Klasifikacija slika pasa

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Dušica Krstić, Nikola Živković
dusicamkrstic@gmail.com, nmzivkovic@gmail.com

5. juni 2019

Sažetak

U ovom radu je ilustrovan rad konvolutivne neuronske mreže koja slike pasa razvrstava u jednu od deset mogućih rasa.

Sadržaj

1 Uvod

Problem klasifikacije je problem učenja, gde se nepoznate instance razvrstavaju u jednu od unapred ponuđenih grupa[11]. Ove grupe se nazivaju klase ili kategorije. Učenje ovih kategorija se ostvaruje pomoću modela, koji se kasnije koristi za procenu odgovarajuće klase do sada neviđenih podataka[2]. Klasifikacija pripada problemu učenja koji se naziva nadgledano učenje. Nadgledano mašinsko učenje se karakteriše time da su za sve podatke poznate vrednosti ciljne promenljive.

U ovom radu je prikazano pravljenje modela koji klasifikuje slike pasa u jednu od deset klasa. Svaka klasa predstavlja jednu od deset izabranih rasa, a to su: pudle, nemački ovčari, rotvajleri, šnauceri, aljaski malamuti, biglovi, mađarske vižle, retriveri, čivave i italijanski hrtovi. Svaka rasa je izabrana zbog svojih specifičnih vizuelnih karakteristika. Kao inspiracija za odabir baš ovih rasa pasa je poslužio FCI standard.

Međunarodna kinološka federacija[9] (fr. *Fédération Cynologique Internationale*) je organizacija koja se bavi prepoznavanjem i priznavanjem različitih rasa pasa, kao i propisivanjem standarda za svaku rasu. Standard predstavlja detaljan opis idealnog primerka svake rase. Ovim standardnom je priznato 349 rasa.

Za analizu i klasifikaciju slika se koriste konvolutivne neuronske mreže, koje predstavljaju poseban tip neuronskih mreža. One se danas mogu naći u okviru programa poput Fejsbuka i Instagrama gde služe za označavanje na fotografijama ili čak u programima koji stoje iza automobila na samoupravljanje. Sve više se rasprostranjuju u oblastima poput zdravstva i bezbednosti[3].

Kôd korišćen u ovom projektu, skupove podataka, istreniran model i ostali prateći softver možete naći u literaturi kao i na sledećoj adresi:
https://github.com/dmkrstic/goodboiclassifier.

2 Korišćeni skup podataka

Prvobitno smo pokušali da treniramo sa ukupno 120000 slika u skupu podataka. Želeli smo da klasifikujemo preko 50 različitih rasa. To je bilo praktično nemoguće jer je jedna epoha trajala preko 40 minuta, a za bilo kakve rezultate bilo bi neophodno da se trenira najmanje 100 epoha i da se nakon toga mreža dalje prilagođava u skladu sa dobijenim rezultatima. Zbog ovoga smo se odlučili da klasifikujemo svega deset različitih rasa.

2.1 Prikljupljanje podataka

Deo prikupljenih podataka je preuzet sa sledećeg linka:
http://vision.stanford.edu/aditya86/ImageNetDogs.

Ostatak korišćenih slika predstavlja prvih 100 rezultata *Google Images* pretrage po nazivu svake rase. Za prikupljanje drugog dela instaci, korišćena je skripta preuzeta sa sledećeg linka:
https://github.com/hardikvasa/google-images-download.

Kompletan skup podataka korišćen u ovom radu, može se preuzeti sa sledećeg linka:
http://tiny.cc/goodboiclassifierdataset.

2.2 Pretprocesiranje

Sve prikupljene datoteke su organizovane u foldere po rasama. Nakon raspodele, sve slike su ručno pregledane da bismo se osigurali da se na svim slikama zaista nalaze

psi. Ovo je bilo neophodno zato što Google images ne garantuje da će se na svim slikama naći pas.

