



GilLevi / README.md

Last active 2 days ago

Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns

 README.md

## Gil Levi and Tal Hassner, Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns

Convolutional neural networks for emotion classification from facial images as described in the following work:

Gil Levi and Tal Hassner, Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns, Proc. ACM International Conference on Multimodal Interaction (ICMI), Seattle, Nov. 2015

Project page: [http://www.openu.ac.il/home/hassner/projects/cnn\\_emotions/](http://www.openu.ac.il/home/hassner/projects/cnn_emotions/)

If you find our models useful, please add suitable reference to our paper in your work.

gist\_id: 54aee1b8b0397721aa4b

### Emotion Classification CNN - RGB

caffemodel: VGG\_S\_rgb/EmotiW\_VGG\_S.caffemodel

caffemodel\_url: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_rgb/EmotiW\\_VGG\\_S.caffemodel](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_rgb/EmotiW_VGG_S.caffemodel)

mean\_file\_proto: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_rgb/mean.binaryproto](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_rgb/mean.binaryproto)

### Emotion Classification CNN - LBP

caffemodel: VGG\_S\_lbp/EmotiW\_VGG\_S.caffemodel

caffemodel\_url: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_lbp/EmotiW\\_VGG\\_S.caffemodel](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_lbp/EmotiW_VGG_S.caffemodel)

mean\_file\_proto: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_lbp/mean.binaryproto](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_lbp/mean.binaryproto)

### Emotion Classification CNN - Cyclic LBP

caffemodel: VGG\_S\_cyclic\_lbp/EmotiW\_VGG\_S.caffemodel

caffemodel\_url: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_cyclic\\_lbp/EmotiW\\_VGG\\_S.caffemodel](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp/EmotiW_VGG_S.caffemodel)

mean\_file\_proto: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_cyclic\\_lbp/mean.binaryproto](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp/mean.binaryproto)

### Emotion Classification CNN - Cyclic LBP-5

caffemodel: VGG\_S\_cyclic\_lbp\_5/EmotiW\_VGG\_S.caffemodel

caffemodel\_url: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_cyclic\\_lbp\\_5/EmotiW\\_VGG\\_S.caffemodel](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp_5/EmotiW_VGG_S.caffemodel)

mean\_file\_proto: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_cyclic\\_lbp\\_5/mean.binaryproto](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp_5/mean.binaryproto)

### Emotion Classification CNN - Cyclic LBP-10

caffemodel: VGG\_S\_cyclic\_lbp\_10/EmotiW\_VGG\_S.caffemodel

caffemodel\_url: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_cyclic\\_lbp\\_10/EmotiW\\_VGG\\_S.caffemodel](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp_10/EmotiW_VGG_S.caffemodel)

mean\_file\_proto: [https://dl.dropboxusercontent.com/u/38822310/demodir/VGG\\_S\\_cyclic\\_lbp\\_10/mean.binaryproto](https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp_10/mean.binaryproto)

Copyright 2015, Gil Levi and Tal Hassner

The SOFTWARE provided in this page is provided "as is", without any guarantee made as to its suitability or fitness for any particular use. It may contain bugs, so use of this tool is at your own risk. We take no responsibility for any damage of any sort that may unintentionally be caused through its use.

