

HW 5 Rubrics

Total: 5 points

Late Penalty: 0.5/day, after 29th April 2020 11:59 pm

Files have not been zipped penalty: 0.5

Q1: 1 point

0.5 pt for submitting weights.h5 file

0.5 pt for submitting a screengrab of any dense layer

The screenshot shows a file explorer window with the file 'weights.h5' selected. The file structure is as follows:

- weights.h5
 - activation_26
 - activation_27
 - activation_28
 - activation_29
 - activation_30
 - conv2d_16
 - conv2d_16
 - bias:0
 - kernel:0
 - conv2d_17
 - conv2d_17
 - bias:0
 - kernel:0
 - conv2d_18
 - conv2d_18
 - bias:0
 - kernel:0
 - dense_11
 - dense_11
 - bias:0
 - kernel:0
 - dense_12
 - dense_12
 - bias:0
 - kernel:0
 - dropout_6
 - flatten_6
 - max_pooling2d_16
 - max_pooling2d_17
 - max_pooling2d_18

The table view for the 'kernel:0' at '/dense_12/dense_12/' is shown below:

	0
0	-0.08492154
1	-0.0132735...
2	0.1943794
3	0.0280992...
4	-0.15635538
5	-0.1305849
6	-0.08386392
7	-0.10359289
8	-0.0450458...
9	0.0729683...
10	0.0275401...
11	0.07731976
12	0.04825445
13	-0.08906876
14	0.0617936...
15	-0.0649955
16	0.06401146
17	-0.0158660...
18	-0.06812193
19	-0.0553165...
20	0.0810956
21	-0.08057273
22	-0.05910004
23	0.08653336
24	-0.24019071
25	0.0922715...
26	-0.06354873
27	0.07198966
28	-0.1000175...
29	-0.06385387
30	-0.06385
31	-0.0701839
32	0.20583776
33	-0.0626298...
34	-0.12051652
35	0.15784958
36	0.07647735
37	-0.0947011...

The screengrabs should show the expanded results of any dense layer below weights.h5

-0.5 points if weights.h5 is not submitted

-0.5 pt if the screenshot is not submitted

Q2: 2 points

1 pt for a screengrab showing correct classification. The values with a 20% range are accepted. So, 0.8 and 0.2 are fine.

0.5 pt each for good_cat and good_dog images



colab classify.ipynb ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
from keras.preprocessing import image

myPic = '/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/data/live/good_cat.jpg'
test_image= image.load_img(myPic, target_size = (img_width, img_height))
test_image = image.img_to_array(test_image)
test_image = test_image.reshape(input_shape)
test_image = numpy.expand_dims(test_image, axis = 0)
result = model.predict(test_image,verbose=0)
print(result[0][0])

myPic2 = '/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/data/live/good_dog.jpg'
#myPic2 = '/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/data/live/corgi.jpg'
test_image2= image.load_img(myPic2, target_size = (img_width, img_height))
test_image2 = image.img_to_array(test_image2)
test_image2 = test_image2.reshape(input_shape)
test_image2 = numpy.expand_dims(test_image2, axis = 0)
result = model.predict(test_image2,verbose=0)
print(result[0][0])
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

0.0
1.0

[]

