Primjena strojnog učenja u klasificiranju LEGO kocaka

Hamzić, Ervin

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:859496

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij

PRIMJENA STROJNOG UČENJA U KLASIFICIRANJU LEGO KOCAKA

Završni rad

Ervin Hamzić

Osijek, 2018.

Sadržaj

1. UVOD 1

1.1 ZADATAK ZAVRŠNOG RADA 1

2. TEORIJSKA POZADINA 2

2.1 MOTIVACIJA 2

2.2 UMJETNE NEURONSKE MREŽE 3

2.2.1 STRUKTURA NEURONSKE MREŽE 5

2.2.1.1 KONVOLUCIJSKI SLOJ 5

2.2.1.2 *ReLU* SLOJ 8

2.2.1.3 OSTALE AKTIVACIJSKE FUNKCIJE 11

2.2.1.4 *DROPOUT* SLOJ I UNAKRSNA VALIDACIJA 13

2.2.1.5 SLOJ SAŽIMANJA 14

2.2.1.6 POTPUNO POVEZAN SLOJ 15

2.2.2 TRENIRANJE NEURONSKE MREŽE 15

2.2.2.1 ALGORITAM PROPAGACIJE POGREŠKE UNATRAG 15

3. REALIZACIJA NEURONSKE MREŽE 19

3.1 PRIPREMA PODATAKA 19

3.2 KORIŠTENE TEHNOLOGIJE 20

3.2.1 *TENSORFLOW* 20

3.2.2 *KERAS* 21

3.3 PRIMJER IMPLEMENTIRANE ARHITEKTURE 21

4. ZAKLJUČAK 25

LITERATURA 26

SAŽETAK 27

ABSTRACT 28

ŽIVOTOPIS 29

1. UVOD

U području umjetne inteligencije, posljednjih nekoliko desetljeća donijelo je izniman razvoj metodologija, kao i njihovo usvajanje u primjeni na sve većem broju problema. Na temelju aktualnih trendova, svakako se može reći da se svojevrstan naglasak u okviru umjetne inteligencije kao računarske discipline pomiče u smjeru primjenskog aspekta, što kroz dulje razdoblje nakon početaka razvoja nije bio slučaj, već je ta grana imala prvenstveno akademski predznak. U tom smislu, primjeri poput obrada govora, prepoznavanje ljudskog lica ili pak potpuno ili djelomično autonomne vožnje samo su neki od istaknutih primjera koje pronalazimo u svakodnevnom životu, a koji su zasnovani na pristupima strojnog učenja. Razlog takvim promjenama i napretku leži ponajprije u sve većoj pristupačnosti grafičkih procesorskih jedinica visokih performansi, kao i velikih podatkovnih skupova. Samim time, otvoren je put ka *treniranju* vrlo složenih neuronskih mreža, što ujedno znači i početak u današnje vrijeme vrlo perspektivne discipline poznate pod nazivom duboko učenje, a to je stvorilo nove potencijale u područjima poput računalnog vida i već spomenute obrade govora, ali i mnogih drugih, odnosno svih onih u kojima se rješavanje problema svodi na svojevrsnu identifikaciju određenog uzorka ili trenda iz velike količine nestrukturiranih podataka. Kao rezultat toga, niz velikih tehnoloških kompanija poput *Microsofta, Google-a, YouTube-a* ili *Amazona* provode procese redizajna svojih ključnih proizvoda i usluga imajući u vidu pristupe dubokog učenja. Vrlo sličan trend pojavio se i među vodećim kompanijama iz sektora razvoja i proizvodnje procesorskih jedinica, kao i u okviru *FPGA (*engl. Field Programmable Gate Array) tehnologije na kojoj se danas zasnivaju računalni sustavi visokih performansi za obradu slike i govora implementirani u velikom broju široko dostupnih elektronskih uređaja, a ponajprije u pametnim telefonima i automobilima. Još jedna vrlo važna domena čiji je razvoj dobio snažan zamah jest autonomna vožnja i različiti sustavi asistencije vozaču (*ADAS – Advanced Driver Assistance Systems)* poput asistencije pri održavanju vozila unutar vozne trake ili pak detekcije objekata u mrtvom kutu, odnosno ispred samog vozila ili npr. sustava za prepoznavanje prometnih znakova. Svi navedeni sustavi danas se baziraju na dubokim neuronskim mrežama radi uspješnosti koju takav pristup pokazuje u rješavanju problema iz područja računalnog vida poput klasifikacije slika te detekcije i lokalizacije objekata, kojem ujedno pripada i zadani problem – klasifikacija *LEGO* kocaka primjenom strojnog učenja.

1.1 ZADATAK ZAVRŠNOG RADA

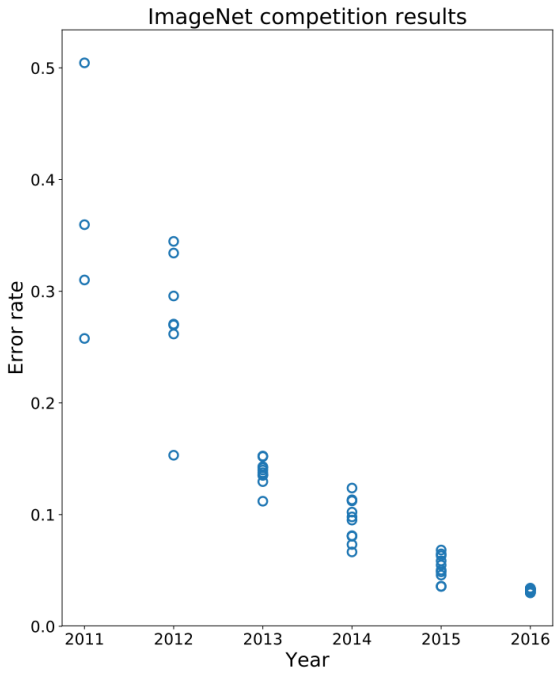
Zadatak ovog rada je primjenom numeričke biblioteke *TensorFlow* i aplikacijskog sučelja *Keras* nad prikladnim podatkovnim skupom razviti, istrenirati i testirati duboku konvolucijsku neuronsku mrežu za klasifikaciju slika *LEGO* kocaka.

2. TEORIJSKA POZADINA

Ovo poglavlje detaljnije obrađuje teorijske pretpostavke i motivaciju koja stoji iza primjene neuronskih mreža na problemima identifikacije objekata gdje ulaznu informaciju predstavlja slika. Naime, ideja o umjetnim neuronskim mrežama nije najnovija – matematički modeli kojima ih se može opisati pojavili su se još sredinom prošlog stoljeća.

2.1 MOTIVACIJA

Jedan od središnjih zadataka koji se postavlja pri donošenju odluka i predviđanju trendova, a o čemu ovise mnogi važni procesi, kao npr. u poslovnom okruženju ili pak kao u slučaju spomenute problematike koja je obrađena u samom radu – prepoznavanje klase objekta bazira se na lokalizaciji i ekstrakciji informacije iz apstraktnog skupa podataka, što je u zadanom problemu slika, a može se raditi i npr. o vremenskom nizu kao informaciji koja ima odgovarajuću interpretaciju poput kretanja cijene određene dionice na tržištu u nekom vremenskom razdoblju. Prije ere strojnog učenja, spomenuti problemi bili su rješivi samo čovjeku na temelju prethodnog iskustva, iako uz određenu (nezanemarivu) vjerojatnost pogreške, ovisno o različitim faktorima. Međutim, ljudski perceptivni sustav vrlo je učinkovit kada govorimo o analizi slike radi prepoznavanja različitih objekata. Točnost koju postiže prosječna osoba u prepoznavanju vizualne informacije u većini je slučajeva svojevrsna referenca koja često još uvijek nije nadmašena metodama strojnog učenja. U tom pogledu, važno je spomenuti *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* – međunarodno natjecanje i referencu za istraživačke timove u problemima računalnog vida, odnosno prepoznavanja i grupiranja objekata na *ImageNet* bazi podataka s više od 14 milijuna slika razvrstanih u klase, na kojoj se evaluiraju implementacije rješenja za klasifikaciju. Rezultati postizani iz godine u godinu znakoviti su za primjenu neuronskih mreža u takvim zadacima te predstavljaju važan doprinos i motivaciju daljnjim istraživanjima u tom znanstvenom području. 2012. godina i rezultati tadašnjeg izdanja natjecanja u kojem je rješenje zasnovano na konvolucijskoj neuronskoj mreži odnijelo uvjerljivu pobjedu (pogreška klasifikacije od 16.4 %, drugoplasirani 26.2 %) smatraju se prekretnicom i početkom revolucije dubokog učenja. Ono što je uslijedilo jest konstantno smanjenje pogreške klasifikacije. 2017. godine gotovo svi timovi postižu vrijednost pogreške manju od 5 %, što je rezultat vrlo sličan, odnosno bolji od onoga koji se uzima za točnost prepoznavanja koju ostvaruje čovjek na istom skupu podataka (pogreška od približno 5.1 %) [1].



