SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5701

Otkrivanje izvandistribucijskih podataka analizom izlaza dubokog modela

Bruno Kovač

Zagreb, srpanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 16. ožujka 2018.

ZAVRŠNI ZADATAK br. 5701

Pristupnik: **Bruno Kovač (0036490749)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Otkrivanje izvandistribucijskih podataka analizom izlaza dubokog modela**

Opis zadatka:

Procjena nesigurnosti predikcije vrlo je važan sastojak mnogih praktičnih primjena konvolucijskih modela računalnog vida. Posebno je zanimljivo procijeniti vjerojatnost da se analizirani podatak nalazi izvan distribucije skupa za učenje. U ovom radu razmatramo pristup koji tu procjenu provodi analizom izlaza dubokog klasifikacijskog modela.
U okviru rada, potrebno je istražiti postojeće pristupe za otkrivanje izvandistribucijskih podataka. Proučiti dokumentacije programskih okvira Tensorflow i PyTorch te biblioteke programskog jezika Python za rukovanje matricama i slikama. Odabrati jednostavnu klasifikacijsku arhitekturu te je naučiti i evaluirati na podatkovnom skupu CIFAR-10. Procijeniti vjerojatnost da je na ulaz modela doveden izvandistribucijski podatak. Prikazati i ocijeniti ostvarene rezultate.
Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 15. lipnja 2018.

Mentor:
Prof. dr. sc. Siniša Šegvić

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Ovim putem posebno bih se htio zahvaliti mentoru prof. dr. sc. Siniši Šegviću na pomoći te uloženom trudu i vremenu pri izradi ovoga rada. Isto tako želim zahvaliti i kolegama Antoniju Anđeliću, Antoniju Borcu te Ivanu Šegi na istome.

SADRŽAJ

1. Uvod

U današnje vrijeme svjedočimo ubrzanom razvoju raznih područja umjetne inteligencije, a izuzetak nisu niti neuronske mreže koje svakoga dana postaju sve dublje te sve bolje u obavljanju svojih zadataka i predikcija temeljenih na ulaznim podacima. Ovaj brzi razvoj tih relativno novih tehnologija pod budnim je okom javnosti gdje se sve više pažnje posvjećuje ovom području te je svaka pogrešna predikcija korištenih neuronskih mreža i modela pod posebnim povećalom. Stoga se kao ključan problem pokazuje sve značajnija potreba za otkrivanjem potencijalnih pogrešaka koje takvi modeli mogu učiniti. Kao što znamo, za definiranje svakog modela koji će koristiti u ovu svrhu prvi korak je svakako treniranje pripremljene neuronske mreže. Nakon ovog ključnog koraka, model je spreman za primanje novih podataka - podataka koje nikada prije nije vidio. Unatoč tome što trenirani model sada testiramo na novim podacima, on će i dalje s određenom sigurnošću predvidjeti izlaz i prikazati ga korisniku. Problemi nastaju kada taj korisnik npr. koristi aplikaciju koja se temelji na konvolucijskoj neuronskoj mreži koja tada na temelju fotografije analizira je li neki madež opasan ili ne. Upravo u ovom slučaju nužno je odrediti mjeru u kojoj možemo vjerovati izlazu modela, a sve u svrhu toga kako krajnji korisnici ne bi dobili netočne informacije koje u ovom slučaju mogu imati dalekosežne posljedice.

Kako je ova problematika očito vrlo važna, u ovom radu pokušat ćemo ponuditi određene metode kojima možemo provjeravati odnos ulaznog podatka i onih na kojima je naš model treniran. Ukoliko se ti podaci dosta razlikuju, možemo reći da je testni podatak izvandistribucijski primjerak te u tom slučaju napomenuti korisnicima kako ne možemo sa dovoljnom sigurnošću vjerovati rezultatu kojega nam je ponudio korišteni istrenirani model.

Kroz sljedećih nekoliko poglavlja bit će opisane konvolucijske neuronske mreže općenito, prikazan konkretan model treniran na skupu podataka CIFAR-10, a za kraj ćemo se posvetiti opisanom problemu otkrivanja izvandistribucijskih podataka koji se mogu naći na ulazu tog istog modela.

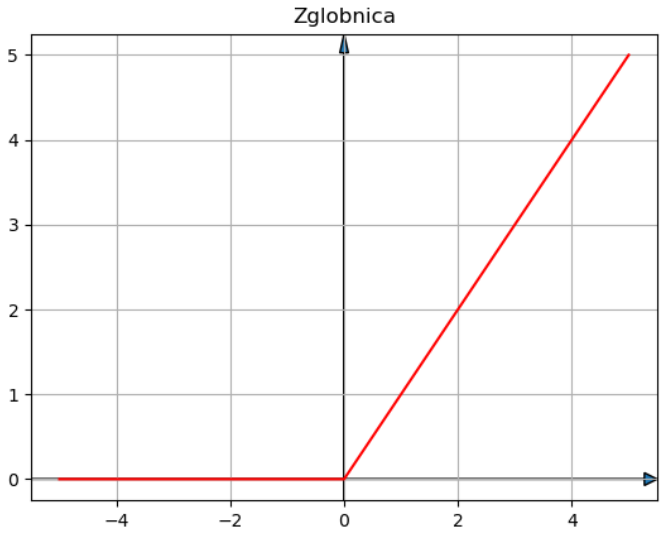
2. Općenito o konvolucijskim neuronskim mrežama

Konvolucijske neuronske mreže jedan su od najpopularnijih načina modeliranja umjetnih neuronskih mreža. Danas se najčešće koriste za analizu i predviđanja sadržaja slika i ostalih vizualnih podataka. Razlog zašto se ovaj tip neuronskih mreža koristi obradu kod obrade slika leži u njihovoj invarijantnosti na translaciju određenih značajki na slici koju obrađujemo. Naime, konvolucijske neuronske mreže uspješno analiziraju određene podatke traženjem pojedinih bitnih značajki, dok njihov položaj na predanim slikama nije ključan za ispravno prepoznavanje i analizu. Naime, korištenjem običnih umjetnih neuronskih mreža koje se sastoje jedino od potpuno povezanih slojeva ne bismo mogli postići jednake rezultate, a i broj težina koje bi morali pamtiti postao bi značajno veći što bi moglo stvoriti i probleme u pohrani istih u memoriji računala.

Osim slojeva neuronske mreže, koje ćemo kasnije detaljnije obraditi, vrlo bitan dio svih neuronskih mreža je i aktivacijska (prijenosna) funkcija koju nalazimo na izlazu iz svakog neurona. Kroz povijest su u ovu svrhu bile korištene razne matematičke funkcije, kao npr. ADALINE, TLU, sigmoidalna funkcija i sl. U zadnjih nekoliko godina najboljom se pokazala ReLU funkcija (zglobnica) koja je stoga sada i najkorištenija. Ova aktivacijska funkcija prikazana je na slici 2.1.

