2017 SDSC Summer Institute
Scalable Machine Learning

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

CNN Transfer Learning Hands-On

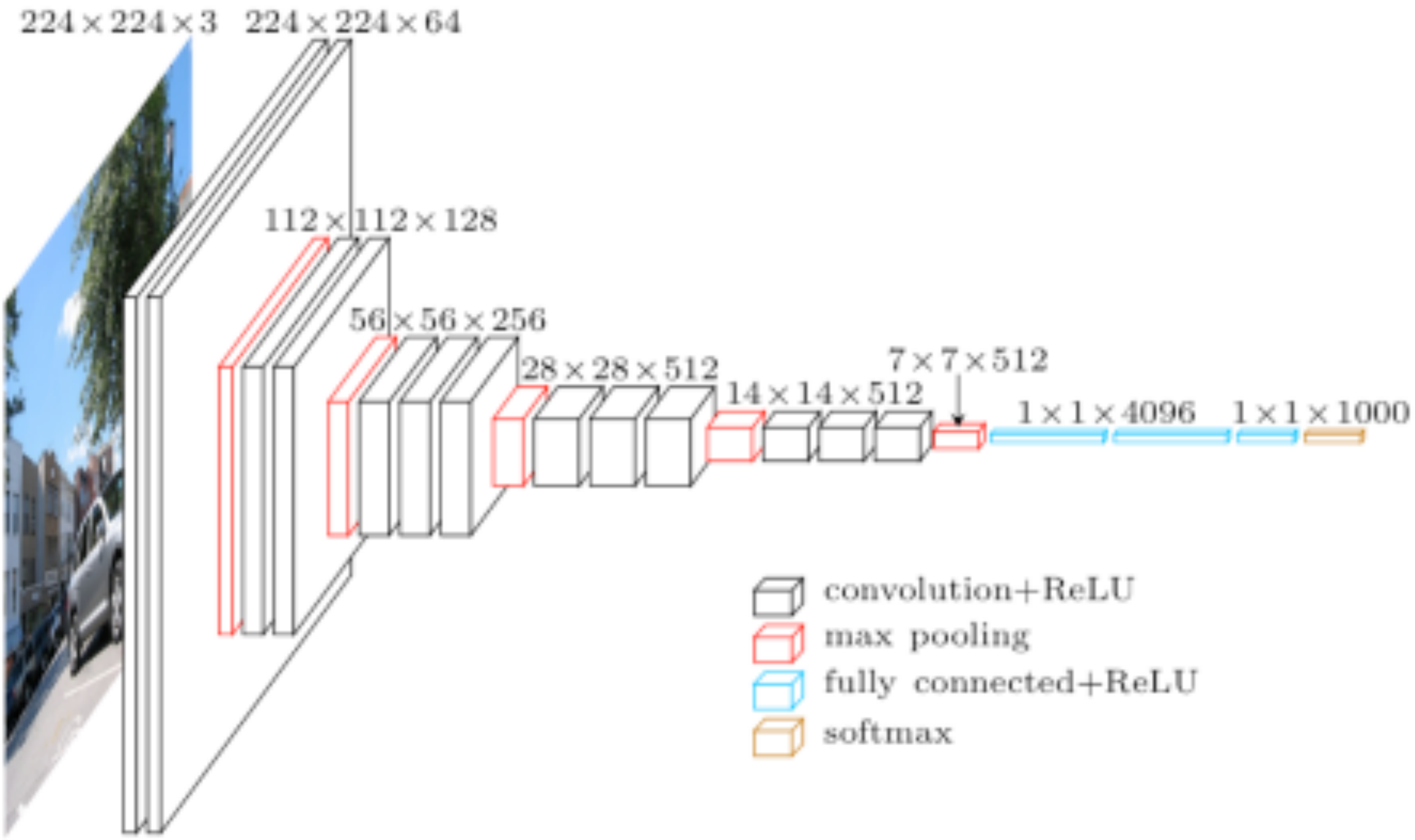
Mai H. Nguyen

Overview

- **Data**
  - Cats and dogs images from Kaggle

- **Method**
  - Use VGG16 trained on ImageNet data as pre-trained model. Remove last fully connected layer.
  - Extract features from pre-trained model and save
  - Neural network then trained on extracted features to classify cats vs. dogs

