SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Deep Dream za vizualizaciju neuronskih mreža

Daniel Bratulić

Voditelj: Tomislav Hrkać

Zagreb, svibanj, 2017

**Sadržaj**

[1. Uvod 3](#_Toc479542515)

[2. Duboko učenje 4](#_Toc479542516)

[2.1 Konvolucijske neuronske mreže 5](#_Toc479542517)

[2.1.1 Lokalna receptivna polja 5](#_Toc479542518)

[2.1.2 Dijeljene težine 6](#_Toc479542519)

[2.1.3 Sloj sažimanja (engl. Pooling layer) 7](#_Toc479542520)

[3. Tensorflow paket 8](#_Toc479542521)

[5. Zaključak 10](#_Toc479542522)

[6. Literatura 11](#_Toc479542523)

[7. Sažetak 12](#_Toc479542524)

# Uvod

Zadatak ovog seminarskog rada je pokušati shvatiti što se događa u skrivenim slojevima dubokih neuronskih mreža. Takav stil strojnog učenja, ili prikladnije nazvanog dubokog učenja je danas sve veći hit u svijetu i jedna od najboljih i najkorištenijih tehnika u području računalnog vida. Uz računalni vid, duboko učenje je vrlo primijenjeno u obradi zvuka i teksta.

Pokušavam riješiti ovaj zadatak jer iako su duboke neuronske mreže vrlo popularne i vrlo uspješne u rješavanju raznih klasifikacijskih zadataka, još uvijek postoji pitanje kako točno izgledaju značajke skrivenih slojeva te što su zapravo te značajke.

Samo nekoliko radova u posljednjih pet godina (u koje je duboko učenje zadobilo veliku popularnost) se dotaklo te teme, a u ovom radu dotaknut ću se blog objave koju su napisali suradnici u Googleu, nazvana Inceptionism (referenca).

Za rješavanje ovog problema koristit ću programski alat Tensorflow(referenca) te jupyter bilježnice(referenca) u kojima ću prikazati svoj kod. Tensorflow je danas jedan od vodećih "open source" programskih paketa za strojno učenje. Dostupan je u jezicima Python i C++ te se trenutno radi na podršci za jezik Java. U ovom radu koristit ću podršku za programski jezik Python.

Organizacija ostatka rada će biti strukturirana na sljedeći način, u prvom poglavlju opisat ću podrobnije opisati principe dubokog učenja i neuronskih mreža te u čemu se oni zapravo razlikuju od dosadašnjih tehnika strojnog učenja, također navesti ću prednosti i mane dubokih neuronskih mreža.

Sljedeće poglavlje biti će zaduženo za pobliži pogled na Tensorflow. Opisat ću na koji način funkcionira te priložiti nekoliko jednostavnih isječaka koda koji predstavljaju neki jednostavan posao. Za to ću koristiti neke prednaučene modele, poput onog modela koji je naučen na skupu podataka za natjecanje ImageNet[[1]](#endnote-1), najveće natjecanje u području računalnog vida. Opisat ću također ukratko i Jupyter Notebook paket/sučelje u sklopu ovog poglavlja pošto je to okolina u kojoj ću raditi.

Treće poglavlje, i posljednje veliko poglavlje će se dotaknuti analize samih pristupa vizualizaciji skrivenih slojeva neuronske mreže, odnosno "Deep Dreaming"-a. Uz to, napravit ću osvrt na nekoliko ostalih radova koji su se bavili tematikom vizualizacije i analize skrivenih slojeva dubokih neuronskih mreža.

Na kraju su, kao i uvijek, zaključak i literatura.

# Duboko učenje

Strojno učenje je u današnje doba zastupljeno svugdje oko nas, vrlo često nismo niti toga svjesni. Svaki put kada utipkamo neki izraz u najpoznatiju svjetsku tražilicu Google, u pozadini se vrti algoritam strojnog učenja koji rangira rezultate pretrage, ako zatim odaberemo neku web trgovinu koja nam se prikazala u rezultatima, vrlo vjerojatno slijedi još jedan algoritam strojnog učenja koji nam preporučuje razne proizvode na temelju naših dosadašnjih odabira.

Postoji i velika primjena u području računalnog vida, od prepoznavanja objekata na slikama, segmentacije slika, grupiranja slika, prepoznavanje rubova, uglova i slično. Gotovo svaki problem se može efikasnije riješiti korištenjem metoda strojnog učenja.

