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/

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

mean_file_proto: 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

mean_file_proto: 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

mean_file_proto: 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

mean_file_proto: 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

mean_file_proto: https://dl.dropboxusercontent.com/u/38822310/demodir/VGG_S_cyclic_lbp_10/mean.binaryproto

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

```
name: "CaffeNet"
input: "data"
input_dim: 1
input_dim: 3
input_dim: 224
input_dim: 224
layers {
  name: "conv1"
  type: CONVOLUTION
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 96
    kernel_size: 7
    stride: 2
  }
}
layers {
  name: "relu1"
  type: RELU
  bottom: "conv1"
  top: "conv1"
}
layers {
  name: "norm1"
  type: LRN
  bottom: "conv1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0005
    beta: 0.75
  }
}
layers {
  name: "pool1"
  type: POOLING
  bottom: "norm1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 3
  }
}
layers {
  name: "conv2"
  type: CONVOLUTION
  bottom: "pool1"
  top: "conv2"
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
  }
}
layers {
  name: "relu2"
  type: RELU
  bottom: "conv2"
  top: "conv2"
}
layers {
  name: "pool2"
  type: POOLING
  bottom: "conv2"
```

```
    top: "pool2"
    pooling_param {
      pool: MAX
      kernel_size: 2
      stride: 2
    }
  }
  layers {
    name: "conv3"
    type: CONVOLUTION
    bottom: "pool2"
    top: "conv3"
    convolution_param {
      num_output: 512
      pad: 1
      kernel_size: 3
    }
  }
  layers {
    name: "relu3"
    type: RELU
    bottom: "conv3"
    top: "conv3"
  }
  layers {
    name: "conv4"
    type: CONVOLUTION
    bottom: "conv3"
    top: "conv4"
    convolution_param {
      num_output: 512
      pad: 1
      kernel_size: 3
    }
  }
  layers {
    name: "relu4"
    type: RELU
    bottom: "conv4"
    top: "conv4"
  }
  layers {
    name: "conv5"
    type: CONVOLUTION
    bottom: "conv4"
    top: "conv5"
    convolution_param {
      num_output: 512
      pad: 1
      kernel_size: 3
    }
  }
  layers {
    name: "relu5"
    type: RELU
    bottom: "conv5"
    top: "conv5"
  }
  layers {
    name: "pool5"
    type: POOLING
    bottom: "conv5"
    top: "pool5"
    pooling_param {
      pool: MAX
      kernel_size: 3
      stride: 3
    }
  }
  layers {
    name: "fc6"
    type: INNER_PRODUCT
    bottom: "pool5"
    top: "fc6"
    inner_product_param {
```

```
    num_output: 4048
  }
}
layers {
  name: "relu6"
  type: RELU
  bottom: "fc6"
  top: "fc6"
}
layers {
  name: "drop6"
  type: DROPOUT
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layers {
  name: "fc7"
  type: INNER_PRODUCT
  bottom: "fc6"
  top: "fc7"
  inner_product_param {
    num_output: 4048
  }
}
layers {
  name: "relu7"
  type: RELU
  bottom: "fc7"
  top: "fc7"
}
layers {
  name: "drop7"
  type: DROPOUT
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layers {
  name: "fc8_cat"
  type: INNER_PRODUCT
  bottom: "fc7"
  top: "fc8"
  inner_product_param {
    num_output: 7
  }
}
layers {
  name: "prob"
  type: SOFTMAX
  bottom: "fc8"
  top: "prob"
}
```