Sve slike su raspoređene u tri skupa: podatke za trening (70%), podatke za validaciju (15%) i podatke za testiranje (15%). Validacioni skup je korišćen kako bismo u toku samog treninga imali uvid u to kako se naš model ponašao nad podacima nad kojima još nije trenirao.

Celokupan skup podataka je zauzimao mnogo memorijskog prostora, stoga je bilo neophodno dodatno obraditi slike. Na slike je prvo primenjena kompresija (smanjenje veličine) sa određenim koeficijentom kvaliteta.

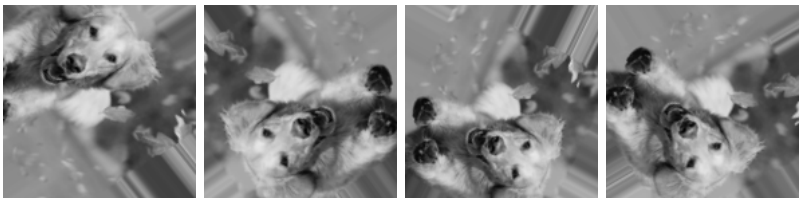
Zatim su u cilju poboljšanja i proširivanja podataka primenom transformacija na već postojeće napravljene "nove" slike[12]. Primenjene su nasumične rotacije između 0 i 359 stepeni, vertikalna i horizontalna translacija za nasumični faktor ne veći od 0.1 sa fokusom u centru slike i odbacivanjem sadržaja van ivica, kao i nasumična primena horizontalnih i vertikalnih refleksija. Iako se na dobijenoj slici nalazi isti pas, ta slika predstavlja novu instancu koju konvolutivna neuronska mreža još uvek nije videla. Prilikom poboljšanja i proširivanja nije za sve klase korišćen isti faktor uvećanja. Ovo je urađeno kako bismo izbegli da model lošije klasifikuje klasu koja ima manje podataka prilikom treninga.

Sve slike koje se prosleđuju konvolutivnoj neuronskoj mreži moraju biti iste rezolucije jer je arhitektura modela konstantna i može primati samo ulaze iste dimenzije. Zbog ovoga su sve slike smanjene na rezoluciju od 144 x 144.

Pošto su najvažnije razlike između rasa pasa anatomske, sve slike su prebačene iz RGB modela u nijanse sive boje.

Slika 1: Originalna slika i grayscale

Slika 2: Resize



Slika 3: Augmentation

Naš model sada sadrži 34840 slika u trening skupu, 5780 slika u test skupu i 7080 slika u validacionom skupu.

Celo pretprocesiranje je izvršeno pomoću python skripti koje se mogu naći na GitHub repozitorijumu projekta[6].

3 Poglavlje sa eksperimentalnim rezultatima

3.1 Tensorflow

Tensorflow[13] je open source biblioteka za numerička izračunavanja koja koristi grafove za tok podataka. Ova biblioteka ima veliku primenu za razvijanje aplikacija zasnovane na mašinskom učenju. Napravljena je 2015. godine od strane Google Brain tima. Može biti pokrenuta na raznim centralnim procesorskim jedinicama (CPU), grafičkim procesorskim jedinicama (GPU), mobilnim i embedded sistemima, kao i na hardveru koji je specijalizovan da se na njemu radi tenzorska matematika (en. Tensor Processing Unit). Za frontend se koriste programski jezici Python i C++, a od API-a Keras i Estimator API.

3.2 Keras

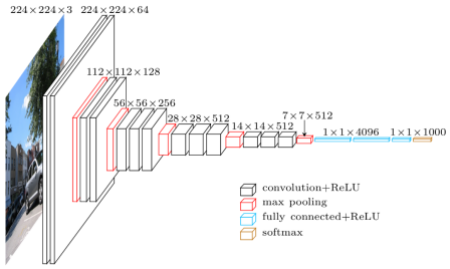
Zbog kompleksnosti tensorflow framework-a koristi se Keras API[10], koji je pristupačan korisnicima, lak za primenu, a veoma moćan u kombinaciji sa programskim jezikom Python, zbog čega smo se i opredelili za korišćenje ovog API-a.