 **deploy.txt**

```
1  name: "CaffeNet"
2  input: "data"
3  input_dim: 1
4  input_dim: 3
5  input_dim: 224
6  input_dim: 224
7  layers {
8    name: "conv1"
9    type: CONVOLUTION
10   bottom: "data"
11   top: "conv1"
12   convolution_param {
13     num_output: 96
14     kernel_size: 7
15     stride: 2
16   }
17 }
18 layers {
19   name: "relu1"
20   type: RELU
21   bottom: "conv1"
22   top: "conv1"
23 }
24 layers {
25   name: "norm1"
26   type: LRN
27   bottom: "conv1"
28   top: "norm1"
29   lrn_param {
30     local_size: 5
31     alpha: 0.0005
32     beta: 0.75
33   }
34 }
35 layers {
36   name: "pool1"
37   type: POOLING
38   bottom: "norm1"
39   top: "pool1"
40   pooling_param {
41     pool: MAX
42     kernel_size: 3
43     stride: 3
44   }
45 }
46 layers {
47   name: "conv2"
48   type: CONVOLUTION
49   bottom: "pool1"
50   top: "conv2"
51   convolution_param {
52     num_output: 256
53     pad: 2
54     kernel_size: 5
55   }
56 }
57 layers {
58   name: "relu2"
59   type: RELU
60   bottom: "conv2"
61   top: "conv2"
62 }
63 layers {
64   name: "pool2"
65   type: POOLING
66   bottom: "conv2"
```

```
67     top: "pool2"
68     pooling_param {
69       pool: MAX
70       kernel_size: 2
71       stride: 2
72     }
73   }
74   layers {
75     name: "conv3"
76     type: CONVOLUTION
77     bottom: "pool2"
78     top: "conv3"
79     convolution_param {
80       num_output: 512
81       pad: 1
82       kernel_size: 3
83     }
84   }
85   layers {
86     name: "relu3"
87     type: RELU
88     bottom: "conv3"
89     top: "conv3"
90   }
91   layers {
92     name: "conv4"
93     type: CONVOLUTION
94     bottom: "conv3"
95     top: "conv4"
96     convolution_param {
97       num_output: 512
98       pad: 1
99       kernel_size: 3
100   }
101 }
102 layers {
103   name: "relu4"
104   type: RELU
105   bottom: "conv4"
106   top: "conv4"
107 }
108 layers {
109   name: "conv5"
110   type: CONVOLUTION
111   bottom: "conv4"
112   top: "conv5"
113   convolution_param {
114     num_output: 512
115     pad: 1
116     kernel_size: 3
117   }
118 }
119 layers {
120   name: "relu5"
121   type: RELU
122   bottom: "conv5"
123   top: "conv5"
124 }
125 layers {
126   name: "pool5"
127   type: POOLING
128   bottom: "conv5"
129   top: "pool5"
130   pooling_param {
131     pool: MAX
132     kernel_size: 3
133     stride: 3
134   }
135 }
136 layers {
137   name: "fc6"
138   type: INNER_PRODUCT
139   bottom: "pool5"
140   top: "fc6"
141   inner_product_param {
```

```
142     num_output: 4048
143   }
144 }
145 layers {
146   name: "relu6"
147   type: RELU
148   bottom: "fc6"
149   top: "fc6"
150 }
151 layers {
152   name: "drop6"
153   type: DROPOUT
154   bottom: "fc6"
155   top: "fc6"
156   dropout_param {
157     dropout_ratio: 0.5
158   }
159 }
160 layers {
161   name: "fc7"
162   type: INNER_PRODUCT
163   bottom: "fc6"
164   top: "fc7"
165   inner_product_param {
166     num_output: 4048
167   }
168 }
169 layers {
170   name: "relu7"
171   type: RELU
172   bottom: "fc7"
173   top: "fc7"
174 }
175 layers {
176   name: "drop7"
177   type: DROPOUT
178   bottom: "fc7"
179   top: "fc7"
180   dropout_param {
181     dropout_ratio: 0.5
182   }
183 }
184 layers {
185   name: "fc8_cat"
186   type: INNER_PRODUCT
187   bottom: "fc7"
188   top: "fc8"
189   inner_product_param {
190     num_output: 7
191   }
192 }
193 layers {
194   name: "prob"
195   type: SOFTMAX
196   bottom: "fc8"
197   top: "prob"
198 }
```

