

Photo-calculator

Nemanja Šobo

Motivacija

- ◆ Aplikacija photo-calculator ima za cilj izračunavanje matematičkih izraza srednje složenosti, koristeći matematičke funkcije kao što su \sin , \cos , integral itd.
- ◆ Kao mali smo uvek bili u situaciji kada smo da dobijemo neke matematičke izraze za domaći zadatak iz knjige bez rešenja, gde nikada nismo mogli proveriti da li su naša rešenja tačna.

- ◆ Ova aplikacija bi mogla pomoći osnovcima da prilikom rada na domaćim zadacima imaju mogućnost provere tačnosti njihovih rezultata.

Slična rešenja

◆ MySriptCalculator

$$\frac{8 \times \pi}{48(\sin \pi)e^2}$$



MyScript Calculator



$$\frac{\sqrt{81}}{3} = 3$$

$$1 + 15 + 100$$

$$5 \times 10$$

- ◆ Ovaj kao i njemu slična rešenja su uglavno Android aplikacije, koje prepoznaju rukom pisane izraze, sa sposobnošću prepoznavanja matematičkih operacija kao što su sabiranje, oduzimanje, deljenje, množenje, sin, cos, tan itd.

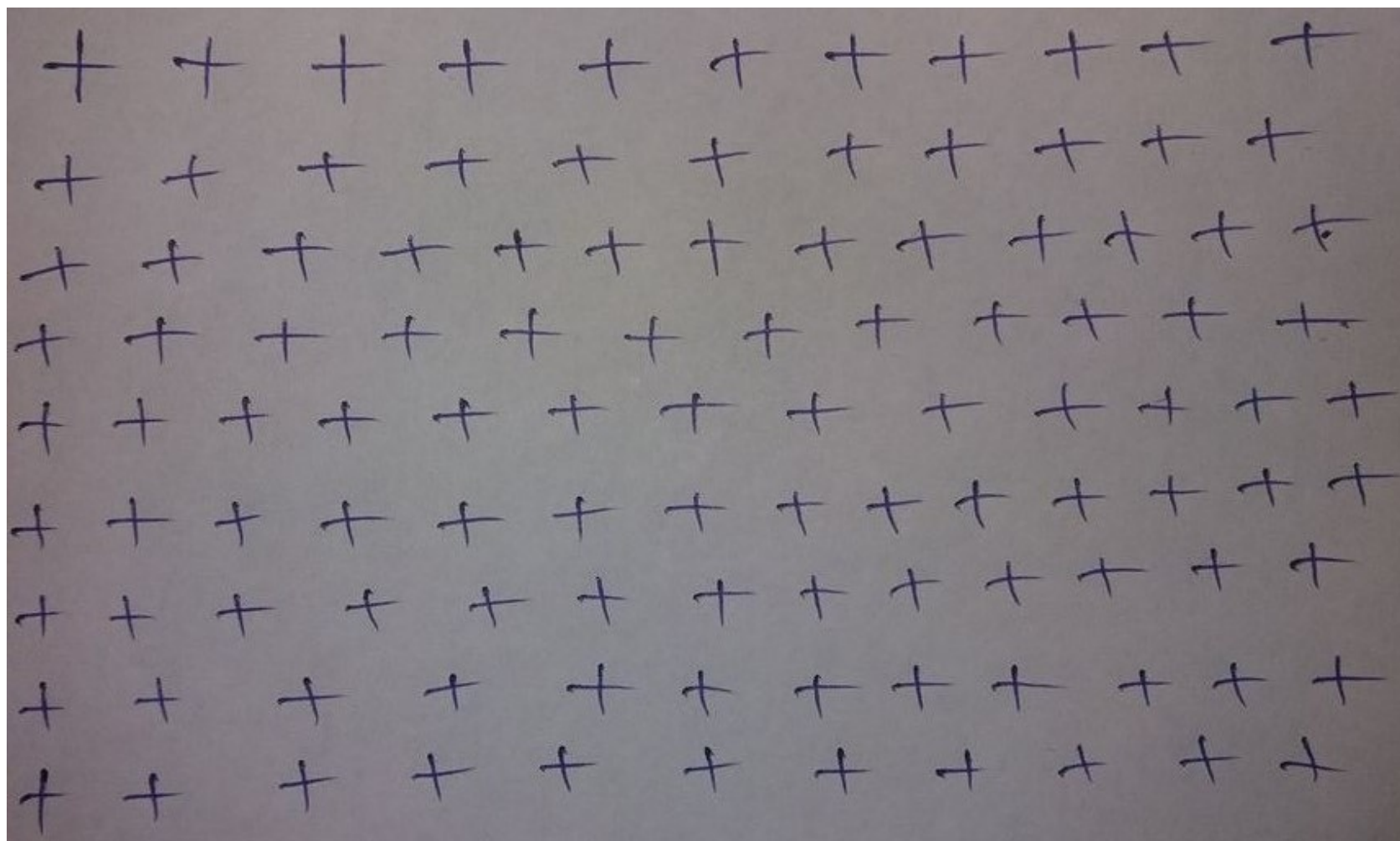
Koraci implementacije

- ◆ Korišćenjem datog koda na vezbama I njegovoj izmeni I nadogradnji, projekat je pisan u python-u.
- ◆ 1.Obučavanje neuronske mreže:
- ◆ Korištena baza podataka MNIST iz keras-ove biblioteke koja sadrži 60.000 slika obuku i 10.000 slika za testiranje