Osnovna struktura u kerasu je model. Mi smo koristili sekvencijalni model, koji je jedan od dva osnovna tipa modela. Sekvencijalni model je skup linearno naslaganih slojeva.

3.3 VGG arhitektura

VGG arhitekturu[1] su 2014. godine predložili profesori K. Simonyan i A. Zisserman, sa Oksfordskog univerziteta. Glavna i najznačajnija ideja ovakvog pristupa je da filteri (neuroni) zadrže najveću moguću jednostavnost. Sastoji se od nekoliko konvolutivnih blokova jedan iza drugog nakon kojih idu dva skrivena FC (en. *fully connected*) sloja i jedan izlazni sloj. Svaki od blokova čine nekoliko konvolutivnih slojeva nad podacima iste dimenzije i MaxPooling sloj posle njih. Najčešće se koristi VGG-16 arhitektura od 16 parametarskih slojeva, čiji se parametri podešavaju u procesu učenja. Ova arhitektura je prikazana na slici 4.

Ova arhitektura je odabrana zbog relativno dobrih rezultata koje daje kao i jednostavnosti implementacije. U VGG-16 varijanti ulazni sloj prima slike rezolucije 224x224 piksela koje se posle prolaska kroz pet kovolutivnih blokova transformišu u



Slika 4: VGG-16 arhitektura

matricu najvažnijih osobina veličine 7x7 koja se dalje propušta kroz dva FC sloja od po 4096 neurona od kojih je drugi spojen sa izlaznim slojem. Prvi konvolutivni blok ima dva sloja od po 64 neurona, drugi dva sloja od po 128 neurona, treći tri sloja od 256 neurona dok četvrti i peti blok imaju po tri sloja od po 512 neurona.

U ovom radu je korišćena modifikacija ove arhitekture. Korišćena su 4 konvolutivna bloka sa manjim brojem slojeva i manjim brojem neurona po slojevima kao i manjem broj neurona u dva FC sloja.

3.4 Google Colaboratory

Pošto je mašinsko učenje vrlo zahtevno što se tiče hardverskih performansi računara, opredelili smo se da treniranje vršimo na platformi Google Colaboratory. Ovo je besplatno Jupyter notebook okruženje koje ne zahteva nikakvu postavku i u potpunosti se izvršava u oblaku. Treniranje je vršeno na NVidia Tesla K80 grafičkoj sa 2496 CUDA jezgara i 12GB GDDR5 VRAM memorije. Jedna sesija korišćenja ovog servisa je ograničena vremenski na 12 sati.

Naš Jupyter notebook možete naći na sledećoj adresi:
http://tiny.cc/goodboiclassifieripynb

3.5 Preliminarni rezultati

Do zaključka koja veličina mreže nam je potrebna smo došli eksperimentalnim putem. Krenuli smo od samo jednog konvolutivnog bloka koji je sadržao 2 sloja od po 32 neurona, zatim smo postepeno povećavali brojeve neurona po sloju i broj blokova u mreži. Do odredjene veličine mreže model je previše brzo konvergirao. Kada smo prešli trenutnu veličinu modela počeo je da se javlja novi problem. Vrednost funkcije grečke je bila prevelika. Na slici 5 se nalazi optimalna veličina mreže, korišćena u ovom projektu.

Pri pravljenju modela korišćene su sledeći parametri:

- Za optimizaciju se koristi ADAM optimizator [4]
- Funkcija aktivacije na konvolutivnim slojevima je *Rectified Linear Unit*[5] funkcija, a za izlazni sloj je *softmax*[14] funkcija
- Za izračunavanje greške je korišćena funkcija kategoričke unakrsne entropije[8]

```
model = Sequential()

model.add(Conv2D(32, (3, 3), padding = 'same', input_shape = (144, 144, 1), kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding='same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(128, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding='same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(256, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding = 'same', kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Flatten())

model.add(Dense(512, kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024, kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(10))
model.add(Activation('softmax'))

opt = keras.optimizers.Adam(lr = 0.0001)

model.compile(loss = 'categorical_crossentropy', optimizer = opt, metrics = ['accuracy'])
```