Konvolucijske neuronske mreže najčešće se sastoje od tri vrste slojeva koji su uglavnom poredani upravo u sljedećem redoslijedu:

- konvolucijski sloj
- sloj sažimanja
- potpuno povezani sloj



Slika 2.1: Aktivacijska funkcija zglobnica

2.1. Konvolucijski sloj

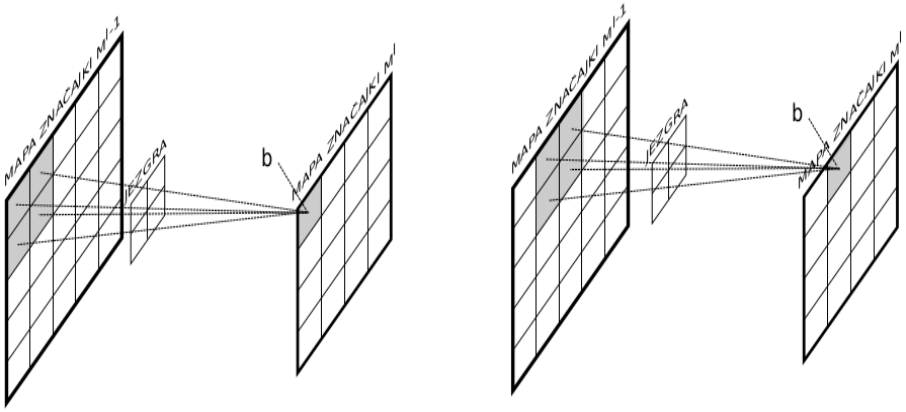
Prvi sloj koji nalazimo u konvolucijskim neuronskim mrežama upravo je konvolucijski sloj (*engl. convolutional layer*). Svrha ovog tipa slojeva ekstrakcija je pojedinih značajki ulaznih podataka, što je upravo i jedan od razloga njegove lokacija na samom početku mreže. Kako je ovo ključan sloj konvolucijskih mreža, potrebno je podrobnije opisati funkcionalnost i način rada ovog sloja prilikom obrade ulaznih podataka.

Najbitniji pojmovi koje moramo spomenuti kao glavne sastavnice ove vrste sloja svakako su filtri (jezgre) i korak.

Filtar (jezgru) možemo shvatiti kao matricu težina (*engl. weights*) čije vrijednosti učimo prilikom treniranja modela. Njegove dimenzije, točnije visina i širina, uglavnom su relativno male, dok je dubina određena dubinom ulaza (npr. za RGB slike dubina je 3). Kada tako definiran filtar preklopimo s dijelom ulaza istih dimenzija, rezultat konvolucije računa se kao suma umnožaka vrijednosti na korespondentnim pozicijama filtra i ulaza. Svakoj od dobivenih vrijednosti potrebno je još dodati prag (*engl. bias*) koji je inicijalno postavljen na nulu, no s vremenom ga također učimo.

Korak (*engl. stride*) definiramo kao pomak za koji pomičemo filtar duž cijele širine i visine ulazne matrice podataka. Nakon svakog pomaka obavi se opisani postupak konvolucije te se rezultantna vrijednost spremi u izlaznu aktivacijsku mapu (mapu

značajki). Broj aktivacijskih mapa jednak je broju različitih filtara koje koristimo u pojedinom konvolucijskom sloju. Izlaz takvog sloja ima dubinu jednaku broju aktivacijskih mapa, dok širina i visina ovise o veličini filtra i koraku, ali se najčešće koristi nadopunjavanje nulama kako bismo zadržali ulaznu visinu i širinu podataka.



Slika 2.2: Prikaz postupka konvolucije u kojem jezgra prolazi kroz ulaznu aktivacijsku mapu i stvara izlaznu [1]

Na slici 2.2 možemo jednostavnije uočiti opisani postupak koji se događa u konvolucijskom sloju. Pri tome redom s lijeva na desno slijede: ulaz (ulazna mapa značajki), jezgra te izlazna mapa značajki. Jezgra dimenzija $2 \times 2 \times 1$ klizi preko ulaza te ažurira izlaznu mapu značajki koja se tada prosljeđuje dalje kroz mrežu. Korak za koji pomičemo jezgru ovdje iznosi 1, a kada filtar prijeđe cijeli redak tada se spušta u redak niže gdje se cijeli postupak ponavlja.

2.2. Sloj sažimanja

Sljedeći sloj koji nalazimo u konvolucijskim mrežama je sloj sažimanja (*engl. pooling layer*). On uglavnom slijedi konvolucijski sloj (ili više njih u nizu), a obavlja ekstrakciju jednog podatka iz određenog podskupa ulaza čime smanjujemo dimenzionalnost podataka koji se kreću idućim slojevima konvolucijske mreže. Bitne sastavnice sloja sažimanja također su jezgra i korak pomaka.

Jezgra je definirana slično kao i ona konvolucijskog sloja, ali svoju funkcionalnost obavlja značajno drugačije. Kada jezgru poklopimo s podskupom ulaza odgovarajućih dimenzija, tada kao rezultat izlazne aktivacijske mape dobivamo jedan broj koji je

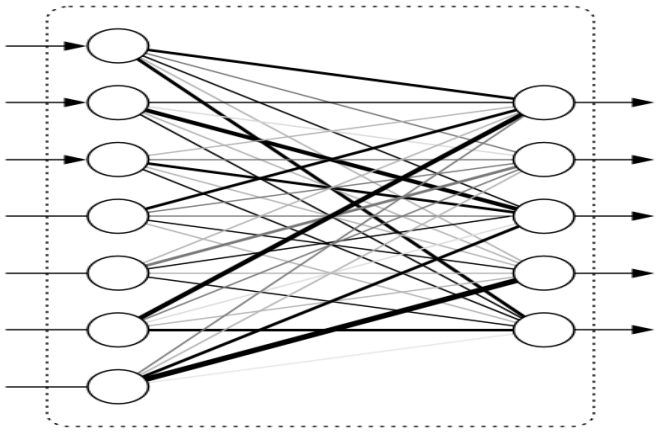
[1] Slika je preuzeta iz prezentacije s kolegija Duboko učenje, `www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf`

danas najčešće maksimalna vrijednost tih podataka (*engl. max-pooling*), dok se nekada koristila i prosječna vrijednost (*engl. average-pooling*). Izvlačenjem maksimalne vrijednosti iz podskupa ulaza zapravo dobivamo informaciju o najistaknutijom karakteristici, a ujedno i uklanjamo potencijalni šum u podacima.

