Seminarski rad iz Računarske Inteligencije

Prepoznavanje cifara upotrebom neuronskih mreža

Stefan Stevović 151/2015
Đorđe Vučković 31/2015

SADRŽAJ:

UVOD

Seminarski rad se zasniva na problemu prepoznavanja cifara sa slika korišćenjem konvolutivnih neuronskih mreža (CNN – Convolutional Neural Network). CNN sačinjava jedan ili više konvolucionih slojeva i opciono jedan ili više potpuno povezanih (Dense) slojeva, koji se mogu sresti u konvolutivnim višeslojnim neuronskim mrežama. CNN su projektovane tako da prednost postižu u radu sa 2D strukturama, kao što su slike ili ulazi poput govornog signala.

Za izradu projekta smo koristili MNIST bazu podataka, koja se sastoji od 60.000 slika za treniranje i 10.000 slika za testiranje preciznosti modela. Uz MNIST bazu podataka za pisanje samog projekta koristili smo programski jezik Python 2.7, biblioteku Numpy koja nam je služila za brzu linearnu algebru, kao i biblioteke Matplotlib, PIL i cv2 koje su nam služile za rad sa slikama, pre svega u testiranju modela sa slikama koje smo mi napravili. Što se tiče API-ja korisćen je KERAS koji predstavlja API za neuronske mreže visokog nivoa i koji je sposoban da radi na TensorFlow, Python biblioteci otvorenog koda koju je razvio Google Brain za potrebe deep learning-a.

Neuronske mreže predstavljaju širok pojam i njihovo polje delovanja je jako široko, ovaj problem koji smo mi obradili je samo deo svega onog sto neuronskim mrežama može da se predstavi. Postoji više načina da se ovaj problem predstavi i reši ali svaki od njih ima skoro pa isti sastav uz neznatne izmene. Čitav posao se deli u nekoliko celina:

1) Odabir okruženja
2) Priprema baze podataka
3) Izgradnja mreže
4) Kompajliranje i trening podataka
5) Testiranje i ocenjivanje performansi modela

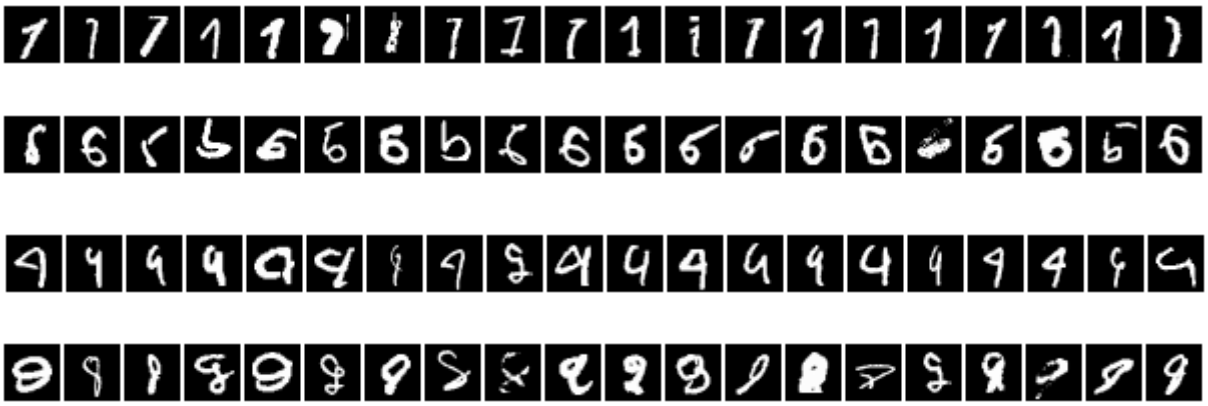
Ovih 5, gore navedenih stavki predstavlja kostur rešavanja ovog problema. Postoje ralzličiti načini da se svaki od ovih stavki implementira i izabere. To predstavlja glavni zadatak svakog ko želi da se bavi ovim problemom.

OPIS REŠENJA

Ovaj problem je razložen na dva dela. Prvi obuhvata proces treniranja mreže i čuvanja naučenog modela. Drugi deo obuhvata proces testiranja, odnosno učitavanja proizvoljne slike, njenu obradu i primenu modela na datu sliku.

