UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

TEMA

MASTER RAD

Studijski Program: Računarstvo i informatika

Mentor: Kandidat:

Prof. dr Leonid Stoimenov Aleksandra Stanojević 918

Niš, oktobar 2020. godine

Master rad
TEMA

Zadatak:
Proučiti….

Student:

Aleksandra Stanojević 918

Datum prijave rada: _____

Datum predaje rada: _____

Datum odbrane rada: _____

Komisija za odbranu:

_____

Predsednik

_____

Član

_____

Član

TEMA

SAŽETAK

//TODO

Ključne reči: Duboko učenje, Klasifikacija, Koronavirus, COVID-19, Konvolucijske neuronske mreže, Rendgenske slike pluća

TEMA NA ENGLESKOM

ABSTRACT

//TODO

Key words: Deep learning, Classification, coronavirus, COVID-19, Convolutional Neural Network, Chest X-Ray images

Sadržaj

Uvod

COVID-19 je novi koronavirus koji se pojavio krajem 2019. godine u Kineskoj oblasti Wuhan, a u 2020. godini proširio na ceo svet i postao svetski zdravstveni problem [1-5]. Ovaj virus inficira pluća i uzrokuje potencijalno smrtonosne respiratorne sindrome [6]. Prema poslednjim podacima ukupan broj potvrđenih slučajeva širom sveta je dostigao 33,502,322, od kojih je 1,005,281 smrtnih slučajeva, a države koje su najviše pogođene ovom pandemijom su USA, Indija, Brazil i Italija [7]. Glavna metoda za detekciju COVID-19 je lančana reakcija polimeraznom reverznom transkripcijom (skr. RT-PCR) koja može detektovati SARS-CoV-2 RNA iz respiratornih uzoraka (nazofaringealni i orofaringealni brisevi) [8]. Međutim, ova metoda je dosta specifična, dugotrajna (od nekoliko sati do celog dana), predstavlja naporan ručni postupak, a javlja se i problem nedostatka kompleta za testiranje usled veoma velike potražnje. Kako je testu potrebno više vremena za dobijanje rezultata u poređenju sa vremenom za širenje virusa među ljudima, uz ponekad i pogrešne rezultate (lažni negativni ili lažni pozitivni), mnogo istraživača radi na automatskoj detekciji virusa korišćenjem rendgenskih slika pluća (engl. Chest X-Ray - CXR) ili slika pluća korišćenjem metode kompjuterizovane tomografije (engl. CT scan). Neki radovi su čak i došli do zaključka da dijagnostikovanje korišćenjem slika pluća daje bolje rezultate od RT-PCR metode i da se treba uzeti u razmatranje da bude primarna tehnika za detekciju COVID-19 virusa [9]. Ovoj ideji doprinosi i činjenica da su medicinska istraživanja već pokazala da je infekcija pluća koju izaziva COVID-19 drugačija u odnosu na druge infekcije i da je na snimcima pluća radiolozi mogu detektovati [9-12].

Klasifikacija rendgenskih slika pluća, međutim, nije novi problem u polju veštačke inteligencije. Naime, konvolucijske neuronske mreže (*skr. CNN*) su jedan od najpopularnijih modela dubokog učenja koji se koristi za digitalnu obradu slike. Do prodora ove metode došlo je sa *ImageNet* takmičenjem [13], na kome je stopa greške za prepoznavanje objekata prepolovljena u odnosu na dotadašnje metode. U najnovijim istraživanjima akcenat je na korišćenju CNN-a, ili radi kreiranja metoda koji će najkorektnije detektovati segmente na snimcima pluća koji ukazuju na COVID-19 ili radi klasifikacije slika pluća na ona sa i bez COVID-19 infekcije, a sve sa ciljem, rane i brze detekcije i izdvajanja pacijenata koji su pozitivni na novi virus radi njihovog daljeg lečenja i izolacije koja bi sprečila širenje zaraze.

Ovaj rad se bavi proučavanjem digitalne obrade rendgenskih slika pluća korišćenjem dubokog učenja, odnosno konvolucijske neuronske mreže radi klasifikacije slika pluća sa COVID-19 infekcijom, bakterijskom infekcijom i slikama čistih pluća. Najpre je objašnjeno koje su to karakteristike koje se na slikama pluća mogu uočiti, a na osnovu kojih se mogu razlikovati pluća sa COVID-19 infekcijom. Zatim je dat osvrt na ranije radove na istu temu, a potom je

predstavljeno duboko učenje kao metoda za digitalnu obradu slika i detaljno obrađen metod konvolucijske neuronske mreže i načina njenog rada od prihvatanja ulaznih podataka do izdavanja rezultata. Takođe, predstavljene su tehnike augmentacije podataka i transfera znanja, jer je zbog male količine podataka bilo neophodno, na neki način, izbeći preterano podudaranje sa podacima za treniranje (*engl. overfitting*).

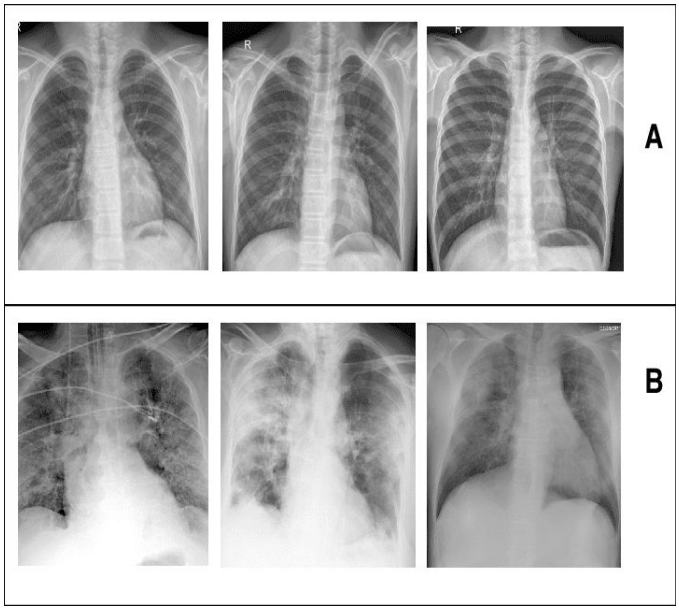
Teorijski koncepti obrađeni u radu su zatim korišćeni prilikom realizacije praktičnog dela zadatka. Najpre su predstavljene korišćene tehnologije, a zatim i sama implementacija, od skupa podataka koji su korišćeni, preko preprocesiranja, kreiranja konvolucijske neuronske mreže do prikaza matrica, dijagrama i evaluacije korišćenjem standardne metrike. Korišćena je tehnika augmentacije podataka, kao i tehnika transfera učenja i to za nekoliko modela CNN-a, a zatim vršeno i poređenje njihovih performansi.

Detekcija virusa COVID-19

Kao što je već rečeno, trenutno je RT-PCR osnovna metoda za detekciju virusa COVID-19, ali mnogo istraživača radi na automatskoj detekciji virusa korišćenjem radiografskih slika pluća (CXR i/ili CT slike) kao alternativnom metodu. Praksa je da je prvi korak korišćenje RT-PCR metode, a zatim se dalje lečenje, nakon što je virus detektovan, nastavlja uz stalno praćenje radiografskih slika od strane radiologa.

Medicinska istraživanja su pokazala da se kod pacijenata, koji su inficirani, mogu pomoću radiografskih snimaka uočiti karakteristične abnormalnosti. Na primer, Huang i drugi [11] identifikovali su da većina COVID-19 pozitivnih slučajeva iz njihovog istraživanja imaju bilateralne radiografske abnormalnosti u CXR slikama, dok su Guan i ostali [12] takođe uočili abnormalnosti u vidu "mlečnog stakla" (*engl. ground-glass opacity - GGO*), bilateralnih i intersticijskih (međuprostorna tečnost) abnormalnosti na CXR i CT slikama.

Naime, kada bolest napreduje u telu pacijenta pozitivnog na COVID-19 virus, dolazi do pojave tečnosti u malim vazdušnim vrećicama u plućima zvanima alveole [14]. Prisustvo ove tečnosti izaziva zapaljenje pluća, a napredak upale može se pratiti preko CXR i CT slika. Upala pluća se pojavljuje u vidu "mlečnih stakala", a moguće je i stvaranje "mlečnih konsolidacija" (spajanje više "mlečnih stakala" u jedno).



Slika XX: *Poređenje slika čistih i pluća sa COVID-19 infekcijom. Skup A su čista, a skup B COVID-19 pluća.*

Medicinsko osoblje mora imati određeni kriterijum na osnovu koga će doneti odluku da li je pacijentu potrebna terapija kiseonikom ili priključenje na respirator ili isključenje sa respiratora

(ako je u fazi oporavka). Vizuelizacija “mlečnih stakala” u CT slikama igra veoma važnu ulogu i od velike je pomoći medicinskom osoblju prilikom donošenja odluka. Takođe je veoma važno tačno utvrditi o kojoj vrsti plućne bolesti se radi pre primene bilo koje terapije, pa je stoga analiza korišćenjem CXR i CT slika pluća od višestrukog značaja.

Prednosti CXR slika u odnosu na CT slike pluća

Kao što je već prethodno rečeno, postoje dve mogućnosti kada govorimo o radiografskim snimcima, a to su CXR i CT slike pluća. U slučaju CT slika, imamo detaljniji prikaz grudnog koša u odnosu na CXR. Na primer, na CXR slikama se mogu uočiti abnormalnosti, dok CT treba da bude u stanju da pokaže tačnu lokaciju i ispita formaciju. CXR je 2D, a CT slike mogu biti i 3D. Prve se koriste radi ispitivanja gustih tkiva, dok CT može istovremeno da prikaže i kosti, meka tkiva i krvne sudove. Rentgenska oprema je mnogo manja i manje složena od CT skeniranja, jer se CT skener mora okretati oko pacijenta koji se skenira. [15]
Zbog ovih razlika se kroz dosta radova, od kojih će neki biti spomenuti u narednom poglavlju, provlači polemika oko izbora vrste radiografskih snimaka za detekciju COVID-19 virusa. U praktičnom delu ovog rada ispitivanja će se vršiti na CXR slikama, pa je stoga potrebno napomenuti koje su to karakteristike CXR snimaka koje su bile presudne prilikom izbora.

- **Brza trijaža**[1] - CXR snimci omogućavaju brzu trijažu pacijenata za koje se sumnja da su zaraženi COVID-19 infekcijom i može se vršiti paralelno sa virusnim testiranjem (za koje je potrebno više vremena) kako bi se broj pacijenata smanjio, naročito kada su u pitanju područja sa popunjenim kapacitetima (npr. New York, Italija, Španija). Takođe, ukoliko virusno testiranje nije moguće zbog nedostatka zaliha, CXR snimci predstavljaju dobru alternativu.
- **Dostupnost i pristupačnost** - CXR snimanje je dostupno i pristupačno u većini kliničkih centara i centara za snimanje, a pored toga smatra se standardnom opremom zdravstvenih sistema. Konkretno, CXR snimanje je mnogo dostupnije u odnosu na CT snimanje, naročito ako govorimo o zemljama u razvoju gde je trošak za opremu i održavanje CT skenera prevelik.
- **Prenosivost** - Postojanje prenosivih CXR sistema znači da se snimanje može izvoditi u domu osobe koja je u kućnoj izolaciji, čime se značajno smanjuje rizik od prenosa COVID-19 infekcije tokom transporta do zdravstvenih ustanova. Ovo nije slučaj sa CT skenerom koji predstavlja fiksni sistem i ne može se prenositi van zdravstvene ustanove.

[1] **Trijaža** je jedan od medicinskih sistema razvrstavanja pacijenata koji se primenjuje u uslovima masovnih događaja (velike nesreće, pandemije i sl), kako bi se omogućilo brzo prepoznavanje osoba koje će imati najveću korist ranog zbrinjavanja i brzog transporta u odgovarajuću zdravstvenu ustanovu.

Povezani radovi

Analiza i detekcija COVID-19 infekcije su obimno istraživane teme u poslednjih nekoliko meseci. Jedan deo radova bavi se procenom broja novih slučajeva infekcije, broja oporavljenih i broja umrlih, odnosno vrši predviđanje daljeg širenja pandemije. Drugi deo radova bavi se pitanjima vezanim za otkrivanje infekcije na osnovu različitih pristupa dubokog učenja koristeći CXR i/ili CT slike pluća. Kako je detekcija virusa COVID-19 koristeći duboko učenje i CXR snimke tema i ovog rada u nastavku će biti dat osvrt na do sada postignute rezultate na ovom polju i to koristeći skup podataka koji će biti korišćen i u praktičnom delu ovog rada.

Kao što je već prethodno rečeno klasifikacija rendgenskih slika pluća nije novi problem u polju dubokog učenja. Objavljeno je dosta skupova podataka i nad njima obučeno puno neuronskih mreža koje su dale jako dobre rezultate. Nedavno su Cohen i saradnici objavili otvoreni skup podataka COVID-19 [16] koji sadrži CXR i CT slike pluća ljudi za koje je potvrđena infekcija ovim virusom ili se sumnja na COVID-19 ili neku drugu bakterijsku ili virusnu infekciju. Od objavljivanja skupa podataka COVID-19, mnogi istraživači pokušali su da klasifikuju te slike i kreirali su test skupove kombinovanjem ovog skupa podataka i drugih skupova CXR slika. Neki od tih pristupa dati su u nastavku.