Slika 5: Konvolutivna neuronska mreža za klasifikaciju rasa pasa

Na slikama 6, 7 i 8 se nalaze rezultati dobijeni nakon pokretanja procesa treniranja ove mreže na 226 epoha.

```
Epoch 1/226
 - 139s - loss: 2.2878 - acc: 0.1869 - val_loss: 2.1551 - val_acc: 0.1740
Epoch 2/226
 - 135s - loss: 2.1152 - acc: 0.2236 - val_loss: 2.0597 - val_acc: 0.2427
Epoch 3/226
 - 135s - loss: 2.0926 - acc: 0.2345 - val_loss: 2.0649 - val_acc: 0.2573
Epoch 4/226
 - 135s - loss: 2.0749 - acc: 0.2516 - val_loss: 2.0328 - val_acc: 0.2594
Epoch 5/226
 - 135s - loss: 2.0555 - acc: 0.2643 - val_loss: 2.0209 - val_acc: 0.2917
```

Slika 6: Prvih 5 epoha

```
Epoch 68/226
 - 135s - loss: 0.0926 - acc: 0.9730 - val_loss: 1.2558 - val_acc: 0.6937
Epoch 69/226
 - 135s - loss: 0.0935 - acc: 0.9734 - val_loss: 1.1645 - val_acc: 0.7125
Epoch 70/226
 - 135s - loss: 0.0985 - acc: 0.9733 - val_loss: 1.4889 - val_acc: 0.6740
Epoch 71/226
 - 135s - loss: 0.0871 - acc: 0.9740 - val_loss: 1.3431 - val_acc: 0.6833
Epoch 72/226
 - 135s - loss: 0.0888 - acc: 0.9752 - val_loss: 1.2903 - val_acc: 0.7073
```

Slika 7: Epohe od 68 do 72

```
Epoch 222/226
 - 135s - loss: 0.0209 - acc: 0.9942 - val_loss: 1.1986 - val_acc: 0.7393
Epoch 223/226
 - 135s - loss: 0.0264 - acc: 0.9931 - val_loss: 1.3438 - val_acc: 0.7115
Epoch 224/226
 - 135s - loss: 0.0296 - acc: 0.9923 - val_loss: 1.3536 - val_acc: 0.7250
Epoch 225/226
 - 135s - loss: 0.0292 - acc: 0.9932 - val_loss: 1.2928 - val_acc: 0.7333
Epoch 226/226
 - 135s - loss: 0.0249 - acc: 0.9935 - val_loss: 1.2692 - val_acc: 0.7531
```

Slika 8: Poslednjih 5 epoha

Možemo uočiti da su finalne tačnosti dostignute između 68 i 72 epohe i da se u narednih 150 epoha nije ništa menjalo.

3.6 Analiza rezultata

Na slici 9 se nalazi matrica konfuzije.

```
Confusion Matrix
[[843   0   4  34   6  12   8  16  20  24]
 [ 39   0  36 124  31  22  28  27  43  49]
 [  8   0 315  26   8   3  21   0   6  39]
 [135   0  17 617  15   6   7  23  42  37]
 [ 34   0   2  23 384  10   4  23  36  29]
 [ 22   0   8   3  14 339   6  17  66  70]
 [ 41   0   8  16   2   8 254  20  13  62]
 [ 38   0   1  14  18  15  13 294  32  17]
 [ 30   0   0  15  17  13  13  10 256  68]
 [ 11   0   6   7   9  18  36   6  29 985]]
```

Slika 9: Matrica konfuzije

Rasa	Dobro klasifikovano	Pogrešno klasifikovano
pudla	843	124
nemački ovčar	0	399
rotvajler	315	111
šnaucer	617	282
aljaski malamut	384	161
bigl	339	206
mađarska vižla	254	170
zlatni retriver	294	148
čivava	256	166
italijanski hrt	985	122

