HW5: assigned 11/24, due 12/2 by 11:59 PM

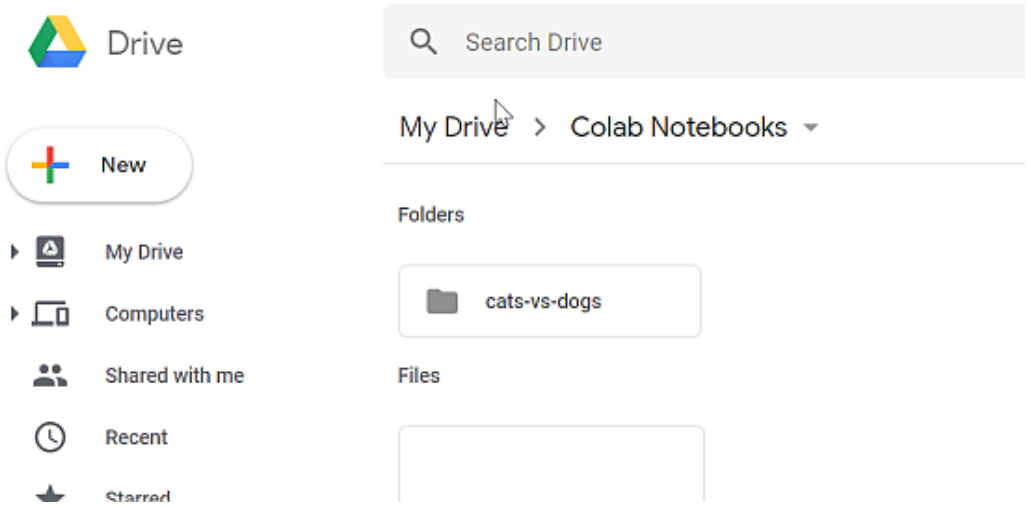
Total points: 6

This last hw is on **machine learning!** It is data-related, after all...

Here is a summary of what you'll do: on Google's Colab (https://colab.research.google.com/), train a neural network on differentiating between a cat pic and dog pic, then use the trained network to classify a new (cat-like or dog-like) pic into a cat or dog. This is a soup-to-nuts (start to finish) assignment that will get your feet wet (or plunge you in!), doing ML - a VERY valuable skill - training a self-driving car would involve much more complexity, but would be based on the same workflow. Below are the steps.

1. Use your GMail/GDrive account to log in, go to https://drive.google.com/ (https://drive.google.com/), click on the '+ New' button at the top left of the page, look for the 'Colab' app [after + New, click on More >, then + Connect more apps] and connect it - this will make the app [which connects to the mighty Google Cloud on the other end!] be able to access (read, write) files and folders in your GDrive.

2. You'll notice that the above step created a folder called Colab Notebooks, inside your GDrive - this is good, because we can keep Colab-related things nicely organized inside that folder. Within the Colab Notebooks subdir/folder, create a folder called cats-vs-dogs, for the hw:



Now we need DATA [images of cats and dogs] for training and validation, and scripts for training+validation and classifying.

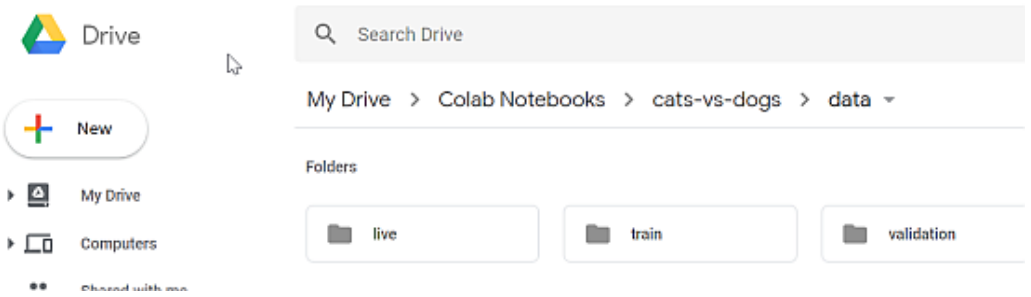
3. Download this (data/data.zip) .zip data file, unzip it. You'll see this structure:

```
data/
  live/
  train/
    cats/
    dogs/
  validation/
    cats/
    dogs/
```