Narin i saradnici [17] su kreirali mali skup podataka sa 50 COVID-19 pozitivnih slika pluća preuzetih iz Cohen-ovog skupa i 50 slika zdravih pluća preuzetih sa Kaggle-a[2] i vršili klasifikaciju na te dve klase. Predložili su CNN model zasnovan na transferu učenja korišćenjem ImageNet skupa podataka (ResNet50, InceptionV3 and Inception-ResNetV2) da bi se klasifikacija mogla izvršiti na tako malom skupu podataka, a da pritom može da da dobre rezultate predviđanja. Istraživanje je pokazalo superiornost ResNet50 modela u pogledu tačnosti i u fazi obuke i u fazi testiranja. Koristili su 5-cross validaciju i dobili tačnost od 98%.

Apostolopoulos i saradnici [18] su kombinovali snimke iz Cohen-ovog skupa i nekoliko drugih izvora da bi kreirali veći skup od 224 COVID-19 pozitivnih slika pluća, 700 slika pluća sa infekcijama koje nisu COVID-19 i 504 slika zdravih pluća. Oni su takođe predložili CNN model zasnovan na transferu učenja korišćenjem ImageNet skupa podataka za pet modela. Rezultati su pokazali da VGG19 dostiže najveću tačnost i to 93.48%, a korišćena je 10-cross validacija.

[2] **Kaggle**, ćerka firma kompanije Google LLC, je online zajednica praktičara mašinskog učenja. Omogućava korisnicima da pronađu, preuzmu i objave skupove podataka, istražuju i grade modele mašinskog učenja, rade sa drugim članovima zajednice i učestvuju u takmičenjima za rešavanje aktuelnih izazova.

Wang i saradnici [19] su trenirali Covid-Net, novu arhitekturu predstavljenu u njihovom radu. Koristili su dosta veliki skup podataka sa 358 COVID-19 pozitivnih slika pluća, 5 538 slika pluća sa infekcijama koje nisu COVID-19 i 8 066 slika zdravih pluća. Dizajn CNN-a konstuisana je od kombinacije 1×1 konvolucija i residualnih modela da bi se omogućila dublja arhitektura, a korišćen je i obrazac projektovanje-ekspanzija-projektovanje-ekstenzija (*skr. PEPX*). Dostignuta tačnost u ovom radu je 92%.

Hemdan i saradnici [20] su kreirali programski okvir za duboko učenje, Covidx-Net i trenirali ga na skupom podaraka sačinjenom od 50 slika pluća, 25 COVID-19 pozitivnih slika pluća i 25 slika pluća koja nisu zaražena virusom COVID-19. Polovina slika preuzeto je iz Cohen-ovog skupa. Ovaj rad koristi transfer učenja korišćenjem ImageNet skupa podataka za sedam modela i predstavlja uporednu studiju različitih arhitektura dubokog učenja. Rezultati su pokazali da su superiornije VGG19 i DenseNet201 u odnosu na ostale i da dostižu tačnost od 90%. Korišćena je 5-cross validacija.

Naravno postoji još radova koji koriste slike iz Cohen-ovog skupa ili pak onih koji su kreirali sopstveni skup snimaka i koji je ostao zatvoren za javnost, te im se ne može pristupiti, ali ovde su spomenuti samo neki od važnijih radova. Tabela XX predstavlja poređenje opisanih modela.

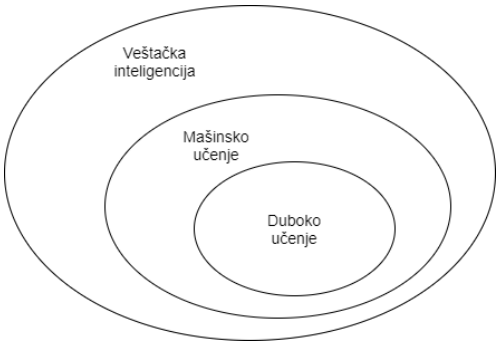
Studija	Skup podataka	Transfer učenja	Korišćeni model	Tačnost
Narin i saradnici [19]	50 COVID-19 50 zdravih pluća	Da (ImageNet skup)	ResNet50	98%
Apostolopoulos i saradnici [20]	224 COVID-19 700 pneumonia 504 zdravih pluća	Da (ImageNet skup)	VGG19	93.48%
Wang i saradnici [21]	358 COVID-19 5538 pneumonia 8066 zdravih pluća	Da (COVIDx skup)	Covid-Net	92%
Hemdan i saradnici [22]	25 COVID-19 25 zdravih pluća	Da (ImageNet skup)	Covidx-Net	90%

Tabela XX: *Poređenje opisanih modela dubokog učenja koji za klasifikaciju koriste CXR slike iz Cohen-ovog skupa podataka COVID-19*

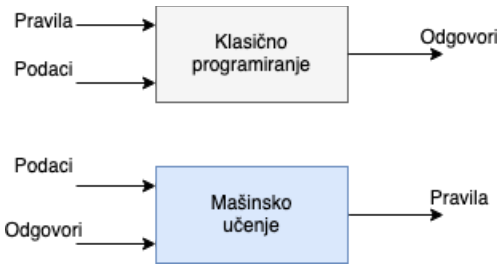
Duboko učenje

U poslednjih nekoliko godina veštačka inteligencija je bila predmet velike medijske pažnje. Mašinsko učenje, duboko učenje i veštačka inteligencija pojavljuju se u bezbroj članaka i to vrlo često izvan tehnološki orijentisanih publikacija. Međutim ova grana računarstva nije mlada i nova, već njeni začeci datiraju iz sredine prošlog veka.

Veštačka inteligencija nastala je još 1950-ih godina kada se prvi put javilo pitanje automatizacije intelektualnih zadataka koje obično obavljaju ljudi. Ona predstavlja širu oblast koja obuhvata mašinsko učenje i duboko učenje, kao i neke od pristupa koji uopšte ne uključuju učenje. Prvi programi bili su za igranje šaha sa računarom i oni su podrazumevali praćenje pravila, što je bila i dominantna paradigma sve do 1980-ih godina, a ovaj pristup poznat je kao *simbolička veštačka inteligencija*. Međutim, kako ovaj pristup nije mogao da reši složenije logičke zadatke, razvio se novi pristup, poznat kao mašinsko učenje. [21]



Slika XX: *Veštačka inteligencija, mašinsko učenje i duboko učenje*



Slika XX: *Mašinsko učenje kao novi pristup u računarstvu*

Mašinsko učenje počinje svoj razvoj sa pitanjem koje je postavio pionir veštačke inteligencije Alan Tjuring, a to je sposobnost mašine da samostalno uči kako da izvrši određen zadatak. U mašinskom učenju čovek dostavlja mašini podatke i odgovore koje očekuje, a ona sama definiše pravila na osnovu kojih će od podataka dobiti odgovore. Ono što je ideja ovog pristupa je pronalaženje transformacije koju je potrebno izvršiti nad podacima tako da se dobije odgovarajuće rešenje (odgovori). [21]

Jedan od načina podele algoritama za mašinsko učenje je na algoritme za :

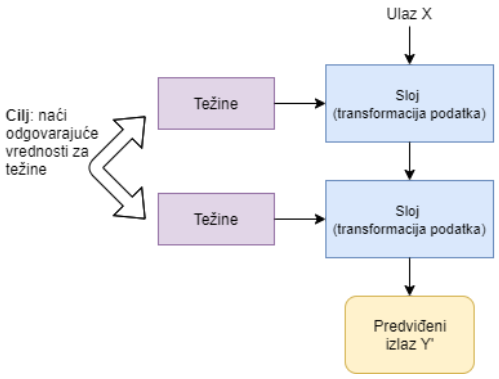
- Nadgledano učenje (*engl. Supervised learning*) - algoritmi koji grade matematički model iz skupa podataka koji sadrže i ulaze i željene izlaze;

- Polunadgledano učenje (*engl. Semi-supervised learning*) – algoritmi koji razvijaju matematičke modele iz nepotpunih podataka, gde deo ulaznih podataka nema ulazne labele;
- Nenadgledano učenje (*engl. Unsupervised learning*) - algoritmi koji razvijaju matematičke modele iz skupova podataka koji sadrže samo ulaze i ne sadrže željene izlazne modele;

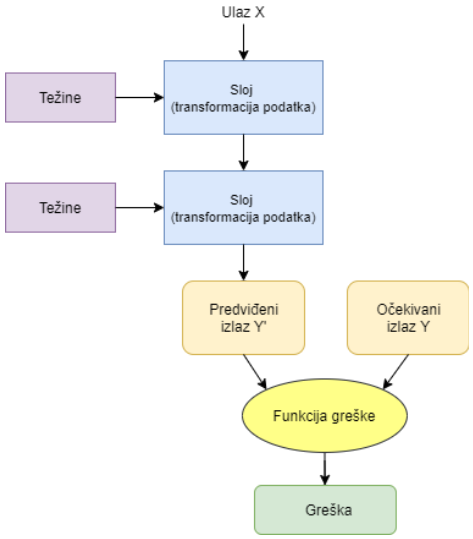
Duboko učenje je specifična podoblast mašinskog učenja koja podrazumeva pristup učenju kroz slojeve odnosno sukcesivne transformacije podataka. Dubina modela predstavlja koliko slojeva čini posmatrani model. Neki modeli uključuju desetine ili čak stotine uzastopnih slojeva, a učenje je automatsko prilikom izlaganja podacima za treniranje. Kod dubokog učenja, ovi slojeviti modeli su poznati pod nazivom neuronske mreže.

Kako funkcioniše duboko učenje?

Način na koji duboko učenje funkcioniše podrazumeva da se specifikacija onoga što sloj radi sa ulaznim podacima čuva u njegovim težinama ili parametrima sloja (Slika XX 1.7). U ovom kontekstu, učenje znači pronalaženje skupa vrednosti za težine svih slojeva u mreži koji će omogućiti pravilno mapiranje ulaznih podataka u odgovarajuće izlazne podatke. Međutim duboka neuronska mreža može sadržati desetine miliona parametara, pa ju je zbog toga nije jednostavno kontrolisati. Da bi mogli upravljati izlazom neuronske mreže, neophodno je izmeriti koliko je on daleko od očekivanog rezultata. Ovo je zadatak funkcije greške (*engl. loss function*) ili kako se drugačije naziva ciljne funkcije (*engl. objective function*). [21]

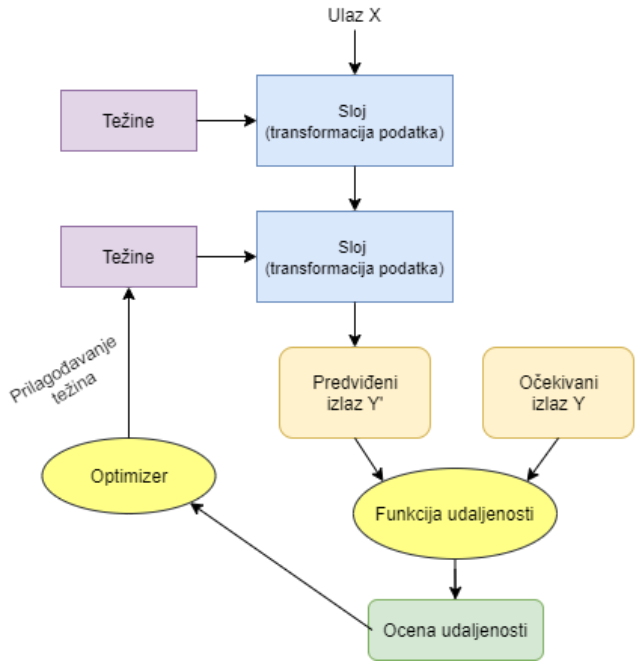


***Slika XX**1.7**:** Parametri neuronske mreže, odnosno težine*



***Slika XX**1.8**:** Funkcija greške meri kvalitet predviđanja neuronske mreže*

Funkcija greške uzima predviđanja koja su izlazne vrednosti mreže i stvarne ciljeve (izlazne vrednosti koje želimo da dobijemo od mreže) i izračunava grešku, beležeći postignuće mreže za konkretan skup ulaznih podataka. (Slika XX 1.8) Osnovni štos u dubokom učenju je upotreba ove greške kao vodilje za prilagođavanje težina u takvom smeru da će to izazvati smanjenje greške za posmatrani skup ulaznih podataka (Slika XX 1.9). Ovo prilagođavanje težina je posao **optimizera** (*engl. optimizer*) koji implementira takozvani ***Algoritam Bekpropagacije***, centralni algoritam u dubokom učenju.



Slika XX1.9***:*** *Optimizer koristi grešku da bi prilagodio težine*

U početku se težinama mreže dodeljuju slučajne vrednosti, tako da mreža sprovodi niz slučajnih transformacija. Naravno, rezultat je tada daleko od onog kakav bi trebao biti, pa je samim tim i greška velika. Ali sa svakim podatkom koji mreža obradi, težine se prilagođavaju i greška smanjuje. Ovo je petlja treninga koja, ponovljena dovoljan broj puta, daje vrednosti težinama koje minimizuju grešku. Mreža sa minimalnom greškom je ona čiji su izlazni podaci najbliži onima koje očekujemo, odnosno predstavlja istreniranu mrežu.

Algoritam Bekpropagacije

Iako su osnovne ideje o neuronskim mrežama nastale još 1950-ih godina, nekoliko decenija je bilo potrebno da se pronađe efikasan način za treniranje velikih neuronskih mreža. Naime ovaj problem rešen je pojavom Algoritma Bekpropagacije.