train_val.prototxt

```
name: "CaffeNet"
layers {
  name: "data"
  type: DATA
  top: "data"
  top: "label"
  data_param {
    source: "/home/ubuntu/EmotiW/lmdb/train_lmdb"
    backend: LMDB
    batch_size: 30
  }
  transform_param {
    crop_size: 224
    mean_file: "/home/ubuntu/EmotiW/mean_image/mean.binaryproto"
    mirror: true
```

```
    }
    include: { phase: TRAIN }
  }
  layers {
    name: "data"
    type: DATA
    top: "data"
    top: "label"
    data_param {
      source: "/home/ubuntu/EmotiW/lmdb/val_lmdb"
      backend: LMDB
      batch_size: 20
    }
    transform_param {
      crop_size: 224
      mean_file: "/home/ubuntu/EmotiW/mean_image/mean.binaryproto"
      mirror: false
    }
    include: { phase: TEST }
  }
  layers {
    name: "conv1"
    type: CONVOLUTION
    bottom: "data"
    top: "conv1"
    blobs_lr: 1
    blobs_lr: 2
    weight_decay: 1
    weight_decay: 0
    convolution_param {
      num_output: 96
      kernel_size: 7
      stride: 2
      weight_filler {
        type: "gaussian"
        std: 0.01
      }
      bias_filler {
        type: "constant"
        value: 0
      }
    }
  }
  layers {
    name: "relu1"
    type: RELU
    bottom: "conv1"
    top: "conv1"
  }
  layers {
    name: "norm1"
    type: LRN
    bottom: "conv1"
    top: "norm1"
    lrn_param {
      local_size: 5
      alpha: 0.0005
      beta: 0.75
    }
  }
  layers {
    name: "pool1"
    type: POOLING
    bottom: "norm1"
    top: "pool1"
    pooling_param {
      pool: MAX
      kernel_size: 3
      stride: 3
    }
  }
  layers {
    name: "conv2"
    type: CONVOLUTION
    bottom: "pool1"
```

```
    top: "conv2"
    blobs_lr: 1
    blobs_lr: 2
    weight_decay: 1
    weight_decay: 0
    convolution_param {
      num_output: 256
      pad: 2
      kernel_size: 5
      weight_filler {
        type: "gaussian"
        std: 0.01
      }
      bias_filler {
        type: "constant"
        value: 1
      }
    }
  }
  layers {
    name: "relu2"
    type: RELU
    bottom: "conv2"
    top: "conv2"
  }
  layers {
    name: "pool2"
    type: POOLING
    bottom: "conv2"
    top: "pool2"
    pooling_param {
      pool: MAX
      kernel_size: 2
      stride: 2
    }
  }
  layers {
    name: "conv3"
    type: CONVOLUTION
    bottom: "pool2"
    top: "conv3"
    blobs_lr: 1
    blobs_lr: 2
    weight_decay: 1
    weight_decay: 0
    convolution_param {
      num_output: 512
      pad: 1
      kernel_size: 3
      weight_filler {
        type: "gaussian"
        std: 0.01
      }
      bias_filler {
        type: "constant"
        value: 0
      }
    }
  }
  layers {
    name: "relu3"
    type: RELU
    bottom: "conv3"
    top: "conv3"
  }
  layers {
    name: "conv4"
    type: CONVOLUTION
    bottom: "conv3"
    top: "conv4"
    blobs_lr: 1
    blobs_lr: 2
    weight_decay: 1
    weight_decay: 0
    convolution_param {
```

```
    num_output: 512
    pad: 1
    kernel_size: 3
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layers {
  name: "relu4"
  type: RELU
  bottom: "conv4"
  top: "conv4"
}
layers {
  name: "conv5"
  type: CONVOLUTION
  bottom: "conv4"
  top: "conv5"
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layers {
  name: "relu5"
  type: RELU
  bottom: "conv5"
  top: "conv5"
}
layers {
  name: "pool5"
  type: POOLING
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 3
  }
}
layers {
  name: "fc6"
  type: INNER_PRODUCT
  bottom: "pool5"
  top: "fc6"
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 4048
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
```

```
      type: "constant"
      value: 1
    }
  }
}
layers {
  name: "relu6"
  type: RELU
  bottom: "fc6"
  top: "fc6"
}
layers {
  name: "drop6"
  type: DROPOUT
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.7
  }
}
layers {
  name: "fc7"
  type: INNER_PRODUCT
  bottom: "fc6"
  top: "fc7"
  blobs_lr: 9
  blobs_lr: 18
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 4048
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layers {
  name: "relu7"
  type: RELU
  bottom: "fc7"
  top: "fc7"
}
layers {
  name: "drop7"
  type: DROPOUT
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.7
  }
}
layers {
  name: "fc8_cat"
  type: INNER_PRODUCT
  bottom: "fc7"
  top: "fc8"
  blobs_lr: 12
  blobs_lr: 24
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 7
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
```

```
    }
  }
  layers {
    name: "accuracy"
    type: ACCURACY
    bottom: "fc8"
    bottom: "label"
    top: "accuracy"
    include: { phase: TEST }
  }
  layers {
    name: "loss"
    type: SOFTMAX_LOSS
    bottom: "fc8"
    bottom: "label"
    top: "loss"
  }
```

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/

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/?
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 KDEF 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 lapcace@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:

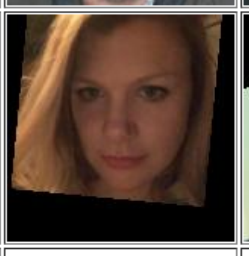
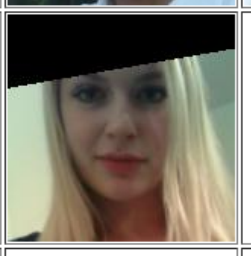
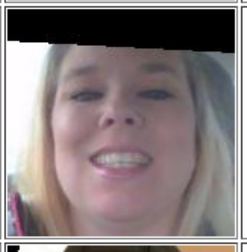
```
import numpy as np
import sys
import caffe

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)

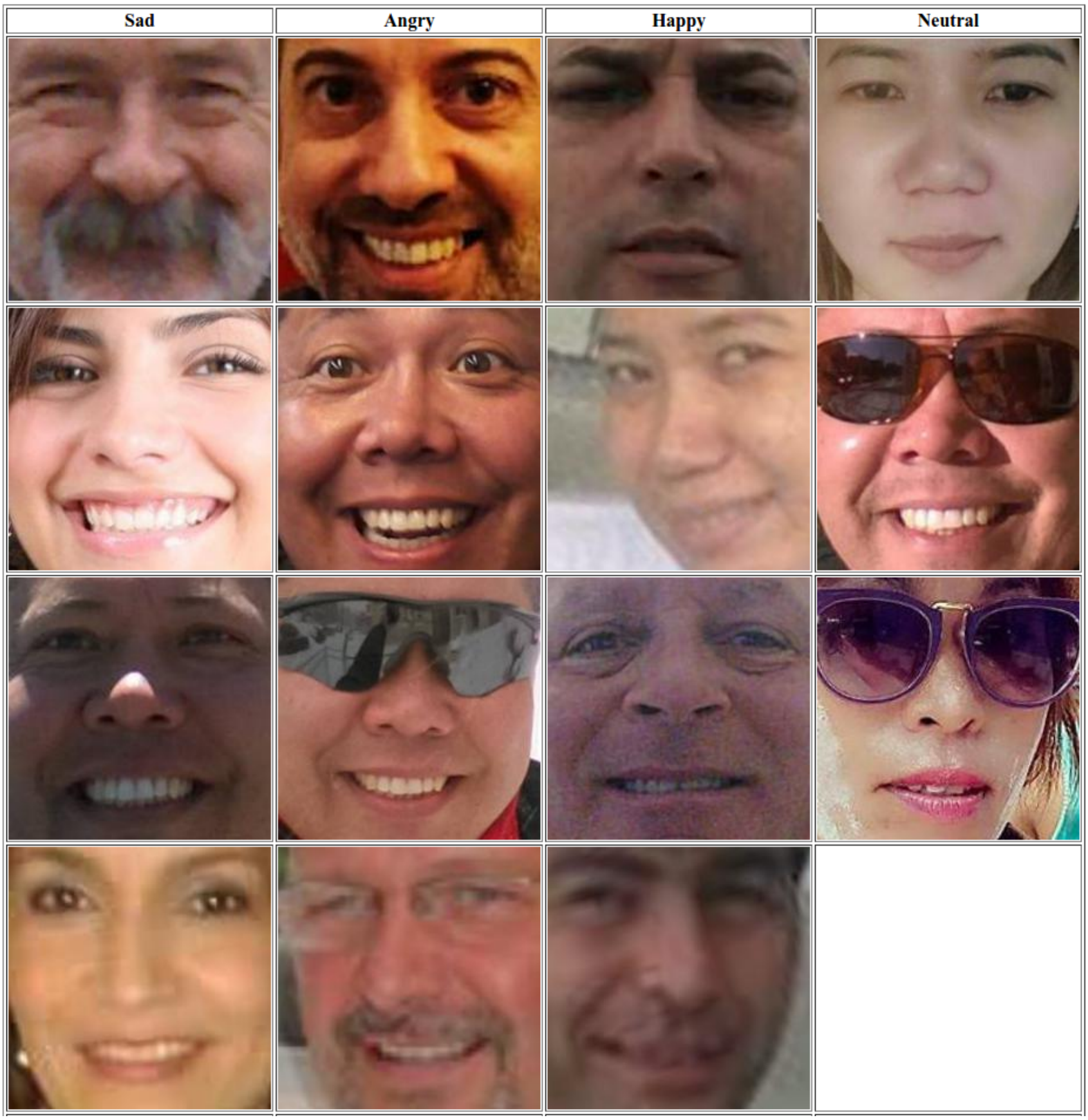
but I see many not correct prediction

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...



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

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()]
```