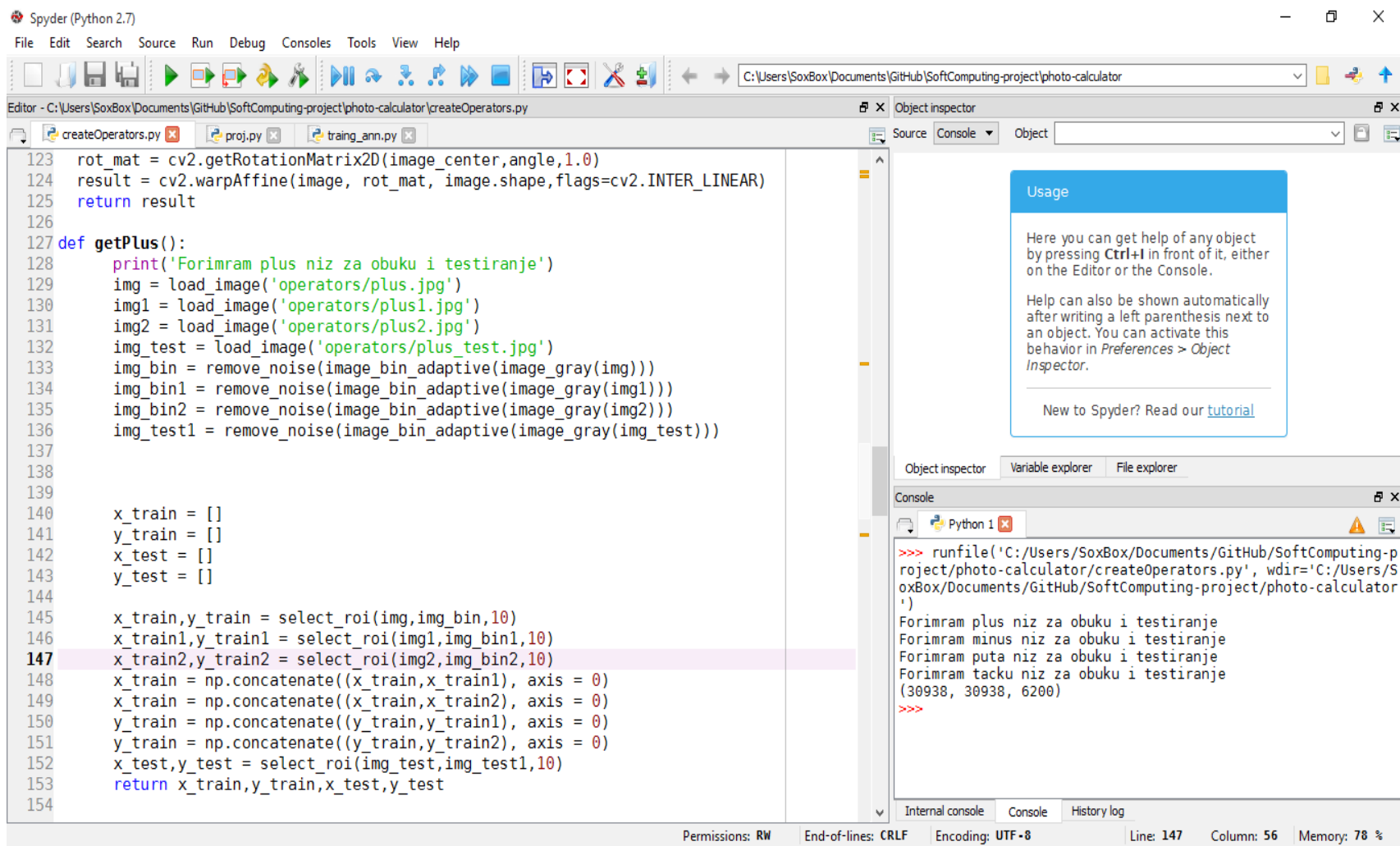
Koraci implementacije

- ◆ Keras je minimalistički , visoko modularni biblioteka neuronske mreže , napisan u Pythonu . Razvijena s naglaskom da omogući brzo eksperimentisanje .
- ◆ Na MNIST-ov obučavajući niz dodati ručno pisani operatori +,-, kao i tačka za prepoznavanje znaka puta i nepoznata promenljiva X.
- ◆ Dobijeni niz preko 90000 u obučavajućem skupu i preko 16000 u test skupu.

Ulazna fotografija za znak +



Prikaz koda za stvaranje obučavajućeg i niza testiranja za znak +



The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script named `createOperators.py` located at `C:\Users\SoxBox\Documents\GitHub\SoftComputing-project\photo-calculator\createOperators.py`. The code defines a `getPlus()` function that loads and processes images for training and testing. The function prints a message and returns a list of training data (`x_train`, `y_train`) and testing data (`x_test`, `y_test`). The code is as follows:

```
123 rot_mat = cv2.getRotationMatrix2D(image_center,angle,1.0)
124 result = cv2.warpAffine(image, rot_mat, image.shape,flags=cv2.INTER_LINEAR)
125 return result
126
127 def getPlus():
128     print('Forimram plus niz za obuku i testiranje')
129     img = load_image('operators/plus.jpg')
130     img1 = load_image('operators/plus1.jpg')
131     img2 = load_image('operators/plus2.jpg')
132     img_test = load_image('operators/plus_test.jpg')
133     img_bin = remove_noise(image_bin_adaptive(image_gray(img)))
134     img_bin1 = remove_noise(image_bin_adaptive(image_gray(img1)))
135     img_bin2 = remove_noise(image_bin_adaptive(image_gray(img2)))
136     img_test1 = remove_noise(image_bin_adaptive(image_gray(img_test)))
137
138
139
140     x_train = []
141     y_train = []
142     x_test = []
143     y_test = []
144
145     x_train,y_train = select_roi(img,img_bin,10)
146     x_train1,y_train1 = select_roi(img1,img_bin1,10)
147     x_train2,y_train2 = select_roi(img2,img_bin2,10)
148     x_train = np.concatenate((x_train,x_train1), axis = 0)
149     x_train = np.concatenate((x_train,x_train2), axis = 0)
150     y_train = np.concatenate((y_train,y_train1), axis = 0)
151     y_train = np.concatenate((y_train,y_train2), axis = 0)
152     x_test,y_test = select_roi(img_test,img_test1,10)
153     return x_train,y_train,x_test,y_test
154
```

The right sidebar contains the Object inspector, Variable explorer, and File explorer. The Console window shows the output of the `runfile` command, which includes the printed message and the dimensions of the training and testing data:

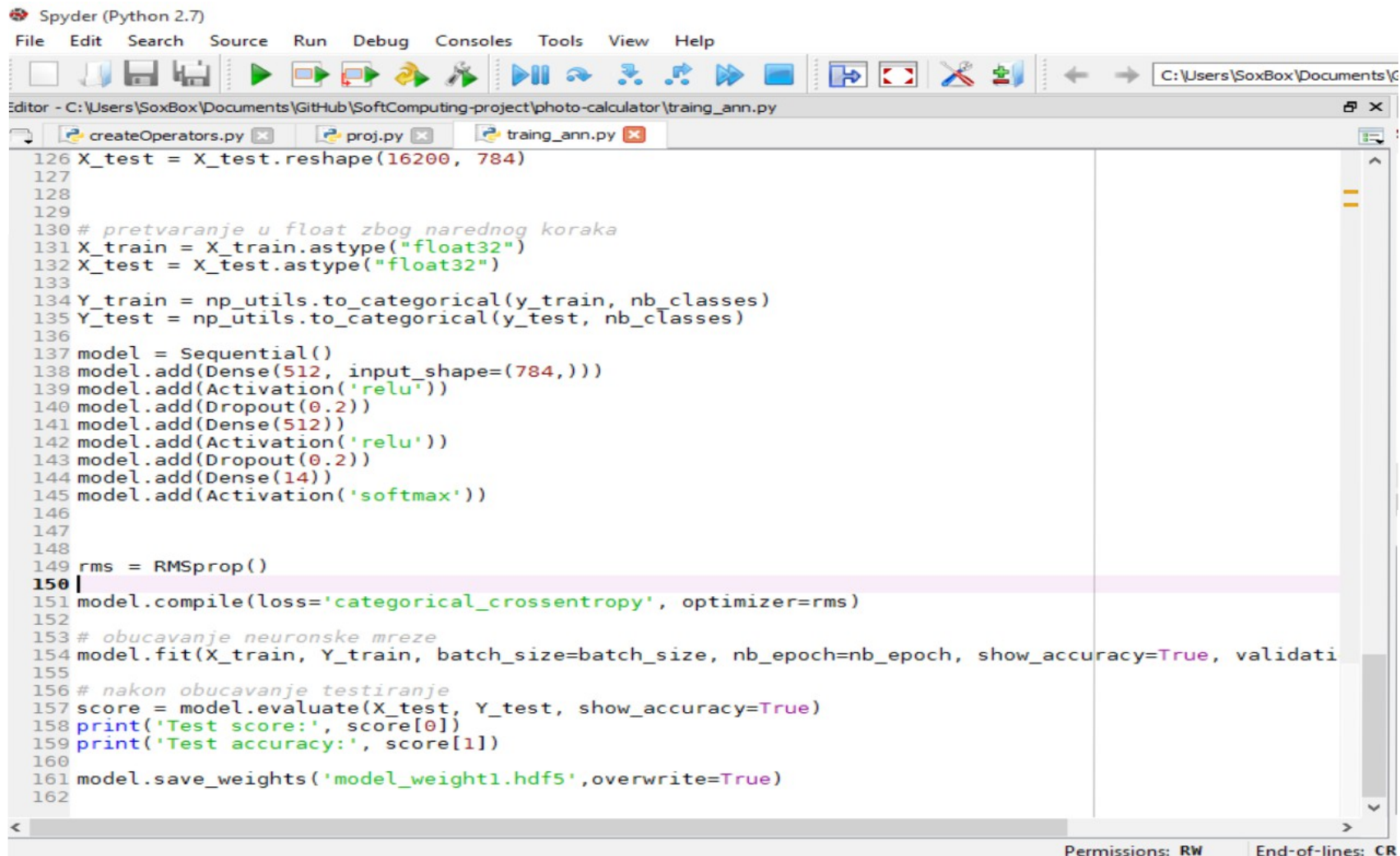
```
>>> runfile('C:/Users/SoxBox/Documents/GitHub/SoftComputing-project/photo-calculator/createOperators.py', wdir='C:/Users/SoxBox/Documents/GitHub/SoftComputing-project/photo-calculator')
Forimram plus niz za obuku i testiranje
Forimram minus niz za obuku i testiranje
Forimram puta niz za obuku i testiranje
Forimram tacku niz za obuku i testiranje
(30938, 30938, 6200)
>>>
```

The status bar at the bottom indicates the file permissions (RW), end-of-lines (CRLF), encoding (UTF-8), and the current position (Line: 147, Column: 56, Memory: 78 %).

Koraci implementacije

- ◆ Model neuronske mreže
- ◆ Keras ima 2 Sequential I Graph, korišćen sequential
- ◆ Ulazni sloj od 784 neurona I dva skrivena sloja od 512 neurona I izlazni sloj od 14 neurona.