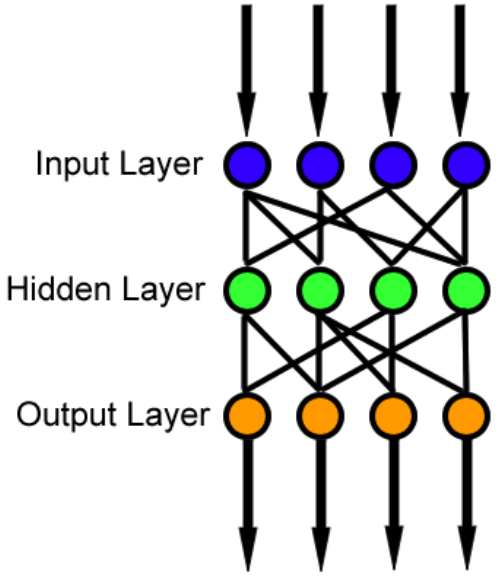
Sl. 2.1: Rezultati *ImageNet* natjecanja u razdoblju od 2011. do 2016. godine. [3]

2.2 UMJETNE NEURONSKE MREŽE

Postavljeni problem zahtijeva da se za određeni podatkovni skup koji se sastoji od niza slika LEGO kocaka osmisli algoritam koji će slike klasificirati prema zajedničkim vizualnim obilježjima. Kao što je već naznačeno, konvolucijska neuronska mreža kao metoda dubokog učenja najefikasniji je, a ujedno i odabrani pristup za rješavanje takvog problema. Klasifikacija slike podrazumijeva da se za određenu sliku kao ulaznu informaciju reproducira klasa kojoj objekt sa slike pripada ili pak niz vjerojatnosti pripadanja najizglednijim klasama. Pod slikom se podrazumijeva JPG slika dimenzija *m x n* koju možemo predočiti kao niz brojčanih vrijednosti koje povezujemo uz pojedini piksel u slici. Taj je niz veličine m x n x 3, pri čemu se posljednja dimenzija odnosi na RGB vrijednosti. Sve su vrijednosti iz raspona 0 do 255 (oboje uključivo). Tu treću dimenziju u ovom kontekstu često nazivamo *dubinom*.
Ideja o neuronskoj mreži i njezinoj strukturi, barem na apstraktnoj razini, naslanja se način funkcioniranja dijela ljudskog mozga odgovornog za vizualnu percepciju. Ukratko, živčane stanice vizualnog korteksa grupirane su po svojoj funkciji i njihovo okidanje događa se samo u prisustvu određene vrste podražaja, ovisno o grupi [2].

Npr. aktivacije se u pojedinoj grupi događaju jedino s obzirom na postojanje okomitih, horizontalnih ili dijagonalnih rubova i sl. Ono što proizlazi iz toga jest da perceptivni sustava čovjeka na svojstven način "kreće" od jednostavnih obilježja u analizi vizualnog podražaja te ih grupira i povezuje prema apstraktnijim objektima.

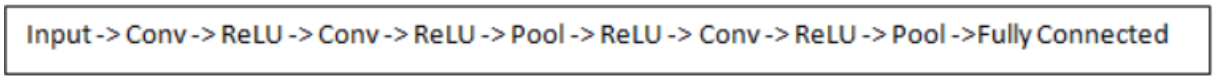
Iako se radi o vrlo apstraktnom opisu funkcioniranja, opisani način analize ulazne informacije vrlo je blizak onome kojeg pronalazimo kod neuronskih mreža, što se ogleda i u njihovoj slojevitoj arhitekturi. Intuitivno govoreći, "provlačenjem" slike kroz slojeve neuronske mreže, sliku kao ulaz sustava pretačemo u niz vjerojatnosti pripadanja pojedinoj klasi objekata na izlazu. Opisan način funkcioniranja odgovara unaprijednoj neuronskoj mreži (eng. *feedforward neural network*) čija je implementacija predmet ovog rada. Atribut unaprijedna označava da se informacija unutar neuronske mreže prenosi isključivo u jednom smjeru, bez petlji i povratnih veza kao što je to slučaj kod povratnih neuronskih mreža (eng. *recurrent neural network*).



Sl 2.2: Pojednostavljeni prikaz unaprijedne neuronske mreže [4]

2.2.1 STRUKTURA NEURONSKE MREŽE

Kao što je spomenuto, duboka neuronska mreža inkorporira niz slojeva – uvijek se radi o određenom broju konvolucijskih, nelinearnih (*ReLU*) i sažimajućih (eng. *undersampling, pooling layer)* slojeva koji su zaključeni jednim ili više potpuno povezanih slojeva. Svi unutarnji slojevi (između prvog konvolucijskog i potpuno povezanog sloja na kraju) nazivaju se i skrivenim slojevima. Slojevi dolaze u određenom slijedu, odnosno grupirani su na odgovarajući način. Značaj i operacije koje podrazumijevamo pod svakim od slojeva bit će pojašnjeni u nastavku.

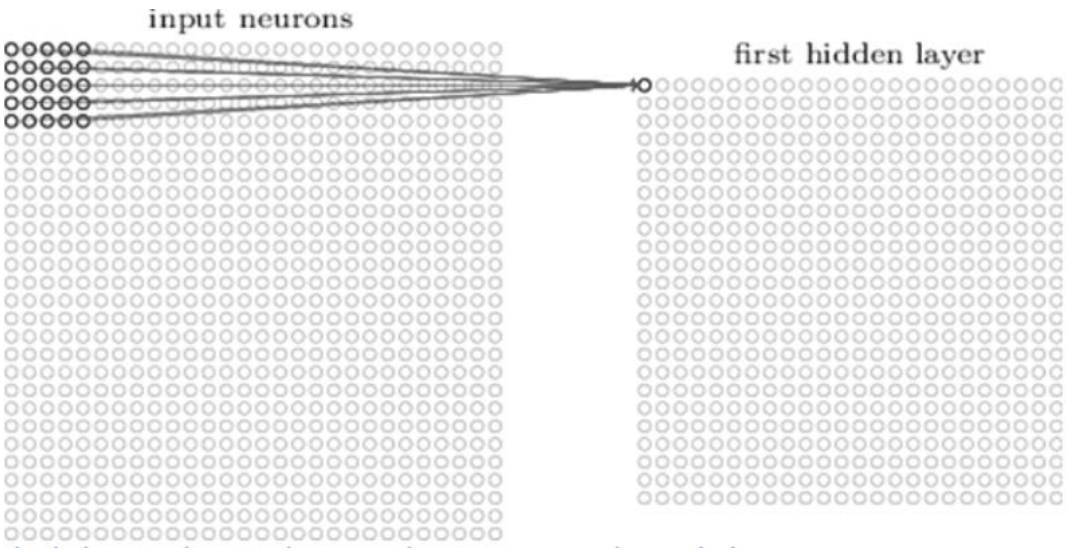


Sl. 2.3: Organizacija slojeva neuronske mreže – zajedničko velikoj većini arhitektura je konvolucijski sloj na početku i potpuno povezani sloj kao završni [2]

2.2.1.1 KONVOLUCIJSKI SLOJ

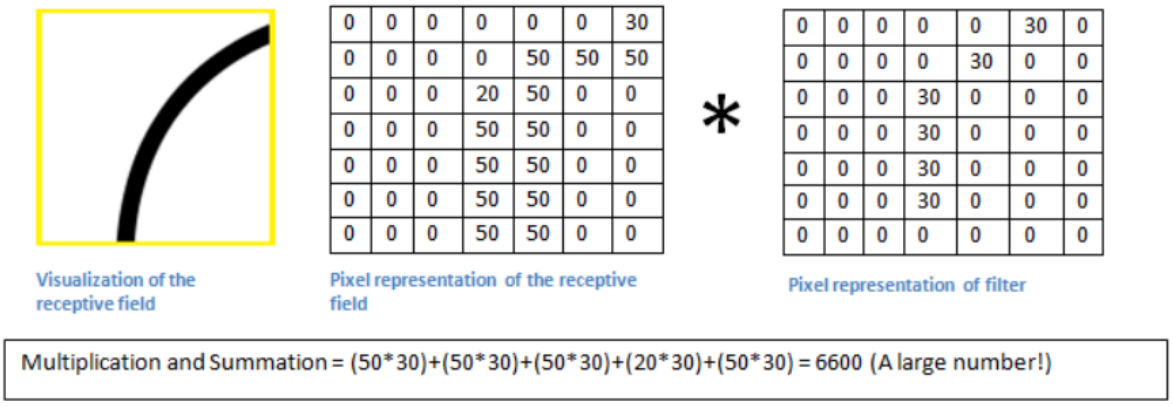
Sloj koji se uvijek pojavljuje kao prvi u konvolucijskoj neuronskoj mreži (eng. *Convolutional Neural Network – CNN)* upravo je konvolucijski sloj. Ovdje je potrebno uvesti pojam filtra (također *neuron* ili *jezgra*) kao manje strukture (matrice) koja se "pomiče" po slici za analizu, odnosno koji je konvoluiran sa slikom prikazanom nizom RGB brojčanih vrijednosti. S apstraktnog stajališta, svrha filtra je identifikacija pojedinih manjih obilježja na slici. Matematički, konvolucija ovdje podrazumijeva niz pojedinačnih operacija množenja elemenata filtra sa spomenutim RGB vrijednostima slike. Uvijek se uzima da konvolucija kreće iz gornjeg lijevog kuta slike. Tada, ako je slika npr. dimenzija 64 x 64, a odabrani filtar 6 x 6, to znači da svaki korak konvolucije imati 108 operacija množenja pojedinačnih elemenata (ne zaboraviti na to da imamo treću dimenziju, dakle potrebno je izvesti 6 * 6 * 3 = 108 operacija). Svi se parcijalni rezultati množenja sumiraju, te se dobiva pojedinačna vrijednost. Filtar se zatim pomiče udesno za određeni broj elemenata (piksela slike) i tako do kraja retka, nakon čega se pozicionira do kraja ulijevo i obično jedno polje niže te se isti postupak ponavlja dok se ne dođe do posljednje pozicije koju filtar takvim pomicanjem može zauzeti nad slikom (filtar u donjem desnom uglu slike). Opisani postupak najslikovitije je predočen na slici 6.

Korak pomaka filtra (eng. *stride*) predstavlja jedan od hiperparametara neuronske mreže i potreban je za određivanje veličine izlazne matrice konvolucijskog sloja [3]. Ta je matrica ujedno i ulaz za idući sloj neuronske mreže te se naziva *mapom aktivacija* (eng. activation map, feature map). U našem slučaju, s obzirom na spomenute dimenzije filtra i slike, rezultat konvolucije bila bi matrica dimenzija 59 x 59.