 [train\\_val.prototxt](#)

```
1  name: "CaffeNet"
2  layers {
3    name: "data"
4    type: DATA
5    top: "data"
6    top: "label"
7    data_param {
8      source: "/home/ubuntu/EmotiW/lmdb/train_lmdb"
9      backend: LMDB
10     batch_size: 30
11   }
12   transform_param {
13     crop_size: 224
14     mean_file: "/home/ubuntu/EmotiW/mean_image/mean.binaryproto"
15     mirror: true
```

```
16     }
17     include: { phase: TRAIN }
18 }
19 layers {
20     name: "data"
21     type: DATA
22     top: "data"
23     top: "label"
24     data_param {
25         source: "/home/ubuntu/EmotiW/lmdb/val_lmdb"
26         backend: LMDB
27         batch_size: 20
28     }
29     transform_param {
30         crop_size: 224
31         mean_file: "/home/ubuntu/EmotiW/mean_image/mean.binaryproto"
32         mirror: false
33     }
34     include: { phase: TEST }
35 }
36 layers {
37     name: "conv1"
38     type: CONVOLUTION
39     bottom: "data"
40     top: "conv1"
41     blobs_lr: 1
42     blobs_lr: 2
43     weight_decay: 1
44     weight_decay: 0
45     convolution_param {
46         num_output: 96
47         kernel_size: 7
48         stride: 2
49         weight_filler {
50             type: "gaussian"
51             std: 0.01
52         }
53         bias_filler {
54             type: "constant"
55             value: 0
56         }
57     }
58 }
59 layers {
60     name: "relu1"
61     type: RELU
62     bottom: "conv1"
63     top: "conv1"
64 }
65 layers {
66     name: "norm1"
67     type: LRN
68     bottom: "conv1"
69     top: "norm1"
70     lrn_param {
71         local_size: 5
72         alpha: 0.0005
73         beta: 0.75
74     }
75 }
76 layers {
77     name: "pool1"
78     type: POOLING
79     bottom: "norm1"
80     top: "pool1"
81     pooling_param {
82         pool: MAX
83         kernel_size: 3
84         stride: 3
85     }
86 }
87 layers {
88     name: "conv2"
89     type: CONVOLUTION
90     bottom: "pool1"
```

```


91   top: "conv2"
92   blobs_lr: 1
93   blobs_lr: 2
94   weight_decay: 1
95   weight_decay: 0
96   convolution_param {
97     num_output: 256
98     pad: 2
99     kernel_size: 5
100    weight_filler {
101      type: "gaussian"
102      std: 0.01
103    }
104    bias_filler {
105      type: "constant"
106      value: 1
107    }
108  }
109 }
110 layers {
111   name: "relu2"
112   type: RELU
113   bottom: "conv2"
114   top: "conv2"
115 }
116 layers {
117   name: "pool2"
118   type: POOLING
119   bottom: "conv2"
120   top: "pool2"
121   pooling_param {
122     pool: MAX
123     kernel_size: 2
124     stride: 2
125   }
126 }
127 layers {
128   name: "conv3"
129   type: CONVOLUTION
130   bottom: "pool2"
131   top: "conv3"
132   blobs_lr: 1
133   blobs_lr: 2
134   weight_decay: 1
135   weight_decay: 0
136   convolution_param {
137     num_output: 512
138     pad: 1
139     kernel_size: 3
140     weight_filler {
141       type: "gaussian"
142       std: 0.01
143     }
144     bias_filler {
145       type: "constant"
146       value: 0
147     }
148   }
149 }
150 layers {
151   name: "relu3"
152   type: RELU
153   bottom: "conv3"
154   top: "conv3"
155 }
156 layers {
157   name: "conv4"
158   type: CONVOLUTION
159   bottom: "conv3"
160   top: "conv4"
161   blobs_lr: 1
162   blobs_lr: 2
163   weight_decay: 1
164   weight_decay: 0
165   convolution_param {
```

```
166     num_output: 512
167     pad: 1
168     kernel_size: 3
169     weight_filler {
170       type: "gaussian"
171       std: 0.01
172     }
173     bias_filler {
174       type: "constant"
175       value: 1
176     }
177   }
178 }
179 layers {
180   name: "relu4"
181   type: RELU
182   bottom: "conv4"
183   top: "conv4"
184 }
185 layers {
186   name: "conv5"
187   type: CONVOLUTION
188   bottom: "conv4"
189   top: "conv5"
190   blobs_lr: 1
191   blobs_lr: 2
192   weight_decay: 1
193   weight_decay: 0
194   convolution_param {
195     num_output: 512
196     pad: 1
197     kernel_size: 3
198     weight_filler {
199       type: "gaussian"
200       std: 0.01
201     }
202     bias_filler {
203       type: "constant"
204       value: 1
205     }
206   }
207 }
208 layers {
209   name: "relu5"
210   type: RELU
211   bottom: "conv5"
212   top: "conv5"
213 }
214 layers {
215   name: "pool5"
216   type: POOLING
217   bottom: "conv5"
218   top: "pool5"
219   pooling_param {
220     pool: MAX
221     kernel_size: 3
222     stride: 3
223   }
224 }
225 layers {
226   name: "fc6"
227   type: INNER_PRODUCT
228   bottom: "pool5"
229   top: "fc6"
230   blobs_lr: 1
231   blobs_lr: 2
232   weight_decay: 1
233   weight_decay: 0
234   inner_product_param {
235     num_output: 4048
236     weight_filler {
237       type: "gaussian"
238       std: 0.005
239     }
240     bias_filler {
```

```
241         type: "constant"
242         value: 1
243     }
244 }
245 }
246 layers {
247     name: "relu6"
248     type: RELU
249     bottom: "fc6"
250     top: "fc6"
251 }
252 layers {
253     name: "drop6"
254     type: DROPOUT
255     bottom: "fc6"
256     top: "fc6"
257     dropout_param {
258         dropout_ratio: 0.7
259     }
260 }
261 layers {
262     name: "fc7"
263     type: INNER_PRODUCT
264     bottom: "fc6"
265     top: "fc7"
266     blobs_lr: 9
267     blobs_lr: 18
268     weight_decay: 1
269     weight_decay: 0
270     inner_product_param {
271         num_output: 4048
272         weight_filler {
273             type: "gaussian"
274             std: 0.005
275         }
276         bias_filler {
277             type: "constant"
278             value: 1
279         }
280     }
281 }
282 layers {
283     name: "relu7"
284     type: RELU
285     bottom: "fc7"
286     top: "fc7"
287 }
288 layers {
289     name: "drop7"
290     type: DROPOUT
291     bottom: "fc7"
292     top: "fc7"
293     dropout_param {
294         dropout_ratio: 0.7
295     }
296 }
297 layers {
298     name: "fc8_cat"
299     type: INNER_PRODUCT
300     bottom: "fc7"
301     top: "fc8"
302     blobs_lr: 12
303     blobs_lr: 24
304     weight_decay: 1
305     weight_decay: 0
306     inner_product_param {
307         num_output: 7
308         weight_filler {
309             type: "gaussian"
310             std: 0.01
311         }
312         bias_filler {
313             type: "constant"
314             value: 0
315         }
316     }
317 }
```




```
316     }
317 }
318 layers {
319     name: "accuracy"
320     type: ACCURACY
321     bottom: "fc8"
322     bottom: "label"
323     top: "accuracy"
324     include: { phase: TEST }
325 }
326 layers {
327     name: "loss"
328     type: SOFTMAX_LOSS
329     bottom: "fc8"
330     bottom: "label"
331     top: "loss"
332 }
```