Korak definiramo kao pomak za koji se krećemo širinom i visinom cijelih ulaznih aktivacijskih mapa. Ovaj korak se uglavnom razlikuje od onog iznosa 1 korištenog kod konvolucijskog sloja jer nam je ipak cilj i smanjiti dimenzije podataka koje ćemo propagirati kroz ostatak mreže.

2.3. Potpuno povezani sloj

Uglavnom posljednji slojevi konvolucijskih mreža su potpuno povezani slojevi (*fully-connected layer*). Ova vrsta sloja sastoji se od proizvoljno odabranog broja neurona u kojima se obavlja operacija nad ulaznim podacima, a izlaz direktno određuju težine i pragovi pojedinih neurona. Svaki neuron prima sve vrijednosti ulaza, tada ih množi sa svojim naučenim težinama te u konačnici dodaje također naučenu vrijednost praga. Specifičnost potpuno povezanih slojeva je, kao što im i samo ime govori, to da je svaki izlaz direktno povezan sa svakim neuronom prethodnog sloja, što i vidimo na slici 2.3. Kroz niz slojeva ovog tipa dolazimo i do posljednjeg sloja neurona čiji broj nužno mora odgovarati broju klasifikacijskih razreda kojima pripadaju podaci na kojima smo cijelu mrežu i trenirali.



Slika 2.3: Prikaz potpunog povezanog sloja - svaki neuron s desne strane povezan je sa svakim iz prethodnog sloja [2]

[2]Slika je preuzeta s lokacije `http://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/`

3. Izvedba neuronske mreže i rezultati

U ovom poglavlju bit će opisana arhitektura korištene konvolucijske neuronske mreže, kao i sam postupak treniranja te testiranja naučenog modela prilagođenog za detekciju slika iz skupa podataka/slika CIFAR-10. Dodatno će biti priloženi i sami kodovi te rezultati dobiveni prilikom treniranja i testiranja modela.

3.1. Definicija modela

Arhitektura mreže koju ćemo koristiti preuzeta je s kolegija *Duboko učenje*, točnije s druge laboratorijske vježbe navedenog kolegija [3]. Slojevi od kojih se korištena konvolucijska neuronska mreža sastoji redom su:

- konvolucijski sloj sa 16 mapa značajki i filtrom veličine $5 \times 5 \times 3$
- sloj sažimanja (*max pooling*) s jezgrom veličine 3×3 te pomakom iznosa 2
- konvolucijski sloj sa 32 mape značajki i filtrom veličine $5 \times 5 \times 16$
- sloj sažimanja (*max pooling*) s jezgrom veličine 3×3 te pomakom iznosa 2
- potpuno povezani sloj s 256 neurona
- potpuno povezani sloj sa 128 neurona

Također je važno napomenuti kako se na izlazu iz konvolucijskog i potpuno povezanog sloja kao aktivacijska funkcija koristi zglobnica opisana ranije u ovom radu.

```
1  init = tf.contrib.layers.xavier_initializer()
2
3  conv1_w = tf.get_variable("conv1_w", shape = [5, 5, 3, 16],
     initializer = init)
4  conv2_w = tf.get_variable("conv2_w", shape = [5, 5, 16, 32],
     initializer = init)
5  fc1_w = tf.get_variable("fc1_w", shape = [8*8*32, 256],
     initializer = init)
```

```
6   fc2_w = tf.get_variable("fc2_w", shape = [256, 128], initializer =
        init)
7   logits_w = tf.get_variable("logits_w", shape = [128, 10],
      initializer = init)
8
9   conv1_b = tf.Variable(tf.zeros([16]))
10  conv2_b = tf.Variable(tf.zeros([32]))
11  fc1_b = tf.Variable(tf.zeros([256]))
12  fc2_b = tf.Variable(tf.zeros([128]))
13  logits_b = tf.Variable(tf.zeros([10]))
14
15  def model():
16    input_layer = tf.reshape(X, [-1, 32, 32, 3])
17
18    conv1 = tf.nn.conv2d(input_layer, conv1_w, strides = [1, 1, 1,
      1], padding = "SAME")
19    conv1 = tf.nn.relu(conv1 + conv1_b)
20
21    pool1 = tf.nn.max_pool(conv1, ksize = [1, 3, 3, 1], strides =
      [1, 2, 2, 1], padding = "SAME")
22
23    conv2 = tf.nn.conv2d(pool1, conv2_w, strides = [1, 1, 1, 1],
      padding = "SAME")
24    conv2 = tf.nn.relu(conv2 + conv2_b)
25
26    pool2 = tf.nn.max_pool(conv2, ksize = [1, 3, 3, 1], strides =
      [1, 2, 2, 1], padding = "SAME")
27
28    pool2_flat = tf.reshape(pool2, [-1, 8*8*32])
29    fc1 = tf.nn.relu( tf.matmul(pool2_flat, fc1_w) + fc1_b )
30    fc2 = tf.nn.relu( tf.matmul(fc1, fc2_w) + fc2_b )
31
32    logits = tf.matmul(fc2, logits_w) + logits_b
33    return logits
34
```

Isječak koda 3.1: Definicija modela korištenog za skup CIFAR-10. U isječku su prikazane inicijalizacije težina i pragova svih slojeva konvolucijske mreže te njihov redoslijed u samoj mreži

U prvih 14 linija prikazanog koda možemo vidjeti definiciju i inicijalizaciju težina te pragova svakog od pojedinig slojeva korištenih u konvolucijskoj mreži. Kod inicijalizacije težina konvolucijskih slojeva prve tri komponente liste `shape` predstavljaju

dimenzije filtra dok je zadnja komponenta broj filtara. Za potpuno povezane slojeve prva komponenta iznosi broj ulaza, a zadnja broj izlaza, tj. broj neurona u tom sloju. Zadnji sloj obavezno mora imati 10 neurona/izlaza kako bi predvidio jednu od 10 klasa u skupu CIFAR-10 kojoj ulazni podatak/slika pripada. Svakom od slojeva potrebno je definirati i pragove koje za početak postavljamo na nulu.