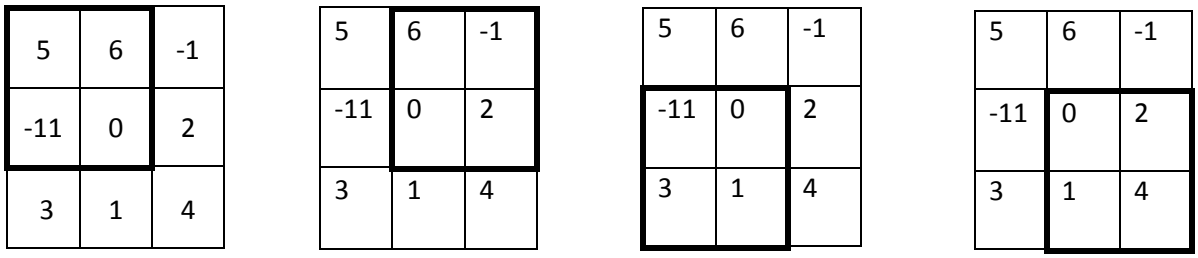
Sl. 2.4: Prikaz operacije konvolucije filtra sa slikom i dobivanja mape aktivacija [2]

Kao što je spomenuto, filtar efektivno izolira pojedine strukture unutar slike. U svezi s time, ako se težine filtra poklope s odgovarajućim područjem slike, odgovarajuća vrijednost u mapi aktivacija bit će izrazito velika.

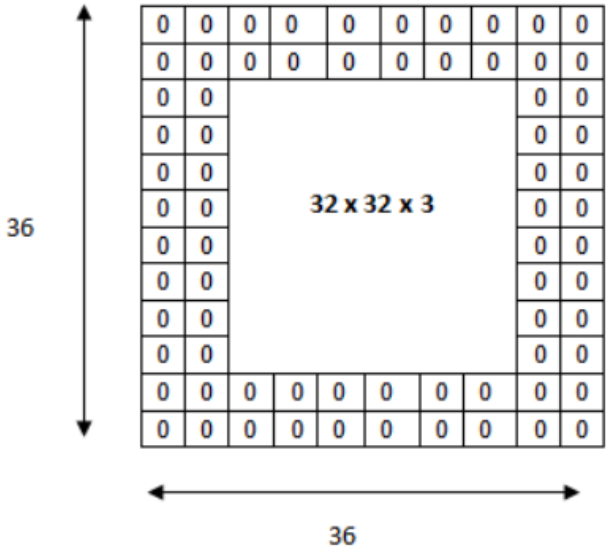


Sl. 2.5: Rezultat jednog koraka konvolucije za slučaj velikog poklapanja segmenta slike s koeficijentima filtra [2]

Ono što je važno istaknuti jest da se prolaskom filtra po ulaznom volumenu na ovaj način prebrzo gube prostorne dimenzije. U gornjem se primjeru vidi da za ulaz dimenzija 64 x 64 proizlazi mapa veličine 59 x 59. Time bi se prolaskom već kroz nekoliko konvolucijskih slojeva dimenzija izlaznog volumena smanjila na neželjeno male veličine. To posebno negativno utječe na ekstrakciju detalja niske razine u ranijoj fazi treniranja neuronske mreže. Kako bi očuvali prostorne dimenzije, ulazni se volumen proširi poljima koja sadrže nule (eng. *zero padding*) kako prikazuje slika 7.



Sl. 2.6: Prelazak filtra po ulaznom uzorku



Sl. 2.7: Povećanje dimenzija ulaznog volumena dodavanjem polja s nulama [3]

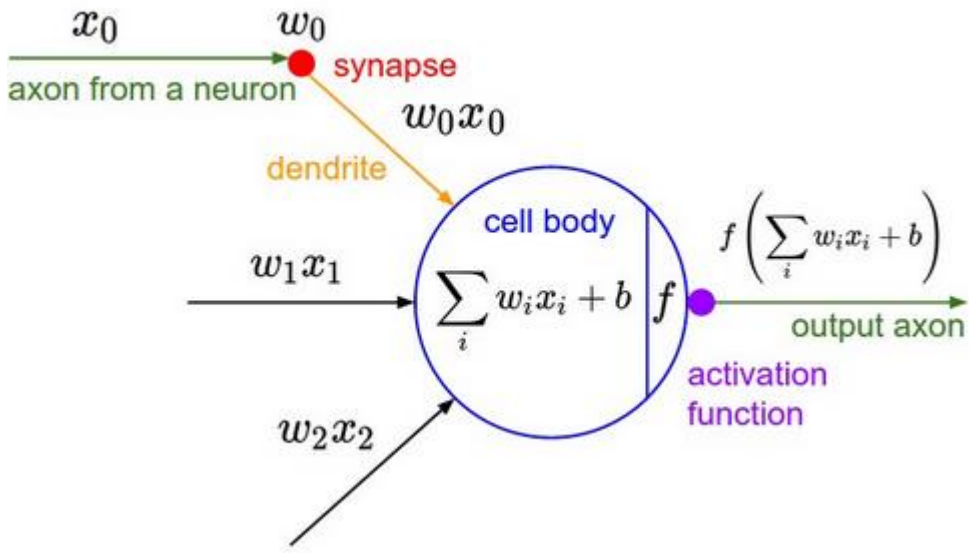
Ovim postupkom uveli smo još jedan hiperparametar neuronske mreže. U konačnici, dimenzija mape aktivacija određuje se prema sljedećem izrazu [3]:

$$O = \frac{W-K+2P}{S} + 1, \qquad (2.1)$$

gdje je O dimenzija ulazne matrice, W je dimenzija mape aktivacija (izlaz konvolucijskog sloja), K je dimenzija filtra, P proširenje nulama (u gornjem primjeru iznosi 2), a S je korak filtra. Jednostavnim računom može se vidjeti da za W = 32, K = 5, P = 2 i S = 1 dimenzija koja je bila prisutna na ulazu ostaje sačuvana.

2.2.1.2 *ReLU* SLOJ

Kako bi se moglo razumjeti funkciju koju *ReLU* (eng. *Rectified Linear Unit*) sloj ima u neuronskoj mreži, potrebno je uvesti pojam aktivacijske funkcije. Ovdje je zgodno nastaviti spomenutu paralelu između neuronskih mreža i ljudskog perceptivnog sustava. Neuroni u živčanom sustavu preuzimaju signale preko dendrita i generiraju električni potencijal na svom aksonu koji se zatim grana i taj se signal prenosi sinaptičkim vezama na dendrite drugih neurona. Jakost sinaptičke veze određuje faktor pojačavanja ili potiskivanja privedenog signala, što je ključno za interpretaciju podražaja, a ujedno i ekvivalent težinama filtra kojima je modeliran konvolucijski sloj neuronske mreže. Unutar samog neurona događa se, uvjetno rečeno, sumacija doprinosa pojedinih signala koji su stigli preko dendrita živčane stanice. Ukoliko ta suma prekoračuje određeni prag (eng. *threshold*), događa se aktivacija, okidanje neurona, što rezultira pojavom izrazitog potencijala na aksonu živčane stanice. Procesom učenja kod čovjeka utječe se na jakost sinaptičkih veza. Slično tome, *treniranjem* neuronske mreže prilagođavaju se vrijednosti koeficijenata filtra.



Sl. 2.8: Pojednostavljeni matematički model neurona [5].

Osnovni matematički model neurona bio bi ovakav:

$$y = \begin{cases} 0, & \sum_j w_j x_j \leq b \\ 1, & \sum_j w_j x_j > b \end{cases}$$

Sljedeći je zapis istog izraza međutim učestaliji:

$$y = \begin{cases} 0, & \sum_j w_j x_j - b \leq 0 \\ 1, & \sum_j w_j x_j - b > 0 \end{cases}$$

Parametri prikazanog modela su sljedeći: *x* – ulazna veličina, svojstvo (eng. *input feature*), *w* – težina (eng. *weight*), b – prag (eng. *bias*). Ulazna veličina u pojednostavljenom slučaju predstavlja binarnu vrijednost (0 ili 1) koja odgovara tome je li neki preduvjet ili događaj ispunjen, težina je faktor koji naznačava relevantnost pojedine ulazne veličine, a *bias* je efektivno prag okidanja neurona. Predstavljeni model neurona naziva se TLU – perceptronom (eng. *Threshold Logic Unit*). Nažalost, ovakav model pokazuje izrazite nedostatke utoliko što rezultira mrežama koje su vrlo teške za treniranje. Naime, u velikom broju slučajeva, mala promjena jednog od ulaza takvog perceptrona rezultirat će promjenom njegovog izlaza iz 0 u 1 ili obrnuto, što jako utječe na rezultat. Potrebna je drukčija aktivacijska funkcija od opisane.

