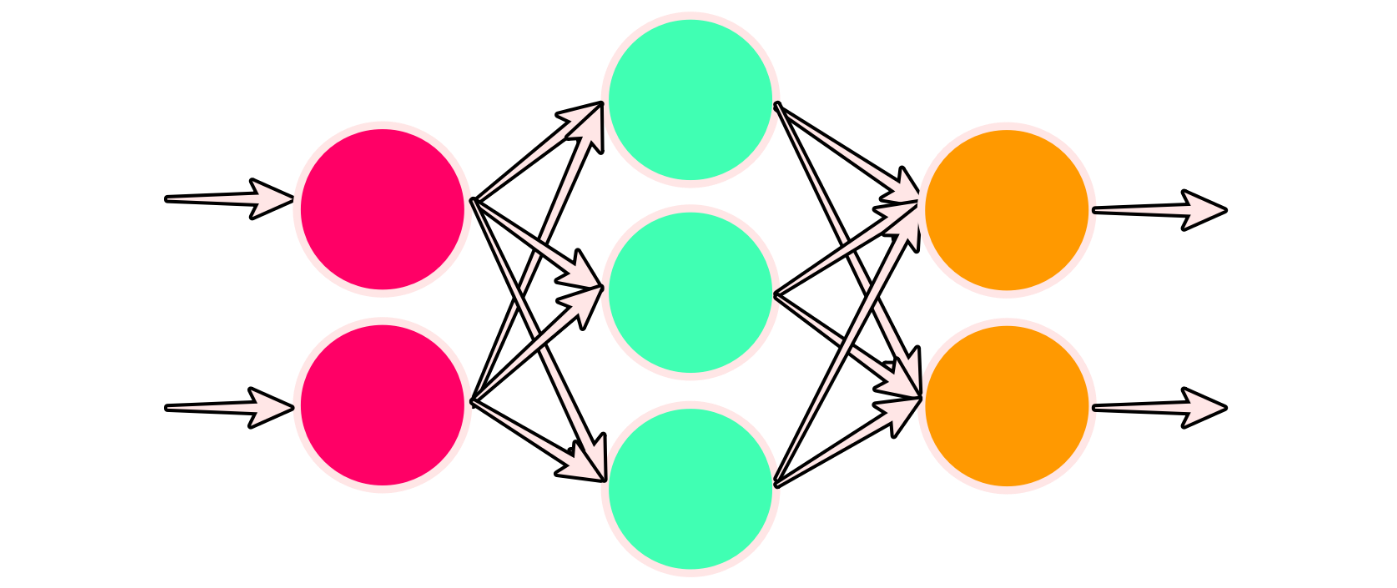
Nakon toga se provjerava sadrži li sloj *bias*-e za svaki čvor. Ako sadrži, broj *bias*-a se jednostavno zbroji s prije dobivenim umnoškom (broj težina). Broj *bias*-a je jednak broju čvorova u danom sloju.

Konačni rezultat jest broj parametara u danom gustom sloju. Isti proces se ponavlja za svaki sloj u neuralnoj mreži. Konačan broj parametara se računa tako što se zbroje parametri za svaki pojedini sloj.

### PRIMJER IZRAČUNA BROJA PARAMETARA KOJI SE MOGU NAUČITI

Neka je dana potpuno povezana neuralna mreža ima tri sloja sa sljedećim brojem čvorova:

1. Ulazni sloj s 2 čvora
2. Skriveni sloj s 3 čvora
3. Izlazni sloj s 2 čvora



Slika : primjer potpuno povezane neuralne mreže s tri sloja

Na kraju, neka mreža sadrži pristranosti. To znači da svaki čvor u skrivenom i izlaznom sloju sadrži pristranost.