VGG Architecture



Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Set Link to Data

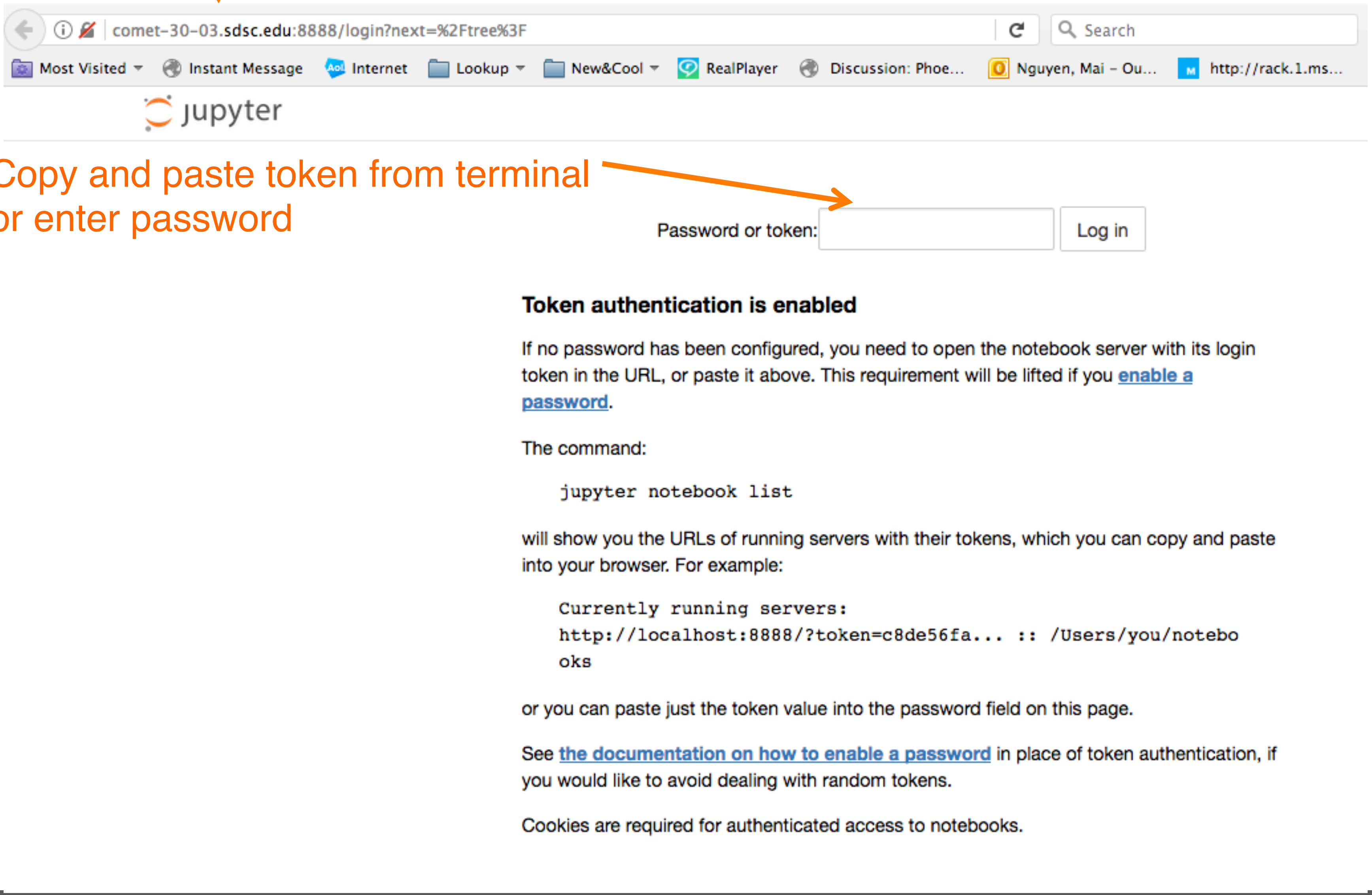
- **Go to Keras directory**
  - cd SI2017/scalableML/keras
- **Create soft link to data (if not already there)**
  - ln –s /oasis/scratch/comet/mhnguyen/temp_projects/data/kaggle_cats_dogs data
- **Look at dataset**
  - ls –l data/train/cats/* | wc
  - ls –l data/train/dogs/* | wc
  - ls –l data/validation/cats/* | wc
  - ls –l data/validation/dogs/* | wc

Server Setup

- **Request GPU node**
  - getgpu
    - alias for (long) command to request interactive session on GPU node
    - Prompt should change to <user>@comet-xx-xx
- **Start up Keras-TensorFlow Singularity image**
  - module load singularity
  - singularity shell keras.img
    - Prompt should change to Singularity.keras.img> $
- **Start up Jupyter server**
  - jupyter notebook --no-browser --ip="*" &

Browser Setup

comet-xx-xx.sdsc.edu:8888



Copy and paste token from terminal or enter password

Password or token:

Log in

Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you enable a password.