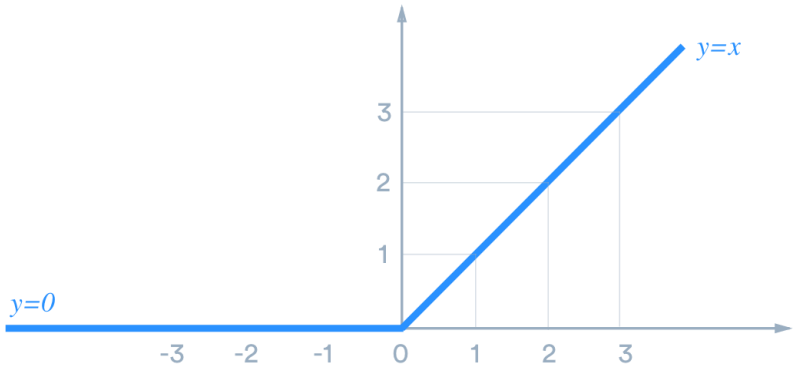
Općenito, aktivacijskom funkcijom ostvarujemo mapiranje izlaza funkcije iz modela neurona

$$f(x) = \sum_i (w_i x_i + b) \qquad (2.2)$$

u određeno područje vrijednosti koje se razlikuje između pojedinih aktivacijskih funkcija. Aktivacijska funkcija za koju smo ustanovili problem zapravo je *step* funkcija, a da bi taj problem bio izbjegnut, prvenstveno je potrebna "analogna" funkcija.
ReLU funkcija jedna je od najzastupljenijih i definirana je na sljedeći način:

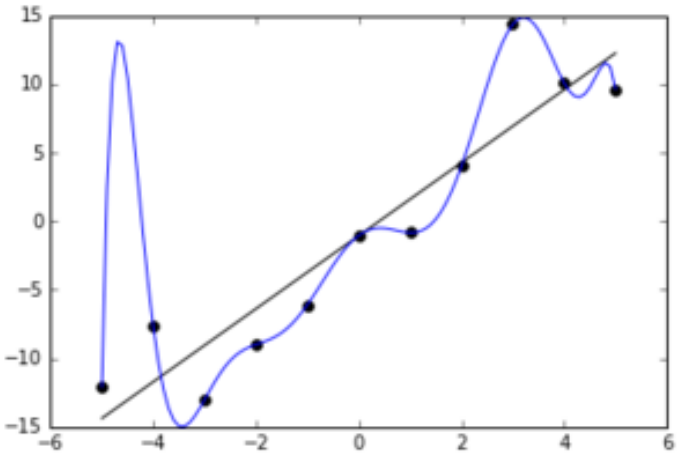
$$f(x) = \max(0, x), \qquad (2.3)$$

gdje je *x* vektor ulaznih vrijednosti neurona.



Sl. 2.9: Graf *ReLU* aktivacijske funkcije [9]

ReLU je u velikoj većini slučajeva preferirana vrsta aktivacijske funkcije, prvenstveno radi toga što omogućuje bolje performanse neuronske mreže u odnosu na sigmoidnu, odnosno tangens hiperbolnu funkciju [11]. To podrazumijeva bržu konvergenciju težina i pragova vrijednostima koje daju minimalnu pogrešku klasifikacije objekta, što znači da je treniranje mreže time brži proces (budući da je dostizanje globalnog minimuma vrijednosti pogreške i konačan cilj treniranja neuronske mreže). Kao matematička operacija, *ReLU* je vrlo nezahtjevna funkcija za izvođenje. Također, još jedno od istaknutih svojstava te funkcije jest efikasnost s obzirom na broj okinutih neurona pri obradi pojedinog inputa. Drugim riječima, različiti ulazi (slike) rezultiraju okidanjem različitih (manjih) grupa neurona uz mala preklapanja između grupa. Naime, kao što se vidi iz grafa funkcije, svi neuroni čija je suma po funkciji (2.2) negativna bit će isključeni. Ta separiranost aktivacija donosi bolju sposobnost predikcije utoliko što se smanjuje prenaučenost (eng. *overfitting*) nad podacima za trening. Prenaučenost ili pretreniranost neuronske mreže karakterističan je problem kod određenih algoritama strojnog učenja, pri čemu je posljedica nedovoljna sposobnost generalizacije od strane istreniranog modela kada taj model izložimo novim podacima, različitim od onih na kojima je model treniran.

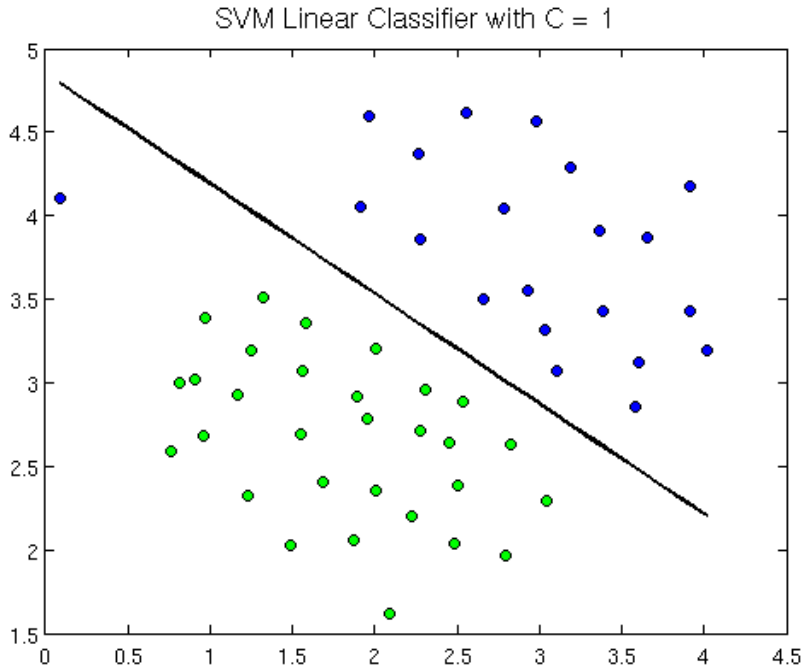


Sl. 2.10: Pravac – dobra generalizacija, krivulja – jako loša generalizacija [12]

Slika 10 jednostavan je primjer koji prikazuje razliku između ciljane aproksimacije podatkovnog skupa i prenaučenog modela koji na danom skupu čak savršeno oponaša trend, ali kada bi se izvodila ekstrapolacija izvan prikazanog područja, model predstavljen pravcem dao bi puno točnije rezultate. Bolju generalizaciju u pravilu predstavlja što jednostavnija i "glađa" krivulja.

Općenito govoreći, ono na što prvenstveno ciljamo primjenom aktivacijske funkcije jest da neuronsku mrežu učinimo nelinearnom, što omogućuje da mreža aproksimira nelinearne zakonitosti između ulaza i izlaza s visokom preciznošću. Bez slojeva aktivacijskih funkcija, neuronska bi mreža mogla precizno mapirati samo linearne procese.

Naime, funkcija $f(x) = wx + b$ kojom je opisan pojedini neuron predstavlja polinom prvog reda, a linearna kombinacija više takvih funkcija ponovno daje polinom prvog reda. Dakle, neovisno o tome koliko bi slojeva neuronska mreža imala, efektivno bi se radilo o mreži sa samo jednim slojem, što je naravno vrlo ograničavajuće.



Sl. 2.11: Primjer linearnog klasifikatora [8]

Takva bi mreža mogla biti upotrijebljena jedino u slučaju binarne klasifikacije i to kada uzorci u koordinatnoj ravnini imaju takvu distribuciju da ih je moguće razdvojiti povlačenjem pravca. Time bi bio realiziran linearni klasifikator. Problem klasifikacije slike s velikim brojem razreda izrazito je nelinearan pa je primjena (nelinearne) aktivacijske funkcije svakako neizostavna.

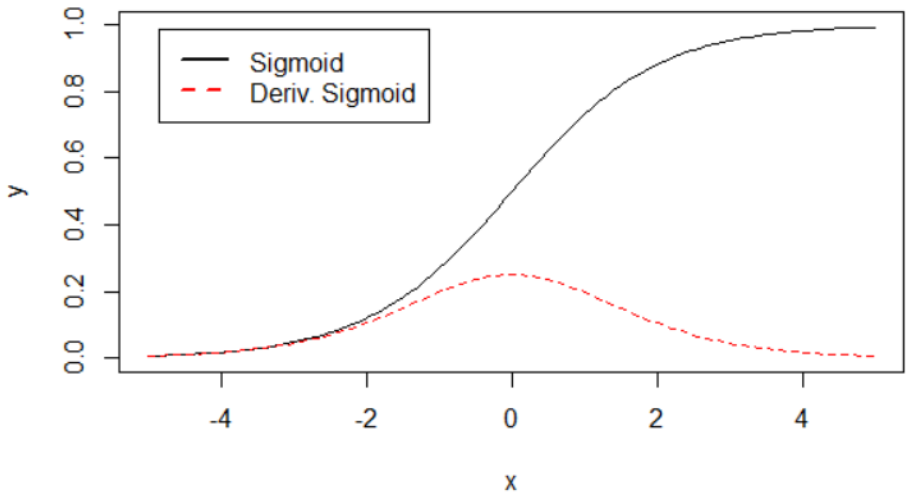
2.2.1.3 OSTALE AKTIVACIJSKE FUNKCIJE

Kako bi se još bolje razumjelo radi čega je *ReLU* preferirana aktivacijska funkcija, zgodno je dodatno pojasniti svojstva drugih dviju često spominjanih nelinearnih aktivacijskih funkcija – sigmoidne i tangens hiperbolne funkcije.

Sigmoidna funkcija definirana je na sljedeći način:

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \qquad (2.4)$$

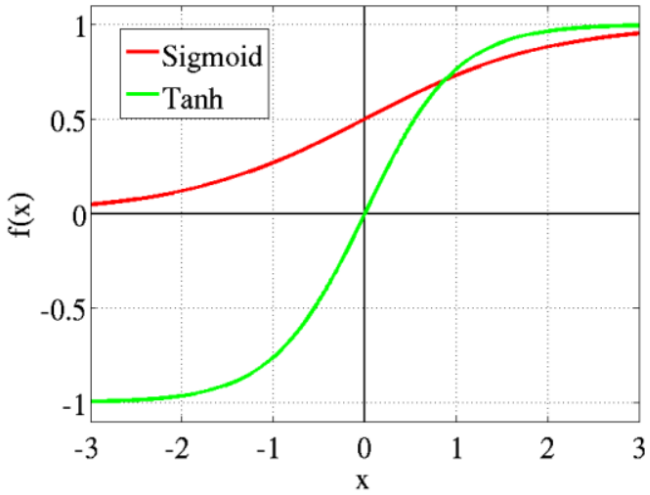
Sigmoidnu funkciju možemo promatrati kao "zaglađenu" step funkciju, što se na prvi pogled čini kao pogodna karakteristika jer je time prevladan glavni nedostatak step funkcije (binarna aktivacija radi koje je treniranje mreže vrlo zahtjevan proces). Iako je u slučaju sigmoidne funkcije interval aktivacija ograničen između 0 i 1, što je svojevrsna prednost u odnosu na *ReLU* koja je neograničena u smjeru pozitivnih vrijednosti aktivacija (a jako izrazite aktivacije nisu poželjne), glavni je problem što se za aktivacije bliske 0 i 1 gradijent jako smanjuje, odnosno učenje se jako usporava. Navedeni problem poznat je kao nestajući gradijent (engl. *vanishing gradient*) To otežava treniranje mreže jer, osim sporosti, posljedica je i "upadanje" u lokalne minimume funkcije pogreške predikcije.