Broj parametara se računa na sljedeći način:

1. Ulazni sloj ne sadrži parametre koji se mogu naučiti jer se on sastoji samo od ulaznih podataka. Izlaz iz tog sloja će se samo smatrati ulazom u sljedeći sloj.
2. Broj ulaza u skriveni sloj jest broj čvorova u prethodnom, odnosno ulaznom, sloju. To je broj 2. Broj izlaza iz danog, skrivenog, sloja jednak je broju čvorova u danom sloju. To znači da imamo tri izlaza iz danog sloja. Pomnože se broj ulaza u sloj i broj izlaza iz sloja kako bi se dobio broj težina veza. Dobiven je broj 6. Kako bi se dobio ukupan broj parametara za dani sloj, umnošku se doda broj pristranosti koji je jednak broju čvorova u danom sloju, odnosno 3. Ovaj sloj ima ukupno 9 parametara koji se mogu naučiti.
3. Broj ulaza u izlazni sloj je 3 jer je broj čvorova u prethodnom, skrivenom, sloju 3. Broj izlaza iz izlaznog sloja je 2 jer ima toliko izlaznih čvorova. Umnožak ta dva sloja 6. Na kraju se umnošku doda broj pristranosti i dobije se da je ukupan broj parametara u izlaznom sloju broj 8.
4. Ukupan broj parametara koji mreža može naučiti se dobije zbrajanjem broja parametara skrivenog sloja i broja parametara izlaznog sloja. Mreža može naučiti ukupno 17 parametara. Tijekom procesa treniranja, SGD će učiti i optimizirati svih 17 parametara.

## PARAMETRI KOJI SE MOGU NAUČITI U KONVOLUCIJSKOJ NEURALNOJ MREŽI

U prethodnom poglavlju je opisano kako se računa broj parametara koje može naučiti potpuno povezana neuralna mreža s gustim slojevima. U ovom poglavlju će se opisati kako se taj broj računa kada se radi s konvolucijskom neuralnom mrežom.

Prvo će se objasniti što su to parametri koje može naučiti konvolucijska neuralna mreža te kako se ukupan broj tih parametara može izračunati. Na kraju će biti dan jednostavan primjer kako se taj broj računa u knovolucijskoj neuralnoj mreži.

### KAKO SE RAČUNA BROJ PARAMETARA U KONVOLUCIJSKOJ NEURALNOJ MREŽI

Kao i kod standardne neuralne mreže, broj parametara u konvolucijskoj neuralnoj mreži se računa za svaki sloj posebno te se ukupan broj parametara dobije tako da se zbroji broj parametara za svaki pojedinačni sloj.

Kod gustih slojeva, broj parametara se računao na sljedeći način:

ulazi \* izlazi + pristranosti

Međutim, konvolucijska mreža obično ne posjeduje guste slojeve. Isto tako, za razliku od standardnih neuralnih mreža, konvolucijska mreža sadrži filtere, koji se još nazivaju kernelima. Kada se strukturira kovnolucijska mreža, potrebno je odrediti broj i veličinu filtera koji će se koristiti i taj će broj poslije ući u izračun broja parametara.

Potrebno je voditi računa o tim, gore navedenim, stvarima jer se, također gore navedena, formula modificira s obzirom na to:

1. je li prethodni sloj konvolucijskog sloja bio gusti sloj. Ako je, ulaz u konvolucijski sloj je jednostavno broj čvorova u prethodnom gustom sloju.
2. Je li prethodni sloj konvolucijskog sloja bio konvolucijski sloj. Ako je, broj ulaza u sloj je broj filtera koje posjeduje taj prethodni konvolucijski sloj.

Isto tako, izlaz iz konvolucijskog sloja je broj koji se dobije množenjem broja filtera s njihovim dimenzijama. Konačno, broj pristranosti će biti jednak broju filtera u danom konvolucijskom sloju.

Tako je sačuvana općenita postavka za broj parametara koji se mogu naučiti, umnožak broja ulaza s brojem izlaza koji se potom zbroji s brojem pristranosti.

### PRIMJER RAČUNANJA BROJA PARAMETARA U KONVOLUCIJSKOJ NEURALNOJ MREŽI

Neka se konvolucijska neuralna mreža sastoji od ulaznog sloja, dva skrivena sloja i gustog izlaznog sloja:

1. ulazni sloj
2. skriveni konvolucijski sloj
3. skriveni konvolucijski sloj
4. gusti izlazni sloj.