Upravo je zbog računalnog vida područje dubokog učenja sve više zaživjelo i doživjelo veliki procvat u posljednjih pet godina. Metode dubokog učenja su vrlo uspješne u raznim zadacima iz područja računalnog vida zbog toga što mogu puno lakše baratati sa sirovim podacima, odnosno slikama.

Klasične metode strojnog učenja iziskuju pronalaženje i odabir najkvalitetnijih značajki unutar slika, poput rubova, histograma i slično. Tim značajkama bismo predstavili sirovi podatak – sliku, u vektorskom obliku i na taj način bismo mogli krenuti na sljedeći uobičajeni korak u strojnom učenju, odabiru modela.

Duboko učenje je podgrana strojnog učenja. Metode dubokog učenja koriste niz slojeva koji imaju nelinearne jedinice za transformaciju podataka koji se dobivaju iz prethodnih slojeva.

Glavno oružje dubokog učenja su umjetne neuronske mreže koje praktički funkcioniraju na spomenuti način u prethodnom odlomku. O njihovoj uporabi u metodama dubokog učenja ću govoriti kroz ovaj rad te će na njima biti glavni fokus, točnije na još specifičnijim neuronskim mrežama koje su nazvane konvolucijske neuronske mreže.

U slučaju slika, razni slojevi odrađuju različit posao te na taj način simuliraju ekstrakcije značajki kod konvencionalnih, već spomenutih metoda strojnog učenja. Na primjer, prvih nekoliko slojeva neuronske mreže prepoznaju pozicije rubova i njihove orijentacije, a neki dublji slojevi mogu slagati te značajke te sve ostale otkrivene iz prethodnih slojeva i iskoristiti ih za (npr.) detekciju objekata, koji je jedan od čestih zadataka dubokog učenja.

Već spomenute nelinearne jedinice, koje su u neuronskim mrežama prikladno nazvane neuroni, su najveća prednost dubokog učenja. Sama činjenica što se značajke i veze između njih pronalaze u skrivenim slojevima, umjesto da programer mora razmišljati kako i na koji način odabrati te značajke, uklanja puno posla prije samog odabira nekog modela.

Ovakav postupak i pristup pokazao se kao trenutno najbolji u svijetu na natjecanju u prepoznavanju objekata na slikama, ImageNet, spomenuto u uvodnom poglavlju.

Najčešći oblik strojnog učenja u današnje vrijeme jest nadgledano učenje koje zahtjeva postojanje podataka i njihovih oznaka, npr. Zamislimo da imamo dvodimenzionalni prostor i nekoliko točaka unutar njih te je naš zadatak odrediti pripadnost razredima budućih točaka koje se pojavljuju. Tada će se svaki podatak sastojati od koordinata (x,y) te pripadnosti razreda z, koji se mijenja od 1 do N, ovisno koliko postoji razreda. Ovo je vrlo jednostavan primjer nadgledanog učenja.

Neuronske mreže imaju težine koje se podešavaju u ovisnosti o ulaznom podatku i njihovoj oznaci. Težine se podešavaju vrlo poznatim postupkom koji se naziva gradijentni spust, te je standardan postupak u učenju neuronskih mreža danas. On funkcionira na način da se primjeri propuste kroz neuronsku mrežu te se na izlazu odredi pripadnost jedne od mogućih N klasa, najčešće u zadnjem neuronu bude upravo N izlaznih neurona te onaj koji ima najveću vjerojatnost određuje klasu. Zatim u ovisnosti izlaza neuronske mreže počinje prostiranje unatrag gdje se podešavaju težine svih neurona. Postoji više varijanti gradijentnog spusta, primjerice skupni i stohastički gradijentni spust.

Kod arhitektura dubokih neuronskih mreža te jedinice su nelinearne te na taj način omogućuju prepoznavanje sitnih razlika između pojedinih slika, ili objekata na tim slikama.

Ono što je imalo jako puno utjecaja na uspješnost dubokog učenja u posljednjih nekoliko godina jesu grafičke kartice. Mogućnost da se treniranje takvih ogromnih neuronskih mreža vrši na grafičkim karticama znatno ubrzava cjelokupni proces. Na taj način se vrijeme učenja smanjilo s nekoliko tjedana na svega dan ili čak manje. Mnogi programski paketi imaju podršku za učenje modela na grafičkim karticama te je taj pristup danas uobičajen.