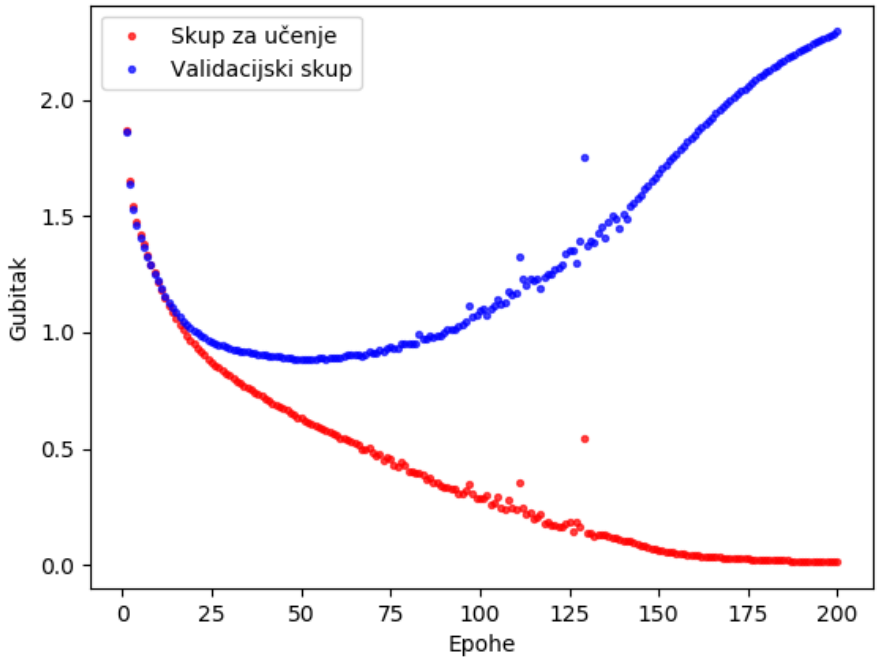
Od reda pod brojem 15 započinje definicija potpune arhitekture korištene mreže te povezanost svih slojeva. Kako je ova arhitektura već prethodno opisana ovdje nećemo davati dodatna pojašnjenja te funkcija `model` služi za demonstraciju stvaranja specifičnog modela korištenjem programskog okvira TensorFlow.

3.2. Treniranje modela

Nakon što smo definirali arhitekturu neuronske mreže, sljedeći korak je treniranje iste kako bismo dobili odgovarajuće težine i pragove modela. Ranije definiran model mreže treniran je na 50000 slika iz skupa CIFAR-10. Težine i pragovi ažurirani su nakon svakog prolaska kroz podskupove podataka (*engl. batch*), a pri tome je korišten postupak gradijentnog spusta sa stopom učenja $5 \cdot 10^{-3}$. Cilj ovog postupka je minimizirati gubitak definiran unakrsnom entropijom predviđenih i stvarnih klasa ulaznih podataka. Mreža je trenirana kroz 200 epoha s veličinom podskupova podataka (*batcheva*) od 200. Ovdje je bitno i napomenuti da epohu definiramo kao prolaz kroz cijeli skup podataka za učenje.

Postupak treniranja modela najbolje je pratiti kroz informacije o prosječnom gubitku i točnosti na skupu za treniranje te validacijskom skupu, a upravo to možemo vidjeti na slikama 3.1 te 3.2.

Na slici 3.1 možemo pratiti promjenu iznosa prosječnog gubitka na skupu na kojem treniramo te ne skupu koji nam koristi za provjeru dosad naučenog modela (validacijski skup). Glavna stvar koju možemo uočiti je kako kroz epohe gubitak na skupu za treniranje konstantno pada, dok je na validacijskom skupu situacija nešto drugačija - gubitak do jednog trenutka pada, ali onda počinje rasti. Do toga dolazi upravo zbog činjenice da mreža svakim prolaskom kroz cijeli skup podataka na kojima uči postaje sve više i više naučena na te konkretne podatke pa je gubitak sve manji. Za razliku od toga, gubitak na validacijskom skupu je sve veći baš zbog te prenaučenosti na skupu za treniranje te sve manjoj mogućnosti generalizacije na novim podacima.



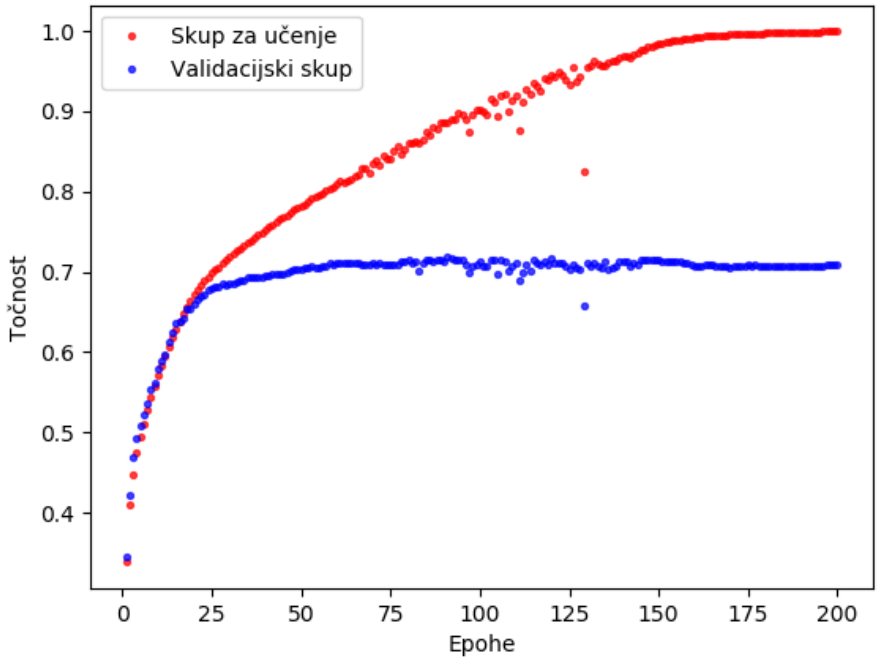
Slika 3.1: Gubitak prilikom treniranja modela

Slične efekte vidimo i na slici 3.2, smo što ovdje promatramo točnost naučenog modela na oba skupa. Model na skupu za učenje postaje sve bolji i bolji pa zbog toga točnost stalno raste, tj. model gotovo da i ne griješi na tom skupu podataka/slika. No, na validacijskom skupu situacija je ponešto drugačija, model postiže točnost oko 70% te uz manja variranja ostaje pri toj vrijednosti. Razlozi ovakvog ponašanja jednaki su ranije navedenim razlozima za pad/rast prosječnog gubitka.