Neka ulazni sloj sadrži ulazne podatke slika čije su dimenzije 20x20x3, gdje 20x20 definira širinu i visinu slike, a broj 3 definira broj kanala. Tri kanala indiciraju da su slike u RGB-u te ova tri kanala predstavljaju ulazne značajke u ulalzni sloj.

Prvi konvolucijski sloj sadrži 2 filtera dimenzija 3x3. Drugi konvolucijski sloj sadrži 3 filtera dimenzija 3x3. Izlazni sloj je gusti sloj s 2 čvora.

Neka konvolucijska mreža sadrži pristranosti i neka je pretpostavljeno popunjavanje nulama kako bi se sačuvale dimenzije slika.

Do sada vrijedi:

1. ulazni sloj: slike dimenzija 20x20x3
2. skriveni konvolucijski sloj: 2 filtera dimenzija 3x3
3. skriveni konvolucijski sloj: 3 filtera dimenzija 3x3
4. gusti izlazni sloj: 2 čvora

Ulazni sloj ne sadrži parametre koji se mogu naučiti jer oni samo sadrži ulazne podatke.

Prvi skriveni sloj ima tri ulaza iz ulaznog sloja. Broj izlaza iz tog sloja jest umnožak broja filtera s njihovim dimenzijama. Kako taj sloj sadrži dva filtera, svaki dimenzija 3x3, broj izlaza iz konvolucijskog sloja će biti 2\*3\*3 = 18. Množenjem broja ulaza s brojem izlaza dobije se broj od 3\*18 = 54 težine. Na kraju se taj broj zbroji s brojem pristranosti i dobije ukupan broj od 56 parametara koji se mogu naučiti u prvom skrivenom konvolucijskom sloju.

Drugi skriveni sloj ima dva ulaza iz prethodnog sloja jer prethodni sloj sadrži dva filtera. Broj izlaza iz trenutnog sloja jest umnožak broja filtera s njihovim dimenzijama, to je 3\*3\*3 = 27.

Umnoškom broja ulaza s brojem izlaza se opet dobije broj od 54 težine. Na kraju se tom broju doda broj pristranosti i dobije broj od 57 parametara koje mreža može naučiti u drugom skrivenom konvolucijskom sloju.

S obzirom da je pretpostavljeno da mreža koristi popunjavanje nulama, sačuvane su dimenzije slika koje ulaze u mrežu (20x20). Tako se množenjem ta dva broja dobije broj od 1200 ulaza u izlazni sloj. Sada, s obzirom da je izlazni sloj gusti sloj, broj izlaza je jednak broju čvorova u danom sloju. Množenjem broja ulaza s brojem izlaza, dobije se broj od 2400 težina koje povezuju izlazni i njemu prethodni sloj. Kada se doda broj pristranosti čvorova, dobije se ukupan broj 2402 parametra koje konvolucijska neuralna mreža može naučiti u izalznom sloju.

Kada se zbroje svi parametri iz svakog pojedinog sloja mreže, dobije se ukupan broj od 2515 parametara koje mreža može naučiti za danu strukturu.

Može se vidjeti da je, općenito, proces određivanja broja parametara isti kao i kod standardne neuralne mreže, samo što se mora voditi računa od dodatnim stvarima kao što su broj kanala, broj filtera i njihovih dimenzija.

## REGULARIZACIJA U KONVOLUCIJSKOJ NEURALNOJ MREŽI

U ovom poglavlju će se objasniti što je to regularizacija, kada se koristi i u kojim situacijama bi mogla biti od pomoći ako se doda modelu.

U poglavlju o *overfitting*-u se spomenula metoda ispuštanja (eng. *dropout*) za koju je napomenuto da je to regularizacijska tehnika.