## Konvolucijske neuronske mreže

Konvolucijske neuronske mreže su danas najučestaliji tip neuronskih mreža koje se koriste u području računalnog vida. Koriste posebnu arhitekturu koja je prilagođena upravo za klasifikaciju slika i imaju barem jedan sloj koji koristi konvoluciju umjesto matričnog produkta, otkud su dobile ime. Konvolucija je operacija definirana nad dvije funkcije realne varijable.

Optimiranje konvolucijskih mreža se vrši na isti način kao i optimiranje neuronskih mreža, dakle stohastičkim (ili grupnim) gradijentnim spustom, no jednadžbe unutar algoritma nisu jednake zbog različitih arhitektura mreža i različitih priroda tih mreža.

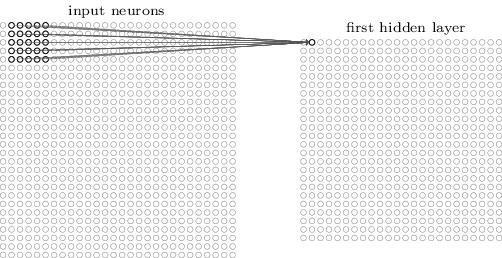
Takve mreže imaju tri osnovne ideje: lokalna receptivna polja (engl. *local receptive fields*), dijeljene težine (engl. *shared weights*) i udruživanje (engl. *pooling*).

### Lokalna receptivna polja ili konvolucije

Svaki skriveni neuron u prvom skrivenom sloju nije spojen sa svim ulaznim neuronima već samo nekim dijelom podataka, odnosno piksela. Svaki skriveni neuron u tom prvom skrivenom sloju pokazuje na dijelove ulazne slike, koji su dimenzija npr. 3x3, 4x4, 5x5 piksela ili nekih drugih dimenzija. Svaki takav dio ulazne slike (koji je obuhvaćen spomenutim dimenzijama) naziva se lokalno receptivno polje za neki skriveni neuron.

Skriveni neuron će pokušati naučiti analizirati to specifično receptivno polje s kojim je povezan. Zbog ovakvog pristupa, ostvaruje se jedna velika prednost konvolucijskih mreža, a to je što skriveni sloj neće imate jednak broj neurona kao ulazni podaci, niti će biti potpuno povezan sa svim ulaznim pikselima, već će tih skrivenih neurona biti manje i na taj način će model biti jednostavniji od običnih potpuno povezanih mreža (barem na tom dijelu mreže).

Taj prozor dimenzija 5x5 ili nekih sličnih dimenzija se pomiče kroz sliku i mapira na ostale skrivene neurone. Na taj način, kao što je u prethodnom odlomku spomenuto, imamo puno manje težina i veza, tj. imamo puno manje parametara.

  
Slika : Lokalna receptivna polja(iz Literatura: 2)

### Dijeljene težine

U konvolucijskim mrežama postoji još jedno poboljšanje u usporedbi s običnim unaprijednim mrežama, za svaki skriveni neuron u prvom konvolucijskom sloju (na konvolucijski sloj se odnose ovo i prethodno svojstvo konvolucijske mreže) vrijedi da ima dijeljene težine i pristranost (engl. *bias)* sa svim ostalim neuronima u tom sloju.

Dakle ako imamo situaciju kao na slici 1 gdje postoje 24x24 skrivena neurona, gdje na svakog od skrivenog neurona pokazuje 5x5 piksela iz ulaznog sloja, imali bismo upravo 5x5 težina koje dijele svi neuroni. To nam kazuje kako će neuroni u skrivenom sloju detektirati istu značajku, ali na drugim lokacijama na slici.

Poanta dijeljenih težina jest da ako skriveni neuron detektira (npr.) neke rubove na specifičnom lokalnom polju, vrlo vjerojatno je da će i neki drugi skriveni neuron htjeti detektirati rubove u svom specifičnom lokalnom polju, a dijeljene težine olakšavaju to. Drugim riječima, ako postoji slika psa te je pomaknemo u neku stranu, ili je zarotiramo, to je još uvijek slika psa. Neće se izgubiti informacije o toj slici.