Zanimljivost koju možemo uočiti usporeduću grafove na slikama 3.1 i 3.2 je ta da iako gubitak na validacijskom skupu prema kraju sve više raste, točnost na istom skupu podataka je i dalje ista (uz manja variranja). Kako se model kroz epohe sve više prilagođava skupu za učenje i na njemu postiže sve veću točnost i manji gubitak, tako je vrlo moguće da se u isto vrijeme mijenja i distribucija klasifikacijskih vrijednosti na validacijskom skupu podataka. Razlozi takvog ponašanja mogu biti sljedeći:

- neki već ranije netočno klasificirani podaci i dalje su netočno klasificirani, ali sada s većom klasifikacijskom vjerojatnosću (npr. $0.7 \rightarrow 0.9$)
- neki prethodno ispravno klasificirani podaci i dalje su točno klasificirani, no sada im je klasifikacijska vjerojatnost nešto manja (npr. $0.9 \rightarrow 0.8$)

U oba navedena slučaja točnost modela i dalje ostaje ista, dok sam gubitak raste, što je i vidljivo u našem slučaju prikazanom na spomenutim grafovima.



Slika 3.2: Točnost prilikom treniranja modela

3.3. Rezultati testiranja modela

Posljednji korak nakon definicije arhitekture i učenja modela je testiranje istog te provjera njegovih klasifikacijskih performansi. U nastavku su prikazani rezultati koji postiže ranije navedeni model na skupu CIFAR-10:

- točnost - 71.53%
- preciznost - 71.95%
- odziv - 71.53%

Navedene vrijednosti su relativno slabe za današnje kriterije kada postaje mnogo kompleksnije i naprednije arhitekture neuronskim mreža, no za ovako jednostavan model rezultati su zadovoljavajući.

Kako sve ove vrijednosti zapravo proizlaze iz matrice zabune (*engl. confusion matrix*), zanimljivo bi bilo promotriti i samu matricu kako bismo dobili bolji uvod u karakteristike i najčešće pogreške našeg modela.

Matrica zabune										
Klase	0	1	2	3	4	5	6	7	8	9
0	**729**	24	57	22	43	8	14	11	51	41
1	15	**826**	17	12	10	4	16	5	21	74
2	59	10	**607**	47	132	40	51	39	7	8
3	15	7	76	**552**	97	118	74	37	8	16
4	9	4	77	47	**752**	19	30	53	6	3
5	14	5	67	206	81	**528**	25	69	3	2
6	4	7	42	67	50	19	**798**	5	4	4
7	4	8	34	44	72	48	8	**775**	0	7
8	56	40	20	15	18	8	6	4	**799**	34
9	32	92	10	18	11	5	6	19	20	**787**

Tablica 3.1: Matrica zabune

Tablica 3.1 predstavlja upravo matricu zabune gdje oznake razreda u tablici označene brojevima odgovaraju pojedinim objektima prikazanim na slikama skupa podataka: 0 - zrakoplov, 1 - automobil, 2 - ptica, 3 - mačka, 4 - jelen, 5 - pas, 6 - žaba, 7 - konj, 8 - brod, 9 - kamion. Pri tome redci predstavljaju točne oznake razreda, dok su predikcije modela prikazane u stupcima. Sada iz ovih podataka možemo uočiti neke zanimljivosti:

- psi su čak 206 puta bili klasificirani kao mačke
- ptice su 132 puta bile svrstane u razred jelena
- razred konja ističe se kao jedini s brojem 0 u svojem retku, što znači da nikada nije bio netočno svrstan u razred brodova
- slike automobila po svojim značajkama se dosta razlikuju od svih ostalih razreda pa su daleko najviše puta bili točno klasificirani

4. Pokazatelji klasifikacijskih performansi

Kako bismo mogli procjenjivati i određivati performanse naučenih modela potrebno je bilo standardizirati određene pokazatelje koji bi se tada uvijek koristili u ovu svrhu. Neki od tih pokazatelja klasifikacijskih performansi *engl. evaluation metrics* trebat ćemo za analizu i detekciju izvandistribucijskih podataka na ulazu dubokog modela pa ih je sada potrebno pojasniti.

Za definiciju pojedinih pokazatelja na početku je potrebno definirati i neke pojmove koje će se koristiti u njihovoj definiciji. Vrijednosti koje će nama biti potrebne su: točni pozitivi (*engl. true positives TP*), lažni pozitivi (*engl. false positives FP*), točni negativi (*engl. true negatives TN*) te lažni negativi (*engl. false negatives FN*).

Točni pozitivi (*TP*) definirani su kao broj ispravno klasificiranih podataka u točan razred kojem zapravo pripadaju. U matrici zabune broj točnih pozitiva za svaki razred odgovara vrijednostima na dijagonali same matrice.

Lažni pozitivi (*FP*) definirani su kao broj podataka koji su svrstani u određeni razred kojem zapravo ne pripadaju. U matrici zabune broj lažnih pozitiva za svaki razred odgovara sumi vrijednosti u odgovarajućem stupcu umanjenoj za vrijednost na dijagonali (broj točnih pozitiva).

Točni negativi (*TN*) definirani su kao broj podataka kod kojih je ispravno detektirano da ne pripadaju određenom razredu. U matrici zabune broj točnih negativa za svaki razred odgovara sumi vrijednosti na dijagonali umanjenoj za konkretnu vrijednost na dijagonali koja odgovara razredu kojeg trenutno promatramo (broj točnih pozitiva).

Lažni negativi (*FN*) definirani su kao broj podataka koji bi u stvarnosti pripadaju određenom razredu, ali model ih nije predvidio kao pripradnike tog istog razreda. U matrici zabune broj točnih negativa za svaki razred odgovara sumi vrijednosti u odgovarajućem retku umanjenoj za vrijednost na dijagonali (broj točnih pozitiva).

Nakon što smo definirali sve pojmove, zanimljivo bi bilo analizirati te vrijednosti na konkretnoj matrici zabune prikazanoj u tablici 3.1. Promotrimo situaciju za razred pod oznakom 0 (razred zrakoplova):

- *TP* = 729
- *FP* = 208
- *TN* = 6424
- *FN* = 271

4.1. Točnost

Točnost (*engl. accuracy*) jedan je od najbitnijih pokazatelja kvalitete modela jer nam upravo on izravno govori o postotku testnih primjera koji su ispravno klasificirani. Koristeći prethodno definirane pojmove i oznake, točnost definiramo prema sljedećem izrazu:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Kao što je i vidljivo iz samog izraza, točnost nam određuje postotak ispravno klasificiranih podataka u odnosu na cijeli skup testnih podataka (predstavljen sumom u nazivniku izraza).

4.2. Preciznost

Preciznost (*engl. precision*) drugi je bitan pokazatelj klasifikacijskih performansi naučenog modela, a matematička definicija prikazana je sljedećim izrazom:

$$\frac{TP}{TP + FP}$$

Iz samog izraza možemo uočiti što točno predstavlja podatak o precoznosti, a to je udio točno klasificiranih razreda u svim situacijama kada je određeni podatak bio svrstan kao pripradnih pojedinog razreda.