The command:

```
jupyter notebook list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

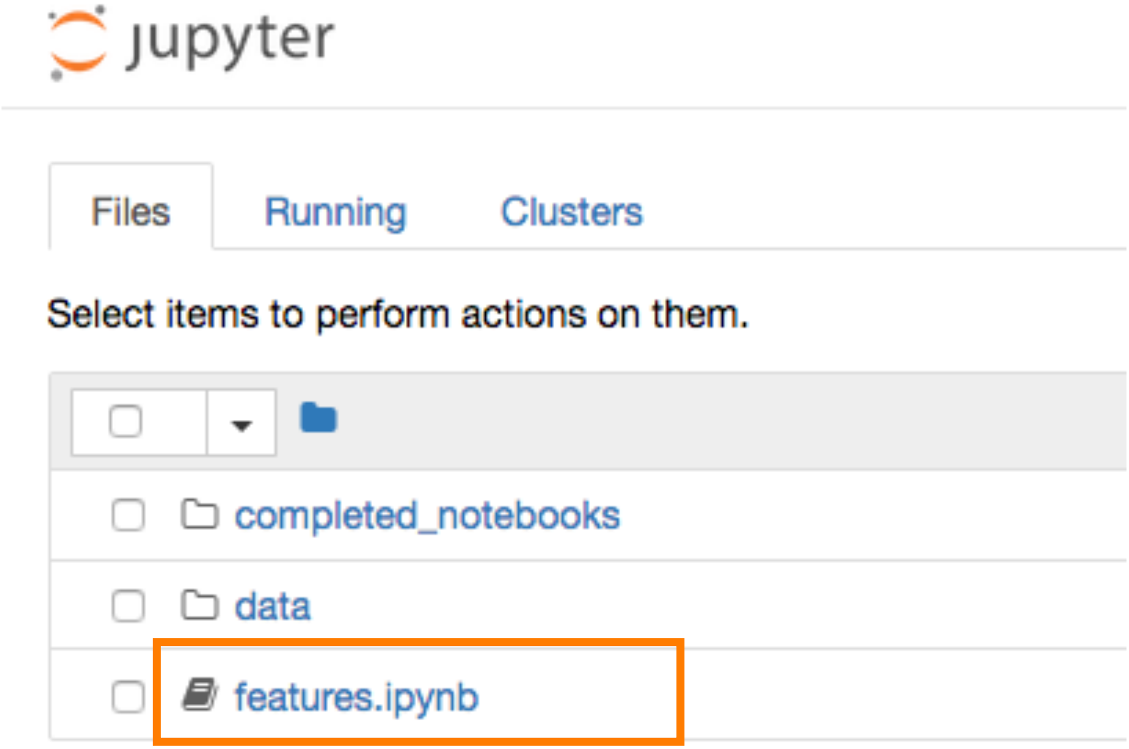
```
Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebo
oks
```

or you can paste just the token value into the password field on this page.