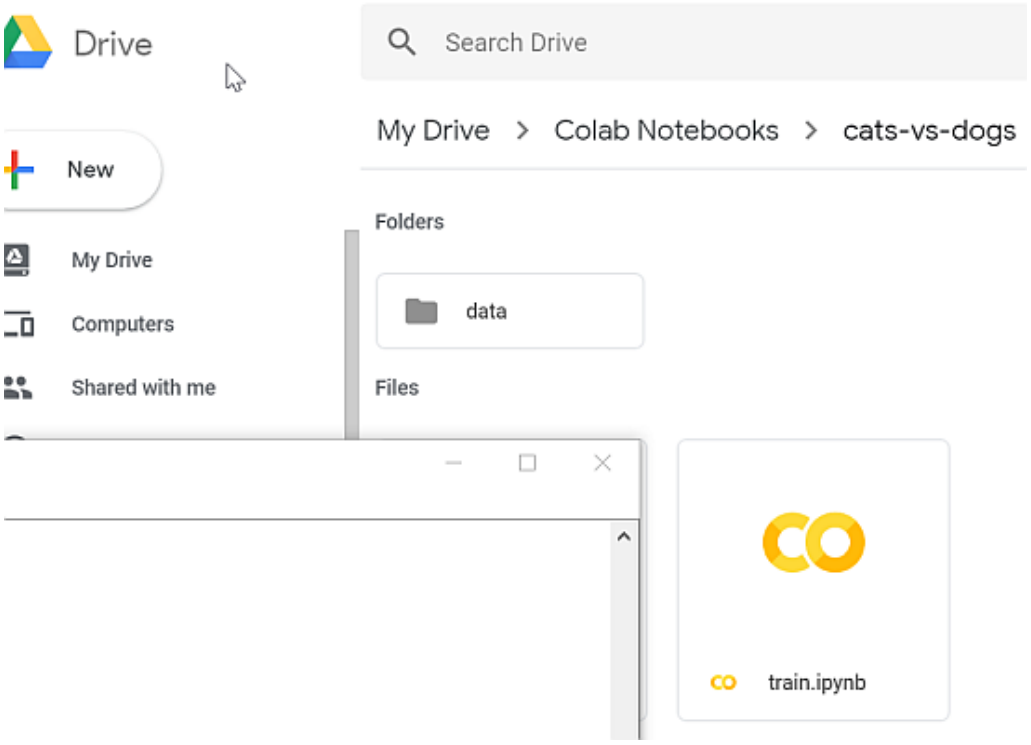
The train/ folder contains 1000 kitteh images under cats/, and 1000 doggo/pupper ones in dogs/. Have fun, looking at the little furballs :) Obviously **you** know which is which :) A neural network is going to start from scratch, and learn the difference, just based on these 2000 images. The validation/ folder contains 400 images each, of more cats and dogs - these are to feed the trained network, compare its classification answers to our own correct answers so we can compute the accuracy of the training. Finally, live/ is where you'd be placing new images of cats and

dogs [that are not in the training or validation datasets], and use their filenames to ask the network to classify them: an output of 0 means 'cat', 1 means 'dog'. Fun!

Simply drag and drop the data/ folder on to your My Drive/Colab Notebooks/cats-vs-dogs/ area, and wait for about a half hour for the 2800 (2*(1000+400)) images to be uploaded. After that, you should be seeing this:



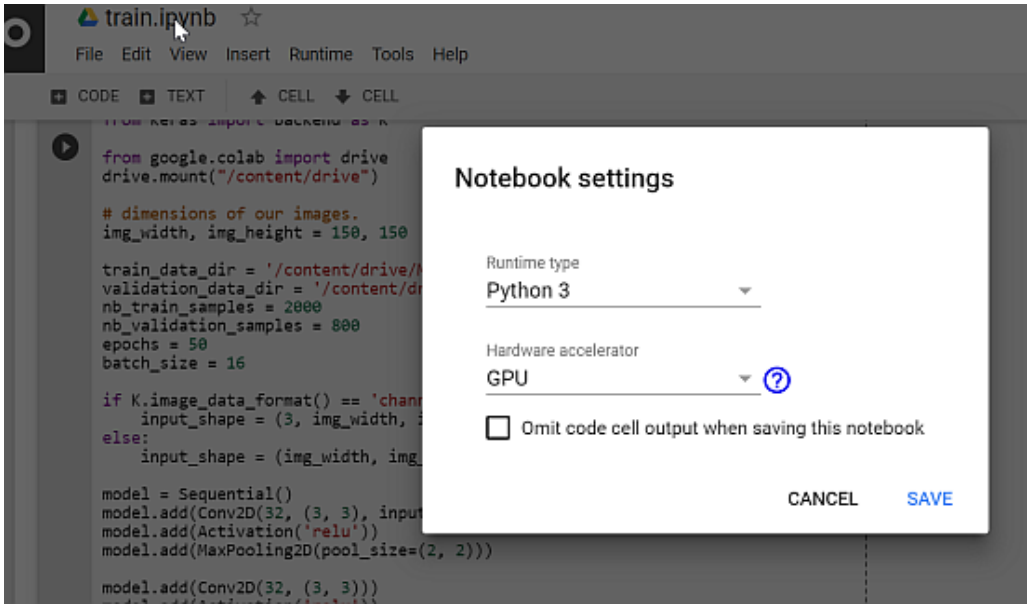
4. OK, time to train! Download this (nb/train.ipynb) Jupyter notebook. A Jupyter notebook (.ipynb extension, 'Iron Python Notebook') is a JSON file that contains a mix of two types of "cells" - text cells that have Markdown-formatted text and images, and code cells that contain, well, code :) Drag and drop the notebook into cats-vs-dogs:



Double click on the notebook, that will open it so you can execute the code in the cell(s).

As you can see, it is a VERY short piece of code [not mine, except annotations and mods] where a network is set up [starting with 'model = Sequential()'], and the training is done using it [model.fit_generator()]. In the last line, the RESULTS [learned weights, biases, for each neuron in each layer] are stored on disk as a weights.h5 file [a .h5 file is binary, in the publicly documented .hd5 file format (https://en.wikipedia.org/wiki/Hierarchical_Data_Format) (hierarchical, JSON-like, perfect for storing network weights)].

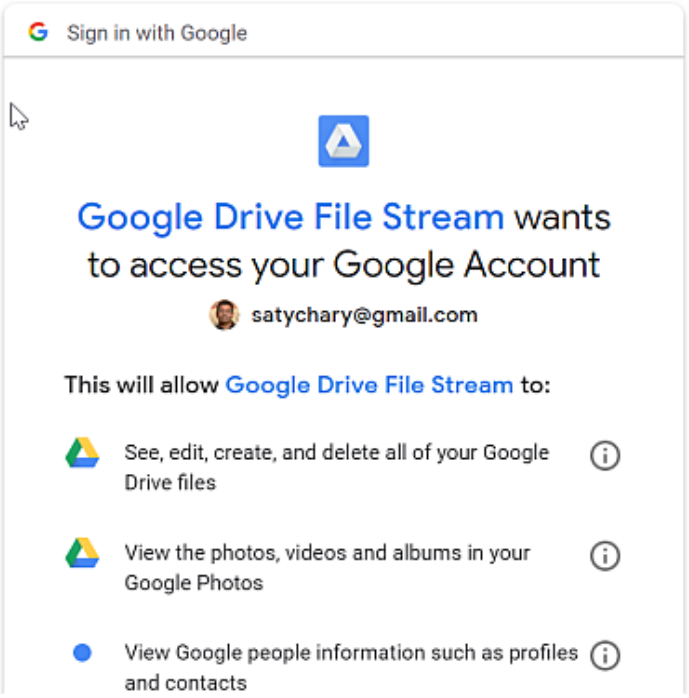
Before you run the code to kick off the training, note that you will be using GPU acceleration on the cloud (results in ~10x speedup) - cool! You'd do this via 'Edit->Notebook settings'. In this notebook, this is already set up (by me), but you can verify that it's set:

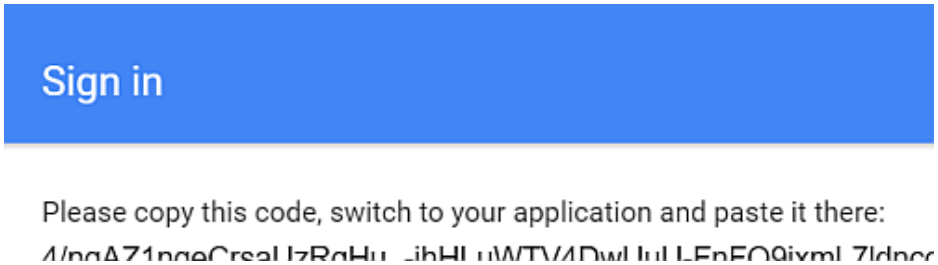


When you click on the circular play button at the left of the cell, the training will start - here is a sped-up version of what you will get (your numerical values will be different):

The backprop loop runs 50 times ('epochs'). The acc: column shows the accuracy [how close the training is, to the expected validation/ results] - NOT BAD, for having learned from just 1000 input images for each class!

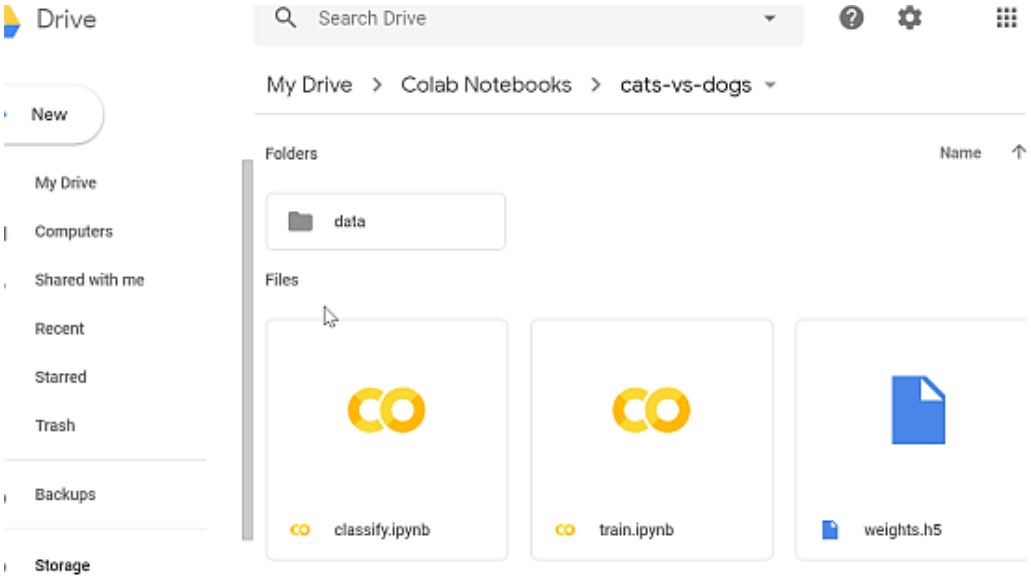
OK, run! The first time you run (any anytime after logging out and logging back in), you'd need to authorize Colab to access GDrive - so a dialog will pop up, asking you to click on a link, and copy and paste an authorization code that appears on the clicked page. Once you do this, the rest of the code (where the training occurs) will start to run.