Sl. 2.12: Zasićenje derivacije sigmoidne funkcije [9]

Tangens hiperbolna funkcija je pak samo skalirana sigmoidna funkcija:

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

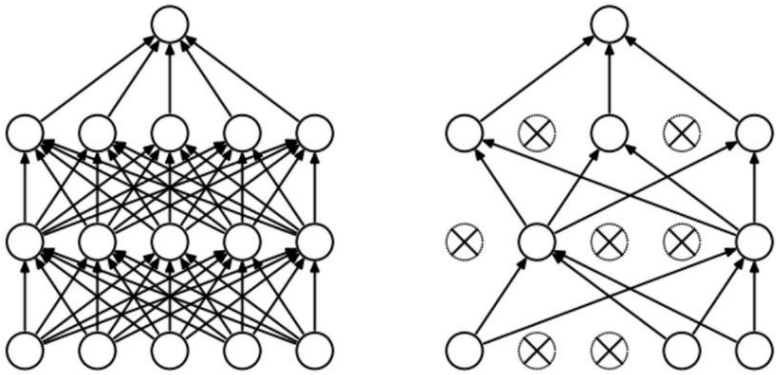


Sl. 2.13: Usporedba sigmoidne i tangens hiperbolne aktivacijske funkcije [10]

Prednost tangens hiperbolne funkcije nad sigmoidnom je prvenstveno izraženiji gradijent u ulaznom intervalu između -1 i 1, što pogoduje brzini treninga, ali se nailazi na isti problem nestajućeg gradijenta izvan tog područja.

2.2.1.4 *DROPOUT* SLOJ I UNAKRSNA PROVJERA

Jedan od jednostavnijih postupaka za smanjenje sklonosti mreže da razvije pretreniranost je primjena *dropout* slojeva u fazi treniranja neuronske mreže, a to donosi nasumično isključivanje određenog broja neurona s obzirom na oba prolaza kroz mrežu (unaprijed i unatrag) kako bi se mrežu učinilo redundantnijom i smanjilo usvajanje svojstava koja predstavljaju šum, kao i suzbilo stvaranje međuovisnosti između pojedinih neurona. S druge strane, primjenom *dropout* slojeva produljuje se vrijeme koje je mreži potrebno za konvergenciju.



Sl. 2.14: Deaktivacija (*dropout*) pojedinih neurona [11]

Često korištena metoda kojom se može ocijeniti sklonost neuronske mreže prenaučenosti jest k-struka unakrsna provjera (eng. *k-fold cross validation*). Tom se metodom može dati procjena testne pogreške. Sam postupak sastoji se u dijeljenju ukupnog podatkovnog skupa na k podskupova i potom se u svakoj od k iteracija izvodi eksperiment gdje se jedan od tih k podskupova odabere kao testni, a nad preostalih k-1 mreža se trenira. Nakon toga, rezultati iz k eksperimenata se usrednjavaju kako bi se dobila ocjena performansi prediktivnog modela. Na taj se način maksimiziraju dostupni podaci u odnosu na standardnu statičku podjelu podataka na podskup za trening i validaciju. Naime, svaki se podatak iskoristi i za učenje i za evaluaciju modela, no to se učini samo jednom za svaki od uzoraka.

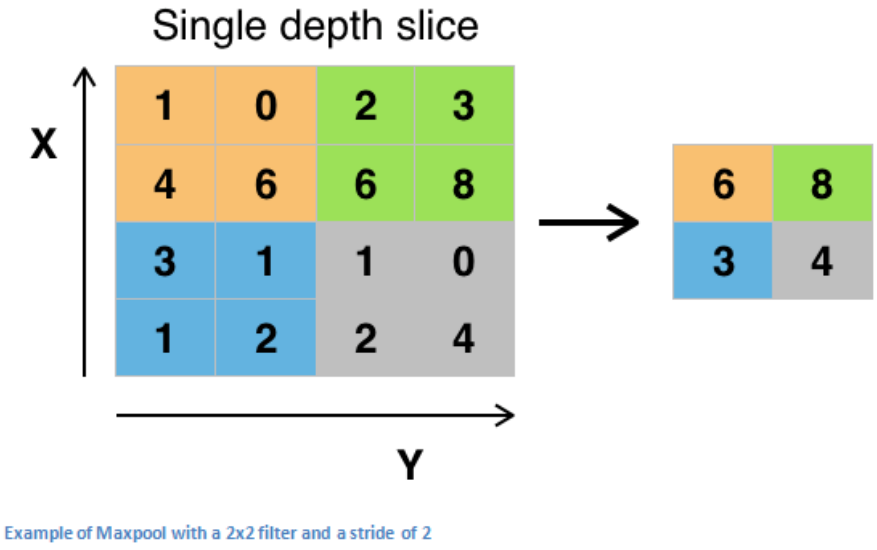
Metoda se uglavnom koristi na manjim podatkovnim skupovima kada podataka nema dovoljno da bi se skup podijelio na isključive podskupove za učenje i validaciju kao u uobičajenim slučajevima. Ukupan podatkovni skup najčešće se pritom dijeli na 10 podgrupa (k = 10). Značajno je i to da je sam postupak skup za veliku vrijednost parametra k radi toga što metoda zahtijeva k treniranja modela.

2.2.1.5 SLOJ SAŽIMANJA

U okviru sloja sažimanja (eng. *pooling layer*), manji segmenti mape aktivacija proizašle iz prethodnog konvolucijskog sloja svode se na pojedinačne elemente, što rezultira smanjenjem dimenzija mape, a to pak povlači manji broj potrebnih računskih operacija u kasnijim slojevima. Također, tim se postupkom povećava prostorna neosjetljivost neuronske mreže (neosjetljivost s obzirom na prostorne pomake objekata unutar uzorka). Sloj sažimanja pritom se može realizirati na različite načine.

Najčešće korišteno je sažimanje usrednjavanjem gdje se pojedinačna vrijednost nad malim segmentom volumena izvede kao aritmetička sredina grupiranih elemenata ili se pak uzme maksimalna vrijednost među elementima (eng. *max pooling*). Na donjem primjeru radi se o sažimanju maksimalnom vrijednošću gdje je polje nad kojim se sažimanje radi dimenzija 2 x 2, te je izlazna mapa značajki 4 puta manja (faktor skaliranja jednak je 2).

Ono u čemu se ove dvije metode sažimanja također razlikuju jest to što je kod sažimanja maksimalnom vrijednošću potrebno memorirati lokaciju elemenata s maksimalnim vrijednostima (onih koji su izvučeni iz podsegmenata), dok kod sažimanja srednjom vrijednošću to nije slučaj.



Sl. 2.15: Postupak sažimanja maksimalnom vrijednošću [2]

2.2.1.6 POTPUNO POVEZAN SLOJ

Potpuno povezan sloj (eng. *fully connected layer*) uvijek dolazi na kraju mreže kao posljednji sloj. Nakon što je stvoren prostor značajki (eng. *feature space*) prolaskom ulaznog volumena kroz sve prethodeće slojeve, taj završni sloj ima ulogu konačne klasifikacije ulazne slike te kao izlaz daje vektor duljine N, gdje je N broj klasa objekata definiranih u podatkovnom skupu. Pojedina vrijednost unutar vektora predstavlja vjerojatnost da za ulazni volumen pripada odgovarajućoj klasi od tih N klasa. Primjerice, za N = 10, gdje se radi o problemu prepoznavanja pojedine znamenke, gdje je izlazni vektor [.05 .05 0 0 .8 0 0 .05 0 .05], najveća je vjerojatnost da se na ulazu pojavila znamenka 4 i ona iznosi 80%.

2.3 TRENIRANJE NEURONSKE MREŽE

Kao što je već naznačeno, proces učenja neuronske mreže nad određenim podatkovnim skupom podrazumijeva konstantna ažuriranja vrijednosti težina te mreže. Ta podešavanja težinskih faktora vode ka smanjenju greške klasifikacije na minimum. Težine filtra prije samog učenja imaju slučajne vrijednosti koje ni na koji način nisu uvjetovane podatkovnim skupom za učenje ili validaciju. Slično kao što čovjek u najranijoj dobi upoznaje okolinu oko sebe povezujući određeni naziv predmeta s njegovom vizualnom pojavom, tako i neuronska mreža uči parametre prolaskom kroz veliku količinu slikovnih podataka uparenih s odgovarajućim nazivom klase. Spomenuta metoda spada u klasu metoda nadziranog učenja (eng. *supervised learning*). Algoritam čija je uporaba najraširenija za treniranje neuronskih mreža i općenito najpopularniji za nadzirano učenje kod višeslojnih perceptrona naziva se algoritmom propagacije pogreške unatrag (eng. *error backpropagation*). Glavni razlog radi kojeg je algoritam toliko zastupljen je što se njime postiže najbrži spust u prostoru težina. Algoritam je detaljnije opisan u nastavku.