See the documentation on how to enable a password in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to notebooks.

Open features.ipynb Notebook



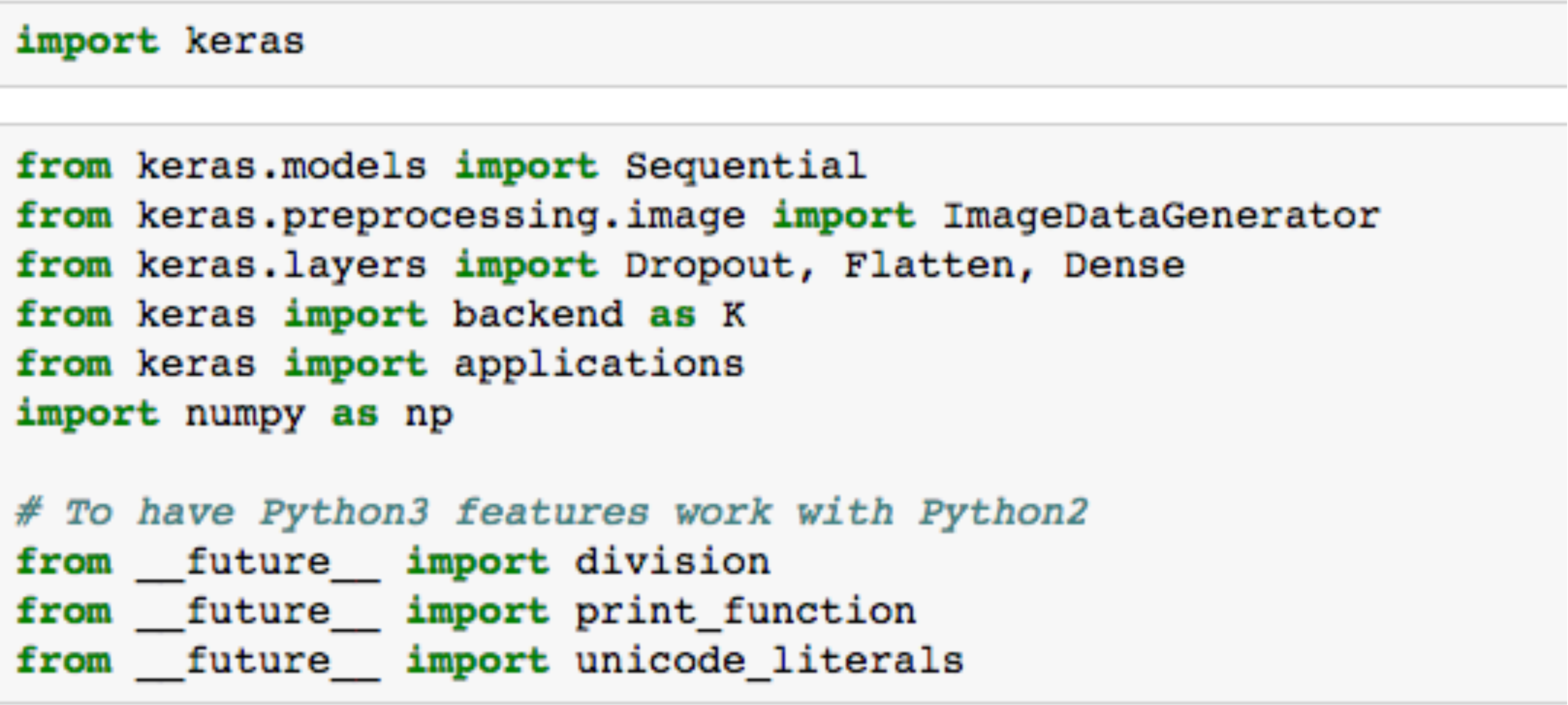
Select items to perform actions on them.

□ completed_notebooks

□ data

□ features.ipynb

Import Libraries

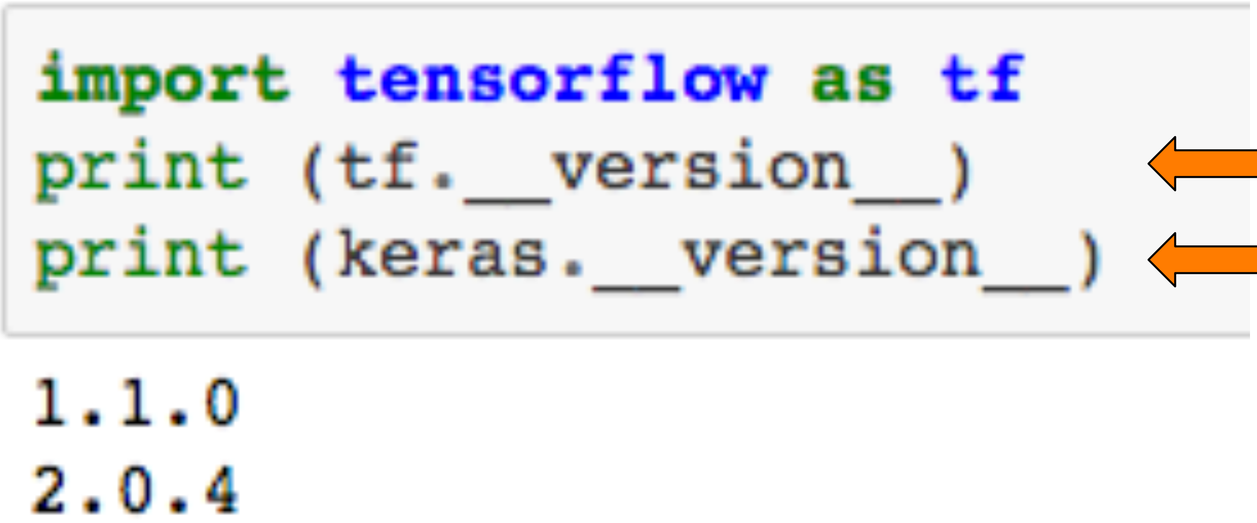