Gzzgz commented on 16 Jun 2016 • edited

---OK I has finished it.


Can you help me ? TKS






Gzzgz commented on 25 Jun 2016

Other problem. I finded the deploy file is not fit to the CNN-LBP CNN-Cyclic-LBP CNN-Cyclic-LBP-5 and CNN-Cyclic-LBP-10 . Can you help me ? THX



singarajus commented on 27 Jul 2016


Gil Levi, your jupyter notebook gives categories = [ 'Angry' , 'Disgust' , 'Fear' , 'Happy' , 'Neutral' , 'Sad' , 'Surprise' ]. Are these the same for each model you have published here? I am using VGG\_S\_rgb/EmotiW\_VGG\_S.caffemodel above. I wanted to confirm what labels are being used here. Would appreciate if you put the labels along with it.



GilLevi commented on 30 Jul 2016

Owner


Hi Gzzg, sorry for the late response (I didn't get an email about your comment). The deploy file attached should work with all models. What seems to be the problem?



GilLevi commented on 30 Jul 2016

Owner

Hi singarajus , indeed the labels are the same for all models.



adramesh commented on 31 Aug 2016

Hi Gil

I downloaded the demo dir and when I go through the demo script even i see the error reported above:

```
I0830 17:48:38.695304 32182 net.cpp:761] Ignoring source layer loss
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nutanix/workspace/caffe/python/caffe/classifier.py", line 34, in __init__
    self.transformer.set_mean(in_, mean)
  File "/home/nutanix/workspace/caffe/python/caffe/io.py", line 259, in set_mean
    raise ValueError('Mean shape incompatible with input shape.')
ValueError: Mean shape incompatible with input shape.
```

any ideas how can I fix this?