Takva poveznica između ulaznog sloja i prvog skrivenog sloja se često naziva mapa značajki. Velika prednost korištenja dijeljenih težina jest smanjeni broj parametara, kao što sam prethodno spomenuo, postojati će 5x5 težina te *bias* što zajedno daje 26 parametara.

To je puno manje u usporedbi s potpuno povezanom neuronskom mrežom, gdje bi ukupan broj parametara bio jednak 28 x 28 (ulazne dimenzije slike, lijeva matrica na slici 1) x 28 (neki proizvoljan broj skrivenih neurona) naspram 26 x 10 (proizvoljan broj mapa značajki).  
Uvijek se koristi više mapa značajki kako bi omogućile što bolju potragu i ekstrakciju značajki sa slika.

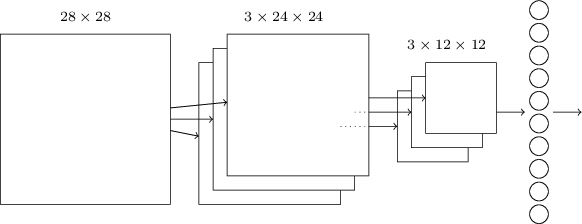
### Sloj sažimanja (*engl. Pooling layer*)

Slojevi sažimanja se nalaze odmah iza konvolucijskih slojeva, oni pojednostavljuju informacije s izlaznih slojeva konvolucijskog sloja. Uzimaju izlaz iz mape značajki te sažmu informacije iz nekog manjeg područja na slici (npr. 2x2) u jedan neuron u svojem sloju. Postoji nekoliko vrsta sažimanja, a najpopularnije tehnike se zasnivaju na osnovnom statističkom agregiranju.

Najpopularnija tehnika jest *max-pooling,* odnosno maksimalno sažimanje, gdje se uzima maksimalna vrijednost iz tog manjeg područja u prethodnom sloju te spremi u neuron u sloju sažimanja.

Sloj sažimanja se sastoji od još manjeg broja neurona, no što ih ima u konvolucijskom sloju (prvi skriveni sloj nakon ulaznog). *Max pooling* možemo zamisliti kao način na koji mreža pita je li pronađena neka značajka negdje u promatranoj regiji na slici.

Postoje ostali pristupi osim *max poolinga*, poput *mean pooling* koji se uzima prosjek vrijednosti unutar regije te *L2 pooling*, koji uzima korijen sume svih kvadrata unutar 2x2 prozora na slici te ostala agregatne funkcije.

  
Slika : Jednostavna konvolucijska neuronska mreža

Na slici 2 je prikazana jednostavna konvolucijska neuronska mreža koja se sastoji od prethodno nabrojanih slojeva. Prvi je ulazni sloj, čije su dimenzije 28x28, zatim slijedi konvolucijski sloj koji ima 3 mape značajki te nakon njega slijedi sloj sažimanja ( *pooling* ). Na kraju je izlazni sloj koji je potpuno povezan sa slojem sažimanja kao i u običnim neuronskim mrežama.

### Sloj ispadanja (engl. Dropout layer)

Sloj ispadanja je praktički postao standard u arhitekturama konvolucijskih neuronskih mreža nakon što je odlično poboljšanje spomenuo Geoffrey Hinton zajedno sa suradnicima u [5]. Taj sloj bi zapravo služio kao oružje protiv prenaučenosti, princip rada tog sloja jest da se nasumično odabere neuroni koji bi se prilikom treniranja jednostavno ugasili, odnosno vrijednost njihovih aktivacijskih funkcija bi se postavila na 0. Udio neurona koji bi se ugasili određen je nekom vrijednošću između 0 i 1. Na taj način mreža nema redundantnih reprezentacija. Dakle, sloj ispadanja služi kao jedna vrsta regularizacije.

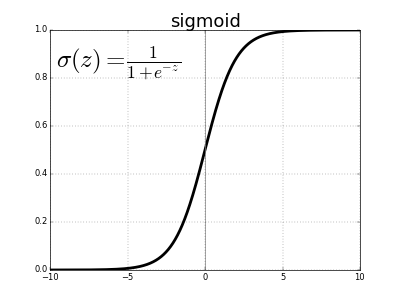
Prilikom testiranja mreže, vrijednost vjerojatnosti ispadanja nekog neurona postavila bi se na 0, dakle niti jedan neuron se ne bi ispustio kako bi se maksimalno iskoristila mreža.