Prikaz koda za obučavanje neuronske mreže



The image shows a screenshot of the Spyder Python IDE interface. The title bar indicates it is running Python 2.7. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, and Help. The toolbar contains various icons for file operations and code execution. The editor window displays a Python script for training a neural network. The script includes steps for reshaping data, converting it to float32, creating categorical labels, building a sequential model with three dense layers and dropout, compiling it with categorical crossentropy loss and RMSprop optimizer, and finally training and evaluating the model. The file path in the editor is C:\Users\SoxBox\Documents\GitHub\SoftComputing-project\photo-calculator\traing_ann.py. The status bar at the bottom shows 'Permissions: RW' and 'End-of-lines: CR'.

```
126 X_test = X_test.reshape(16200, 784)
127
128
129
130 # pretvaranje u float zbog narednog koraka
131 X_train = X_train.astype("float32")
132 X_test = X_test.astype("float32")
133
134 Y_train = np_utils.to_categorical(y_train, nb_classes)
135 Y_test = np_utils.to_categorical(y_test, nb_classes)
136
137 model = Sequential()
138 model.add(Dense(512, input_shape=(784,)))
139 model.add(Activation('relu'))
140 model.add(Dropout(0.2))
141 model.add(Dense(512))
142 model.add(Activation('relu'))
143 model.add(Dropout(0.2))
144 model.add(Dense(14))
145 model.add(Activation('softmax'))
146
147
148
149 rms = RMSprop()
150 model.compile(loss='categorical_crossentropy', optimizer=rms)
151
152
153 # obucavanje neuronske mreze