Thanks!

Aditya



NoorShaker commented on 13 Sep 2016 • edited

Hi Aditya,

Try

```
mean = caffe.io.blobproto_to_array(a)[0]
mean=mean.mean(1).mean(1)
```

This worked for me.

Good luck.



GilLevi commented on 25 Oct 2016

Owner

Hi Aditya,

The issue is explained and fixed here (update from July 15th, 2015):  
[http://www.openu.ac.il/home/hassner/projects/cnn\\_agegender/](http://www.openu.ac.il/home/hassner/projects/cnn_agegender/)



ilyamaslo commented on 31 Oct 2016 • edited

Hi

I tried this attached deploy.prototxt for all LBP models and had same problem as Gzzgz (kernel died). It seems logical since RGB model and LBP models had different size.

from jupyter log:  
F1031 17:28:42.965265 7498 net.cpp:765] Cannot copy param 0 weights from layer 'fc6'; shape mismatch. Source param shape is 4096 18432 (75497472); target param shape is 4048 25088 (101556224). To learn this layer's parameters from scratch rather than copying from a saved net, rename the layer.  
\*\*\* Check failure stack trace: \*\*\*



GuitarZhang commented on 12 Dec 2016

Hi, GilLevi

Does the align step same as [http://www.openu.ac.il/home/hassner/projects/cnn\\_agegender/](http://www.openu.ac.il/home/hassner/projects/cnn_agegender/)?

Thanks.



GilLevi commented on 3 Jan

Owner

Sorry for the late response, I didn't get notifications for some reason.

@ilyamaslo , try replacing the size of fc6 from 4048 to 4096.

@GuitarZhang , no. In this project the aligned faces were provided by the challenge authors.



rahulkulhalli commented on 1 Feb

Hey @GilLevi,

I downloaded your RGB caffemodel, along with the deploy.prototxt that you've attached, and it's working really well!

I had a few requests, if you don't mind:

1. Could you please provide the labels.txt? I don't know the exact order of the labels, so it'll be great if you could provide that.
2. Could you also provide the solver.prototxt file? I'd like to fine-tune the caffemodel to my own dataset, if you don't mind.

Thanks in advance! :)



user65432 commented on 7 Feb

Hi @GilLevi,

I downloaded RGB caffemodel, and execute it as you showed at python notebook example. I works really good at happy, fear and disgusted images but it can't predict sad images. I used KDEP database as a testing database. It can detect all emotions except sad and suprise. Can you give an example with any sad or suprise person picture like the angry one on the python notebook example?

Thanks.



GilLevi commented on 12 Feb • edited

Owner

Hi @rahulkulhalli,

Thank you for your interest in our work.

The solver is:  
net: "/home/ubuntu/EmotiW/VGG\_S\_David/train\_val.prototxt"  
test\_iter: 1000  
test\_interval: 1000  
base\_lr: 0.001  
lr\_policy: "step"  
gamma: 0.1  
#stepsize: 20000  
stepsize: 2000000  
display: 100  
max\_iter: 400000  
momentum: 0.9  
weight\_decay: 0.0005  
snapshot: 1000  
snapshot\_prefix: "EmotiW\_VGG\_S"  
solver\_mode: GPU  
device\_id:0

There is no labels file, the labels are just :[ 'Angry' , 'Disgust' , 'Fear' , 'Happy' , 'Neutral' , 'Sad' , 'Surprise'] in that order.

Best,  
Gil



GilLevi commented on 12 Feb

Owner

Hi @user65432,

Thank you for your interest in our work.