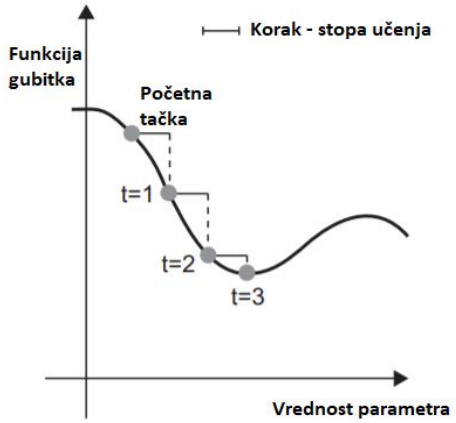
Najteži zadatak dubokog učenja je prilagođavanje težina modela (neuronske mreže) u takvom smeru da će to izazvati smanjenje greške. Međutim, postavlja se pitanje kako odrediti za pojedinačnu težinu u neuronskoj mreži da li je potrebno smanjiti je ili povećati i za koliko. Odgovor na ovo pitanje leži u činjenici da su sve operacije koje se koriste u neuronskoj mreži diferencijabilne, te se može izračunati gradijent na osnovu težina mreže. [21]

Stohastički opadajući gradijent

Ukoliko radimo sa diferencijabilnom funkciojom teoretski je moguće pronaći njen minimum, a to je tačka u kojoj je njen izvod jednak nuli. Potrebno je pronaći sve tačke u kojoj je izvod funkcije jednak nuli i zatim utvrditi za koju od tih tačaka funkcija ima najmanju vrednost i pronašli smo minimum funkcije.
U slučaju neuronske mreže, ukoliko želimo da pronađemo njen minimum potrebno je da odredimo vrednosti za težine mreže takve da daju najmanju moguću vrednost za funkciju greške. Međutim ako uzmemo u obzir da se neuronska mreža sastoji od nekoliko hiljada do desetina miliona parametara, razumećemo da je nemoguće na ovaj način pronaći njen minimum.
Umesto toga, koristi se algoritam u četiri koraka koji malo po malo modifikuje parametre mreže na osnovu trenutne vrednosti funkcije greške na slučajnoj grupi podataka [21]:

1. Određivanje x skupa podataka za obuku mreže i y skupa podataka koji predstavlja očekivane rezultate (ciljeve);
2. Treniranje neuronske mreže na x skupu podataka i dobijanje y_pred skupa podataka koji predstavljaju predviđene rezultate;
3. Izračunavanje greške, odnosno razlike između y i y_pred skupova podataka;
4. Izračunavanje gradijenta greške na osnovu parametara mreže (bekpropagacija);
5. Pomeranje parametara suprotno od gradijenta (W -= step * gradient), što izaziva smanjenje greške.

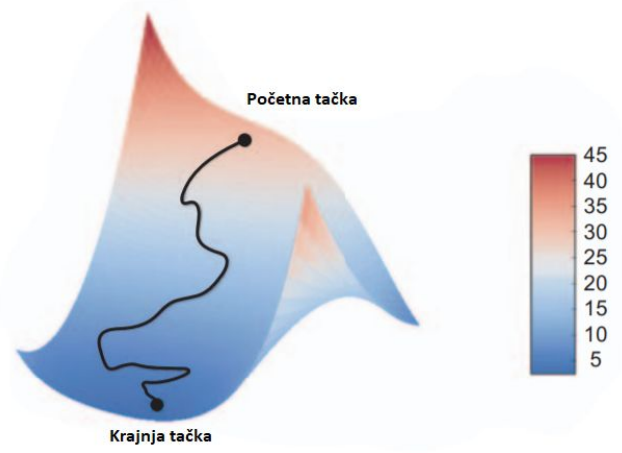
Opisani algoritam je poznat kao **stohastički opadajući gradijent mini-serije podataka**, a termin stohastički se odnosi na činjenicu da je svaka serija podataka koja se koristi za treniranje slučajna (stohastički je sinonim za slučajni). Slika XX ilustruje šta se dešava kada mreža ima samo jedan parametar i kada imamo jedan uzorak za treniranje (jednodimenzionalni prostor).
Ono što se može zaključiti na osnovu slike je da je potrebno pažljivo odabrati korak, odnosno stopu učenja. Ukoliko korak bude premali, biće potrebno mnogo iteracija da se dođe do minimuma ili se možemo zaglaviti kod lokalnog minimuma. Sa druge strane, ako je korak preveliki svaka iteracija nas može odvesti do kompletno slučajne lokacije na krivoj funkcije. [22]



Slika XX: *Stohastički opadajući gradijent mini serije u jednodimenzionalnom prostoru(jedan uzorak za treniranje)*

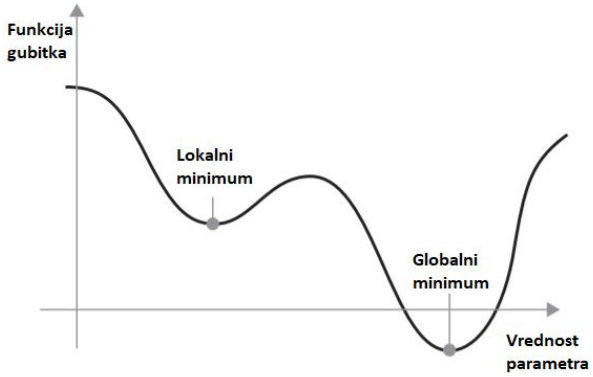
Varijanta sa korišćenjem stohastičkog opadajućeg gradijenta mini-serije podataka podrazumevala bi crtanje svakog uzorka (jedan element iz x) i cilja (jedan element iz y) za svaku iteraciju, umesto crtanja serije podataka (istinski stohastički opadajući gradijent). Prelazeći u drugu krajnost, postoji stohastički opadajući gradijent serije podataka, koji podrazumeva crtanje svakog uzorka za svaki korak, što bi dalo bolje rezultate, ali bi to bila jako skupa operacija. Najefikasniji kompromis između ove dve krajnosti je upotreba serija podataka razumne veličine. [22]

Kada govorimo o neuronskim mrežama, međutim, govorimo o **višedimenzionalnom prostoru**: svaka težina u neuronskoj mreži predstavlja novu dimenziju (od nekoliko hiljada do desetina miliona dimenzija). Na Slika XX data je vizuelizacija stohastičkog opadajućeg gradijenta u dvodimenzionalnom prostoru, da bi se mogla stvoriti neka slika o samom konceptu, jer je nemoguće nacrtati 1.000.000-dimenzionalni prostor na takav način da to za čoveka ima smisla.



Slika XX: *Stohastički opadajući gradijent mini serije u dvodimenzionalnom prostoru(jedan uzorka za treniranje)*

Postoji više vrsta stohastičkog opadajućeg gradijenta, a razlikuju se po tome koliko uzimaju u obzir prethodna ažuriranja težina prilikom izračunavanja sledećeg ažuriranja. Postoji varijanta sa impulsom koji se bavi problemima brzine konvergencije i lokalnog minimuma. [21]



Slika XX: *Lokalni i globalni minimum*

Funkcije najčešće imaju više minimuma, a onaj koji je najbolje pronaći je globalni minimum. Na osnovu Slika XX jasno se vidi da bi se u slučaju malog koraka, odnosno stope učenja, došlo do lokalnog minimuma, a svako dalje povećanje ili smanjenje bi rezultovalo uvećanjem greške. To bi dalje rezultovalo ne pronalaženjem globalnog minimuma. Rešenje ovog problema je impuls, a može se zamisliti kao spuštanje lopte niz krivu - ukoliko je impuls dovoljno jak loptica će proći lokalni minimum i završiti u globalnom minimumu. U praksi, to znači da ažuriramo parametre mreže na osnovu prethodnog ažuriranja parametara, a ne samo na osnovu vrednosti gradijenta.

Lančanje izvoda funkcija

U prethodnom algoritmu smo pretpostavili da pošto je funkcija diferencijabilna, možemo izračunati njen izvod. Međutim, u slučaju neuronskih mreža govorimo o mnogo ulančanih operacija sa tenzorima, od kojih svaka ima poznati izvod. Na primer, imamo mrežu sa tri operacije sa tenzorima, a, b i c i matricama težina W1, W2 i W3:

$$f(W1, W2, W3) = a(W1,\ b(W2,\ c(W3)))$$

Matematika nas uči da se za takav lanac funkcija može pronaći izvod ukoliko koristimo sledeći identitet, poznat kao izvod složene funkcije: $f(g(x)) = f'(g(x)) * g'(x)$. Primena izvoda složene funkcije na izračunavanje gradijenta neuronske mreže predstavlja algoritam Bekpropagacije

(poznat i kao automatsko diferenciranje[3] u obrnutom režimu). Bekpropagacija započinje finalnom greškom, zatim se kreće od unutrašnjih (dubljih) ka spoljašnjim slojevima i primenjuje izvod složene funkcije radi izračunavanja doprinosa koji je svaki parametar imao za vrednost greške. [21]

U današnje vreme i u budućnosti, ljudi će neuronske mreže kreirati koristeći specijalizovane biblioteke, kao što je TensorFlow, o kome će kasnije biti više reči, koje omogućavaju izvršenje čitavog algoritma Bekpropagacije pozivom samo jedne funkcije. Samim tim, neće nikada biti potrebno implementirati ovaj algoritam ručno.

Primena dubokog učenja

Iako je duboko učenje poprilično stara podoblast mašinskog učenja, njen uspon je krenuo tek početkom 2010-ih godina. Za samo nekoliko godina ova podoblast dostigla je izvanredne rezultate na perceptivnim problemima poput gledanja i slušanja - problema koji uključuju veštine koje su intuitivne za čoveka, ali su dugo bile nedostižne za mašine. Poseban proboj je duboko učenje imalo u istorijski teškim oblastima mašinskog učenja:

- Klasifikacija slika, gotovo na nivou čoveka;
- Prepoznavanje govora, gotovo na nivou čoveka;
- Konvertovanja ručno pisanog teksta u digitalni sadržaj, gotovo na nivou čoveka;
- Poboljšanje mašinskog prevođenja;
- Konverzija teksta u govor;
- Digitalni asistenti poput Google Now i Siri (Apple);
- Autonomna vožnja, gotovo na nivou čoveka;
- Poboljšano distribuiranje ciljanih oglasa (*engl. ad targeting*);
- Poboljšanje pretrage na web-u;
- Mogućnost davanja odgovora na pitanja na prirodnom jeziku;

Još uvek se istražuje šta duboko učenje u potpunosti može da uradi. Ova oblast primenjena je na širok spektar problema, a neke od najvažnijih su one u oblasti nauke, farmacije i medicine.

[3] **Diferenciranje** je postupak nalaženja izvoda u matematici.

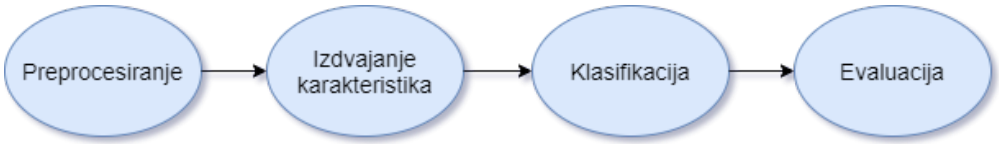
Primena dubokog učenja u medicini - analiza medicinskih slika

Zdravstveni sektor je posebna grana industrije, sa obzirom na to da je veoma visokog pioriteta i da ljudi očekuju najviši nivo nege i usluga bez obzira na troškove. Međutim, ovaj sektor ipak ne ispunjava očekivanja i pored toga što troši veliki deo budžeta. Uglavnom, medicinske podatke interpretira medicinski stručnjak, a to najčešće, naročito ako govorimo o slikama, uključuje dosta subjektivnosti, slike mogu biti složene, postoji mnogo varijacija različitih oboljenja i sl. Mnogi zadaci dijagnostikovanja podrazumevaju početno istraživanje radi identifikacije abnormalnosti, određivanja mera i praćenja promena. Automatski alati za analizu slika bazirani na mašinskom učenju su ključni za poboljšanje kvaliteta dijagnostikovanja na osnovu medicinskih slika. Široko primenjena tehnika u ovom polju je duboko učenje i to za veliki spektar problema [23]:

- Detekcija dijabetične retinopatije (izlečenje je usko povezano sa ranim otkrivanjem oboljenja);
- Otkrivanje histoloških i mikroskopskih elemenata (nivo ćelije i tkiva);
- Detekcija bolesti probavnog trakta;
- Detekcija bolesti srca;
- Otkrivanje ćelija tumora;
- Otkrivanje Alchajmerove i Parkinsonove bolesti.

Koraci u procesu klasifikacije medicinskih slika

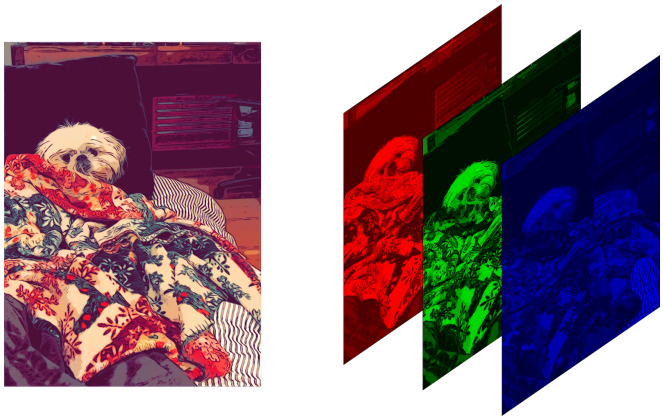
Ono što je najvažnije u slučaju zdravstvenog sektora je da sistem za dijagnostikovanje treba da da što tačnije rezultate, a za to je neophodno proći kroz osnove korake digitalne obrade medicinskih slika. Koraci obrade prikazani su na Slika XX.