154 model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch, show_accuracy=True, validation_data=(X_test, Y_test))
155
156 # nakon obucavanje testiranje
157 score = model.evaluate(X_test, Y_test, show_accuracy=True)
158 print('Test score:', score[0])
159 print('Test accuracy:', score[1])
160
161 model.save_weights('model_weight1.hdf5', overwrite=True)
162
```

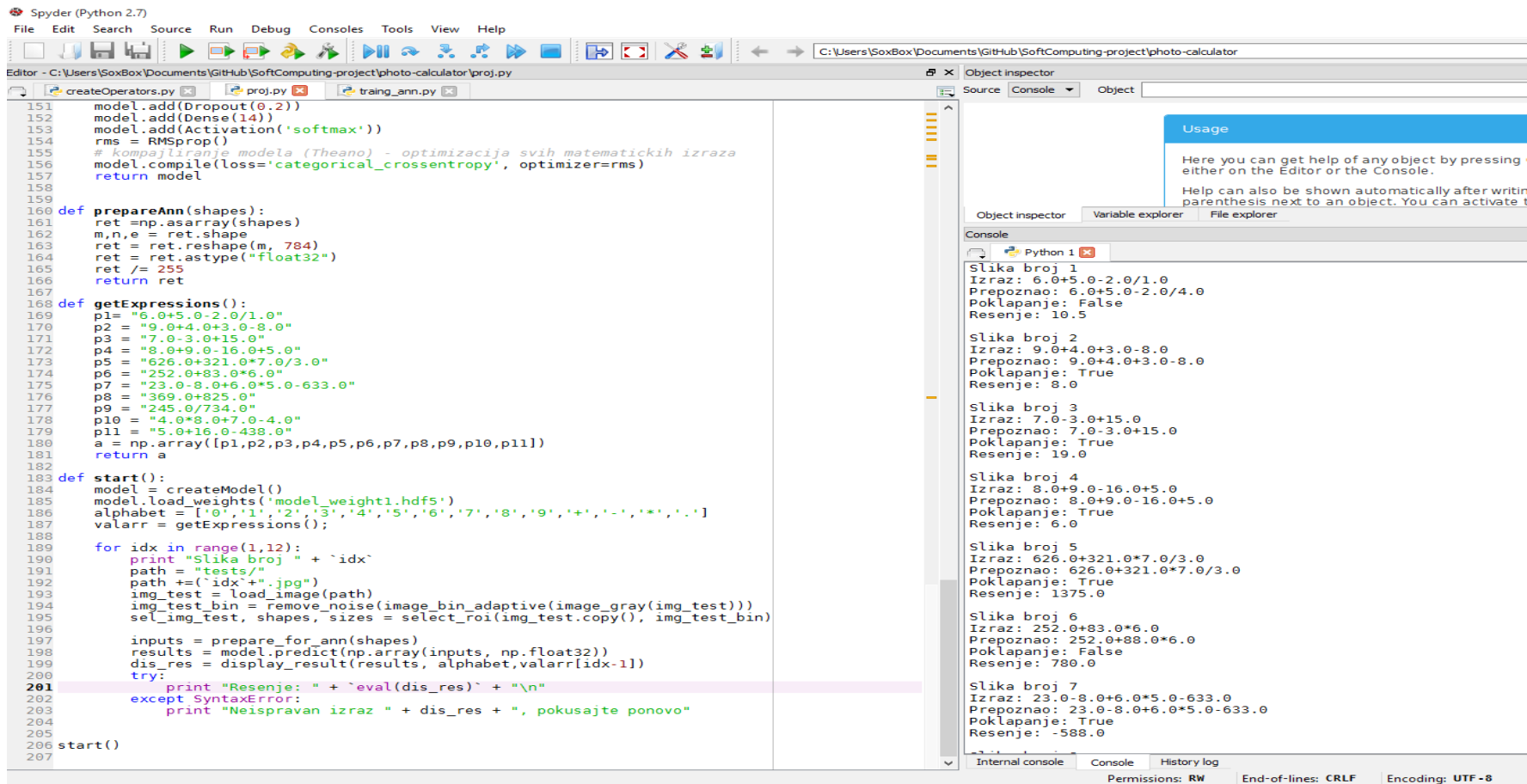
Prikaz tačnosti obučenosti mreže

```
Train on 90938 samples, validate on 16200 samples
Epoch 1/10
90938/90938 [=====] - 72s - loss: 0.2847 - acc: 0.9164 - val_loss: 0.2770 - val_acc: 0.9157
Epoch 2/10
90938/90938 [=====] - 70s - loss: 0.1041 - acc: 0.9679 - val_loss: 0.2853 - val_acc: 0.9154
Epoch 3/10
90938/90938 [=====] - 72s - loss: 0.0697 - acc: 0.9786 - val_loss: 0.2984 - val_acc: 0.9185
Epoch 4/10
90938/90938 [=====] - 71s - loss: 0.0483 - acc: 0.9846 - val_loss: 0.2784 - val_acc: 0.9314
Epoch 5/10
90938/90938 [=====] - 71s - loss: 0.0344 - acc: 0.9887 - val_loss: 0.2955 - val_acc: 0.9269
Epoch 6/10
90938/90938 [=====] - 71s - loss: 0.0247 - acc: 0.9921 - val_loss: 0.3948 - val_acc: 0.9136
Epoch 7/10
90938/90938 [=====] - 71s - loss: 0.0192 - acc: 0.9938 - val_loss: 0.2714 - val_acc: 0.9381
Epoch 8/10
90938/90938 [=====] - 71s - loss: 0.0159 - acc: 0.9948 - val_loss: 0.2429 - val_acc: 0.9423
Epoch 9/10
90938/90938 [=====] - 71s - loss: 0.0127 - acc: 0.9958 - val_loss: 0.3061 - val_acc: 0.9360
Epoch 10/10
90938/90938 [=====] - 77s - loss: 0.0105 - acc: 0.9965 - val_loss: 0.3423 - val_acc: 0.9322
16200/16200 [=====] - 2s
Test score: 0.34233859475
Test accuracy: 0.932222222222
```