I've attached an example of an angry face and a sad face which are classified correctly by our model. Perhaps you can get better results by finetuning our network on the KDEF dataset.



zhanglaplace commented on 13 Feb

@user65432,can you share me KDEF dataset? I download it always 0kb.



user65432 commented on 13 Feb

@zhanglaplace, sure. I tried to share it from there but i can't because it's bigger than 10MB.



zhanglaplace commented on 14 Feb

@user65432 , my email is [laplace@gmail.com](mailto:laplace@gmail.com) , can you email me ? thx



shinchanyox commented on 15 Feb • edited

Hi GilLevi ,  
I am using the Emotion Classification CNN - RGB model configured on EC2 AWS server(preconfigured with caffe and CUDA).The model is giving the same emotion for every input image.  
categories[prediction.argmax()] always gives categories[0] and so I always get the 0th index emotion for every input.

<https://cloud.githubusercontent.com/assets/6024900/22856133/3a70dcec-f094-11e6-9bf2-7dd793ffa605.png>

This is the image we used as our input.The dimensions of the image we used was 256\*256.



**GilLevi** commented on 17 Feb

Owner

Hi **@shinchanyox**, can you pleas attache the images that you used and gave the same emotion each time?

Best,  
Gil



**shinchanyox** commented on 17 Feb

sir , i was using incorrect images , now it is working fine . Can you please suggest a method to connect to your model on ec2 via android application ?



**GilLevi** commented on 20 Feb

Owner

Hi **@shinchanyox**, I have no experience in android programming so I really don't know.

Best,  
Gil



**user65432** commented on 11 Mar

Hi GilLevi,  
I'm still trying to fine tune your CNN :). Could you provide "train\_val.prototxt" , if you don't mind ?  
Thanks.



**GilLevi** commented on 20 Mar

Owner

Hi **@user65432**,

Thanks you for your interest in our project, I uploaded the "train\_val.prototxt".

Best,  
Gil



**iftekharanam** commented on 26 Apr

Hi GilLevi,  
Thanks for sharing the models and prototxt files. To compare the results using different pretrained models, I tried VGG\_S\_rgb and VGG\_S\_cyclic\_lbp\_10. While the first one finished finetuning with the new data, the latter model shows the following error:

```
F0425 11:27:24.064520 62685 net.cpp:774] Cannot copy param 0 weights from layer 'fc6'; shape mismatch. Source param shape is 4096 18432 (75497472); target param shape is 4048 25088 (101556224). To learn this layer's parameters from scratch rather than copying from a saved net, rename the layer.
```

I looked up online and found that that the difference in feature map size could be the reason ([link](#))  
Would you please let me know what are the differences between the above two models?  
Thank you again.



**dayinji** commented on 27 Apr

Hi GilLevi,  
The deploy.txt above is not suit for [ Emotion Classification CNN - RGB ], even I change the [ num\_output ] of [ fc6 ] from 4048 to 4096. I hope you can upload a correct deploy.txt that suit for [ Emotion Classification CNN - RGB ]. Anyway, Thank you for sharing this nice work! Hope for your response. : )



**vishal733** commented on 7 Jun • edited

Hi GilLevi,  
Is it possible for you to share a complete Python Caffe code which can obtain emotion for an input image of size 224x224

I'm getting the following error while trying out the RGB model:

Cannot copy param 0 weights from layer 'conv3'; shape mismatch. Source param shape is 384 256 3 3 (884736); target param shape is 512 256 3 3 (1179648). To learn this layer's parameters from scratch rather than copying from a saved net, rename the layer.

And here's my current python code:

```
import numpy as np
import sys
import caffe

MODEL_FILE = '/opt/caffe/caffe/models/cnn_emotions/deploy.prototxt'
PRETRAINED = '/opt/caffe/caffe/models/agenet/age_net.caffemodel'

caffe.set_mode_cpu()

net = caffe.Classifier(MODEL_FILE, PRETRAINED,
                      mean=np.load('/opt/caffe/caffe/models/cnn_emotions/VGG_S_rgb/mean.npy').mean(1).mean(1),
                      channel_swap=(2,1,0),
                      raw_scale=255,
                      image_dims=(224, 224))
print "successfully loaded classifier"

IMAGE_FILE = '/opt/caffe/caffe/models/vgg_face/ak.png'
input_image = caffe.io.load_image(IMAGE_FILE)
pred = net.predict([input_image])
print pred
```

And in case direct assistance on Python is not possible, I'd appreciate if you can share the caffe command line for running emotion RGB model.