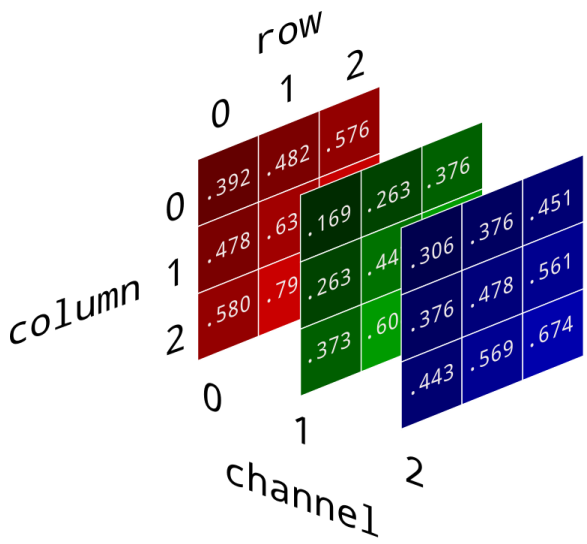
Slika XX: *Osnovni koraci u digitalnoj obradi medicinskih slika*

Preprocesiranje

Medicinske slike mogu biti pod uticajem smetnji odnosno izložene takozvanoj "buci" (*engl. noise*), koja se može javiti tokom prenosa slika. Prilikom preprocesiranja (*engl. preprocessing*) ove smetnje se smanjuju, a neželjeni podaci se ručno odsecaju i prerađuju. Poboljšanje podataka je neophodno za postizanje najboljeg mogućeg rezultata, što dalje pomaže pri dostizanju krajnjeg cilja. Obično se preprocesiranje odvija kroz sledeće korake: pročišćavanje podataka (*engl. data cleaning*), integracija podataka, smanjenje skupa podataka (*engl. data reduction*) i transformacija skupa podataka. [24]



SlikaXY*. Slika prikazana kroz 3 kanala (RGB)*

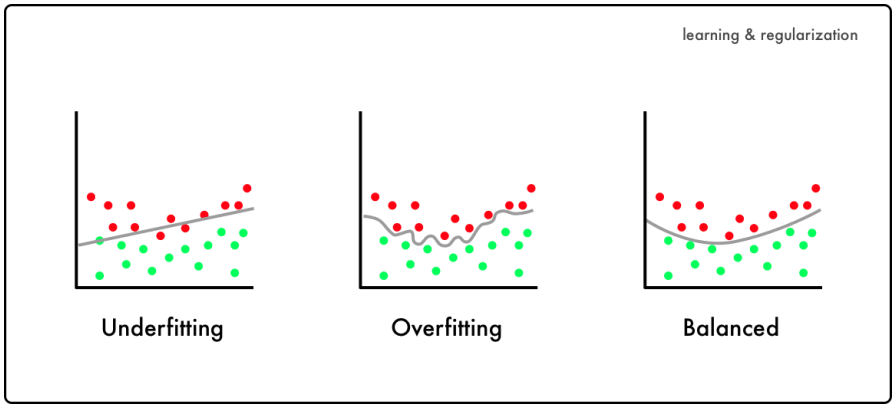


SlikaXY. *Predstavljanje slike preko 3 2D matrice kanala sa vrednostima piksela*

Slika je predstavljena matricom piksela. Komponenta slike naziva se kanal (*engl. channel*) i kod standardnih slika postoje 3 kanala, po jedan za crvenu, zelenu i plavu boju (*engl. RGB*). Svaki kanal se može predstaviti posebnom 2D matricom u kojoj pikseli uzimaju vrednost iz opsega [0-255]. Pikseli koji imaju vrednost 0 predstavljaju crnu boju, dok oni koji iznose 255 predstavljaju belu boju. Slike koje imaju samo jedan kanal su slike na sivoj skali (*engl. grayscale*) i one su predstavljene samo jednom matricom piksela što ih čini pogodnim za obradu. Kako bi se slika konvertovala u sivu skalu potrebno je uzeti prosečnu vrednost svakog piksela za svaki kanal (RGB) i smestiti je na odgovarajuće mesto u novoj 2D matrici koja će predstavljati prikaz sive slike.

Podaci koji se koriste kao ulazni moraju biti formatirani na odgovarajući način pre pohranjivanja neuronske mreže. Učitavanje podataka iz fajl sistema i dekodiranje JPEG slika u RGB mreže piksela su prvi koraci pripreme podataka. Sledeći bitan korak je skaliranje piksela sa vrednosti iz opsega [0-255] na opseg [0-1]. Biblioteke mašinskog učenja uglavnom sadrže module koji služe kao pomoćni alati pri preprocesiranju slika i prethodno pomenuti koraci se mogu automatizovati, te ih gotovo nikada nije potrebno ručno implementirati.

Overfitting



***SlikaXZ**. Prikaz underfitting, overfitting i dobro balansiranog skupa podataka*

Kada model mašinskog učenja upozna sve detalje skupa trening podataka uz sve smetnje koje sadrži nastaće problem *overfitting*. *Overfitting* znači da model mašinskog učenja ima odlične rezultate nad skupom trening podataka, ali generalno loš rezultat pri obradi podataka koje nije prethodno video. Ovo je jako čest problem u domenu mašinskog učenja i postoji dosta metoda koje pokušavaju da utiču na neutralizaciju ili sprečavanje nastanka ovog problema. Model mašinskog učenja u nekom trenutku počinje da prepoznaje obrasce koji su specifični za skup trening podataka, ali navode na pogrešne zaključke ili su potpuno irelevantni kada je reč o novim podacima. Najbolji način koji pomaže u prevazilaženju ovog problema je veći skup trening podataka, ali kako to nije uvek moguće obezbediti jedno od rešenja je moduliranje količine informacija koje model sme da čuva ili dodavanje ograničenja za čuvanje informacija. Ukoliko ima manje obrazaca za pamćenje, mreža će se prilagoditi da pamti što relevantnije podatke, koji pomažu pri generalizaciji. Ovako se opisuje regularizacija, kao jedan od načina za prevazilaženje problema *overfitting-a*. Osim toga augmentacija podataka je još jedan od načina rešavanja problema *overfitting-a*, posebno kod modela koji obrađuju slike.

Augmentacija podataka

Kao što je prethodno pomenuto, u slučaju da postoji mali skup podataka overfitting će se sigurno javiti, a augmentacija podataka (*engl. data augmentation*) predstavlja jedan od načina prevazilaženja ovog problema. Ovaj metod posebno daje značaj kod obrade slika uz pomoć nekog od modela dubokog učenja.

Augmentacija podataka koristi postojeće podatke iz skupa trening podataka i utiče do određene mere na njih kako bi proizvela nove podatke za skup trening podataka primenom određenih

transformacija. Cilj augmentacije je da uoči treniranja model podataka uvek dobije različitu sliku na kojoj se trenira, a jednostavne transformacije bazirane na slučajnom odabiru to omogućavaju.

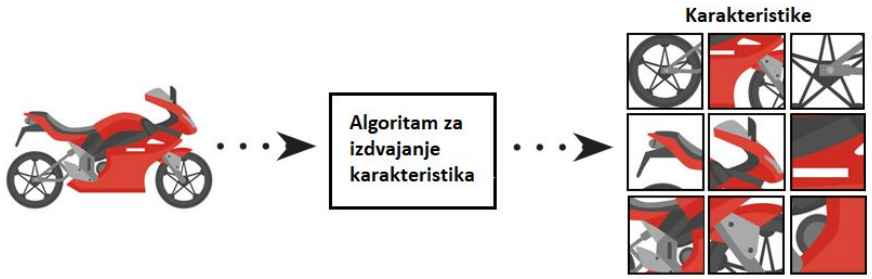
***SlikaYZ**. Primer augmentacije na slici mačke*

Kada treniramo mrežu uz pomoć posebne konfiguracije koja omogućava augmentaciju podataka, ona nikada neće videti isti ulazni podatak i biće naučena da bolje vrši generalizaciju. Pošto se u slučaju augmentacije podataka koristi ista slika kao osnova, to će uticati do određene mere na model, ali on neće biti u stanju da donosi nove zaključke i zbog toga sama metoda nije dovoljna u svim slučajevima za prevazilaženje problema *overfitting-a* skupa podataka. U tom slučaju dodaje se i *Dropout* sloj unutar modela, pre samog klasifikatora.

Izdvajanje karakteristika

Izdvajanje karakteristika je proces smanjenja dimenzionalnosti kojim se početni skup podataka smanjuje na grupe koje se mogu dalje obrađivati. Odlika velikih skupova podataka je veliki broj varijabli za čiju obradu je potrebno mnogo resursa. Izdvajanje karakteristika je naziv za metode koje biraju i/ili kombinuju varijable u karakteristike, efikasno smanjujući količinu podataka za obradu, a istovremeno tačno i u potpunosti opisujući izvorni skup podataka.

Proces izdvajanja karakteristika je koristan kada treba smanjiti broj resursa potrebnih za obradu bez gubljenja važnih informacija. Izdvajanje karakteristika takođe može smanjiti količinu suvišnih podataka za datu analizu. Takođe, smanjenje podataka i potrebnih resursa olakšavaju brzinu učenja i korake generalizacije u procesu mašinskog učenja.



Slika XX: *Izdvajanje karakteristika*

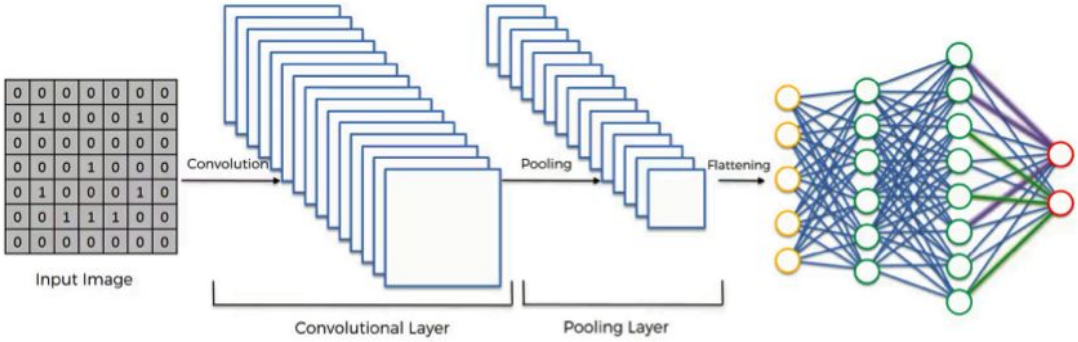
Nakon preprocesiranja ulaznih podataka oni se ulivaju u konvolucijsku neuronsku mrežu koja vrši izdvajanje karakteristika i smanjuje redundantnost. Način na koji konvolucijska neuronska mreža vrši izdvajanje karakteristika iz slika koje predstavljaju ulazne podatke predstavljen je u nastavku ovog pogavlja.

Konvolucijske neuronske mreže

Do nedavno računari nisu mogli da izvrše neke trivijalne zadatke poput prepoznavanja objekta na slici ili prepoznavanja prirodnog govora. Razlog je taj što se kod čoveka percepcija odvija izvan područja svesti, unutar specijalizovanih slušnih, vizuelnih i drugih modula u mozgu, te zapravo nije ni malo jednostavan proces.

Međutim, zahvaljujući David H. Hubel-u i Torsten Wiesel-u koji su izveli serije eksperimenata na mačkama [25-26], a kasnije i majmunima [27] i dali ključni uvid u strukturu vizuelnog

korteksa (autori su dobili Nobelovu nagradu za fiziologiju ili medicinu 1981. godine za svoj rad), rodila se ideja koja je vremenom evoluirala do danas široko poznatih Konvolucijskih neuronskih mreža. Važna prekretnica bio je rad Yann LeCun-a, Léon Bottou-a, Yoshua Bengio-a, i Patrick Haffner-a iz 1998. godine koji je predstavio poznatu arhitekturu **LeNet-5** [28].



Slika XX: *Gradivni blokovi konvolucijske neuronske mreže*

U poslednjih nekoliko godina, zahvaljujući povećanju računarske snage, količine podataka za treniranje i novim pronalascima za problem treniranja neuronskih mreža, konvolucijske neuronske mreže su uspele da postignu nadljudske performanse na velikom broju kompleksnih vizuelnih zadataka. Danas konvolucijske neuronske mreže predstavljaju model dubokog učenja koji se gotovo univerzalno koristi za probleme računarskog vida, a gradivni blokovi i princip funkcionisanja ovog modela prikazani su na Slika XX. [22]

Neuronska mreža predstavlja sistem koji se sastoji od određenog broja međusobno povezanih procesnih elemenata ili čvorova koje nazivamo veštačkim **neuronima**. Arhitektura neuronske mreže predstavlja specifično povezivanje neurona u jednu celinu. Struktura neuronske mreže se razlikuje po broju slojeva. Konvoluciona neuronska mreža sastoji se od ulaznog i izlaznog sloja, kao i više skrivenih slojeva. Skriveni slojevi CNN-a obično se sastoje od niza konvolucionih slojeva, slojeva sažimanja i potpuno povezanih slojeva, a svaki od ovih gradivnih elemenata će biti detaljnije predstavljen u nastavku ovog poglavlja.