Treniranje modela:

Naš model se sastoji od 60.000 slika za treniranje. Međutim, koristeći "ImageDataGenerator" uspeli smo da "razmrdamo" naše podatke tako što smo svaku sliku rotirali za slučajan broj stepeni od 0 do 10, zumirali za proizvoljnu vrednost od 0 do 0.1, pomerili horizontalno, pa zatim vertikalno za slučajnu vrednost od 0 do 0.1. Time smo dobili znatno raznovrsniju bazu podataka. Podatke proširujemo zbog raloga datih na sledećim slikama, gde vidimo da postoje veliki problemi sa prepozavanjem sličnih cifara (pogledati prvu sliku gde su 1 i 7 pisani maltene isto).



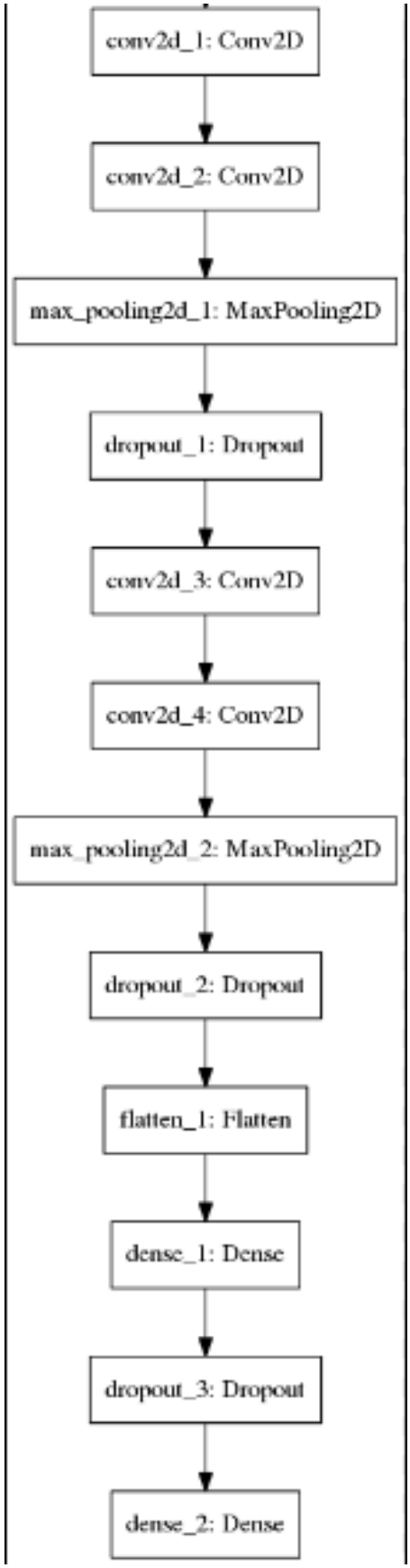
U našem modelu smo koristili, redom, dva "Conv2D" sloja sa po 32 filtera, oba sa matricom veličine 5x5 i "relu" aktivacionom funkcijom. Zatim sledi jedan "MaxPool2D" sloj sa 2x2 matricom.

Nakon njih slede dva nova "Conv2D" sloja sa po 64 filtera, oba sa matricom veličine 3x3 i "relu" aktivacionom funkcijom, pa nakon njih jos jedan "MaxPool2D" sa veličinom matrice 2x2.

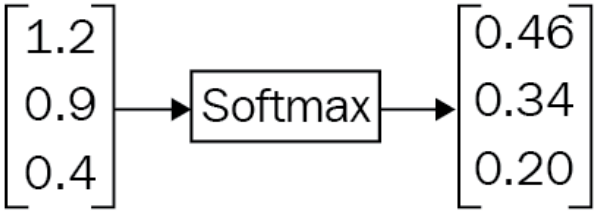
Nakon toga sledi sloj "Flatten" koji služi za povezivanje "Conv2D" slojeva sa "Dense" slojevima.

Na kraju konvolutivnog modela gotovo uvek stoje "Dense" slojevi, kod kojih važi pravilo da je svaki čvor novog sloja povezan sa svakim čvorom prethodnog sloja. Prvi sloj je veličine 256, a drugi 10, jer na izlazu imamo 10 mogućnosti, tj. 10 različitih cifara. Kod prvog "Dense" sloja smo koristili "relu" aktivacionu funkciju, a kod drugog, poslednjeg sloja u mrezi, smo koristili "softmax" aktivacionu funkciju. "Sofmax" je karakteristična po tome sto je suma svih elemenata na izlazu jednaka 1, tako da ćemo na izlazu dobiti neki niz [0.1 0 0 0 0 0.1 0 0 0 0.8] i to znači da je našoj slici broj 9.