2.3.1 ALGORITAM PROPAGACIJE POGREŠKE UNATRAG

Algoritam propagacije pogreške unatrag ima četiri karakteristične faze – unaprijedna faza (eng. *forward pass*), izračun pogreške, unazadna faza (eng. *backward pass*) i ažuriranje težina. Vratimo li se na spomenuti primjer s klasifikacijom znamenki, mreža na samom početku neće davati izrazitu preferenciju bilo kojoj klasi (znamenki), što znači da će izlazni vektor vjerojatnosti imati ujednačene vrijednosti, dok će ciljani rezultat imati vjerojatnosti jednake nuli za sve znamenke osim one koja se pojavila na ulazu. Radi toga pogreška u početnoj fazi treniranja ima veliku vrijednost.

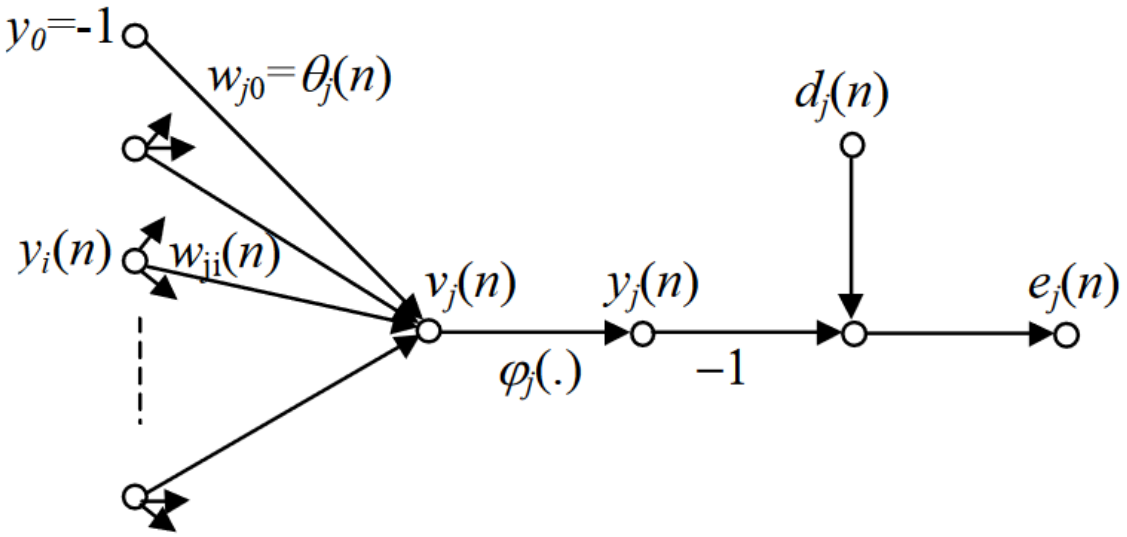
Jedan od češće korištenih načina za određivanje pogreške jest prema srednjem kvadratnom odstupanju (eng. *mean square error*):

$$E_{total} = \sum_i \frac{1}{n} * \left(x_{i,target} - x_{i,output}\right)^2 \qquad (2.5)$$

U spomenutom primjeru klasifikacije znamenki, vektori xtarget i xoutput bi na početku faze učenja mogli izgledati ovako (znamenka na ulazu je 5): x_{target} = [0 0 0 0 0 1 0 0 0 0], x_{output} = [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1].

Dakle, ovaj se optimizacijski problem sastoji u pronalasku minimuma pogreške između predikcije koju radi neuronska mreža i ciljane vrijednosti s obzirom na težinske faktore. Prve dvije od spomenutih faza obavljene su prolaskom slike kroz mrežu i određivanjem pogreške nastale pogreške u predikciji prema izrazu (2.5), nakon čega slijedi ključan korak, a taj korak podrazumijeva propagaciju izračunate pogreške u obrnutom smjeru (prema prvom sloju) i određivanje gradijenta pogreške.
Gradijent pogreške, odnosno parcijalne derivacije pogreške po težinama unutar slojeva neuronske mreže, govori koliko se intenzivno mijenja vrijednost pogreške s obzirom na težine, a određuje se kako bi se znalo na koji način podesiti težine da bi se greška postupno odvela u minimum. Korekcija težine pritom se odvija u pravcu negativnog gradijenta [7].

Sada uzmimo sljedeći primjer mreže kako je prikazano:



Sloj s neuronom *j* uzimamo kao izlazni te gledano iz tog sloja počinje određivanje pogreške rekurzivnim računanjem za svaki neuron pojedinačno sloj po sloj.
Težinski faktori veza između neurona *j* i onih iz prethodnog sloja označeni su s w_{ji}.

Ulaz bloka aktivacijske funkcije označen s v_j jednak je:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{2.6}$$

pri čemu y_i predstavlja izlaz i-tog neurona prethodnog sloja.

Zatim imamo konačni izlaz y_j neurona nakon aktivacijskog bloka:

$$y_j(n) = \varphi_j\left(v_j(n)\right) \tag{2.7}$$

gdje aktivacijska funkcija djeluje na izlaz neurona v_j.

Gradijent se potom računa na sljedeći način:

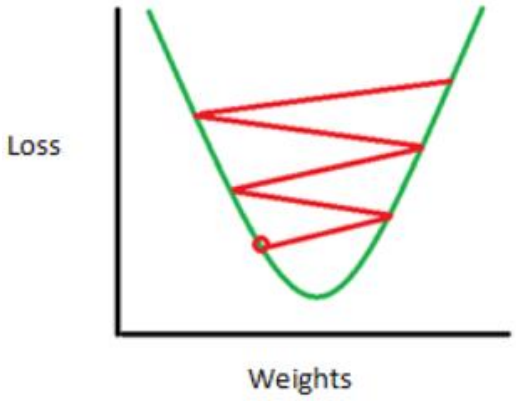
$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$
$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \qquad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$
$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \qquad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.8}$$

Kada se gradijent odredi, sljedeći korak je konačno ažuriranje težina, što je općenito dano sljedećim izrazom (metoda najbržeg spusta):

$$w_{ji} = w_{ji-1} - \eta * \frac{dE}{dW_{ji-1}} \tag{2.9}$$

Parametar η predstavlja brzinu učenja (eng. *learning rate*) i radi se o veličini koju sami određujemo.

Veća vrijednost znači bržu konvergenciju težinskih faktora prema vrijednosti minimuma pogreške (dakle postupak treniranja neuronske mreže kraće traje), no to ujedno znači veću nestabilnost i može značiti da točka minimuma nikada neće biti dosegnuta zbog prevelikih skokova [2]. Međutim, nestabilniji proces učenja ima jednu karakterističnu prednost, a to je da je takav proces manje sklon upadanju u lokalne minimume gdje bi se onda završilo treniranje bez da je pronađen stvarni minimum pogreške.



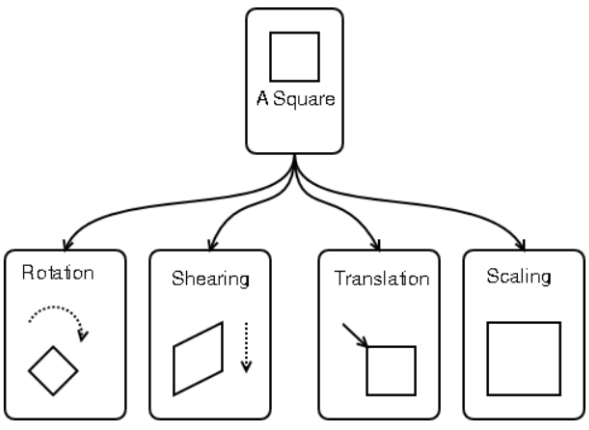
Sl. 2.16: Problem previsoke brzine učenja [1]

3. REALIZACIJA NEURONSKE MREŽE

3.1 PRIPREMA PODATAKA

Kao što je već naznačeno, konvolucijske neuronske mreže zahtijevaju velike količine podataka ako se računa na imalo reprezentativne rezultate. Pod reprezentativnim rezultatom smatra se točnost predikcije od strane istreniranog modela nad novim podacima iskazana kao udio ispravno prepoznatih uzoraka. Veličina tog udjela naravno ovisi o vrsti uzoraka i klasifikacije koja se izvodi. U principu je poželjno raspolagati skupom reda veličine tisuću ili deset tisuća uzoraka po klasi. Te su količine podataka potrebne jer je postupak ekstrakcije svojstava i struktura iz podataka spor, a posebno je to istaknuto kod visoko dimenzionalnih ulaznih uzoraka kao što su slikovni podaci. Klasifikacija slika u pravilu zahtijeva kompleksne modele s velikim brojem parametara, no da bi razvoj tako kompleksnih modela imao smisla, ključno je raspolagati izrazito velikim podatakovnim skupom jer je uspješnost optimizacije parametara mreže proporcionalna broju iteracija u procesu treniranja mreže. Tako na primjer jedna od poznatijih dubokih neuronskih mreža za klasifikaciju slika *VGGNet* (pobjednik natjecanja *ILSVRC* 2012. godine) ima 140 milijuna parametara.

Može se reći kako je, barem u većini slučajeva, sinteza kvalitetnog podatkovnog skupa na kojem će neuronska mreža trenirati veći i prioritetniji zadatak od samog dizajna mreže. Međutim, u nekim slučajevima dovoljno veliki podatkovni skupovi nisu izvorno dostupni ili bi izgradnja takvog skupa "ručnim" metodama bila predugotrajan postupak. Ručni postupak podrazumijevao bi npr. ručno (pojedinačno) uzimanje novih slika, te pridruživanje oznake klase svakoj slici. Radi toga se vrlo često nad slikama primjenjuju određene transformacije kako bi se uvećala izvorna količina podataka (engl. *data augmentation*) [16]. Pritom se može raditi o vrlo raznolikim metodama u ovisnosti o tome s kakvim podacima se radi, no za slikovne podatke karakteristične su afine transformacije poput skaliranja, rotacije ili smika (engl. *shearing*).