Općenito, regularizacija je tehnika koja pomaže reducirati pojavu *overfitting*-a ili reducirati varijancu u mreži na način da se penalizira složenost. Ideja iza ove tehnike je da određene složenosti u modelu mogu smanjiti sposobnost modela da dobro generalizira, iako model ispravno preslikava ulaze iz skupa podataka za treniranje.

Korištenjem neke od tehnika za regularizaciju, žrtvuje se sposobnost modela da se dobro prilagodi modelu kako bi model bolje generalizirao podatke kojih neka u skupu za treniranje.

### L2 REGULARIZACIJA

Najčešće korištena tehnika regularizacije naziva se 'L2 regularizacija' (eng. *L2 regularization*). Napomena da regularizacija radi tako da funkciji gubitka pridruži pojam koji penalizira velike težine. L2 regularizacija funkciji gubitka pridružuje sumu kvadriranih normi matrica težina

Koja je pomnožena malom konstantom

Norma je funkcija koja pridjeljuje strogo pozitivnu duljinu ili veličinu svakom vektoru u prostoru vektora. U ovom slučaju prostor vektora ovisi o veličinama matrica težina koje se koriste za dani model.

Ako *v* označava vektor u prostoru vektora, njegova norma se označava s ||v|| i mora vrijediti da je

Tehnikom L2 regularizacije dobiva se sljedeći izraz za gubitak:

Donja tablica daje definiciju za svaku varijablu u gornjem izrazu.

|  |  |
| --- | --- |
| **varijabla** | **definicija** |
| n | broj slojeva |
| w[j] | matrica težina za sloj j |
| m | broj ulaznih podataka |
| λ | regularizacijski parametar |

Λ je regularizacijski parametar i jedan od hiperparametara koji se koristi kako bi se testirao i ugodio specifični model.

Ukratko, regularizacija je tehnika koja penalizira relativno velike težine u modelu na način da pridruži neki dodatni pojam postojećoj funkciji gubitka.

### UTJECAJ REGULARIZACIJE

Neka je za tehniku regularizacije odabrana L2 regularizacijska tehnika i neka je vrijednost λ postavljena na neki veliki broj. Regularizacija bi utjecala na model tako da bi on postavio težine veza na vrijednost koja je blizu 0 jer je cilj SGD-a minimizirati funkciju gubitka.

Ako je vrijednost λ velika, tada će izraz ostati relativno velik. Kada se pomnoži sa sumom kvadriranih normi, rezultat će biti relativno velik, ovisno o tome koliko su velike težine u modelu. To znači da će model biti potaknut da stvara male težine kako bi vrijednost cijele funkcije gubitka ostala relativno mala kako bi minimizirala gubitak.

## VELIČINA PODSKUPOVA U UMJETNOJ NEURALNOJ MREŽI

U ovom poglavlju će se opisati što to znači specificirati veličinu podskupa iz sveukupnog skupa podataka za treniranje.

Veličinu podskupa će činiti broj uzoraka iz cjelokupnog skupa podataka za treniranje koji će se kao cjelina proslijediti mreži u jednom trenutku. Podsjetnik, epohu čini jedan prolazak cijelog skupa podataka za treniranje. Podskup i epoha nisu ista stvar.

### PODSKUPOVI U EPOHI

Neka skup podataka za treniranje čini 1000 slika automobila pomoću koji će se model naučiti da razlikuje različite modele automobila. Ako je batch veličina jednaka 10, to znači da će se skup od 10 slika automobila proslijediti modelu u jednom trenutku.

Kako epohu čini jedan prolazak cijelog skupa za treniranje, bit će potrebno 100 podskupova kako bi se završila jedna epoha.

### RAZLOG KORIŠTENJA PODSKUPOVA

Općenito vrijedi da, što je veća veličina podskupa, model će brže završiti svaku epohu tijekom treniranja. To je zbog toga što, ovisno o računalnim resursima, računalo može procesirati toliko više od jednog uzorka u jednom trenutku.