Koraci implementacije

- ◆ Implementacija koda za korišćenje neuronske mreže i prepoznavanje matematičkih izraza u python-u.
- ◆ Uneto je 11 fotografija na kojima su napisani proizvoljni matematički izrazi, rezultat toga je preko 80 znakova, što cifara što operatora koje je mreža morala da prepozna.

Implementacija u python-u



```
151 model.add(Dropout(0.2))
152 model.add(Dense(14))
153 model.add(Activation('softmax'))
154 rms = RMSprop()
155 # kompajliranje modela (Theano) - optimizacija svih matematickih izraza
156 model.compile(loss='categorical_crossentropy', optimizer=rms)
157 return model
158
159 def prepareAnn(shapes):
160     ret = np.asarray(shapes)
161     m,n,e = ret.shape
162     ret = ret.reshape(m, 784)
163     ret = ret.astype("float32")
164     ret /= 255
165     return ret
166
167 def getExpressions():
168     p1= "6.0+5.0-2.0/1.0"
169     p2 = "9.0+4.0+3.0-8.0"
170     p3 = "7.0-3.0+15.0"
171     p4 = "8.0+9.0-16.0+5.0"
172     p5 = "626.0+321.0*7.0/3.0"
173     p6 = "252.0+83.0*6.0"
174     p7 = "23.0-8.0+6.0*5.0-633.0"
175     p8 = "369.0+825.0"
176     p9 = "245.0/734.0"
177     p10 = "4.0*8.0+7.0-4.0"
178     p11 = "5.0+16.0-438.0"
179     a = np.array([p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11])
180     return a
181
182 def start():
183     model = createModel()
184     model.load_weights('model_weight1.hdf5')
185     alphabet = ['0','1','2','3','4','5','6','7','8','9','+','-','*','.']
186     valarr = getExpressions();
187
188     for idx in range(1,12):
189         print "Slika broj " + `idx`
190         path = "tests/"
191         path +=(`idx`+".jpg")
192         img_test = load_image(path)
193         img_test_bin = remove_noise(image_bin_adaptive(image_gray(img_test)))
194         sel_img_test, shapes, sizes = select_roi(img_test.copy(), img_test_bin)
195
196         inputs = prepare_for_ann(shapes)
197         results = model.predict(np.array(inputs, np.float32))
198         dis_res = display_result(results, alphabet, valarr[idx-1])
199         try:
200             print "Resenje: " + `eval(dis_res)` + "\n"
201         except SyntaxError:
202             print "Neispravan izraz " + dis_res + ", pokušajte ponovo"
203
204 start()
205
206
207
```