Slika 3 Sloj ispadanja (engl. Dropout layer)

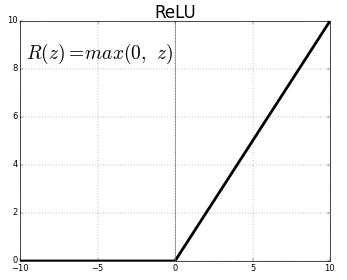
### Razne aktivacijske funkcije

Uz prethodno navedena svojstva konvolucijskih mreža još je jedna velika stvar imala utjecaj na njihovu uspješnost u raznim izazovima računalnog vida, to je raznovrsnost aktivacijskih funkcija. Sve do prije nekoliko godina uobičajena je bila sigmoidalna aktivacijska funkcija.



Slika 4 Sigmoidalna aktivacijska funkcija

Trenutno najbitnija aktivacijska funkcija za duboke neuronske mreže jest ReLU, odnosno zglobnica (*engl. Rectified Linear Unit*). Za razliku od sigmoide ona ne dovodi do zasićenja neurona koji su inicijalizirani na jako niske negativne vrijednosti ili jako visoke pozitivne vrijednosti te konvergira brže. Uz to, računalno je puno efikasnije od sigmoide. Formula je puno jednostavnija i glasi f(x) = max(0, x). Problem kod ove aktivacijske funkcije su mrtvi neuroni koji se mogu javiti ako se vrijednost inicijalizira na vrijednost manje od 0, no to se lako izbjegne manipulacijom inicijalizacije vrijednosti.



Slika 5 Funkcija zglobnice, ReLU aktivacijska funkcija

# TensorFlow

TensorFlow je programski paket za primjenu algoritama strojnog i dubokog učenja koji je izašao u uporabu tokom 2015. godine. Prvo stabilno izdanje, dakle verzija TensorFlow 1.0 izašlo je u 2. mjesecu 2017. godine.

Autor paketa TensorFlow jest kompanija Google, točnije jedan od timova unutar Google-a, Google Brain Team. Google je za svoju unutarnju uporabu prethodno koristio paket DistBelief u istu svrhu, dakle za razvoj modela strojnog i dubokog učenja. Nakon uvida u nedostatke tog paketa, odlučili su nadograditi i pojednostaviti neke dijelove te je iz toga nastao TensorFlow.

TensorFlow je danas vodeći paket u području dubokog učenja, a konkurenti su mu Theano, Caffe, Torch, Keras i mnogi drugi. Ono što ga posebice izdvaja od konkurencije jest njegova srž, računalni graf koji se sastoji od mnoštvo čvorova i bridova. U takvom jednom grafu, čvorovi predstavljaju operacije, dok bridovi između čvorova predstavljaju podatke koji teku između pojedinih operacija. Takav graf se jednostavno izgrađuje povezivanjem više operacija i time možemo razbiti izgradnju modela na više jednostavnijih dijelova.

Sljedeća pozitivna stvar TensorFlowa jest mogućnost jednostavnog i vrlo učinkovitog prikaza složenih modela neuronske mreže pomoću alata koji se zove TensorBoard, taj alat je instaliran zajedno s TensorFlowom. Uz samu vizualizaciju modela, moguće je i pratiti stanje u modelima tokom učenja.

Uz to, TensorFlow je moguće pokretati na više CPU i što je još važnije, na više GPU uređaja. Kao što je spomenuto u poglavlju 2, jačanje grafičkih jedinica je značajno pogodilo razvoju dubokog učenja, TensorFlow automatski prepoznaje instalirano razvojno okruženje za grafičke kartice (nVIDIA predvodi to područje, s CUDA tehnologijom) te automatski omogućava učenje na grafičkim karticama.

Za kraj, posljednja velika stvar koja ističe TensorFlow od ostatka konkurencije jest mogućnost pokretanja na Linuxu, macOS i nekim mobilnim platformama poput Androida i iOS-a. TensorFlow API dostupan je za više programskih jezika, što je također vrlo zgodna stvar. Trenutno dostupni jezici su Python, C++, Haskell, Java i Go. U ovom radu će fokus i primjeri biti napisani u Pythonu zbog brzine i jednostavnosti samog jezika kao i lakoće čitanja.