4.3. Odziv

Odziv (*engl. recall*) još je jedan od bitnih pokazatelja kvalitete modela, a definiran je izrazom:

$$\frac{TP}{TP + FN}$$

Izraz i definicija odziva nam direktno daje informaciju o tome koliko je puta podatak bio točno klasificiran u određeni razred u odnosu na ukupan broj pojava tog pojedinog razreda.

5. Otkrivanje izvandistribucijskih podataka

Kao što je to već ranije navedeno, otkrivanje izvandistribucijskih podataka analizom izlaza korištenog modela vrlo je bitna sastavnica klasifikacije podataka te je samim time jedan od najvećih problema odrediti kada je naš model netočno predvidio klasu podatka koji se trenutno analizira. Ova problematika relativno je nova, a u nastavku ćemo analizirati razne metode određivanja izvandistribucijskih podataka spomenute u radu [5], poput ROC i PR krivulja te površina ispod istih (AUROC i AUPR). Prilikom sljedećih analiza pozitivnim razredom smatramo originalne podatke (unutar distribucije), dok je negativan razred predstavljen izvandistribucijskim podacima, ukoliko nije navedeno drugačije. Prilikom crtanja oba tipa krivulja iterativno se pomiče granica (*engl. threshold*) koja svaki put uzima sve više podataka te tada računa tražene vrijednosti samo na podacima koji su zahvaćeni granicom.

(a) **(b)** **(c)**

Slika 5.1: Slike korištene za testiranje

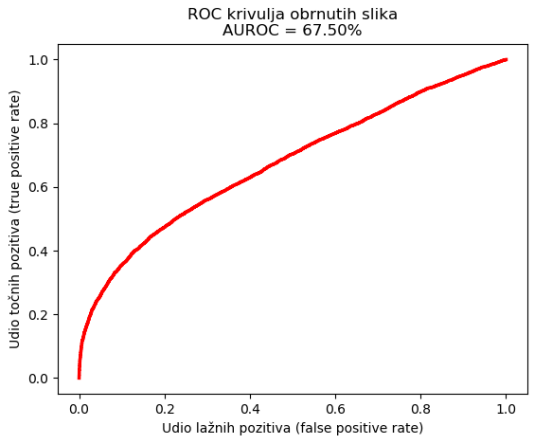
Slika 5.1 prikazuje slike korištene za potrebe ovih testiranja. Kao podaci unutar distribucije na kojoj smo trenirali model koriste se originalne slike iz CIFAR-10 testnog skupa, a primjer takve slike vidimo na slici zrakopolova pod a). Slike pod b) i c) predstavljaju podatke koji će biti korišteni kao izvandistribucijski. Slika b) nastala je rotacijom originalne slike za 180°, dok je slika pod c) nastala slučajnim generiranjem

Gaussovim šumom očekivanja 0 te standardne devijacije 0.5.

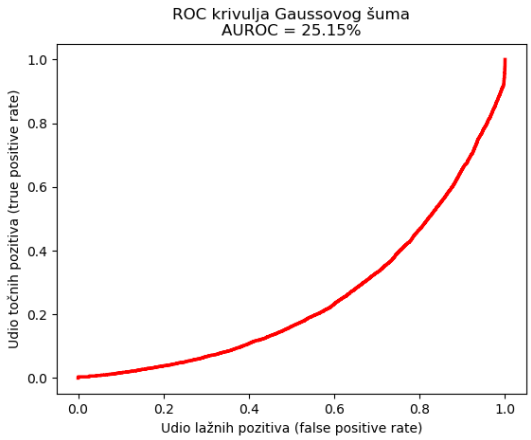
5.1. AUROC

ROC krivulja ili krivulja odnosa udjela lažnih pozitiva i udjela točnih pozitiva (*engl. Receiver Operating Characteristic curve*), tj. površina ispod te krivulje (*engl. AUROC - Area under ROC curve*) koristi se kao jedna od mjera za detekciju primjeraka podataka koji su u ili van distribucije onih podataka na kojima smo trenirali model te za koju imamo veću šansu za ispravnu klasifikaciju. Nadalje, AUROC ili površina ispod ROC krivulje može se protumačiti kao vjerojatnost da će pozitivan podatak (podatak unutar distribucije) imati veću klasifikacijsku vjerojatnost (maksimalnu softmax vrijednost) nego li je to slučaj za negativne podatke (podatke van distribucije). Navedenu detekcijsku vrijednost možemo svrstati u više kategorija u ovisnosti o površini ispod ROC krivulje [5]:

- 90-100% - izvrsna
- 80-90% - vrlo dobra
- 70-80% - dobra
- 60-70% - slaba
- < 60% - nezadovoljavajuća



(a) ROC krivulja za obrnute slike



(b) ROC krivulja za slike kreirane Gaussovim šumom

Slika 5.2: ROC krivulje

Na slici 5.2 a) prikazana je ROC krivulja za slučaj originalnih podataka rotiranih za 180° oko horizontalne osi. Iznad prikaza krivulje prikazana je i izračunana površina

ispod iste, a ta površina u postotcima iznosi 67.50%. Prema ovoj informaciji model možemo svrstati u slabe modele po pitanju odvajanja originalnih podataka i onih izvandistribucijskih.

Slika 5.2 b) predstavlja ROC krivulju za slučajno generirane slike korištenjem Gaussovog šuma. Analiziramo ove krivulje uočavamo značajno manju površinu ispod krivulje (25.15%) te samim time i vrlo lošu sposobnost detekcije izvandistribucijskih podataka (Gaussovog šuma).

Kako bismo si bolje pojasnili dobivene rezultate i AUROC površine, korisno bi bilo analizirati prosječne klasifikacijske vjerojatnosti za sve korištene ulaze:

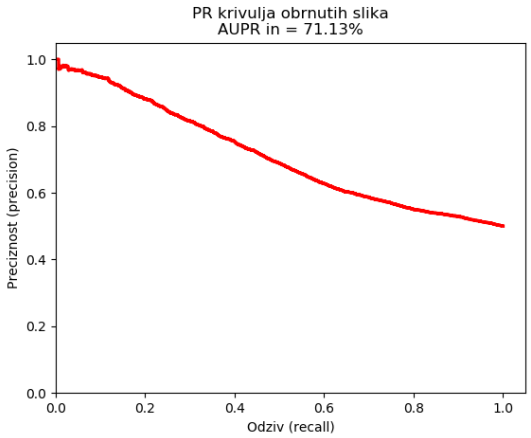
- originalne slike - 83.19%
- obrnute slike - 72.69%
- Gaussov šum - 94.94%

Pogledom na ove rezultate, prethodne površine ispod krivulja mogu nam biti puno jasnije. Razlika između prosječnih klasifikacijskih vjerojatnosti za originale i obrnute slike je oko 10% u korist originalnih podataka pa ipak imamo određenu šansu za prepoznavanje podataka koji su izvan naučene distribucije. Kod usporedbe originalnih slika i Gaussovog šuma, klasifikacijske vjerojatnosti su u prosjeku 11% veće pa ispada da naš model ima manju vjerojatnost za prepoznavanje izvandistribucijskih podataka nego li bi to bio slučaj da koristimo slučajni klasifikator.