Sl.3.1: Afine transformacije (engl. *affine transformation*) [14]



Sl. 3.2: Primjer transformacija nad izvornom slikom [15]

Pored spomenutih metoda, nad slikovnim se podacima često primjenjuju i modifikacije parametara poput osvjetljenja ili pak dodavanje šuma. Osim samog povećanja podatkovnog volumena, dodatna prednost koja se ostvaruje primjenom transformacijskih postupaka na izvornim podacima jest to da model treniran na takvom podatkovnom skupu postaje neosjetljiviji upravo na spomenute varijacije, što je u realnim uvjetima vrlo poželjno svojstvo modela. Može se reći da su na taj način uklonjene irelevantne karakteristike iz podataka.
Naime, kada je cilj ispravna klasifikacija slike, na samu klasifikaciju ne bi trebao utjecati npr. položaj ili orijentacija objekta unutar slike. Osim toga, na taj se način ujedno i smanjuje sklonost modela razvijanju problema pretreniranosti, odnosno takav model ima veću sposobnost generalizacije. Kao što se može i pretpostaviti, upravo su modeli trenirani na malim podatkovim skupovima loši u pogledu generalizacije. U konačnici, ono što model čini kvalitetnim, a što proizlazi iz podatkovnog skupa na kojem je treniran jest balans u pogledu količine i relevantnosti karakteristika ekstrahiranih iz podataka.

3.2 KORIŠTENE TEHNOLOGIJE

3.2.1 *TENSORFLOW*

TensorFlow je biblioteka otvorenog koda (engl. *open source*) namijenjena za numeričke proračune izvorno razvijena od strane *Google Brain* istraživačkog tima za potrebe internih projekata. Jedna od istaknutih primjena ovog *frameworka* upravo je u području strojnog učenja, a posebno se to odnosi na duboko učenje. Karakteristika ove biblioteke je i mogućnost razvoja na različitim platformama, što podrazumijeva opciju iskorištenja grafičke procesorske jedinice, a to je od velikog značaja budući da su višeslojne neuronske mreže i treniranje pripadajućih modela računski vrlo zahtjevni pa je time vrijeme razvoja znatno skraćeno. Jezgru računskog modela oko kojeg je *TensorFlow* baziran čini struktura koja se naziva tenzor.

Ostavljajući po strani stroge matematičke definicije, može se reći da je tenzor višedimenzionalno polje podataka nad kojim su definirana određena transformacijska svojstva. Ta svojstva opisuju promjene spomenutog polja kao rezultat matričnih operacija. Modeliranje neuronske mreže primjenom *TensorFlow-a* podrazumijeva definiranje instance *dataflow* grafa kojeg čini niz objekata koji predstavljaju jedinice izračuna (engl. *units of computation*). *TensorFlow* grafovi prevode se u visoko optimiziran *C++* programski kod. U radu se *TensorFlow* ne koristi izravno već se pozivi njegovih konstrukata ostvaruju kroz *Keras* sučelje.

3.2.2 *KERAS*

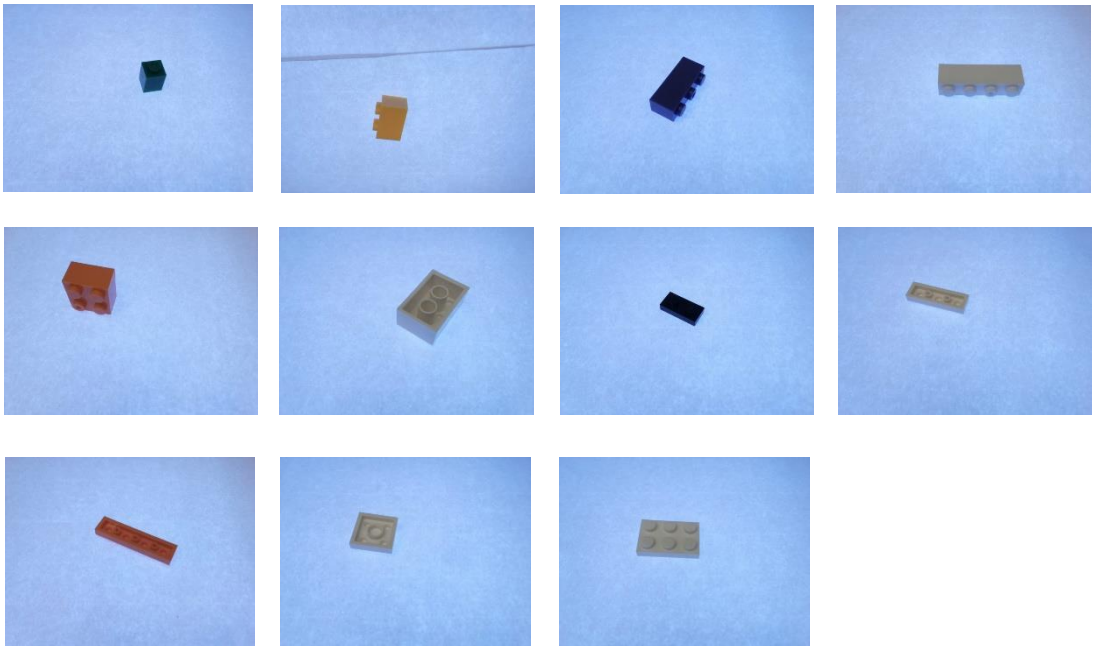
Keras je aplikacijsko programsko sučelje (engl. *application programming interface*) visoke razine pisano u programskom jeziku Python, a omogućuje definiciju i treniranje kompleksnih neuronskih mreža te funkcionira kao sloj iznad *TensorFlow-a* [17]. *Keras* ima važnu primjenu u području dubokog učenja jer je njime ostvariv jednostavniji dizajn neuronske mreže gdje je fokus izravno na realizaciji apstraktnih struktura koje su predstavljene u teorijskom dijelu o konvolucijskim neuronskim mrežama (slojevi, aktivacijske funkcije, itd.). Dakle, sam proces dizajna je time intuitivniji i brži, a programer na raspolaganju ima velik broj pripremljenih implementacija spomenutih elemenata mreže.

3.3 PRIMJER IMPLEMENTIRANE ARHITEKTURE

U ovom će dijelu biti izneseni detalji arhitekture i parametara konvolucijske neuronske mreže realizirane spomenutim tehnologijama. Za treniranje i validaciju mreže korišten je relativno mali podatkovni skup podijeljen na 11 klasa s približno 200 do 250 uzoraka po klasi. Validacijski skup sastoji se od ukupno 200 slika.

Budući da se radi o ograničenom skupu podataka, nad skupom su primijenjena 4 tipa transformacija – skaliranje slike, uvećanje (engl. *zoom*), zrcaljenje slike po horizontalnoj osi i smik, čime je izvorni podatkovni volumen upeterostručen. Slike su izvorno velikih dimenzija (640x480), te im je prije bilo kakve obrade i treniranja mreže smanjena veličina na dimenzije 64 x 48.

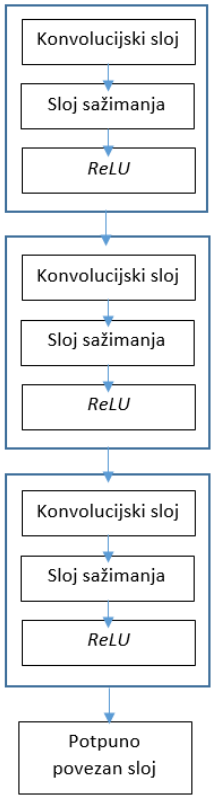
Što se tiče arhitekture same mreže, u početku je odabrana relativno jednostavna izvedba s tri povezane kombinacije konvolucijskog i sloja sažimanja maksimalnom vrijednošću nakon kojeg dolazi aktivacijska funkcija, a potom dolazi potpuno povezan sloj. Konvolucijski slojevi imaju po 32 filtra dimenzija 3 x 3, sažimanje se izvodi po blokovima veličine 2 x 2, te se koristi *ReLU* aktivacijska funkcija.



Sl. 3.3: Uzorci skupa za treniranje mreže iz 11 klasa

Za prvu iteraciju treniranja mreže, evaluacija točnosti klasifikacije odvija se nakon obrade svakih 10 ulaznih uzoraka (engl. *batch*), te se izvodi 10 *epoha* treniranja, odnosno mreža se 10 puta trenira na ukupnom podatkovnom skupu za učenje. Od ostalih hiperparametara, korak pomaka filtra (engl. *stride*) iznosi 1, te se ne primjenjuje proširenje ulaznog volumena dodatnim poljima s nulama (engl. *zero padding*).