```
import keras
```

```
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dropout, Flatten, Dense
from keras import backend as K
from keras import applications
import numpy as np

# To have Python3 features work with Python2
from __future__ import division
from __future__ import print_function
from __future__ import unicode_literals
```

Print Keras & TensorFlow Versions



```
import tensorflow as tf
print (tf.__version__)
print (keras.__version__)
```

```
1.1.0
2.0.4
```

Set Data Parameters

- Set image dimensions
  - *img_width, img_height = 150, 150*

- Set data location
  - *train_data_dir = ‘data/train’*
  - *validation_data_dir = ‘data/validation’*

- Set number of images
  - *nb_train_samples = 2000*
  - *nb_validation_samples = 800*

(150, 150, 3)

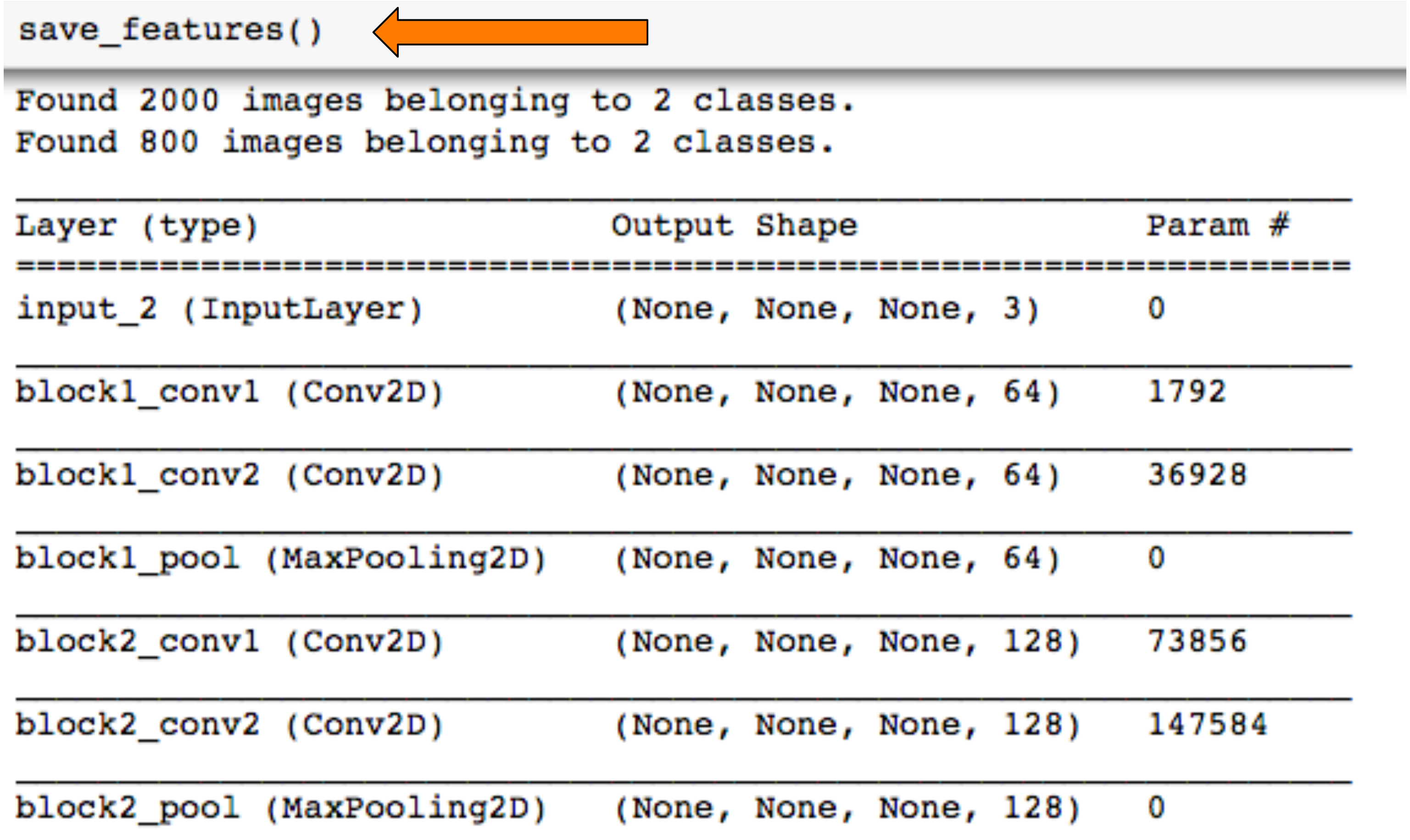
Method to Extract Features from Pre-Trained Network

def save_features():

…

1. Scale pixel values in each image
2. Load weights for pre-trained network without top classifier
3. Generator reads images from subdir, batch_size number of images at a time.
4. Feed images through pre-trained network and extract features
5. Save features
6. Repeat 3-5 for validation data

Call Method to Extract & Save Features



```
save_features()
```

```
Found 2000 images belonging to 2 classes.
Found 800 images belonging to 2 classes.