## Osnove TensorFlowa

Kao što je spomenuto, osnova rada TensorFlowa su računalni grafovi kroz koje se definiraju operacije koje će se izvoditi. Nakon što se definiraju te operacije potrebno je predati vrijednosti na ulaze takvog grafa, ukoliko je to uopće potrebno. Nabrojit ću neke osnovne i neophodne jedinica za rad u TensorFlowu kao i pojmove koji predstavljaju principe po kojima radi TensorFlow.

### Operacije

Operacije su predstavljene svim čvorovima u grafu i zadužene su za transformaciju ili kombiniranje podataka koji se kreću kroz graf. Operacije mogu biti razne stvari, poput običnog zbrajanja, množenja pa zatim matrično množenje ili čak neki dijelovi neuronskih mreža, poput prethodno spomenutih aktivacijskih funkcija. Detaljan prikaz vidljiv je u Tablici 1. Operacije mogu biti nula ili više ulaza te nula ili više izlaza, npr. konstanta je operacija koja nema ulaza i ima jedan izlaz.

|  |  |
| --- | --- |
| **Kategorija** | **Primjer operacije** |
| Matematičke operacije s elementima | Add, Sub, Mul, Div, Exp, Log, Equal, ... |
| Operacije s poljima | Concat, Slice, Split, Rank, Shape, … |
| Operacije s matricama | MatMul, MatrixInverse, … |
| Operacije sa stanjima | Variable, Assign, AssignAdd, … |
| Blokovi za neuronske mreže | SoftMax, Sigmoid, ReLU, MaxPool, ... |

Tablica 1 Prikaz primjera operacija u TensorFlowu

### Tenzori

Tenzori su osnovne jedinice za pohranu podataka unutar TensorFlowa. Tenzori se predaju kao ulaz čvorovima i oni su ujedno izlaz iz pojedinih operacija. To su matrice proizvoljnih dimenzija nekog fiksno određenog tipa. Svaki tenzor ima svoj rang (*engl. rank,* različito od ranga matrice), koji predstavlja dimenziju tenzora. Dakle, matrice bi imale rang 2, vektori bi imali rang 1 dok bi obične skalarne vrijednosti imale rang 0. Tenzori ne drže vrijednosti u svojoj memoriji već pružaju sučelje za dohvat vrijednosti koje referencira taj tenzor.

### Varijable

Tenzori često bivaju uništeni prilikom rada s modelima dubokih neuronskih mreža, što nam je ponekad u redu, no nekad bismo htjeli zapamtiti te vrijednosti kako bismo ih mogli višestruko iskoristit. Za to nam služe varijable (*engl. Variable*), koje su u TensorFlowu označene s tf.Variable. One su operacije koje pamte tenzore te na taj način dozvoljavaju promjenu vrijednosti unutar tenzora.

### Sesije (engl. Sessions)

Nakon što izgradimo naš računalni graf s prethodno spomenutim dijelovima i postupcima, potrebno je izvršiti taj graf. Način na koji je to ostvareno u TensorFlowu je unutar posebnog okruženja, odnosno sesije (*engl. Session*). Graf se izvršava tako što se pozove metoda *run()* nad objektom tipa *tf.Session*. Toj metodi se preda ciljni čvor koji je izlaz grafa te ukoliko treba, predaju se vrijednosti za čvorove koji predstavljaju ulazne vrijednosti(*engl. placeholders*).

### TensorBoard

TensorBoard je alat, te se zapravo logički po opisu razlikuje od prethodnih pojmova, pošto su oni neophodni za rad bilo kojeg programa u TensorFlowu. Prikaz grafova u TensorBoardu je opcionalan, no kao što je spomenuto u samom uvodu u TensorFlow, to svojstvo je vrlo moćno i od velike pomoći. Bez TensorBoarda bi bilo vrlo teško predočiti neke duboke mreže pošto one znaju imati puno više od 10000 čvorova.

Strukture podataka koje TensorBoard može vizualizirati jesu Summary objekti unutar TensorFlowa (tf.summary). To mogu biti slike, tekstovi, histogrami, zvukovi ili obični skalari, poput stope učenja ili vrijednosti funkcije gubitka. Također, svaka operacija je jedan tf.summary objekt čime se olakšava prikaz grafova.