Keras sadrži metod “plot_model” koji nam omogućava da vidimo sliku našeg modela. On izleda ovako:



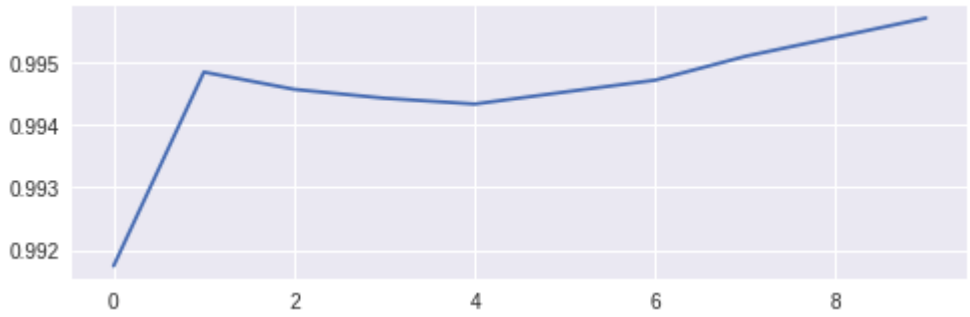
ActivationFunction 1: ReLU - rectified linear unit



ActivationFunction 2: Softmax

Što se tiče samog treniranja modela, bilo ga je nemoguće trenirati na našim mašinama koji su prosečnih specifikacija. Jedna iteracija ove mreže je na našim mašinama trajala oko 22 sata. Iz tog razloga smo bili prinuđeni da koristimo Google-ov server, tačnije “Colaboratory”. Od opcija smo definisali da se prevodi kod za Python 2.7 i da se pri treniranju koristi “GPU” koji generalno daje mnogo bolje performanse. Bez odabira te opcije i na Google-ovom serveru jedna epoha je trajala jako dugo. Nakon ovih podešavanja, na tom serveru je jedna epoha trajala oko pedesetak minuta.

Nakon treniranja modela, model smo sačuvali na Google Drive-u da bismo ga kasnije odatle preuzeli i koristili prilikom testiranja modela.



```
('X_train shape:', (60000, 28, 28, 1))
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 32)        25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               803072
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 887,530
Trainable params: 887,530
Non-trainable params: 0
```

```
_________________________________________________________________
Epoch 1/10
60000/60000 [==============================] - 2515s 42ms/step - loss: 0.0281 - acc: 0.9917
Epoch 2/10
60000/60000 [==============================] - 2521s 42ms/step - loss: 0.0208 - acc: 0.9949
Epoch 3/10
60000/60000 [==============================] - 2572s 43ms/step - loss: 0.0246 - acc: 0.9946
Epoch 4/10
60000/60000 [==============================] - 2563s 43ms/step - loss: 0.0281 - acc: 0.9944
Epoch 5/10
60000/60000 [==============================] - 2579s 43ms/step - loss: 0.0316 - acc: 0.9943
Epoch 6/10
60000/60000 [==============================] - 2601s 43ms/step - loss: 0.0333 - acc: 0.9945
Epoch 7/10
60000/60000 [==============================] - 2606s 43ms/step - loss: 0.0359 - acc: 0.9947
Epoch 8/10
60000/60000 [==============================] - 2602s 43ms/step - loss: 0.0386 - acc: 0.9951
Epoch 9/10
60000/60000 [==============================] - 2597s 43ms/step - loss: 0.0409 - acc: 0.9954
Epoch 10/10
60000/60000 [==============================] - 2582s 43ms/step - loss: 0.0462 - acc: 0.9957
('Test loss:', 0.0478560668170456)
('Test accuracy:', 0.9966)
```

Testiranje modela:

Prilikom testiranja modela, prva stvar koju je potrebno uraditi jeste učitavanje samog modela kog smo sačuvali nakon procesa treniranja. Nakon toga sledi učitavanje slike na koju želimo da primenimo učitani model. Nakon učitavanja slike, sledi binarizacija, gde slike definišemo tako da cifre budu ispisane belom bojom na crnoj pozadini. Svaki piksel ima vrednost od 0 do 255, a naš prag za binarizaciju je 128. To znači da će svaki piksel koji ima vrednost manju od 100 biti obojen u crno, a svaki piksel koji ima vrednost veću od 128 biti obojen u belo.

Nakon binarizacije sledi promena veličine same slike jer naš model prima samo one slike koje su veličine 28x28 piksela.