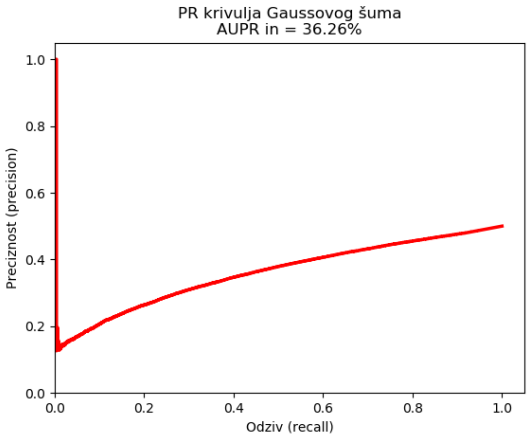
5.2. AUPR

PR krivulja ili krivulja odnosa odziva i preciznosti modela (engl. Precision-Recall curve), kao i površina ispod iste (*engl. AUPR - Area under Precision-Recall curve*) također se koristi kao jedna od mjera da otkrivanje izvandistribucijskih podataka na ulazu modela. AUPR se često pokazuje informativnijim od AUROC-a, a ujedno je i bolje prilagođen za slučajeve kada se razdiobe klasifikacijskih vjerojatnosti pozitivnog i negativnog razreda značajnije razlikuju. AUPR nam zapravo govori o prosječnoj preciznosti modela te je još jedan od dobrih pokazatelja kvalitete samog naučenog modela. Korištenje i prikaz ovog tipa krivulja prikazat ćemo i analizirati na nekoliko primjera, kao što je to bio slučaj i za AUROC.

Prva situacija koju obrađujemo prikazana je na slici 5.3 za dvije prethodne opisane mogućnosti izvandistribucijskih podataka. Kao što vidimo iz naslova ovih



(a) PR krivulja za obrnute slike

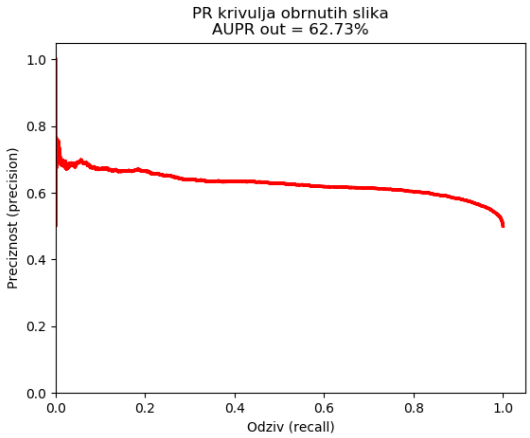


(b) PR krivulja za slike kreirane Gaussovim šumom

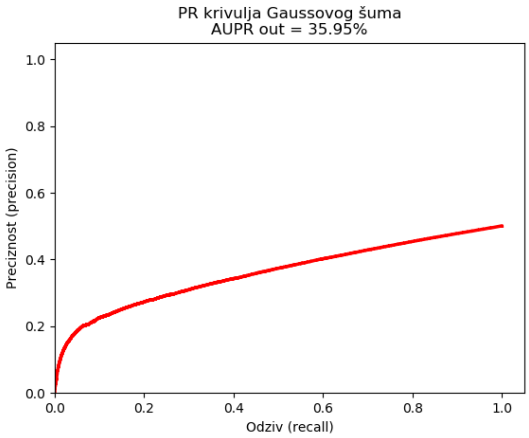
Slika 5.3: PR In krivulje

grafova, ovdje se obrađuje situaciju AUPR In, tj. situacija kada su originalne slike korištene kao pozitivi, dok su kao negativi kosrišteni izvandistribucijski podaci.
Na slici 5.3 a) prikazana je PR krivulja nastala tako da se originalni podaci koriste kao pozitivi, a obrnute slike su korištene kao negativi. Uočavamo da je prosječna preciznost ovdje 71.13%, što znači da naš model dovoljno dobro i precizno može odvojiti izvandistribucijske podatke od onih unutar naučene distribucije.
Graf na slici 5.3 b) napravljen je u slučaju kada su kao vandistribucijski podaci korištene slike generirane slučajnim Gaussovim šumom. Kao što smo to primijetili i ranije, i ovdje model pokazuje svoje nedostatke. Na samom početku je preciznost vrlo dobra, no kako dohvaćamo sve više izvandistribucijskih slika koje imaju visoku klasifikacijsku vjerojatnost (u prosjeku i višu od one originalnih podataka) tako preciznost naglo pada i u prosjeku kasnije samo malo raste.

Sljedeća situacija koju ćemo promotriti prikazana je na slici 5.4. Ovdje se kao pozitivi koriste izvandistribucijski podaci, dok su originalne slike korištene kao negativi. No u ovoj situaciji vrlo je bitno napomenuti kako se sada u cijelom procesu obrade podataka i kreiranja grafova koriste negativne vrijednosti klasifikacijskih vjerojatnosti. Površine ispod krivulja pod a) i b) nešto su manje od onih ispod PR In krivulja, te je i dalje primjetan problem kod rada s Gaussovim šumom već prethodno opisan u nekoliko prilika.