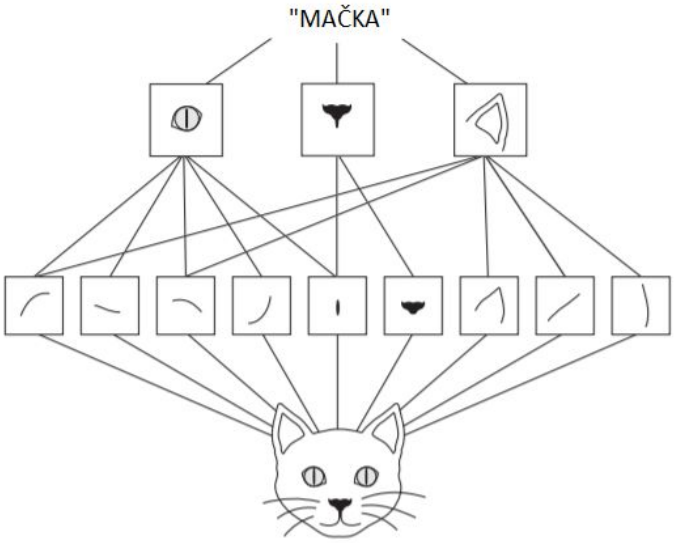
Sloj konvolucije

Najvažniji gradivni blok konvolucijske neuronske mreže je sloj konvolucije. Prvi skriveni sloj mreže uči lokalne obrasce, odnosno koncentriše se na najmanje karakteristike (elemente) sa slike koja se obrađuje (ulazni podatak), a zatim u narednim skrivenim slojevima ih kombinuje u veće karakteristike (elemente) i tako na dalje. Ova hijerarhijska struktura je uobičajena kod stvarnih

slika, pa je to i razlog zbog koga ove mreže jako dobro rade na prepoznavanju slika. Ova ključna odlika daje konvolucionim neuronskim mrežama dve zanimljive osobine [21]:

- **Naučeni obrasci su nezavisni u odnosu na svoj položaj:** Nakon što nauči određeni obrazac u donjem desnom uglu slike, konvolucjska neuronska mreža ga može prepoznati bilo gde (na primer, u gornjem levom uglu). To čini konvolucijske neuronske mreže efikasnim pri obradi slika (vizuelni svet je u osnovi nezavistan u odnosu na svoj položaj);

- **Mogu naučiti prostorne hijerarhije obrazaca** (videti Slika XX): Prvi sloj konvolucije naučiće male lokalne obrasce kao što su ivice, a drugi sloj veće obrasce napravljene od karakteristika prvih slojeva itd. Tako mreža efikasno uči sve složenije i apstraktnije vizuelne koncepte (jer je vizuelni svet u osnovi prostorno hijerarhijski).

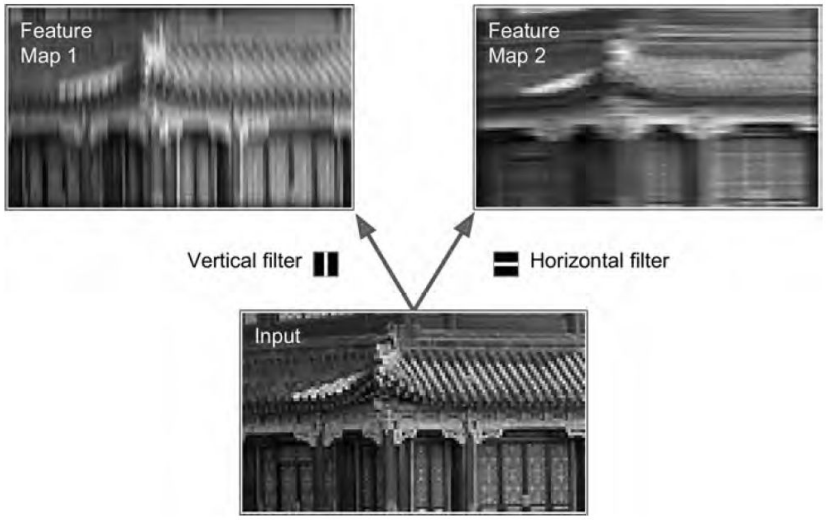


Slika XX: *Prostorna hijerarhija: lokalni obrasci (oči, uši) se kombinuju u veće obrasce (mačka)*

Operacija konvolucije radi sa trodimenzionalnim tenzorima, takozvanim **mapama karakteristika**, sa dve prostorne ose (visina i širina), kao i dubinskom osom (osom kanala). Za RGB sliku, dimenzija dubinske ose je 3, jer slika ima tri kanala u boji - crvenu, zelenu i plavu. Za crno-belu sliku dubina je 1 (nivoi sive boje). Operacija konvolucije izvlači parčiće iz ulazne mape karakteristika i na njima primenjuje istu transformaciju, stvarajući izlaznu mapu karakteristika. Ova mapa izlaznih karakteristika i dalje je trodimenzionalni tenzor, a njegova dubina može biti proizvoljna, jer dubina više ne predstavlja određene boje kao na RGB ulazu; već predstavlja **filtere**. [21]

Filteri

Težine neurona mogu se predstaviti kao mala slika veličine receptivnog polja. Na primer, Slika XX prikazuje dva moguća skupa težina, koji se nazivaju filteri. Prvi je predstavljen kao crni kvadrat sa vertikalnom belom linijom u sredini - neuroni koji koriste ove težine ignorisaće sve u svom receptivnom polju, osim centralne vertikalne linije (jer će se svi ulazi pomnožiti sa 0, osim onih koji se nalaze u centralnoj vertikalnoj liniji). Drugi filter je crni kvadrat sa vodoravnom belom linijom u sredini. Još jednom, neuroni koji koriste ove težine ignorisaće sve u svom receptivnom polju, osim centralne horizontalne linije. Sada, ako svi neuroni u sloju koriste isti filter vertikalne linije, sloj će dati gornji levi izlaz na Slika XX. [22]



Slika XX: *Upotreba dva različita filtera radi dobijanja dve različite mape karakteristika*

Vertikalne bele linije se poboljšavaju dok se ostatak zamućuje. Slično tome, slika u gornjem desnom uglu se dobija, ako svi neuroni koriste filter horizontalne linije - vodoravne bele linije poboljšavaju dok su ostale zamućene. Dakle, sloj neurona koji koriste isti filter vam daje mapu karakteristika, koja ističe područja na slici koja su najsličnija filteru. Tokom treninga, CNN pronalazi najkorisnije filtere za svoj zadatak i uči da ih kombinuje u složenije obrasce.

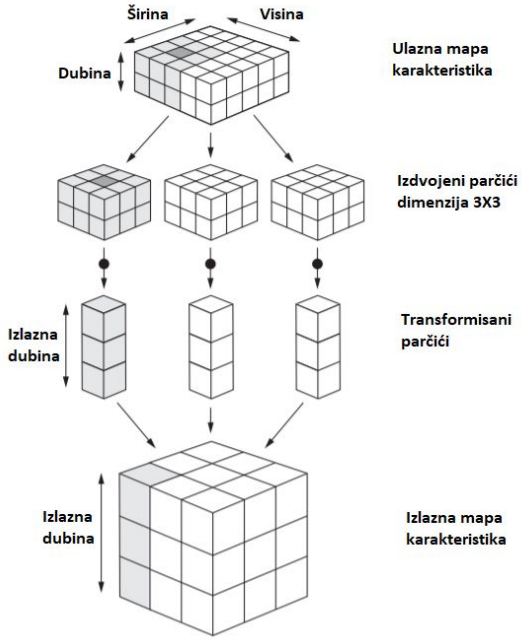
U primeru kada prvi sloj konvolucije uzima mapu karakteristika veličine (28, 28, 1) i daje mapu karakteristika veličine (26, 26, 32) imamo izračunavanje za 32 filtera. Svaki od ova 32 izlazna kanala sadrži mrežu 26 × 26 vrednosti, koja je mapa rezultata rada filtera, ukazujući na odgovor za taj filter na različitim lokacijama na ulaznoj mapi karakteristika. To je ono što pojam mapa karakteristika znači - svaka dimenzija u osi dubine je karakteristika (filter).

Parametri konvolucije

Konvoluciju definišu dva parametra:

1. Veličina parčiča izdvojenih iz ulazne mape karakteristika;
2. Dubina izlazne mape karakteristika (broj primenjenih filtera)

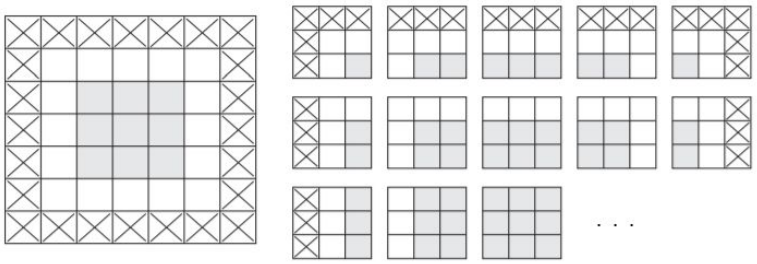
Konvolucija funkcioniše klizanjem parčića preko trodimenzionalne mape ulaznih karakteristika, zaustavljanjem na svim mogućim lokacijama i izvlačenjem parčića iz okolnih obeležja. Svako takvo trodimenzionalno parče se zatim transformiše u jednodimenzionalni vektor. Svi ovi vektori su zatim prostorno ponovo sastavljeni u trodimenzionalnu izlaznu mapu. Svaka prostorna lokacija na mapi izlaznih karakteristika odgovara istoj lokaciji na ulaznoj mapi karakteristika (na primer, donji desni ugao izlaza sadrži informacije o donjem desnom uglu ulaza). Opisani proces konvolucije predstavljen je na Slika XX. [21]



Slika XX: *Prikaz procesa konvolucije*

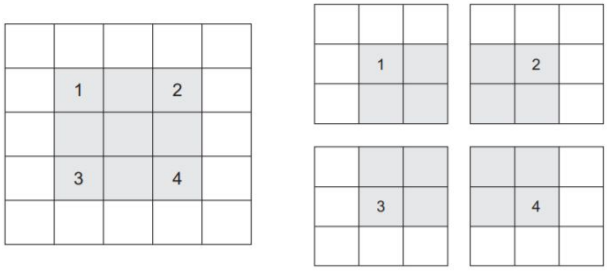
Ono što je važno napomenuti je da se izlazna visina i širina mogu razlikovati od ulazne visine i širine, iz dva razloga:

1. **Efekti ruba**: Ovaj efekat možemo primetiti na prethodnom primeru gde su dimenzije ulazne mape karakteristika bile 28x28, a izlazne 26x26 nakon sloja konvolucije. Ukoliko želimo da ulazna i izlazna mapa karakteristika imaju iste dimenzije možemo koristiti *padding*. On podrazumeva dodavanje odgovarajućeg broja redova i kolona sa svake strane ulazne mape.



Slika XX: *Dodavanje paddinga na 5x5 ulaznoj mapi rezultovalo je izdvajanjem 25 3x3 parčića*

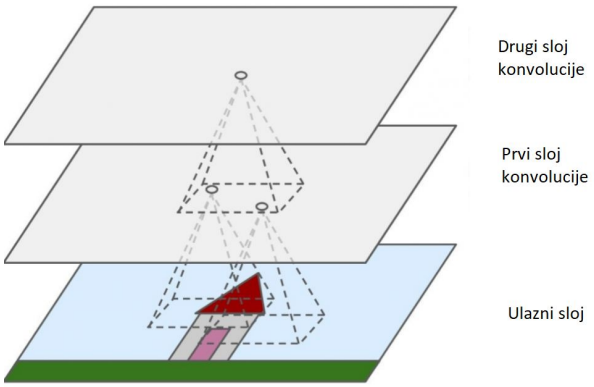
2. **Korišćenje koraka**: Drugi faktor koji može uticati na veličinu izlaza je pojam koraka. Naime, rastojanje između dva uzastopna prozora je parametar konvolucije, koji se naziva njenim korakom. Moguće je imati stepenaste konvolucije,odnosno konvolucije sa korakom većim od 1. Na Slika XX prikazana je konvolucija sa korakom 2, što znači da se širina i visina mape umanjuju za faktor 2. Međutim ovo se retko koristi u praksi, jer se implementira sloj sažimanja o kome će biti reči kasnije.



Slika XX: *Konvolucija sa parčićima dimenzija 3x3 i korakom 2*

<u>Matematička jednačina konvolucije</u>

Ukoliko posmatramo na nivou neurona, neuroni u prvom konvolucionom sloju nisu povezani sa svakim pojedinačnim pikselom na ulaznoj slici, već samo sa pikselima u svojim receptivnim poljima. Zatim, svaki neuron u drugom konvolucionom sloju povezan je samo sa neuronima smeštenim unutar malog pravougaonika u prvom sloju itd. Neuron smešten u redu i, koloni j datog sloja povezan je sa izlazima neurona u prethodnom sloju smeštenom u redovima i do $i + f_h - 1$, kolonama j do $j + f_w - 1$, gde su f_h i f_w visina i širinu receptivnog polja.