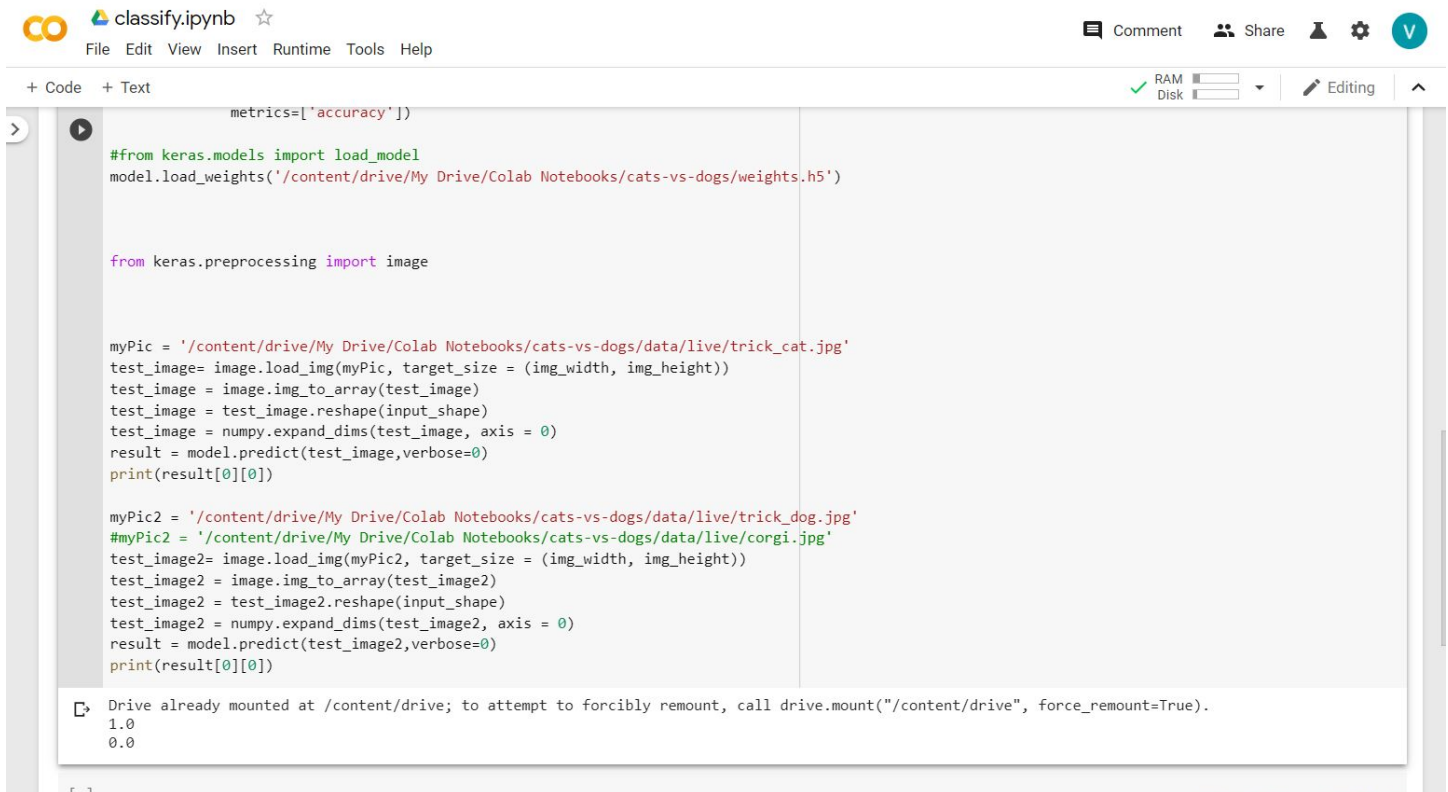
-1 pt if the classification is wrong or screenshot is not submitted

-0.5 pt each for wrong dog and cat images

Q3: 2 points

1 pt for a screengrab showing misclassification. The values with a 20% range are accepted. So, 0.8 and 0.2 are fine.

0.5 pt each for trick_cat and trick_dog images



```
metrics=['accuracy'])

#from keras.models import load_model
model.load_weights('/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/weights.h5')

from keras.preprocessing import image

myPic = '/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/data/live/trick_cat.jpg'
test_image= image.load_img(myPic, target_size = (img_width, img_height))
test_image = image.img_to_array(test_image)
test_image = test_image.reshape(input_shape)
test_image = numpy.expand_dims(test_image, axis = 0)
result = model.predict(test_image,verbose=0)
print(result[0][0])

myPic2 = '/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/data/live/trick_dog.jpg'
#myPic2 = '/content/drive/My Drive/Colab Notebooks/cats-vs-dogs/data/live/corgi.jpg'
test_image2= image.load_img(myPic2, target_size = (img_width, img_height))
test_image2 = image.img_to_array(test_image2)
test_image2 = test_image2.reshape(input_shape)
test_image2 = numpy.expand_dims(test_image2, axis = 0)
result = model.predict(test_image2,verbose=0)
print(result[0][0])
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

1.0
0.0



-1 pt if the classification is wrong or screenshot is not submitted

-0.5 pt each for wrong cat and dog images

Bonus Question: No points

The dataset should contain at least 300 images

Screengrab of correct classification

Readme.txt file containing the changes made to the code

Readme file should contain the following changes:

1) `model.add(Dense(3))`

Indicates the dimensionality of the output space. Our output space must be of the form [0, 0, 1]
In other words, they can be considered as the output nodes for each possible class.

2) `model.add(Activation('softmax')) :`

It is the last layer activation function, it returns an array of 3 probability scores summing to 1.
The sigmoid function is used for the two-class classification, whereas the softmax function is used for the multiclass classification.

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

Since we are dealing with multiple classes (non-binary), we change the loss function from `binary_crossentropy` to `categorical_crossentropy`

4) `train_generator = train_datagen.flow_from_directory(train_data_dir, target_size=(img_width, img_height), batch_size=batch_size, class_mode='categorical')`

Since we are trying to train the neural network with multiple classes (non-binary), we are using the categorical class mode instead of the binary class mode.

5) `validation_generator = test_datagen.flow_from_directory(`

`validation_data_dir,`

`target_size=(img_width, img_height),`

`batch_size=batch_size,`

`class_mode='categorical')`

Since we are trying to test the neural network with multiple classes (non-binary), we are using the categorical class mode instead of binary class mode.