Classification Report

	precision	recall	f1-score	support
00-Caniche	0.70	0.87	0.78	967
01-Deutscher Schaeferhund	0.00	0.00	0.00	399
02-Rottweiler	0.79	0.74	0.77	426
03-Schnauzer	0.70	0.69	0.69	899
04-Alaskan malamute	0.76	0.70	0.73	545
05-Beagle	0.76	0.62	0.68	545
06-Magyar vizsla	0.65	0.60	0.62	424
07-Golden retriever	0.67	0.67	0.67	442
08-Chihuahua	0.47	0.61	0.53	422
09-Picolo levriero italiano	0.71	0.89	0.79	1107
accuracy			0.69	6176
macro avg	0.62	0.64	0.63	6176
weighted avg	0.65	0.69	0.67	6176

Slika 10: Izveštaj klasifikacije

Na slici 10 se nalazi izveštaj klasifikacije.

Kao što možemo uočiti najlošije se klasifikuju nemački ovčari, a najveću preciznost u klasifikaciji smo ostvarili kod rotvajlera.

Nelogičnost sa kojom smo se susreli je da nijedan pas nije pogrešno klasifikovan kao nemački ovčar, niti je makar jedan pas te rase dobro klasifikovan.

Srednja preciznost na celom skupu podataka iznosi 69%, što je zadovoljavajuće.

4 Zaključak

Za optimalne rezultate, neophodan je znatno veći broj slika, koje pritom treba da budu što raznovrsnije. Veći broj različitih klasa povećava potreban broj slika po klasi. Metodi primenjeni za prevazilaženje ovog problema[7] nisu dali željene krajnje rezultate. Pored toga, bilo bi potrebno mnogo više vremena za treniranje jer bi povećanje skupa podataka povećalo kompleksnost mreže, kao i trajanje epohe.

Podaci sa kojima smo mi raspolagali su bili dosta repetitivni. Proces prikupljanja podataka je takođe neophodno produžiti i unaprediti za prikupljanje reprezentativnih podataka. Slike dostupne po rezultatima pretrage na sajtu https://images.google.com već oko stotog rezlatata prestaju da budu slike pasa i postaju slike drugih stvari (ljudi, šolja, čarapa, torbi...).

Literatura

[1] Intel AI Academy. Hands-On AI Part 16: Modern Deep Neural Network Architectures for Image Classification, 2017. on-line at: https://software.intel.com/en-us/articles/hands-on-ai-part-16-modern-deep-neural-network-architectures-for-image-classification.

[2] Charu C. Aggarwal. *Data Mining The Textbook*. Springer, New York, 2015.

[3] Anne Bonner. Wtf is image classification? on-line at: https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb.

[4] Jason Brownlee. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, 2017. on-line at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[5] Francois Chollet. A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks, 2016. on-line at: https://ksrs.rs/fci-standardi.

[6] Nikola Živković Dušica Krstić. preprocessing. on-line at: https://github.com/dmkrstic/goodboiclassifier/tree/master/src/preprocessing.

[7] FCI. Building powerful image classification models using very little data, 2019. on-line at: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html.

[8] Raúl Gómez. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names, 2018. on-line at: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[9] Fédération Cynologique Internationale. Fédération Cynologique Internationale, 2019. on-line at: http://www.fci.be/en/.

[10] Keras. Keras, 2019. on-line at: https://keras.io/.

[11] Mladen Nikolić Predrag Janičić. Veštačka inteligencija, 2018. on-line at: http://poincare.matf.bg.ac.rs/~janicic//courses/vi.pdf.

[12] Bharath Raj. How to use Deep Learning when you have Limited Data, 2018. on-line at: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced.

[13] TensorFlow. Tensorflow, 2019. on-line at: https://www.tensorflow.org/.

[14] Uniqtech. Understand the Softmax Function in Minutes, 2018. on-line at: https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d.