Slika XX: *Ulazni i dva uzastopna konvolucijska sloja konvolucijske neuronske mreže*

Pored navedenih parametara važno je uzeti u obzir i korak konvolucije. Neuron smešten u redu i, koloni j u gornjem sloju povezan je sa izlazima neurona u prethodnom sloju koji se nalaze u redovima $i \times s_h$ do $i \times s_h + f_h - 1$, kolonama $j \times s_w$ do $j \times s_w + f_w - 1$, gde s_h i s_w predstavljaju vertikalni i horizontalni korak. Ovo važi za sve mape karakteristika, odnosno svi neuroni koji se nalaze u istom redu i i koloni j, ali u različitim mapama karakteristika su povezani na izlaze potpuno istih neurona u prethodnom sloju.

Jedna velika matematička jednačina sumira prethodna objašnjenja i pokazuje kako izračunati izlaz datog neurona u konvolucionom sloju [22]:

$$z_{i,j,k} = b_k + \sum_{u=1}^{fh} \sum_{v=1}^{fw} \sum_{k'=1}^{fn'} x_{i',j',k'} \times w_{u,v,k',k}$$

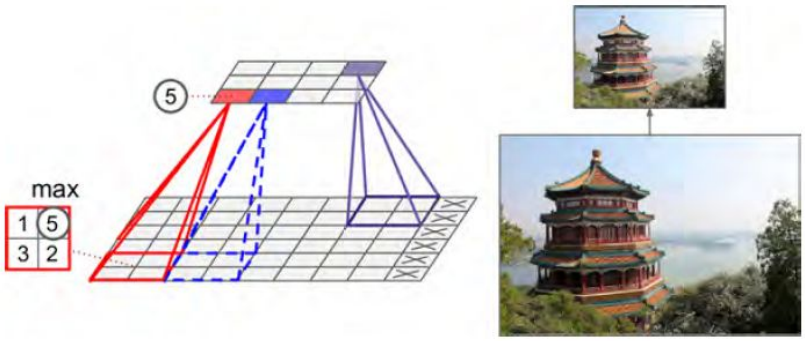
Gde se *i'* i *j'* izračunavaju na sledeći način:

$$i' = u \times s_h + f_h - 1$$
$$j' = v \times s_w + f_w - 1$$

- $z_{i,j,k}$ je izlaz neurona koji se nalazi u redu i, koloni j na mapi karakteristika *k* (sloj *l*);
- b_k je nagib za mapu karakteristika k (sloj *l*);
- s_h i s_w su vertikalni i horizontalni koraci, f_h i f_w su visina i širina receptivnog polja, a f_h' je broj mapa karakteristika u prethodnom sloju (sloj *l* - 1);
- $x_{i',j',k'}$ je izlaz neurona smeštenog u sloju *l* - 1, red *i'*, kolona *j'*, mapa karakteristika k';
- $w_{u,v,k',k}$ je težina veze između bilo kog neurona na mapi *k* sloja *l* i njegovog ulaza koji se nalazi u redu *u*, koloni *v* na mapi *k'*.

Sloj sažimanja

Sledeći sloj u konvolucijskoj neuronskoj mreži, nakon sloja konvolucije je sloj sažimanja. Cilj ovog sloja je da smanji ulaznu sliku radi smanjenja računarskog opterećenja, upotrebe memorije i broja parametara. Baš kao i u konvolucionim slojevima, svaki neuron u sloju sažimanja povezan je sa ograničenim brojem neurona iz prethodnog sloja (receptivno polje). Međutim, neuroni iz ovog sloja nemaju težinu, jer sve što sloj radi je da agregira ulaze koristeći funkciju agregacije kao što je maksimum ili srednja vrednost. Na Slika XX prikazan je sloj za sažimanje koji koristi funkciju agregacije - maksimum, što je i najčešće korišćena funkcija. Dakle, samo maksimalna ulazna vrednost u svakom neuronu prelazi na sledeći sloj, a ostali ulazi su odbačeni.

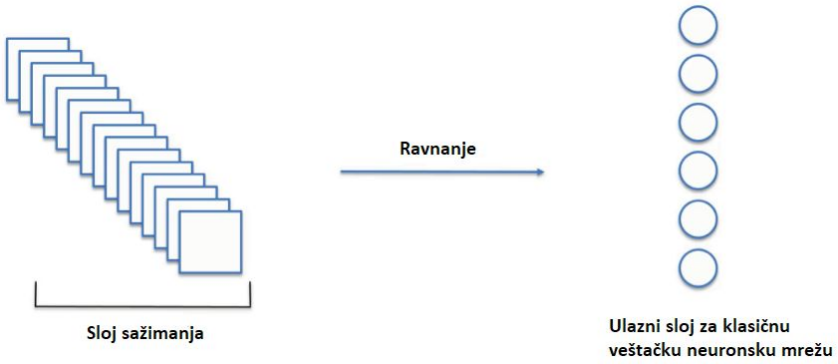


Slika XX: *Sloj sažimanja koji koristi funkciju agregacije - maksimum*

Ovo je vrlo destruktivna vrsta sloja čak i sa sićušnim 2 × 2 parčićima i korakom 2, jer će izlaz biti dva puta manji u oba smera (tako da će njegova površina biti četiri puta manja). Odnosno biće odbačeno 75% ulaznih vrednosti. Sloj sažimanja obično radi na svakom ulaznom kanalu nezavisno, tako da je izlazna dubina jednaka ulaznoj dubini. Alternativno se može vršiti sažimanje preko dimenzije dubine i u tom slučaju prostorne dimenzije slike (visina i širina) ostaju nepromenjene, ali se dubina smanjuje.

Ravnanje

Nakon završetka prethodna dva koraka imamo kao rezultat sažetu mapu karakteristika. Kao što i sam naziv ovog koraka glasi, sledeće što je potrebno uraditi je izravnati našu sažetu mapu karakteristika u jednu kolonu kao što je prikazano na Slika XX. Razlog postojanja ovog koraka je taj što nakon njega sledi ubacivanje ovih podataka u klasičnu veštačku neuronsku mrežu. Ono što se dešava nakon sloja ravnjanja je da na kraju dobijamo jako dugačak vektor ulaznih podataka koji zatim prolaze kroz veštačku neuronsku mrežu da bi se dalje obrađivali.



Slika XX: *Sloj ravnanja*

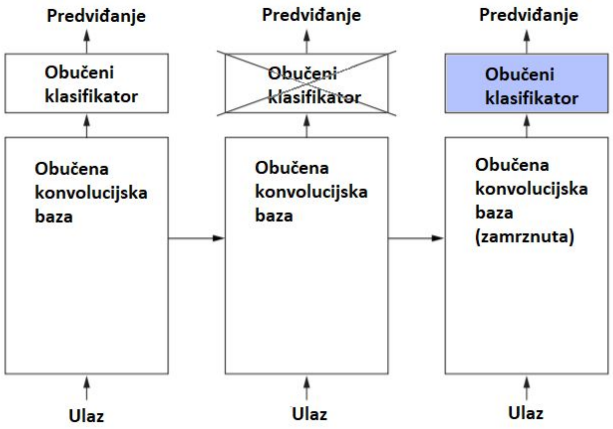
Transfer učenja

U praksi se veoma retko dešava da se trenira cela konvolucijska neuronska mreža od nule, jer duboke neuronske mreže zahtevaju veliku količinu podataka, koju često nije moguće obezbediti, naročito kada govorimo o medicinskim slikama. Zbog toga se u kontekst neuronskih mreža, uvodi pojam transfer učenja (*eng. transfer learning*) koji upućuje na razmenu znanja između različitih modela neuronskih mreža. Podrazumeva se da je prethodno obučena mreža bila trenirana na jako velikom skupu podataka, obično na zadatku klasifikacije slika. Ako je ovaj izvorni skup podataka dovoljno velik i dovoljno opšti, tada naučena prostorna hijerarhija obrazaca (ovaj je pojam već prethodno objašnjen) može efikasno delovati kao generički model vizuelnog sveta, pa se stoga njegove karakteristike mogu pokazati korisnim za mnoge različite probleme računarskog vida, iako novi problemi mogu da uključuju potpuno različite klase od onih iz prvobitnog zadatka. Umesto obučavanja celih mreža, koristi se prethodno navedeno znanje, pri čemu se u zavisnosti od problema, vrše određena podešavanja.

Transfer učenja može se klasifikovati na tri različita:

- **Induktivni** - zadaci izvorni i ciljni su različiti, bez obzira da li su domeni isti ili nisu;
- **Transduktivni** - zadaci izvorni i ciljni su isti, dok su domeni različiti;
- **Nenadgledani** - zadaci izvorni i ciljni su različiti, dok su domeni isti.

Kao što je već rečeno, konvolucijske neuronske mreže koje se koriste za klasifikaciju slika sastoje se iz dva dela: serije slojeva konvolucije i sažimanja, a završavaju se gusto povezanim klasifikatorom. Prvi deo naziva se konvoluciona osnova modela. Izdvajanje karakteristika se sastoji iz uzimanja konvolucijske baze prethodno obučene mreže, provođenju novih podataka kroz nju i treniranju novog klasifikatora na vrhu rezultata kao što je prikazano na Slika XX.



Slika XX: *Zamena klasifikatora uz zadržavanje iste konvolucijske baze*

Važno je napomenuti da nivo opštosti (a samim tim i reupotrebljivosti) prikaza izvučenih određenim slojevima konvolucije zavisi od dubine sloja u modelu. Slojevi koji dolaze ranije u modelu izdvajaju lokalnu, vrlo generičku karakteristiku mape (kao što su vizuelne ivice, boje i teksture), dok viši slojevi izvlače apstraktnije koncepte (poput „mačje uho“ ili „pseće oko“). Dakle, ako se novi skup podataka mnogo razlikuje od skupa podataka na kojem je originalni model treniran, potrebno je koristiti samo prvih nekoliko slojeva modela za izdvajanje karakteristika, umesto cele konvolucijske baze.

Fino podešavanje

Još jedna široko korišćena tehnika za ponovnu upotrebu modela, koja se koristi uz izdvajanje karakteristika, je fino podešavanje (*engl. fine-tuning*). Fino podešavanje vrši odmrzavanje nekoliko gornjih slojeva smrznute osnove prethodno obučenog modela koji se koriste za izdvajanje karakteristika, a zatim se vrši zajedničko treniranje novog modela i ovih gornjih slojeva prethodno obučenog modela. Ova tehnika se naziva fino podešavanje, jer neznatno prilagođava apstraktnije delove modela koji se ponovo koristi, kako bi ih učinio relevantnim za trenutni problem.

Neophodno je zamrznuti osnovu konvolucionog algoritma prethodno podučenog modela kako bi slučajno inicijalizovan klasifikator prešao na vrh i bio treniran. Iz sličnih razloga moguće je fino podešavati slojeve koji se nalaze na vrhu osnove konvolucionog algoritma kada je klasifikator sa vrha već istreniran. U slučaju da pomenuti klasifikator nije istreniran greške koje će propagirati kroz mrežu će biti prevelike i reprezentacija rezultata na osnovu prethodno naučenih slojeva će biti uništena.

Zbog toga neophodno je preduzeti sledeće korake:

1. Dodavanje prilagođene mreže na vrh već istrenirane osnovne mreže
2. Zamrzavanje osnovne mreže
3. Treniranje dela koji se dodaje
4. Odmrzavanje određenih slojeva osnovne mreže
5. Treniranje oba sloja i dodatog dela zajedno

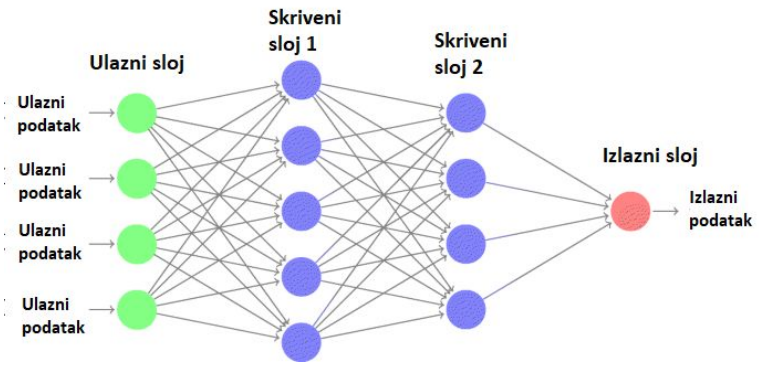
Prva tri koraka su obuhvaćena kroz proces izdvajanja karakteristika (*engl. feature extraction*). Slojevi koji se nalaze pri vrhu mreže sadrže specijalizovane karakteristike, u odnosu na one ispod koje su sve generalnije. Zbog toga se fokus finog podešavanja stavlja na specijalizovane slojeve u konkretnom domenu. Što više parametara je obuhvaćeno tokom treniranja, utoliko je veća šansa da dođe do preteranog podudaranja trening podataka (*engl. overfitting*). Osnovna konvoluciona mreža sadrži preko 15 miliona parametara (neurona), što bi dovelo do dosta grešaka na malom skupu podataka, te se zbog toga biraju 2 ili 3 sloja koja se nalaze na vrhu.

Klasifikacija

Najčešći zadaci mašinskog učenja su regresija (predviđanje vrednosti) i klasifikacija (predviđanje klasa). Klasifikacija podrazumeva određivanje klase kojoj pripadaju elementi iz skupa podataka koji se obrađuje, odnosno predviđa klasu za ulaznu promenljivu. Najjednostavniji vid klasifikacije u mašinskom učenju je **binarna klasifikacija**, kod koje postoje samo dve moguće klase za ulazne podatke, dok postoji i **klasifikacija na veći broj klasa** (br. klasa > 2). Klasifikacija na veći broj klasa predstavlja složeniji problem i uglavnom daje gore rezultate predviđanja, pogotovo ako govorimo o velikom broju klasa.