_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         (None, None, None, 3)     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0         
```

Load Saved Features

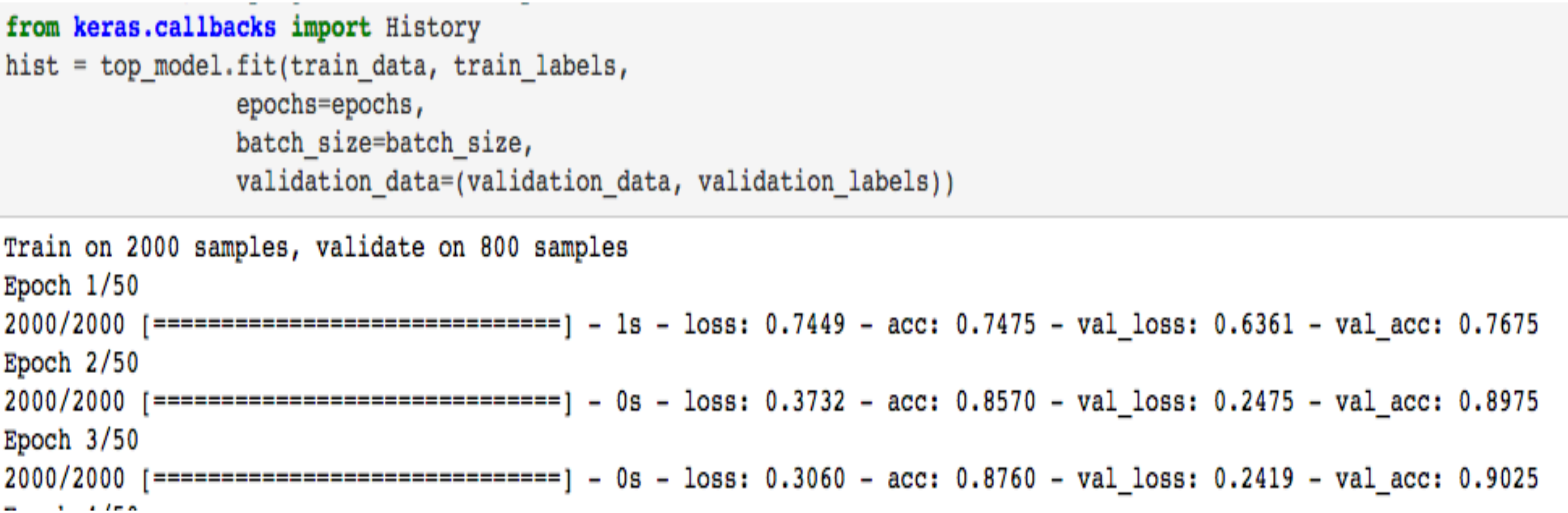
- **Add name of file containing saved features**
  - For train data
    - *train_data = np.load (‘features_train.npy’)*
  - For validation data
    - *validation_data = np.load (‘features_validation.npy’)*

Create Top Model to Classify Extracted Features

- **Model**
  - Fully connected layer from input to hidden
    - 256 nodes in hidden layer
    - Rectified linear activation function
  - Fully connected layer from hidden to output
    - 1 node in output layer (cat or dog)
    - Sigmoid activation function

Train Top Model

- Set number of training iterations
  - epochs = 50
- Train model, keeping track of history

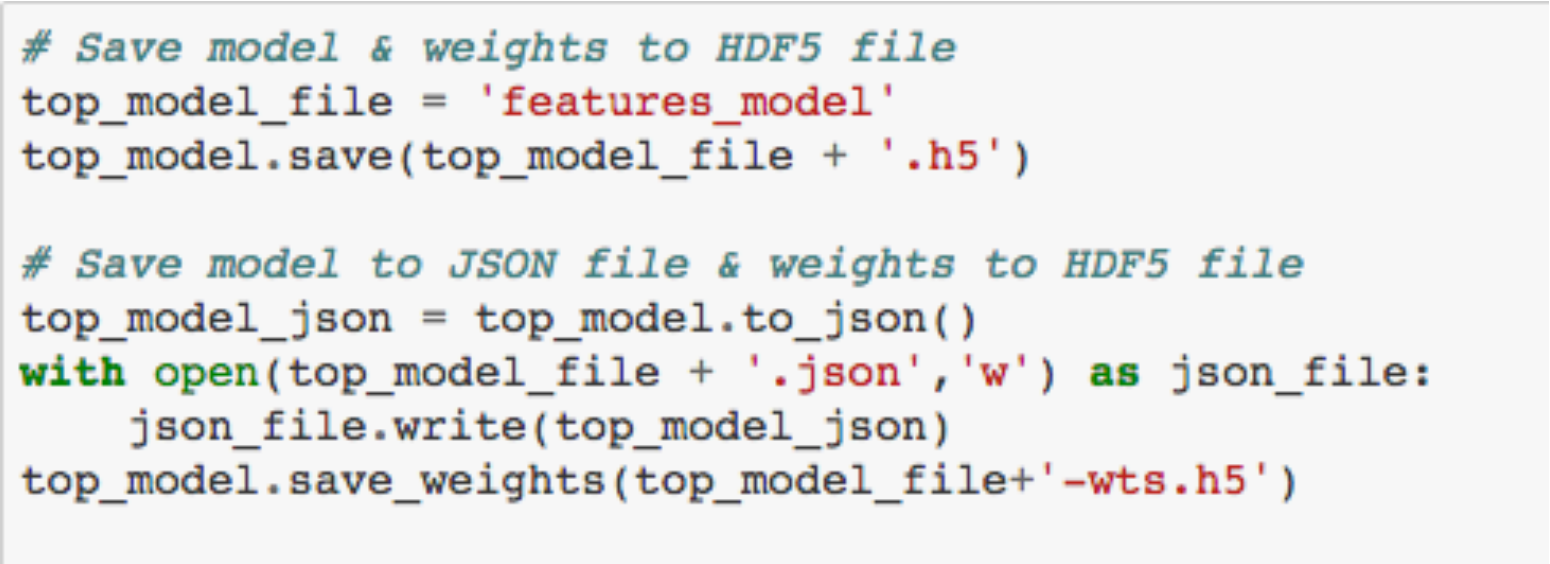