UPOREĐIVANJE MREŽA

Samo vreme treniranja mreže i losa internet konekcija nam nije ostavila mnogo prostora za treniranje većeg broja različitih mreža i njihovo upoređivanje (da podsetimo, jedna epoha je na nasem računaru trajala 21-22 sata, a na Google-ovom serveru oko pedesetak minuta). Ono sto smo mi modifikovali kod naseg programa jeste samo dodavanje dodatnog “Dense” sloja sa 1024 čvorova i preciznost je u tom slučaju bila -0.05 sto je neznačajno. Pored toga, probali smo sa dodavanjem jos 2 ”Conv2D” i jednim ”MaxPool2D” slojem, međutim preciznost je i dalje za nijansu bila gora od preciznosti našeg modela. Što se tiče broja iteracija, u svim navedenim slučajevima smo stavili da bude 10, jer smo primetili da je 10 iteracija sasvim dovoljno da treniranje modela. Sve posle desete iteracije je skoro neznatno poboljšavalo preciznost.

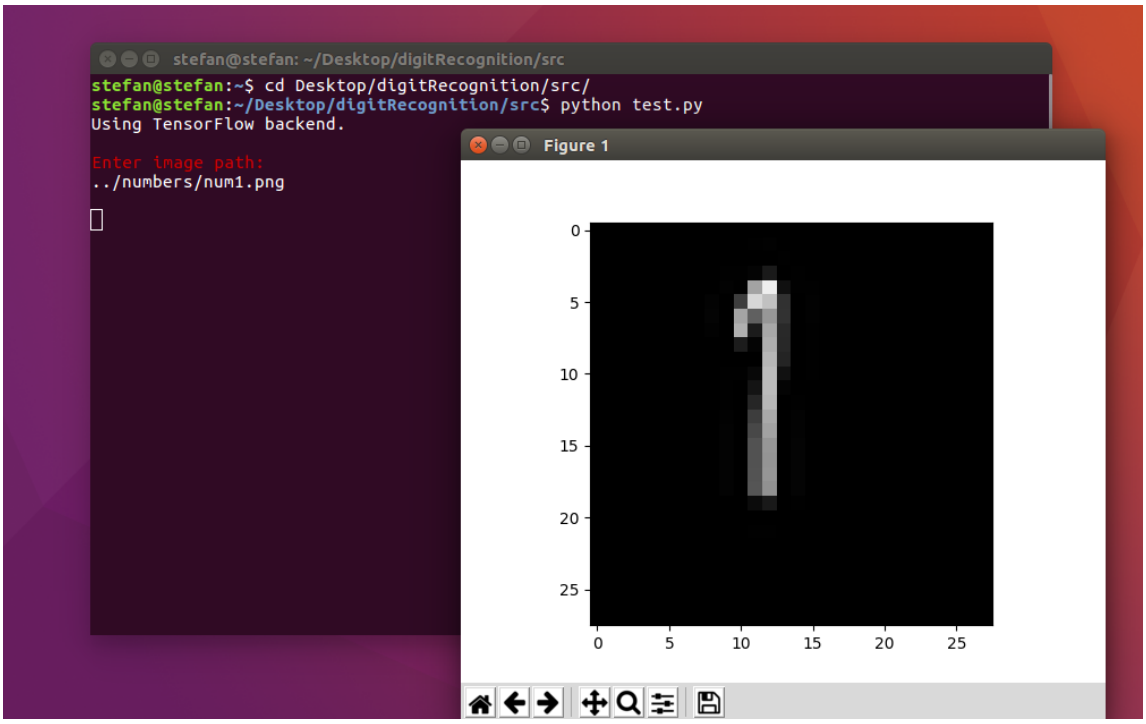
UPOTREBA PROGRAMA

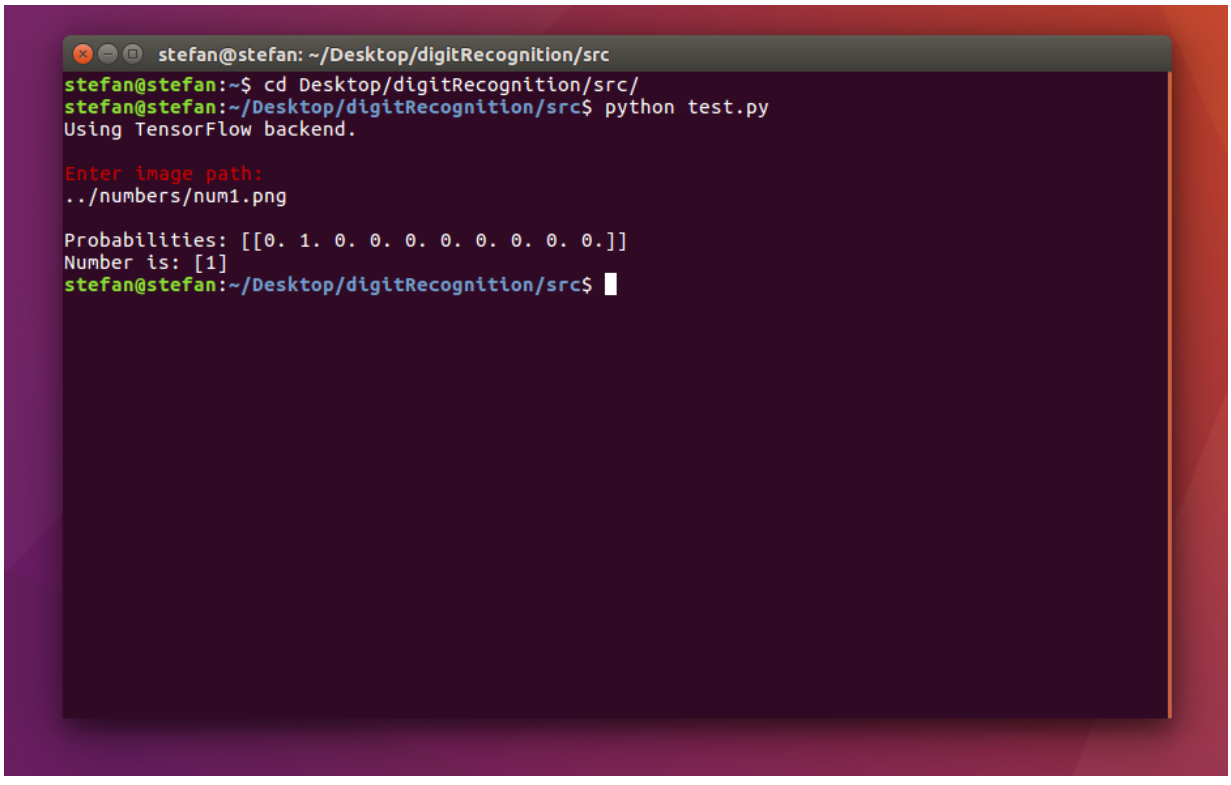
Da bismo kreirali naš model potrebno je pokrenuti program “training.py”. Nakon treniranja mreže dobićemo model “best_model.h5”.

Mi smo definisali dva načina na koji je moguće testirati podatke. Prvi način je definisan specijalno za potrebe odbrane projekta. On podrazumeva obradu svih slika odjednom koji se nalaze u direktorijumu “numbers”. U suštini, ideja je potpuno ista

kao kod testiranja pojedinačnih instanci (slika), samo što imamo jednu for petlju koja obilazi sve slike unutar foldera i primenjuje model na svaku od njih bez prikazivanja same slike radi uštede vremena.