(a) PR krivulja za obrnute slike



(b) PR krivulja za slike kreirane Gaussovim šumom

Slika 5.4: PR Out krivulje

5.3. Programska izvedba

Kako bismo upotpunili ovo poglavlje u kojem su opisane ROC i PR krivulje, potrebno je ukratko prikazati i isječak koda koji se koristi za njihovo iscrtavanje i računanje pojedinih pokazatelja klasifikacijskih performansi koje su nam u tom trenutku potrebne. Ovaj postupak prikazan je u sljedećem isječku koda, pri čemu su korištene biblioteka NumPy i Matplotlib.

```
def draw_ROC(data_type, P, N):
    assert(P.shape == N.shape)

    total = P.shape[0]
    P = [[np.max(x), True] for x in P]
    N = [[np.max(x), False] for x in N]

    union = np.append(P, N, axis=0)
    union = sorted(union, key=lambda x : -x[0])

    t1 =[]; t2 = []
    for i in range(1, len(union)+1):
    used = union[:i]

    tp = sum(x[1] for x in used)
    fp = len(used) - tp

    t1.append(fp / total)
    t2.append(tp / total)

```

```
    area = 0
    for i in range(len(t1) - 1):
    area += np.trapz(np.array(t2[i:i+2]), dx=abs(t1[i] - t1[i+1]))

    plt.title("ROC krivulja {0}\nAUROC = {1:.2f}%".format(data_type,
    area * 100))
    plt.xlabel("Udio laznih pozitiva (false positive rate)")
    plt.ylabel("Udio tocnih pozitiva (true positive rate)")
    plt.plot(t1, t2, 'r-', linewidth=2.5)
    plt.show()

```

Isječak koda 5.1: Postupak crtanja ROC krivulje i računanja površine ispod iste

Funkcija `draw_ROC` kao argumente prima tip podataka (korisno samo u svrhu ispisa) te matrice softmax vrijednosti pozitivnog (`P`) i negativnog (`N`) razreda. Za početak je potrebno odrediti maksimalnu vrijednost softmaxa za svaki podatak koji je bio doveden na ulaz modela, a isto tako obavljamo i sortiranje podataka prema iznosu klasifikacijske vjerojatnosti. Nakon inicijalnih priprema podataka iterativno mijenjamo granicu te postupno zahvaćamo sve više podataka u skup nad kojim trenutno radimo. Tada za svaki podskup podataka `used` računamo broj lažnih pozitiva `fp` te broj točnih pozitiva `tp` kako bismo na kraju dobili njihove udjele u ukupnom broju podataka. Nakon što je ovaj postupak proveden površinu ispod dobivene ROC krivulje računamo kao sumu trapezoidnih aproksimacija površina između susjednih točaka krivulje te zatim sve to prikažemo na grafu. Da smo htjeli prikazati PR krivulju jedina razlika bi bila u tome što bismo tada za svaki podskup podataka računali preciznost i odziv umjesto udjela točnih i lažnih pozitiva.

6. Zaključak

U ovom radu dali smo kratak pregled i opis funkcionalnosti konvolucijskih neuronskih mreža te prikaz aktualnih mogućnosti u području otkrivanjea izvandistribucijskih podataka analizom izlaza istreniranog dubokog modela. Početni korak u realizaciji ovog rada bila je implementacija modela kojeg smo potom trenirali i testirali na CIFAR-10 skupu podataka uz točnost od 71.53%. Nakon što je model bio spreman, krenuli smo na sljedeći korak, a to je bila detekcija vandistribucijskih podataka koje smo simulirali obrnutim originalnim slikama te kreiranjem potpuno slučajnih slika korištenjem Gaussovog šuma. U trenutku kada smo imali definiran model i vandistribucijske podatke, analizirali smo moguće tehnike (npr. AUROC i AUPR) otkrivanja takvih podataka temeljem izlaza dubokog modela. U ovom dijelu su postignuti dosta dobri rezultati, unatoč problemima na koje smo naišli analizom slika Gaussovog šuma.

Za kraj bismo još mogli navesti mogućnosti poboljšanja dobivenih rezultata. Postoji nekoliko mogućih poboljšanja kojima bismo dobili još bolje rezultate, a to je prvenstveno korištenje složenijeg dubokog modela koji bi tada bolje generalizirao podatke te imao bolju mogućnost distinkcije izvandistrubucijskih podataka od onih na kojima smo model trenirali.

LITERATURA

[1] Cs231n convolutional neural networks for visual recognition. URL http://cs231n.github.io/convolutional-networks/.

[2] Enhancing the reliability of out-of-distribution image detection in neural networks | openreview. URL https://openreview.net/forum?id=H1VGkIxRZ.

[3] Stranice predmeta duboko učenje (fer). URL https://dlunizg.github.io/lab2/.

[4] Tutorials | tensorflow. URL https://www.tensorflow.org/tutorials/.

[5] Dan Hendrycks i Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. U *Proceedings of International Conference on Learning Representations*, 2017.

[6] Matthijs Hollemans. Convolutional neural networks on the iphone with vggnet. URL http://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/.

[7] Krešimir Kralj. Klasifikacija slika dubokim konvolucijskim modelima. URL www.zemris.fer.hr/~ssegvic/project/pubs/kralj17bs.pdf.

[8] Sinisa Segvic. Duboko učenje. URL http://www.zemris.fer.hr/~ssegvic/du/.

POPIS SLIKA

POPIS ISJEČAKA KODA

Otkrivanje izvandistribucijskih podataka analizom izlaza dubokog modela

Sažetak

Konvolucijske neuronske mreže danas su često korištene za klasifikacijsku ulaznih podataka u pojedine moguće razrede za koje je model prethodno istreniran. Neovisno o testnom podatku, model će uvijek s određenom sigurnošću predvidjeti pripradnost ulaza u jedan od klasifikacijskih razreda. Ključna problematika ovog pristupa je nemogućnost otkrivanja ulaznih podataka za koje će model najvjerojatnije pogriješiti. Upravo zbog toga potrebno je otkriti izvandistribucijske podatke dovedene na ulaz modela, a to činimo analizom njegovog izlaza. Kod takvih podataka nikad ne možemo sigurno tvrditi jesu li dijelom distribucije na kojoj je model treniran ili su vandistribucijski, no vjerojatnost detekcije izvandistribucijskih podataka moguće je odrediti na više načina, npr. korištenjem AUROC i AUPR površina.

Ključne riječi: duboki model, konvolucijske neuronske mreže, otkrivanje izvandistribucijskih podataka, AUROC, AUPR, skup podataka, treniranje, testiranje, validacija, CIFAR-10, TensorFlow

Discovery of Out-of-Distribution Data by Analyzing Predictions of a Deep Model

Abstract

Convolutional neural networks are nowadays often used for classifying input data into specific possible classes for which the model was previously trained. Whatever test data we provide to the model, it will always predict the result corresponding class with specific certainty. The key problem with this approach is an inability of determining the data for which the model will most likely make a wrong prediction. Specifically because of that, it is a necessity to determine out-of-distribution data provided as input to the model, and the way to do that is by analyzing its output. Despite the fact that we cannot claim if the test data belongs to the distribution on which the model was trained or it is out-of-distribution, the possibility of detecting out-of-distribution data can be determined with various methods, including using AUROC and AUPR areas.

Keywords: deep model, convolutional neural networks, discovery of out-of-distribution data, AUROC, AUPR, dataset, training, testing, validation, CIFAR-10, TensorFlow