Međutim, čak i ako računalo može obraditi velike podskupove u jednom trenutku, kvaliteta modela može degradirati jer, ako je podskup, koji se šalje modelu u jednom trenutku, dovoljno velik, model možda neće moći dobro generalizirati podatke koje do tada nije vidio.

Općenito, veličina podskupa je još jedan u nizu hiperparametara koji se moraju testirati i podesiti na temelju toga koliko ispravno konkretni model preslikava podatke tijekom učenja. Ovaj parametar se treba testirati i u odnosu na to koliko dobru izvedbu ima samo računalo u smislu iskorištavanja vlastitih resursa kada se koriste različite veličine podskupova.

Na primjer, kada bi veličina podskupa bio relativno veliki broj, kao 100, tada računalo možda neće raspolagati dovoljnom računalnom moći da paralelno obradi svih 100 slika i to bi bio znak da se veličina skupa mora postaviti na neki manji broj.

### MINI-BATCH GRADIJENTNI SPUST

Korištenjem *mini-batch* gradijentnog spusta, gradijent će promijeniti svoju vrijednost nakon prolaska svakog podskupa (per-batch basis) kroz model. Ovo je suprotno od stohastičkog gradijentnog spusta koji implementira promjene gradijenta nakon prolaska svakog uzorka i suprotno je od *batch* gradijentnog spusta koji implementira promjenu gradijenta nakon svake epohe.

### FINO PODEŠAVANJE NEURALNE MREŽE – NIJE POTREBNO?

U ovom poglavlju će se objasniti što je to fino podešavanje i kako se ono može iskoristiti prilikom razvoja i treniranje neke određene umjetne neuralne mreže.

Fino podešavanje je usko povezano s pojmom transfera učenja (eng. *transfer learning*). Transfer učenja se odvija kada se iskoristi znanje, koje se steklo rješavanjem jednog problema, na novom problemu koji je sličan prethodnom.

To bi, na primjer, bilo iskorištavanje znanja koje se steklo kako bi se prepoznali modeli automobila na problem gdje treba prepoznati modele kamiona.

Pod pretpostavkom da je originalni zadatak sličan novom zadatku, korištenje umjetne neuralne mreže koja je već dizajnirana i trenirana daje prednost u smislu da se model ne mora dizajnirati i trenirati od nule.

Kada bi se model trenirao od nule, više parametara se podešava na način pokušaja i pogreške. Na primjer, potrebno je odabrati koliko će se slojeva koristiti, koje će se vrste slojeva koristiti, u kojem će se redu postaviti, koliko će svaki sloj imati čvorova, koja će se tehnika regularizacija koristiti, koja će biti stopa učenja itd.

S druge strane, kada bi se primjeru prepoznavanja modela kamiona iskoristio model koji je naučen da prepoznaje modele automobila, on će već znati prepoznati teksture, oblike, rubove i, u kasnijim slojevima, svjetla, kvake, vjetrobrane, gume itd. Sve će te značajke, koje je model naučio na prethodnom modelu, dati prednost u učenju značajki koje su karakteristične za kamione.

Kod finog podešavanja se najčešće ostave prvi slojevi u mreži dok se miču slojevi koji se nalaze pri kraju mreže.

## BATCH NORMALIZACIJA

U ovom poglavlju će se opisati što je to batch normalizacija, još poznata i kao batch norm, i kako se ona primjenjuje kod treniranja umjetnih neuralnih mreža.

### TEHNIKE NORMALIZACIJE

Prije nego se objasni batch normalizacija, opisat će se regularne tehnike normalizacije.

Općenito govoreći, tijekom treniranja neuralne mreže, cilj je na neki način normalizirati ili standardizirati podatke kako bi ih se pripremilo za treniranje.

Normalizacija i standardizacija imaju isti cilj transformiranja podataka kako bi svi podaci pripadali istoj ljestvici.

Obično se normalizacijski proces sastoji od skaliranja numeričkih podataka na ljestvicu od 0 do 1. Obično se proces standardizacije sastoji od oduzimanja srednje vrijednosti od svakog podatka. Na kraju se ta razlika podijeli sa standardnom devijacijom skupa podataka.