```
from keras.callbacks import History
hist = top_model.fit(train_data, train_labels,
                     epochs=epochs,
                     batch_size=batch_size,
                     validation_data=(validation_data, validation_labels))
```

```
Train on 2000 samples, validate on 800 samples
Epoch 1/50
2000/2000 [==============================] - 1s - loss: 0.7449 - acc: 0.7475 - val_loss: 0.6361 - val_acc: 0.7675
Epoch 2/50
2000/2000 [==============================] - 0s - loss: 0.3732 - acc: 0.8570 - val_loss: 0.2475 - val_acc: 0.8975
Epoch 3/50
2000/2000 [==============================] - 0s - loss: 0.3060 - acc: 0.8760 - val_loss: 0.2419 - val_acc: 0.9025
```

Save Model and Weights

- **Add name for model files**
  - top_model_file = ‘features_model’
- **Save model and weights**

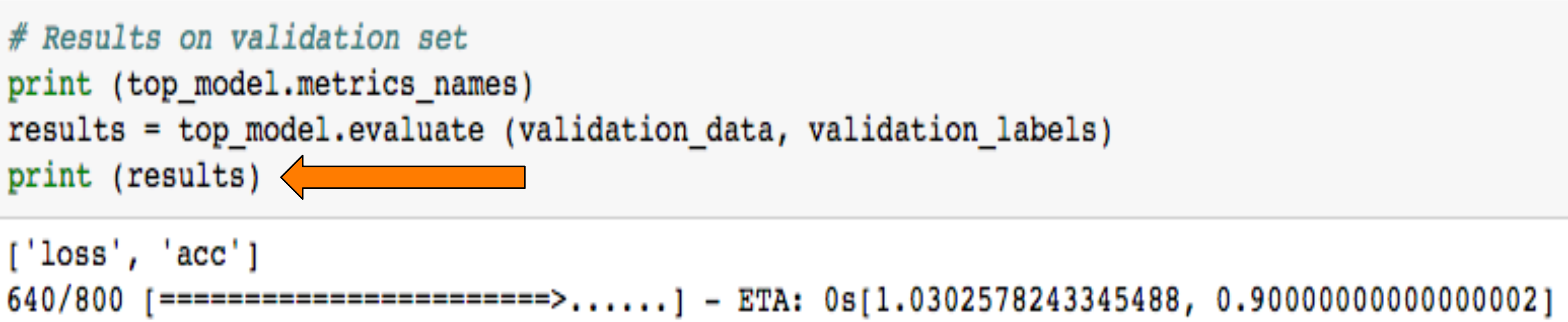