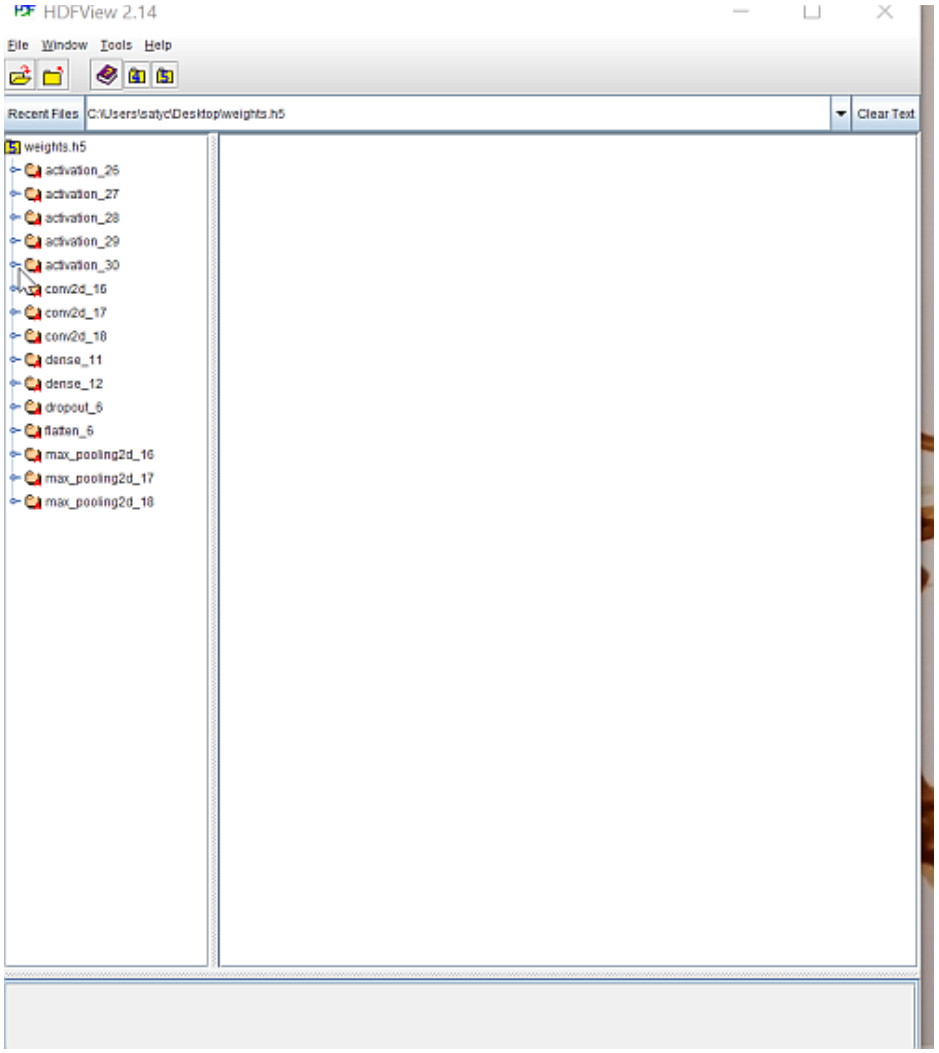


Scroll down to below the code cell, to watch the training happen. As you can see, it is going to take a short while.

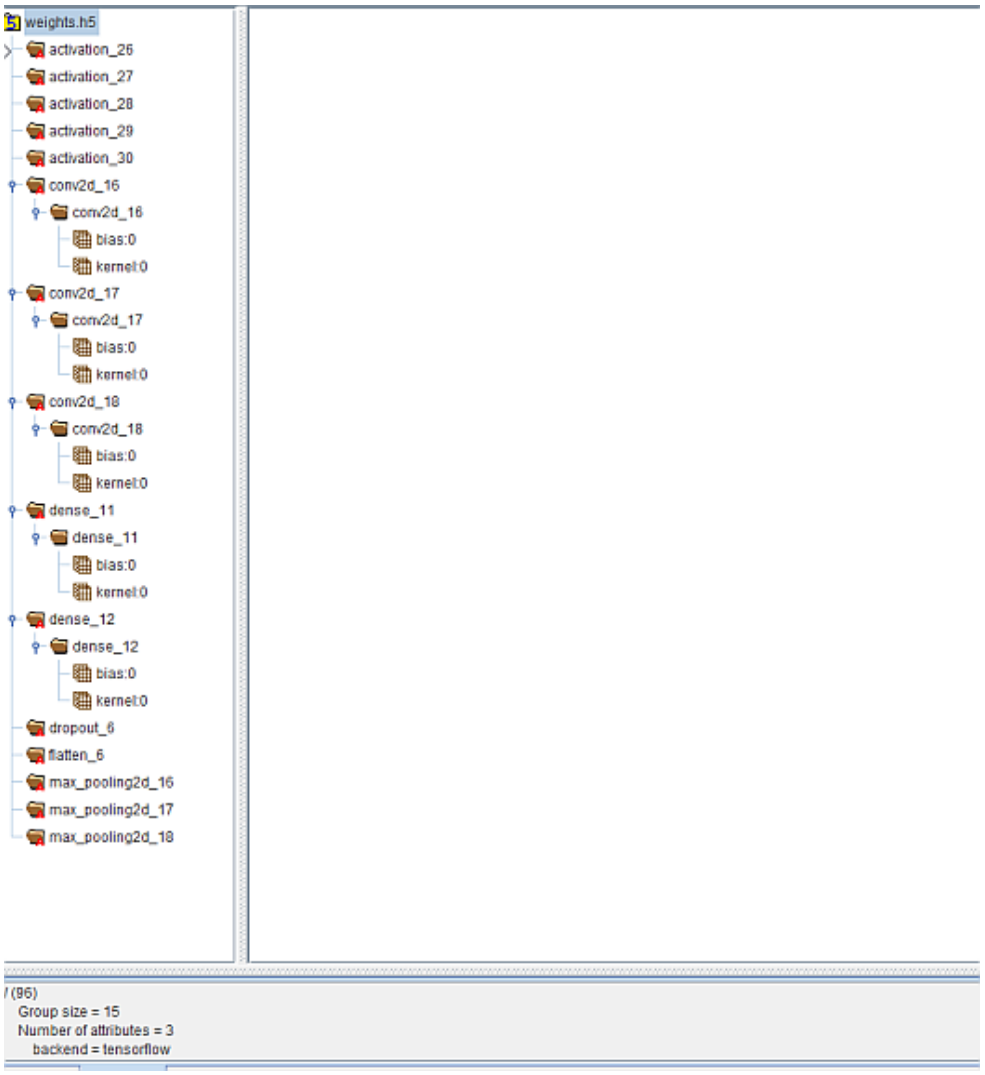
After the 50th epoch, we're all done training. **What's the tangible result, at the end of the training process? It's a 'weights.h5' file!** If you look in your cats-vs-dogs/ folder, it should be there:



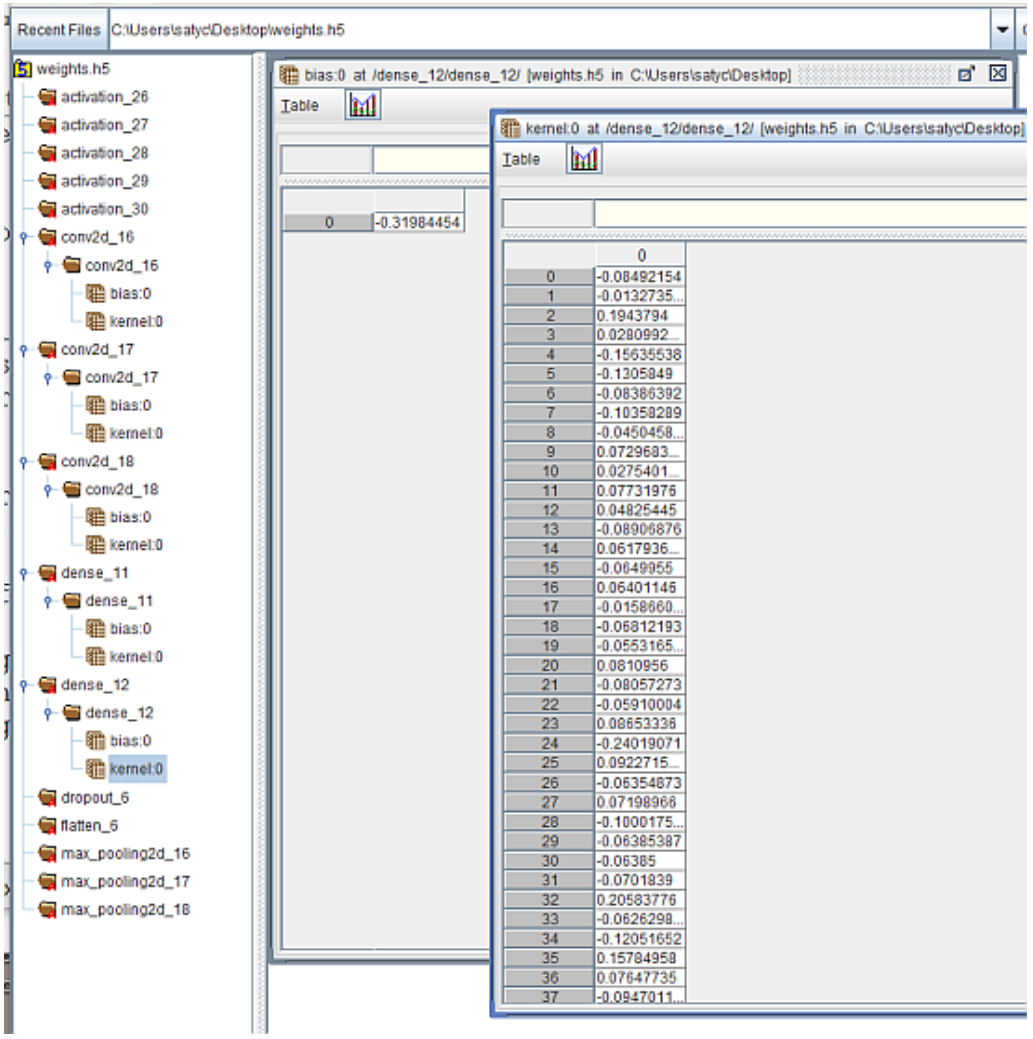
5. Soooo, what exactly [format and content-wise] is in the weights file? You can find out, by downloading HDFView-2.14.0, from https://support.hdfgroup.org/products/java/release/download.html (https://support.hdfgroup.org/products/java/release/download.html) [grab the binary, from the 'HDFView+Object 2.14' column on the left]. Install, and bring up the program. Download the .h5 file from GDrive to your local area (eg. desktop), then drag and drop it into HDView:



Right-click on weights.h5 at the top-left, and do 'Expand All':



Neat! We can see the NN columns, and the biases and weights (kernels) for each. Double click on the bias and kernel items in the dense_12 layer, and stagger them so you can see both:



Computing those floating point numbers is WHAT -EVERY FORM- OF NEURAL NETWORK TRAINING IS ALL ABOUT! Collectively (taking all layers into account), it's those numbers that REPRESENT the network's "learning" of cats and dogs! The "learned" numbers (the .h5 weights file, actually) can be sent to anyone, who can instantiate a new network (with the same architecture as the one in the training step), and simply re/use the weights in weights.h5 to start classifying cats and dogs right away - no training necessary. The weight arrays represent "catness" and "dogness", in a sense :)

Q1 [1+1=2 points]. Submit your weights.h5 file. Also, create a submittable screengrab similar to the above [showing values for dense_12]. For fun, click around, examine the arrays in the other

layers as well. Again, it's all these values that are the end result of training - they result from iterating and minimizing errors through those epochs.

6. Now for the fun part - finding out how well our network has learned! Download this (nb/classify.ipynb) Jupyter notebook, which contains code to read the weights file and associate the weights with a new model (100% identical to what we set up, to train), then take an image filename as input, and **predict (model.predict()) whether the image is that of a cat [output: 0], or a dog [output:1]!** Why 0 for cat and 1 for dog? Because 'c' comes before 'd' alphabetically :)

Supply (upload, into live/) a what1.jpg cat image, and what2.jpg dog image, then execute the cell. Hopefully you'd get a 0, and 1 (for what1.jpg and what2.jpg, respectively). The images can be any resolution (size) and aspect ratio (squarishness), but nearly-square pics would work best. Try this with pics of your pets, your neighbors', images from a Google search, even your drawings/paintings...

Q2 [1+1=2 points]. Create a screenshot that shows the [correct] classification (you'll also be submitting your what{1,2}.jpg images with this).

What about misclassification? After all, we trained with "just" 1000 (not 1000000) images each, for about an 80% accurate prediction. What if we input 'difficult' images, of a cat that looks like it could be labeled a dog, and the other way around? :)

Q3 [1+1=2 points]. Get a 'Corgi' image [the world's smartest dogs!], and a 'dog-like' cat image [hint, it's all about the ears!], upload to live/, attempt to (mis)classify, ie. create incorrect

results (where the cat pic outputs a 1, and the dog's, 0), make a screenshot. Note that you need to edit the code to point myPic and myPic2 to these image filenames.

Here's a checklist of what to submit [as a single .zip file]:

- weights.h5, and a screenshot from HDFView
- your 'good' cat and dog pics, and screenshot that shows proper classification
- your 'trick' cat and dog pics, and screenshot that shows misclassification

---

All done - hope you had fun, and learned a lot!

You can continue using Colab to run all sorts of notebooks (https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks) [on Google's cloud GPUs!], including ones with TensorFlow, Keras, PyTorch... etc. ML code.