U slučaju konvolucijske neuronske mreže klasifikacioni sloj dolazi nakon sloja konvolucije, sloja sažimanja i ravnanja. Ovaj sloj sakuplja krajnje rezultate procesa konvolucije i sažimanja kao svoje ulazne podatke i za njih vrši klasifikaciju.

Klasifikacioni sloj se sastoji od jednog ulaznog sloja, jednog ili više skrivenih slojeva i jednog izlaznog sloja (Slika XX). Svaki sloj osim izlaznog sloja je u potpunosti je povezan sa sledećim slojem. Kada veštačka neuronska mreža ima dva ili više skrivenih slojeva, naziva se duboka neuronska mreža.



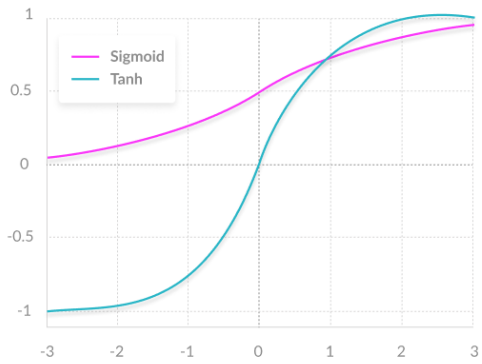
Slika XX: *Klasifikacioni sloj konvolucijske neuronske mreže (potpuno povezani slojevi)*

Za svaki podatak za treniranje algoritam bekpropagacije prvo pravi predviđanje, zatim meri grešku i na kraju prolazi kroz svaki sloj unazad da izmeri doprinos grešci iz svake veze i na kraju malo azurira težine veze kako bi se greška smanjila (algoritam opadajućeg gradijenta).

Matematičke jednačine koje određuju izlaznu vrednost neuronske mreže zovu se aktivacione funkcije. Funkcija je povezana sa svakim neuronom u mreži i određuje da li treba da se okine (aktivira) ili ne, na osnovu relevantnosti ulaznog podatka za predikciju modela.

Aktivacione funkcije

Aktivacione funkcije pomažu pri normalizaciji izlaznih podataka iz svakog neurona na opseg od 0 do 1 ili između -1 i 1. Dodatno, aktivacione funkcije moraju biti efikasne u smislu obrade, jer se računaju kroz hiljade ili čak milione neurona svakog uzorka iz skupa podataka.



SlikaMN. *Dve najčešće aktivacione funkcije neuronskih mreža - Sigmoid i Tan*

Evaluacija

Kako bi se pratila uspešnost algoritma koji se koristi pri analizi i obradi slika potrebno je meriti performanse sistema. Evaluacija je neophodan korak kako bi se rezultat pratio i po potrebi sistem modifikovao i unapređivao.

Tipične mere performansi sistema za obradu slika su stopa tačnosti, F1 skor, stopa preciznosti, stopa opoziva, stepen osetljivosti, stopa specifičnosti i Sorensen-Dice indeks (*engl. Sorensen-Dice coefficient*). Stope preciznosti, tačnosti i opoziva su tri glavne metrike koje se koriste pri evaluaciji klasifikacije. Stopa preciznosti se definiše kroz odnos broja tačnih predviđanja prema ukupnom broju ulaznih uzoraka. Logaritamski gubitak je važan kod višeklasne klasifikacije i koristi se za smanjenje lažnih klasifikacija. Matrica konfuzije je još jedan čest prikaz rezultata koji opisuju performanse modela mašinskog učenja. U nastavku su date matematičke formule za neke od prethodno pomenutih mera evaluacije.

Skraćenice za nazive promenljivih koje se koriste u formulama:

- PPV - stopa preciznosti (*engl. positive predictive value*);
- TPR - stopa opoziva (*engl. true positive rate*);
- F-skor (*engl. F-score*) - harmonijska sredina stope preciznosti i opoziva.

Ostale vrednosti koje se koriste u formulama:

TP - *true positive*, TN - *true negative*, FP - *false positive*, FN - *false negative*, TNR - *true negative rate*, FNR - *false negative rate.*

$$PPV\ (P) = \frac{broj\ tačno\ identifikovanih\ entiteta}{broj\ identifikovanih\ entiteta} = \frac{TP}{TP + FP}$$

$$TPR\ (R) = \frac{broj\ tačno\ identifikovanih\ entiteta}{broj\ entiteta\ u\ skupu\ test\ podataka} = \frac{TP}{TP + FN}$$

$$F - score = 2 * \frac{PPV * TPR}{PPV + TPR}$$

$$TNR = \frac{TN}{TN + FP}$$

$$FNR = \frac{FN}{FN + TP} = 1 - TPR$$

$$NPV = \frac{TN}{TN + FN}$$

$$FPR = \frac{FP}{FP + TN} = 1 - TNR$$

Evaluacija uz korišćenje test podataka za utvrđivanje mera performansi modela je *k-cross validacija*. Ova procedura bira uzorke za evaluaciju modela mašinskog učenja i ima samo jedan parametar - *k*, koji predstavlja broj grupa u koje je podeljen dati skup podataka. Vrednost *k* parametra se mora pažljivo odabrati na osnovu vrste podataka i veličine skupa podataka. Najčešće se uzima vrednost između 5 i 10, a ukoliko postoji dilema obično je dobra praksa opredeliti se za *k = 10*.

Generalna procedura k-cross validacije opisana je u nastavku:

1. Izmešati skup podataka - *random*
2. Podeliti skup podataka u k grupa
3. Za svaku jedinstvenu grupu:
    a. Uzeti grupu kao test skup podataka
    b. Uzeti preostale grupe kao trening skup podataka
    c. Trenirati model korišćenjem trening skupa podataka
    d. Evaluirati model korišćenjem test skupa podataka
    e. Zapamtiti rezultat evaluacije
4. Izvesti zaključke na osnovu evaluacije

Korišćene tehnologije

U ovom poglavlju opisani su programski jezici, platforme, biblioteke i okruženja koji su korišćeni za praktičnu realizaciju rada.

Python

Python je programski jezik interpretatorskog tipa, objektno-orijentisani jezik visokog nivoa, interaktivni, poseduje dinamičku semantiku i opšte je namene. Nastao je 1991. godine, stvorio ga je Gvido van Rosum (*engl. Guido van Rossum*), a ime je dobio po kultnoj britanskoj komediji „Monty Python". Akcenat kod ovog jezika je na njegovoj jednostavnoj sintaksi, preglednosti koda i velikoj fleksibilnosti. Koristi se za veb razvoj, veštačku inteligenciju, mašinsko učenje, operativne sisteme, razvoj mobilnih aplikacija i video igrica.

Anaconda

Anaconda je besplatna distributivna platforma otvorenog koda (*engl. open-source*) za jezike Python i R. Ova platforma donosi puno alata koji se koriste u Nauci o podacima (engl. Data science) i Mašinskom učenju (engl. Machine learning) samo jednom instalacijom, tako da pruža kratku i jednostavnu postavku. Anaconda koristi koncept kreiranja promenljivih okruženja, tako da može da izoluje različite biblioteke i njihove verzije. Obuhvata veliki broj paketa kao što su conda, numpy, scipy, ipython, notebook i mnoge druge, a može se koristiti na Windows, Linux i macOS operativnim sistemima.

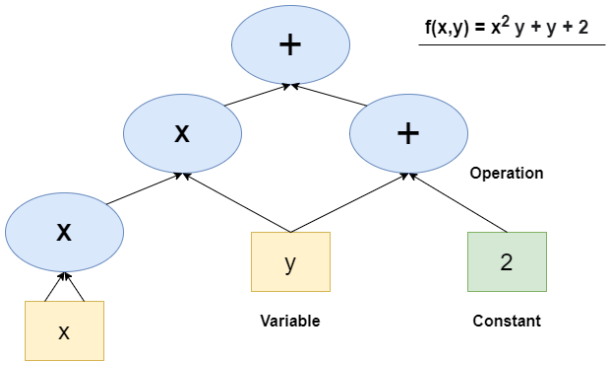
Spyder

Spyder je integrisano razvojno okruženje (*engl. Integrated Development Environment - IDE*) za naučno programiranje u programskom jeziku Python i otvorenog je koda. Inicijalno ga je kreirao i razvio Pierre Raybat 2009. godine, a od 2012. godine na njegovom održavanju i kontinuiranom unapređenju radi tim naučnika i programera, kao i čitava python zajednica.
Spyder uključuje uređivanje, interaktivno testiranje, debagiranje i introspektivne karakteristike, a dostupan je preko Anaconda platforme. Pokreće se ili upotrebom komande - *spyder* - iz Anaconda terminala ili jednostavnim klikom na ikonicu Spyder u Anaconda Navigator-u[4], koji je deo Anaconda platforme. Može se koristiti na Windows, Linux i macOS operativnim sistemima.

[4] **Anaconda Navigator** je grafički-korisnički interfejs (*engl. Graphical User Interface - GUI*) koji je uključen u Anaconda platformu i omogućava pokretanje aplikacija i jednostavno upravljanje paketima i promenljivama okruženja bez potrebe za upotrebom komandi komandne linije (*engl. command-line commands*).

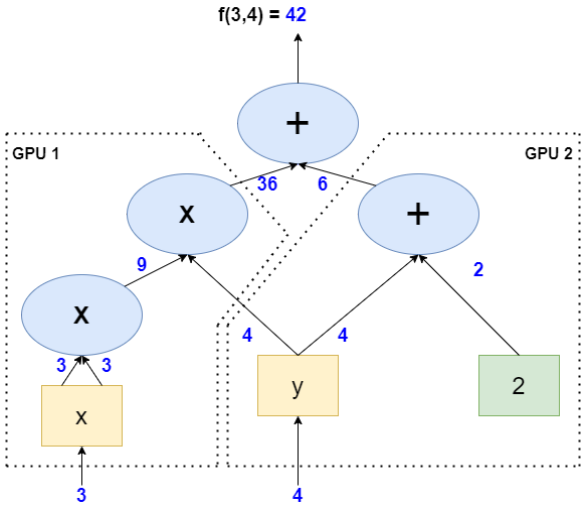
TensorFlow

TensorFlow je softverska biblioteka otvorenog koda (*engl. open-source*) za numerička izračunavanja, a posebno je pogodna i prilagođena za jako velike zadatke iz oblasti mašinskog učenja (*engl. machine learning*). Princip rada ove biblioteke podrazumeva da se u programskom jeziku Python definiše graf operacija koje je potrebno obaviti (kao na primeru sa Slika XX), a zatim TensorFlow uzima taj graf i efikasno ga izvršava koristeći optimizovani C++ kod. [22]



Slika XX: *Jednostavni graf operacija*

Još važnije je to da je moguće rasparčati graf operacija na nekoliko delova i izvšavati ih paralelno na većem broju CPU-a ili GPU-a (kao što je prikazano na Slika XX).



Slika XX: *Paralelno izvršavanje na više CPU-a/GPU-a/servera*

TensorFlow takođe podržava distribuirano izvršavanje, tako da se mogu trenirati velike neuronske mreže na ogromnim skupovima podataka u razumnom vremenskom roku podelom

izračunavanja na nekoliko hiljada servera. Ova biblioteka može da trenira mrežu od nekoliko miliona parametara na skupu obuke sačinjenog od nekoliko biliona instanci gde svaka ima po nekoliko miliona karakteristika. Ovo nije iznenađenje, ako uzmemo u obzir činjenicu da je TensorFlow razvio Google Brain tim stručnjaka i da on pokreće Google-ove servise kao što su Google Cloud Speech, Google Photos i Google Search.

Kada je TensorFlow postao otvorenog koda (*engl. open source*) Novembra 2015. godine, postojalo je već puno popularnih i dostupnih biblioteka za duboko učenje (*engl. Deep Learning*) i one su sadržale dosta funkcionalnosti koje ima i TensorFlow. Ipak, čist dizajn, skalabilnost, fleksibilnost i veoma dobra dokumentacija (da se ne spomene i ime Google-a) su jako brzo podigli TensorFlow do ogromne popularnosti.

Neke od najzanimljivijih odlika TensorFlow-a date su u nastavku [22]:

- Pored Windows-a, Linux-a i macOS-a može se pokrenuti i na mobilnim uređajima i to i na IOS-u i na Android-u;
- Pruža veoma jednostavan Python API - TF.Learn2 (tensorflow.contrib.learn), koji je kompatibilan sa Scikit-Learn[5] bibliotekom;
- Pruža još jedan jednostavan API - TF-slim (tensorflow.contrib.slim) za pojednostavljeno kreiranje, treniranje i evaluaciju neuronskih mreža;
- Pruža i još nekoliko API-a visokog nivoa, koji su kreirani nezavisno, kao što su Keras i Pretty Tensor;
- Njegov osnovni Python API nudi dosta fleksibilnosti u kreiranju svih vrsta izračunavanja, uključujući bilo koju arhitekturu neuronskih mreža;
- Uključuje veoma efikasnu C++ implementaciju za veliki broj operacija mašinskog učenja (*engl. Machine Learning*), kao i C++ API za definisanje korisničkih operacija visokih performansi;
- Pruža nekoliko naprednih čvorova za optimizaciju radi traženja parametara koji minimizuju troškove operacija. Jako su jednostavni za upotrebu, jer TensorFlow automatski računa gradijent funkcija koje korisnik definiše (automatsko diferenciranje);
- Dolazi sa veoma dobrim alatom za vizuelizaciju - TensorBoard, koji omogućava navigaciju kroz graf operacija, krive učenja i sično;
- Za izvršavanje TensorFlow grafova, Google je kreirao klaud servis (*engl. Cloud service*);
- Ima rastuću zajednicu koja radi na njegovom unapređenju i jedan je od trenutno najpopularnijih projekata otvorenog koda.