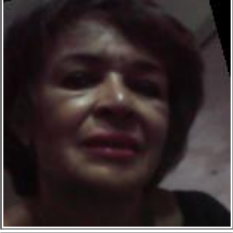


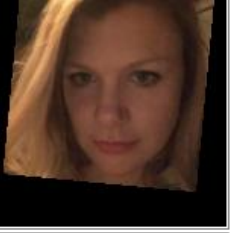
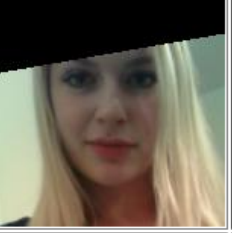


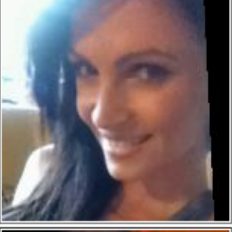




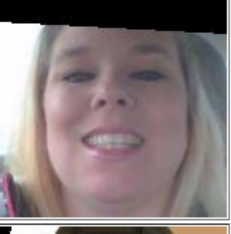




**vinnitu** commented 14 days ago

I tested model on photos from my dataset like in attach (150x150 each)



but I see many not correct prediction


















Sad	Happy	Angry	Surprise	Fear	Neutral	Disgust
						
						
						
						
						

Tell me please - when I am wrong?  
Maybe I need other size of pictures?  
Or it must be only face rect?



vinnitu commented 14 days ago

a make 224x224 photos but result bad again...

Sad	Angry	Happy	Neutral
			
			
			
			

why?



GilLevi commented 2 days ago

Owner

Hi @iftekharanam @dayinji

Thank you for your interest in our work. I think there is a typo there and for the fc6 layer you need to change the value of 4048 to 4096.

@dayinji, what is the error you got?

Gil



GilLevi commented 2 days ago

Owner

Hi @vishal733,

Thank you for your interest in our work. That is strange, are you sure you are using the RGB network?

You can see here and example of usage in python:

[http://nbviewer.jupyter.org/urls/dl.dropboxusercontent.com/u/38822310/DemoDir/EmotiW\\_Demo.ipynb](http://nbviewer.jupyter.org/urls/dl.dropboxusercontent.com/u/38822310/DemoDir/EmotiW_Demo.ipynb)

Gil



GilLevi commented 2 days ago

Owner

Hi @vinnitu,

Thank you for your interest in our work. Can you please upload your code?

Best,  
Gil



0xPr0xy commented 2 days ago

@GilLevi I also seem to get inaccurate results with the provided RGB model.

Changing the fc6 layer num\_output to 4096 does not work:

```
RuntimeError: Caffe model error in layer 'fc6' of type 'Inner Product': 'num_output' (4069) does not match the first dimension of the weight matrix (4048).
```



0xPr0xy commented 2 days ago

It also seems to return pretty confident (above 50%) predictions for images without any face in it. Like a picture of a floor.



vinnitu commented 2 days ago • edited

Hi @GilLevi,

Thank you for reply. This is my code

```
#!/usr/bin/env python

import os
import numpy as np
import sys
import caffe

DEMO_DIR = 'models'
categories = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

cur_net_dir = 'VGG_S_lbp'

mean_filename = os.path.join(DEMO_DIR, cur_net_dir, 'mean.binaryproto')
proto_data = open(mean_filename, "rb").read()
a = caffe.io.caffe_pb2.BlobProto.FromString(proto_data)
mean = caffe.io.blobproto_to_array(a)[0]

net_pretrained = os.path.join(DEMO_DIR, cur_net_dir, 'EmotiW_VGG_S.caffemodel')
net_model_file = os.path.join(DEMO_DIR, cur_net_dir, 'deploy.prototxt')
VGG_S_Net = caffe.Classifier(net_model_file, net_pretrained,
                             mean = mean,
                             channel_swap = (2, 1, 0),
                             raw_scale = 255,
                             image_dims = (256, 256))

input_image = caffe.io.load_image('./image.jpg')
prediction = VGG_S_Net.predict([input_image], oversample=False)
print categories[prediction.argmax()]
```