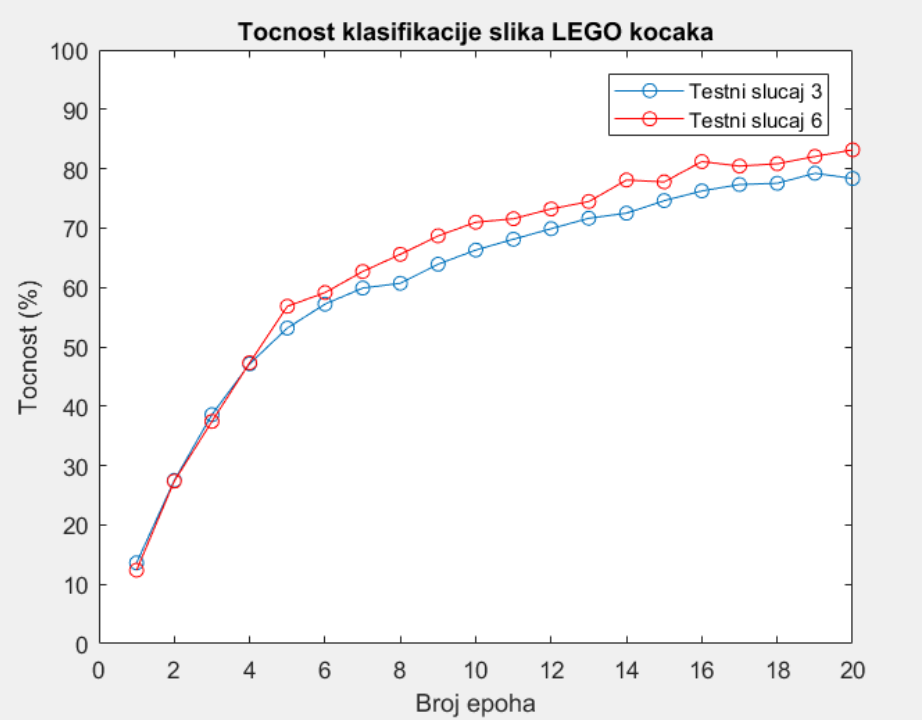
Takvom je mrežom u prvoj trening iteraciji (*testni slučaj 1)* postignuta točnost klasifikacije (engl. *accuracy*) od 65,44 %. Vrijeme treniranja mreže po epohi iznosi nešto više od 1 minute. U idućem eksperimentu, mreža je učinjena nešto kompleksnijom povećanjem broja filtara u konvolucijskim slojevima (*testni slučaj 2*) na 64. Rezultat je iznosio 69,02 %.
Ovo povećanje kompleksnosti gotovo uopće nije usporilo treniranje mreže. Zatim je isproban utjecaj broja epoha treniranja iste mreže. Povećanjem broja epoha na 20, dobiva se točnost od 79,23 % u 19. epohi, no nakon obavljenih 20 epoha, točnost počinje opadati (*testni slučaj 3*). Daljnjim eksperimentima u pravcu povećanja kompleksnosti početne mreže, isprobano je dodavanje još dviju spomenutih kombinacija skrivenih slojeva (konvolucijski sloj – sloj sažimanja – *ReLU*; *testni slučaj 4*) u odnosu na testni slučaj 2 te je zabilježena točnost klasifikacije od 71,10 %, a potom je i uvećan broj filtara konvolucijskog sloja na 128 (*testni slučaj* 5) uz rezultat od 74,69 %. Oba rezultata odnose se na treniranje kroz 10 epoha. Vrijeme treniranja mreže povećalo se time za približno 50 % u odnosu na prve dvije implementacije.
Kao posljednji primjer, mreža iz posljednjeg testnog slučaja trenirana je kroz 20 epoha (*testni slučaj 6*). Tim je eksperimentom u posljednjoj epohi dobivena točnost od 83,17 %
U konačnici, primjetno je da kompleksnija mreža ima veći potencijal za učenje parametara nelinearnog procesa nad ovakvim podatkovnim skupom, što je i očekivano.



Sl. 3.4: Početno odabrana arhitektura mreže za klasifikaciju slika *LEGO* kocaka (prikazano radi jednostavnosti – isprobani kompleksniji testni primjeri s više slojeva donosili su samo dodatne blokove konvolucija – sažimanje – *ReLU*) – testni slučaj

Broj testnog slučaja	Broj kombinacija skrivenih slojeva (konvolucija - sažimanje – *ReLU*)	Broj filtara konvolucijskog sloja	Broj epoha	Točnost klasifikacije - trening [%]	Točnost klasifikacije-validacija [%]
1	3	32	10	65,44	57,23
2	3	64	10	69,02	60,85
3	3	64	20	79,23	68,82
4	5	64	10	71,10	64,21
5	5	128	10	74,69	69,25
6	5	128	20	83,17	74,26

Tablica 3.1: Obrađeni testni slučajevi treniranja mreže i dobiveni rezultati



Sl. 3.5: Usporedba procesa učenja mreže na dva najuspješnija testna slučaja

Konačno, postupkom validacije se istrenirane mreže testiraju na podatkovnom skupu koji je obično nešto različit od testnog jer se na taj način provjerava najvažnija sposobnost neuronske mreže, a to je sposobnost generalizacije. Budući da je izvorni podatkovni skup u ovom slučaju relativno ujednačen po svojim karakteristikama, odnosno same slike se ne razlikuju u znatnoj mjeri s obzirom na svojstva kao što su osvjetljenje, pozadina, položaj objekta na slici i sl., da bi se testirala generalizacijska svojstva istrenirane mreže, korištene su nešto različite postavke transformacija nad podskupom za trening i onim za validaciju.

U konačnici, najbolji validacijski rezultat postignut je u testnom slučaju 6 kao što je i očekivano te iznosi 74,26 %. Primjetna su znatnija odstupanja točnosti prilikom validacije u odnosu na točnost postizanu u toku treninga, što se u najvećoj mjeri pripisuje ipak nedovoljno velikom podatkovnom skupu.

ZAKLJUČAK

Radom su razjašnjene teorijske pretpostavke koje stoje iza primjene konvolucijskih neuronskih mreža na problemima klasifikacije slikovnih podataka. Pritom su obrađene sve značajnije specifičnosti dizajna duboke neuronske mreže, u prvom redu uloge pojedinih slojeva i način funkcioniranja algoritma propagacije pogreške unatrag kojim se ažuriraju težinski faktori filtara tijekom treniranja mreže. Također, pokriven je i problem pripreme podataka za treniranje mreže, što je vrlo značajno za uspješnost mreže u klasifikaciji podataka.
Cilj rada bio je dizajn i učenje mreže za klasifikaciju slika *LEGO* kocaka. U izradi rješenja korištena je numerička biblioteka *TensorFlow* u kombinaciji s aplikacijskim sučeljem *Keras*. Također, korišten je jedan od već dostupnih podatkovnih skupova koji se sastoji od 11 klasa *LEGO* kocaka sa po otprilike 200 uzoraka slika po klasi nad kojima je zatim primijenjen niz afinih transformacija radi uvećanja samog skupa i postizanja boljih rezultata klasifikacije.
Sama implementacija i treniranje mreže obavljeno je kroz 6 testnih slučajeva gdje je najbolji rezultat klasifikacije iznosio 83,17 %, a validacije 74,26 %.
U daljnjem radu fokus bi bio prvenstveno na podatkovnom skupu budući da se taj element pokazao najzahtjevnijim na zadanom problemu. S obzirom da se radi o klasifikaciji *LEGO* kocaka gdje je prisutan vrlo veliki broj različitih klasa kocaka, a reprezentativne količine podataka nisu lako dostupne, sinteza bogatijeg podatkovnog skupa doprinijela bi kvaliteti procesa treniranja mreže. Nadalje, preporučuje se mrežu trenirati na modificiranoj arhitekturi, s drukčijim hiperparametrima i aktivacijskim funkcijama, a ujedno i primijeniti neke dodatne tehnike transformacije izvornog skupa podataka.

LITERATURA

[1] https://en.wikipedia.org/wiki/ImageNet

[2] https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[3] https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[4] https://en.wikipedia.org/wiki/Feedforward_neural_network

[5] https://raghakot.github.io/2017/01/03/Exploring-alternative-neural-computational-models.html

[6] http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

[7] http://neuralnetworksanddeeplearning.com/chap2.html

[8] http://cs231n.github.io/neural-networks-1/

[9] https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class03_tensor_flow.ipynb

[10] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6?gi=df368c769b68

[11] https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[12] http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex7/ex7.html

[13] https://www.tinymind.com/learn/terms/relu

[14] https://www.technologyreview.com/s/530561/the-revolutionary-technique-that-quietly-changed-machine-vision-forever/

[15] https://en.wikipedia.org/wiki/Overfitting

[16] https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

[17] https://people.gnome.org/~mathieu/libart/libart-affine-transformation-matrices.html

[18] https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

[19] https://www.tensorflow.org/guide/keras

SAŽETAK

Rad donosi teorijsku pozadinu potrebnu za razumijevanje funkcioniranja i realizacije konvolucijskih neuronskih mreža, danas najzastupljenije metode dubokog učenja na problemima klasifikacije slikovnih podataka. Cilj rada bio je demonstrirati izradu konvolucijske neuronske mreže za rješavanje problema klasifikacije slika *LEGO* kocaka. U rješavanju zadatka korištene su tehnologije *TensorFlow* i *Keras* pomoću kojih je neuronska mreža dizajnirana, trenirana i evaluirana. Na samom kraju rada opisani dobiveni rezultati i zaključci.

Ključne riječi: duboko učenje, umjetne neuronske mreže, konvolucijske neuronske mreže, klasifikacija slika, algoritam propagacije pogreške unatrag

ABSTRACT

Title: The application of machine learning for *LEGO* brick classification

This work presents the theoretical background required in order to understand the way of functioning and implementation of convolutional neural networks which are the most widely used method of deep learning applied for image classification problems nowadays. The aim of this work was to demonstrate the design of a convolutional neural network applied on a *LEGO* brick image classification problem. *TensorFlow* and *Keras* were the technologies used in solving this task and by means of which the neural network was designed, trained and evaluated. The results and conclusions are described in the final part of the work.

Keywords: deep learning, artificial neural networks, convolutional neural networks, image classification, backpropagation algorithm

ŽIVOTOPIS

Ervin Hamzić rođen je 23. veljače 1994. godine u Zagrebu gdje je pohađao osnovnu i srednju školu (Prva gimnazija Zagreb). Po završetku gimnazije upisuje preddiplomski studij elektrotehnike na Fakultetu elektrotehnike i računarstva u Zagrebu 2012. godine. Studij od 2016. godine nastavlja na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Komunikacije i informatika. Trenutno je student 3. godine.
2018. godine na prvom izdanju natjecanja *STEM Games* za studente osvaja 4. mjesto u finalu *Technology* arene. Aktivno se služi engleskim i njemačkim jezikom u govoru i pismu.

Ervin Hamzić

_____