[5] **Scikit-learn** je besplatna biblioteka za mašinsko učenje u programskom jeziku Python. Sadrži algoritme za klasifikaciju, regresiju, klasterizaciju, kao i mnoge druge poznate algoritme, a prilagođena je saradnji sa Python numeričkim bibliotekama NumPy i SciPy.

Instalacija

Korišćenje TensorFlow biblioteke je malo komplikovanije u odnosu na druge Python biblioteke, jer vrlo često izaziva konflikte ukoliko se ne uklapa sa verzijama drugih prisutnih biblioteka. Da bi se sprečili potencijalni konflikti potrebno je kreirati posebnu promenljivu okruženja na Anaconda platformi. Ta će promenljiva sadržati verziju Pythona koji se uklapa sa TensorFlow bibliotekom i drugim paketima kao što su NumPy, Pandas i MatPlotLib. [29]

Potrebno je otvoriti Anaconda terminal ili komandnu liniju i instalirati TensorFlow okruženje izdavanjem sledeće komande:

conda create -n tensorflow_env tensorflow

Ovo novo okruženje neće imati pristup drugim Python/Anaconda paketima koje smo već instalirali u drugom okruženju, te je potrebna ponovna instalacija svih paketa koje nameravamo da koristimo. Potrebno je najpre aktivirati TensorFlow okruženje, zatim instalirati sve potrebne pakete i nakon toga deaktivirati okruženje. Komanda za aktiviranje je:

conda activate tensorflow_env

Nakon aktiviranja okruženja na početku komandne linije ili terminala trebalo bi da vidimo naziv okruženja. Zatim, sledećom komandom vrši se instalacija najčešće korišćenih paketa:

conda install -n tensorflow_env scikit-learn pandas matplotlib

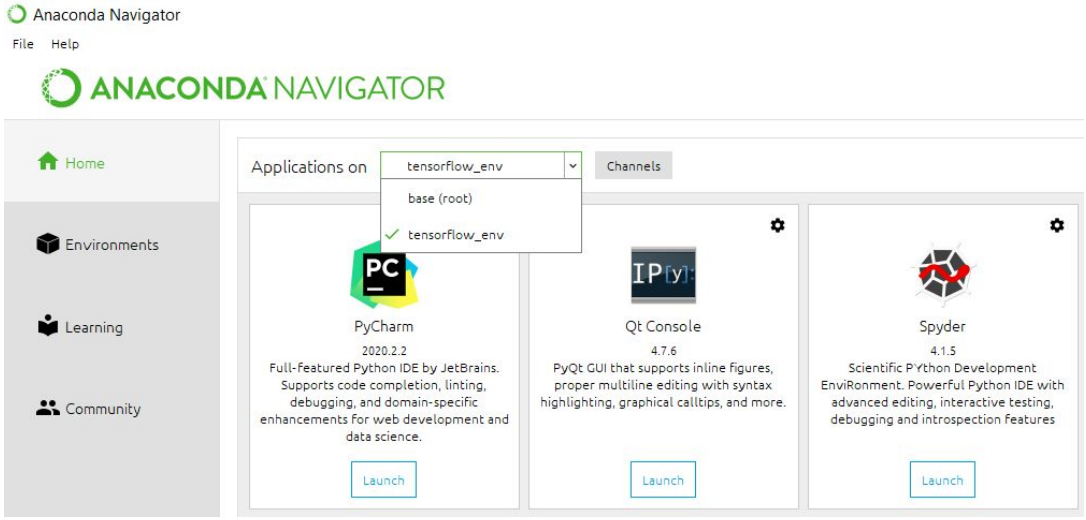
Nakon toga, sledećom komandom vršimo deaktiviranje okruženja, čiju smo instalaciju završili:

conda deactivate

Korišćenje

Moguće je instaliranom okruženju i nadalje pristupati upotrebom Anaconda terminala ili komandne linije korišćenjem komandi za aktivaciju i deaktivaciju, ali daleko jednostavnije je korišćenjem Anaconda Navigator-a.

Na Slika XX prikazan je padajući meni koji nudi dve opcije, jednu za TensorFlow i drugu za okruženje koje smo imali pre instalacije.



Slika XX: *Prikaz dostupnih okruženja u Anaconda Navigator-u*

U novom okruženju potrebno je instalirati Spyder, da bi smo mogli da unosimo kod, a nakon njegovog instaliranja možemo uvesti TensorFlow i otpočeti njegovo korišćenje komandom:

import tensorflow as tf

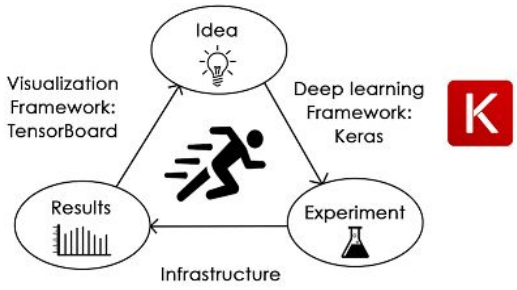
Keras

Keras je biblioteka otvorenog koda za neuronske mreže napisana u programskom jeziku Python. Dizajnirana je tako da omogući brzo eksperimentisanje sa dubokim neuronskim mrežama, a fokus je na tome da bude jednostavna za korišćenje, modularna i proširljiva.

Neke od najzanimljivijih karakteristika ove biblioteke date su u nastavku [30]:

- Keras prati najbolje prakse za smanjenje kognitivnog opterećenja: nudi konzistentne i jednostavne API-je, minimizuje broj korisničkih akcija koje su neophodne za uobičajene slučajeve korišćenja i omogućava jasne i primenljive poruke o greškama. Takođe je prate obimna dokumentacija i niz tutorijala za programere;

- Keras je najviše korišćeno programsko okruženje za duboko učenje od top-5 pobedničkih timova na Kaggle-u. Razlog je taj što Keras olakšava izvođenje novih eksperimenata i omogućava brže isprobavanje više ideja od konkurencije;



Slika XX: *Petlja napretka korišćenjem Keras API-a*

- Kako je Keras izgrađena nad Tensorflow 2.0 bibliotekom, predstavlja jako snažno okruženje koje može da se skalira na ogromne klastere GPU-a ili na čitav TPU pod[6] i to vrlo jednostavno;
- Keras modeli se mogu eksportovati na JavaScript i izvršavati direktno u pretraživaču ili na TF Lite[7] radi izvršavanja na IOS-u, Androidu ili embedded uređajima;
- Keras je centralni deo jako spregnutog TensorFlow 2.0 sistema, jer pokriva svaki korak procesa razvoja modela mašinskog učenja, od upravljanja podacima do hiperparametara za treniranje i krajnjeg rešenja;
- Keras koriste CERN i NASA, kao i mnoge druge naučne organizacije u svetu, jer ima veliku fleksibilnost čak i na najnižim nivoima za implementaciju proizvoljnih istraživačkih ideja, dok nudi opcione pogodnosti na visokom nivou radi ubrzavanja ciklusa eksperimentisanja;
- Zbog činjenice da je jako jednostavan za korišćenje i fokusiran na korisnička iskustvam Keras je rešenje za duboko učenje koje biraju mnogi univerziteti i naširoko se preporučuje kao jedan od najboljih načina za upoznavanje sa dubokim učenjem.

[6] **TPU pod** je konfiguracija u Google-ovom centru za podatke koji ima niz TPU (*engl. Tensor Processing Unit*) uređaja međusobno povezanih veoma brzom mrežom. Kada je koristimo za izvršavanje sva izračunavanja mašinskog učenja koja su nam potrebna se distribuiraju na sve TPU uređaje.

[7] **TF Lite** (*engl. TensorFlow Lite*) je skup alata koji programerima omogućava izvršavaje TensorFlow modela na mobilnim, embedded i IoT (*engl. Internet of Things*) uređajima.

Implementacija

Rezultati istraživanja

Zaključak

Literatura

[1] C.D.C. COVID, R. Team (February 12 - March 16, 2020) 'Severe outcomes among patients with coronavirus disease 2019 (COVID-19)—United States', *MMWR Morb Mortal Wkly Rep*. 69, 343–346.

[2] A. Remuzzi, G. Remuzzi (March 13, 2020) 'COVID-19 and Italy: what next?', *The Lancet*, 395(10231), 1225-1228.

[3] Joe Parkin Daniels (July 25, 2020) 'COVID-19 cases surge in Colombia', *The Lancet*, 396(10246), 227.

[4] The Lancet (September 26, 2020) 'COVID-19 in India: the dangers of false optimism', *The Lancet*, 396(10255), 867.

[5] E. Mahase (2020) 'Coronavirus: covid-19 has killed more people than SARS and MERS combined, despite lower case fatality rate', *BMJ*, 368:m641.

[6] C. Huang, Y. Wang, X. Li, L. Ren, J. Zhao et al. (2020) 'Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China', *The Lancet*, 395(10223), 497–506.

[7] https://www.worldometers.info/coronavirus/, (n.d.).

[8] V.M. Corman, O. Landt, M. Kaiser, R. Molenkamp et al. (2020) 'Detection of 2019 novel coronavirus (2019-nCoV) by real-time RT-PCR', *Eurosurveillance*, 25 2000045.

[9] T. Ai, Z. Yang, H. Hou, C. Zhan et al. (Feb 26, 2020) 'Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases', *RSNA*. 296(2), E32–E40.

[10] M.Y. Ng, E. YP Lee, J. Yang, F. Yang, X. Li, H. Wang et al. (Feb 13 2020) 'Imaging profile of the COVID-19 infection: Radiologic findings and literature review', *Cardiothoracic Imaging*, 2(1).

[11] C. Huang Y. Wang et al. (2020) 'Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China', *The Lancet*, 395, 497–506.

[12] W. Guan, Z. Ni, Y. Hu, W. Liang, C. Ou, J. He et al. (2020) 'Clinical characteristics of coronavirus disease 2019 in China', *N Engl J Med*, 382, 1708-1720

[13] A. Krizhevsky, I. Sutskever, and G. Hinton (2012) 'ImageNet classification with deep convolutional networks', *Advances in Neural Processing Systems*, 25, 1097–1105.

[14] Neha Patha (September 23, 2020) 'What Does COVID-19 Do to Your Lungs?', *WebMD*. Available at: https://www.webmd.com/lung/what-does-covid-do-to-your-lungs#1 (Accessed 30 September 2020)

[15] C. Rachna (April 5, 2017) 'Difference between X-ray and CT Scan', Bio Differences. Available at: https://biodifferences.com/difference-between-x-ray-and-ct-scan.html (Accessed 02 October 2020)

[16] J.P. Cohen, P. Morrison, L. Dao (2020) 'COVID-19 image data collection', *ArXiv Prepr.* ArXiv2003.11597.

[17] A. Narin, C. Kaya, Z. Pamuk (2020) 'Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks', *ArXiv Prepr.* ArXiv2003.10849.

[18] I.D. Apostolopoulos, T.A. Mpesiana (2020) 'Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks', *Phys Eng Sci Med 43*, 635–640.

[19] L. Wang, A. Wong (2020) 'Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest radiography images', *ArXiv Prepr.* ArXiv2003.09871.

[20] E.E.-D. Hemdan, M.A. Shouman, M.E. Karar (2020) 'Covidx-net: A framework of deep learning classifiers to diagnose covid19 in x-ray images', *ArXiv Prepr.* ArXiv2003.11055.

[21] François Chollet (2018) 'Deep Learning with Python', 1st edn. United States of America: Manning Publications Co.

[22] Aurélien Géron (2017) 'Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems', 1st edn. United States of America: O'Reilly Media.

[23] Muhammad Imran Razzak, Saeeda Naz and Ahmad Zaib (2018) 'Deep Learning for Medical Image Processing: Overview, Challenges and Future', *Classification in BioApps. Lecture Notes in Computational Vision and Biomechanics*, vol 26. Springer, Cham.

[24] Sindhu, Vaidhehi V. (2017) 'Classification of Human Organ Using Image Processing', *Orient.J. Comp. Sci. and Tech*, 10(2).

[25] D. Hubel and T. Wiesel (1958) 'Single Unit Activity in Striate Cortex of Unrestrained Cats', *The Journal of physiology*, 147(2), 226–238.

[26] D. Hubel and T. Wiesel (1959) 'Receptive Fields of Single Neurons in the Cat's Striate Cortex', *The Journal of physiology*, 148(3), 574–591.

[27] D. Hubel and T. Wiesel (1968) 'Receptive Fields and Functional Architecture of Monkey Striate Cortex', *The Journal of physiology,* 195, 215–243.

[28] Y. LeCun et al. (1998) 'Gradient-Based Learning Applied to Document Recognition', *Proceedings of the IEEE*, 86(11), 2278-2324.

[29] Michael Allen (2018) 'Python for healthcare modelling and data science - 106. Installing and using TensorFlow in Anaconda' Available at: https://pythonhealthcare.org/2018/12/19/106-installing-and-using-tensorflow-using-anaconda/ (Accessed 26 September 2020)

[30] François Chollet (2015) 'Keras: the Python deep learning API' Available at: https://keras.io/ (Accessed 27 September 2020)