Kako bi se spomenuti objekti tipa tf.Summary mogli prikazati, mora se pozvati objekt tf.summary.FileWriter koji je zadužen za spremanje čvorova koji će se iscrtavati te privremenih vrijednosti koje se dobivaju tokom treniranja.

## Primjer kodova u TensorFlowu

### Jednostavan početni primjer i prikaz grafa



Isječak koda 1 Osnovna funkcionalnost TensorFlowa

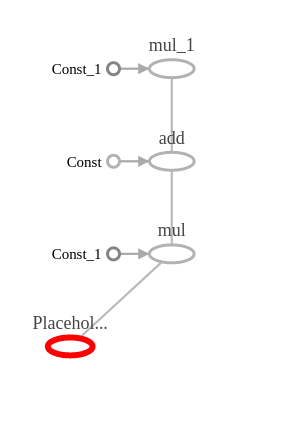
Ovaj primjer pokazuje osnovni rad s TensorFlowom, radi se o običnom zbrajanju i množenju, no kroz funkcionalnost TensorFlowa.

U ovom isječku koda potrebno je objasniti neke notacije koje još nisu spomenute. Tf.constant jest operacija koja sadržava konstantu, to je čvor koji nema ulaza već ima samo jedan izlaz i to skalar koji je zapisan pri definiciji.

Tip podatka koji je vezan uz varijablu x, tf.placeholder je već natuknut u potpoglavlju o sesijama. To je operacija kojoj nije odmah zadana vrijednost već joj se predaje prilikom izvršavanja sesije, dakle ulazna vrijednost, to se može vidjeti prilikom poziva metode run(), ondje se argumentu feed\_dict predaje rječnik koji inicijalizira sve varijable koje su tipa tf.placeholder s njihovim pripadnim vrijednostima, u ovom slučaju to je varijabla x i vrijednost 3.

Na slici 6 je vidljiv graf koji prikazuje ovaj isječak koda. Placeholder je varijabla x kojoj zadajemo vrijednost. Const\_1 je varijabla b, dok je Const varijabla a. Moguće je izmijeniti imena čvorova, no za ovaj osnovni primjer to nije urađeno.

Mul i add su operacije množenja odnosno zbrajanja. Prilikom interaktivnog rada na TensorBoardu moguće je odabrati svaki pojedini čvor te ispitati ulaze i izlaze tih čvorova.



Slika 6 Graf koji prikazuje prethodni isječak koda

### Prikaz jednostavne konvolucijske mreže

# Vizualizacija neuronskih mreža

TODO:

Opisat model Inception koji će se koristi

Nasrat nešto o DeepDreamu

Nasrat nešto općenito o vizualizaciji neuronskih mreža

Prikazat isječak koda

Prikazat kul slikice i objasnit šta se događa

Spomenut ostale načine vizualizacije

# Zaključak

# Literatura

[1] Neural Networks and Deep Learning: <http://neuralnetworksanddeeplearning.com/>, Michael Nielsen

[2] Deep learning, Review: Yann LeCun, Yoshua Bengio, Geoffrey Hinton, <http://pages.cs.wisc.edu/~dyer/cs540/handouts/deep-learning-nature2015.pdf>

[3] Predavanja iz Dubokog Učenja, FER: Siniša Šegvić, Zoran Kalafatić, Sven Lončarić, Marko Subašić, Marko Čupić

[4] Neural Networks github.io post(slike): <https://ml4a.github.io/ml4a/neural_networks/>

[5] Dropout: A Simple Way to Prevent Neural Networks from Overfitting: Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, <http://www.jmlr.org/papers/volume15/srivastava14a.old/source/srivastava14a.pdf>

[6] Andrej Karpathy, Justin Johnson, Fei Fei Li: CS231n Convolutional Neural Networks for Visual Recognition, Stanford course, <http://cs231n.github.io/>

[7] Prva Laboratorijska vježba iz dubokog učenja: Siniša Šegvić, <http://zemris.fer.hr/~ssegvic/du/lab1.shtml>

[8] Paper Tensorflow 1

[9] Paper tensorflow 2

[10] Wikipedia TensorFlow

[11] TensorFlow gettingStarted

# Sažetak

1. ImageNet natjecanje: <http://www.image-net.org/challenges/LSVRC/> [↑](#endnote-ref-1)