Drugi način je unošenje putanje do slike koju je potrebno obraditi i to je standardni način testiranja algoritma koji bi korisnik trebao da koristi. Kada pokrenemo “test.py”, izlazi nam tekst koji nam kaže da unesemo putanju do slike. Nakon toga, prvo ce nam se prikazati binarizovana slika koju smo učitali. Izlaskom iz slike, učitana slika se dalje obrađuje i u terminalu nam se ispisuje predviđanje modela.





ZAKLJUČAK

Preciznost modela neuronske mreže zavisi od podešavanja parametara samog modela poput broja epoha, slojeva, čvorova u slojevima, funkcije gubitka itd. Model koji smo mi dobili nije najbolji mogući, jer se pronalašenjem odgovarajućih vrednosti ovih parametara moze dobiti model koji je za bolji od naseg, ali sa preciznosću od oko 0.97 nas model je sasvim dobar.

Što se tiče samog unapređivanja algoritma, korisno bi bilo doraditi program tako da prepoznaje više od jedne cifre na nekoj slici. Za to je, pored malo bolje istreniranog modela od ključne važnosti preprocesiranje slika. Pod tim preprocesiranjem se najviše misli na primene različitih naprednih metoda za binarizaciju slika, otklanjanje šuma itd. Sve te navedene stvari se tiču rada sa slikama i nisu deo ovog projekta.

LITERATURA

[1] Neural networks, Christos Stergiou and Dimitrios Siganos

[2] Neural Networks and Deep Learning, Michael Nielsen

[3] Materijali sa predavanja profesora Aleksandra Kartelja, Matematički fakultet Beograd

[4] The MNIST database of handwriten digits, Yann LeCun and Corina Cortes and Christopher J.C. Burges