Usage

Here you can get help of any object by pressing either on the Editor or the Console.

Help can also be shown automatically after writing parenthesis next to an object. You can activate t

Object inspector Variable explorer File explorer

Console

Python 1

Slika broj 1
Izraz: 6.0+5.0-2.0/1.0
Prepoznao: 6.0+5.0-2.0/4.0
Poklapanje: False
Resenje: 10.5

Slika broj 2
Izraz: 9.0+4.0+3.0-8.0
Prepoznao: 9.0+4.0+3.0-8.0
Poklapanje: True
Resenje: 8.0

Slika broj 3
Izraz: 7.0-3.0+15.0
Prepoznao: 7.0-3.0+15.0
Poklapanje: True
Resenje: 19.0

Slika broj 4
Izraz: 8.0+9.0-16.0+5.0
Prepoznao: 8.0+9.0-16.0+5.0
Poklapanje: True
Resenje: 6.0

Slika broj 5
Izraz: 626.0+321.0*7.0/3.0
Prepoznao: 626.0+321.0*7.0/3.0
Poklapanje: True
Resenje: 1375.0

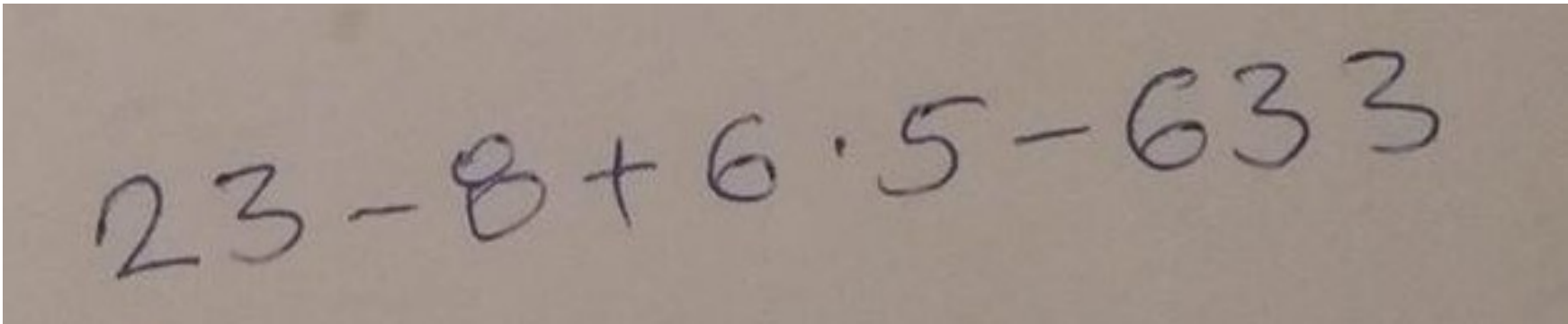
Slika broj 6
Izraz: 252.0+83.0*6.0
Prepoznao: 252.0+83.0*6.0
Poklapanje: False
Resenje: 780.0

Slika broj 7
Izraz: 23.0-8.0+6.0*5.0-633.0
Prepoznao: 23.0-8.0+6.0*5.0-633.0
Poklapanje: True
Resenje: -588.0

Internal console Console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8

Primer ulazne fotografije



23-8+6.5-633

Zaključak

- ◆ Rezultat prepoznavanja nad 11 fotografija koje je mreža trebalo da prepozna je dva nepotpuno prepoznata izraza u kojima je pogresila samo u dva znaka, što znači da je od preko 80 test slučajeva ona pogrešila dva broja, I to u prvom primeru gde je umesto 1 prepoznala 4 I u šestom gde je umesto 3 prepoznala 8.