Ovaj postupak forsira standardizirane podatke kako bi imali srednju vrijednost 0 i standardnu devijaciju 1. U praksi, ovaj proces standardizacije se također naziva normalizacija.

### KORIŠTENJE TEHNIKA NORMALIZACIJE

Općenito se sve svodi na sažimanje podataka na neku poznatu standardiziranu ljestvicu. Kada podaci ne bi bili normalizirani, neki numerički podaci bi u danom skupu podataka jako odstupali od ostalih svojom jako visokom ili jako niskom vrijednosti.

Na primjer, neka podaci govore o tome koliko su pojedinci autom prešli kilometara u razdoblju od 5 godina. Mogu postojati pojedinci koji su prešli 100 000 kilmetara, ali isto tako mogu postojati pojedinci koji su prešli 1000 kilometara. Ovi podaci imaju relativno široki opseg i nisu nužno na istoj ljestvici.

Nadalje, uzorci u skupu podataka mogu međusobno odudarati i po još nekim značajkama. Kada bi se u skupu podataka, koji je maloprije bio spomenut, nalazili podaci i o starosti pojedinaca, podaci o starosti i broju prijeđenih kilometara se neće nalaziti na istoj ljestvici.

Takvi tipovi podataka koji nisu normalizirani mogu uzrokovati nestabilnost u neuralnim mrežama zbog toga što se relativno veliki opseg ulaznih podataka propagira kroz mrežu i uzrokuje neuravnotežene gradijente koji mogu uzrokovati poznati problem explodirajućih gradijenata.

Podaci koji nisu normalizirani i uravnoteženi mogu uzrokovati probleme u mreži zbog kojih ona teže i sporije uči.

### PRIMJENA BATCH NORMALIZACIJE NA SLOJ

Batch normalizacija se primjenjuje samo da odabrane slojeve u neuralnoj mreži.

Prvo što batch normalizacija učini, kada se primijeni na određeni sloj, jest da normalizira izlaz iz aktivacijske funkcije. Kada batch norm normalizira taj izlaz, ona taj normalizirani izlaz pomnoži s nekim proizvoljnim parametrom kojem doda još jedan proizvoljni parametar kako bi se dobio konačni izlaz.

Proces normalizacije prikazan je na donjoj tablici.

|  |  |  |
| --- | --- | --- |
| **korak** | **izraz** | **opis** |
| 1 |  | Normaliziraj izlaz x iz aktivacijske funkcije. |
| 2 |  | Pomnoži normalizirani izlaz z s proizvoljnim parametrom g. |
| 3 |  | Pribroji proizvoljni parametar b produktu (z\*g). |

Ovaj izračun s dva proizvoljna parametra podacima postavlja novu standardnu devijaciju i srednju vrijednost. Dva proizvoljno postavljena parametra, *g* i *b*, mogu se naučiti. To znači da će se optimizirati tijekom procesa učenja.

Ovaj proces čini da težine u mreži ne postanu neuravnotežene zbog ekstremno visokih ili niskih vrijednosti s obzirom da je normalizacija uključena u proces gradijenata.

Primjena batch normalizacije može značajno povećati brzinu treniranja i smanjiti preveliki utjecaj velikih težina tijekom procesa učenja. Ovaj proces normalizacije se odvija prije nego što se podaci proslijede ulaznom sloju jer je to proces pred-procesiranja podataka prije nego započne proces treniranja.

S batch normalizacijom, moguće je normalizirati izlazne podatke iz aktivacijske funkcije za pojedine slojeve u modelu. To znači da u sloj ulaze normalizirani podaci, ali isto tako iz sloja izlaze normalizirani podaci.

Naravno, ovaj proces normalizacije se ponavlja za svaki *batch* našeg skupa podataka. Od tamo dolazi naziv *batch norm.* Ti podskupovi su određeni veličinom skupa (*batch size*) koja je postavljena za svrhu treniranja danog modela.