```
# Save model & weights to HDF5 file
top_model_file = 'features_model'
top_model.save(top_model_file + '.h5')

# Save model to JSON file & weights to HDF5 file
top_model_json = top_model.to_json()
with open(top_model_file + '.json','w') as json_file:
    json_file.write(top_model_json)
top_model.save_weights(top_model_file+'-wts.h5')
```

Test Model on Validation Data

- **Get prediction results on validation data**



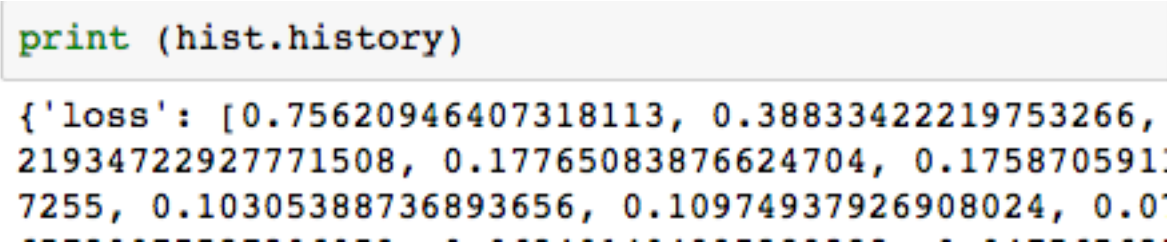
```
# Results on validation set
print (top_model.metrics_names)
results = top_model.evaluate (validation_data, validation_labels)
print (results)
```

```
['loss', 'acc']
640/800 [=======================>......] - ETA: 0s[1.0302578243345488, 0.9000000000000002]
```

- **Load model again and re-test**
  - Results should be the same

Print History & Plot Performance Measures

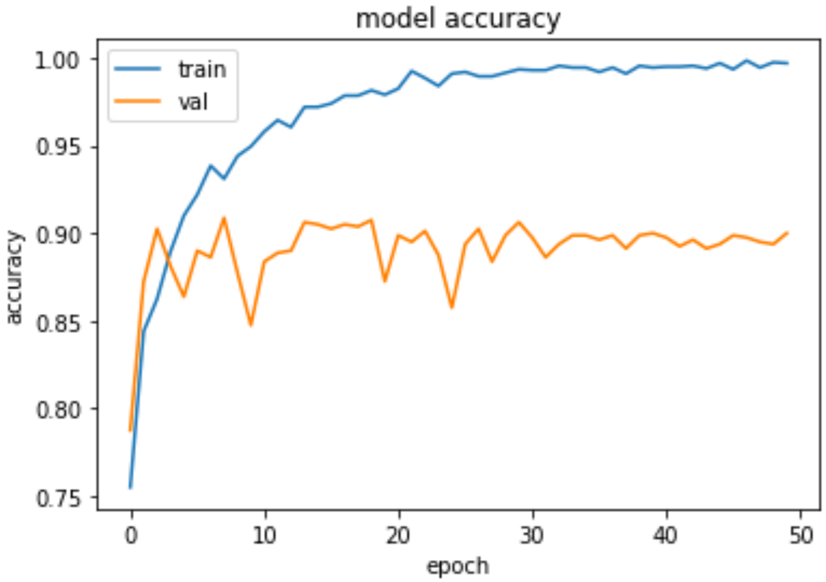
- Print training history



```
print (hist.history)
```

```
{'loss': [0.7562094640731811З, 0.38833422219753266,
21934722927771508, 0.17765083876624704, 0.175870591
7255, 0.10305388736893656, 0.10974937926908024, 0.0
```

- Plot accuracy



References

- **The Keras Blog**
  - https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
- **Code for feature extraction**
  - https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069

Questions?

Scalable Machine Learning Topics

- **R in HPC**
  - Scaling R, running R on HPC
  - Scaling R linear models
- **Machine Learning with Spark**
  - Spark stack, RDDs, MLlib
  - Data exploration & clustering in Spark
- **Deep Learning Overview**
  - Neural network & deep learning overview
  - MNIST tutorial
- **CNN Transfer Learning with Keras**
  - Pre-trained CNN to speed up CNN training
  - Transfer learning to classify cats & dogs images in Keras