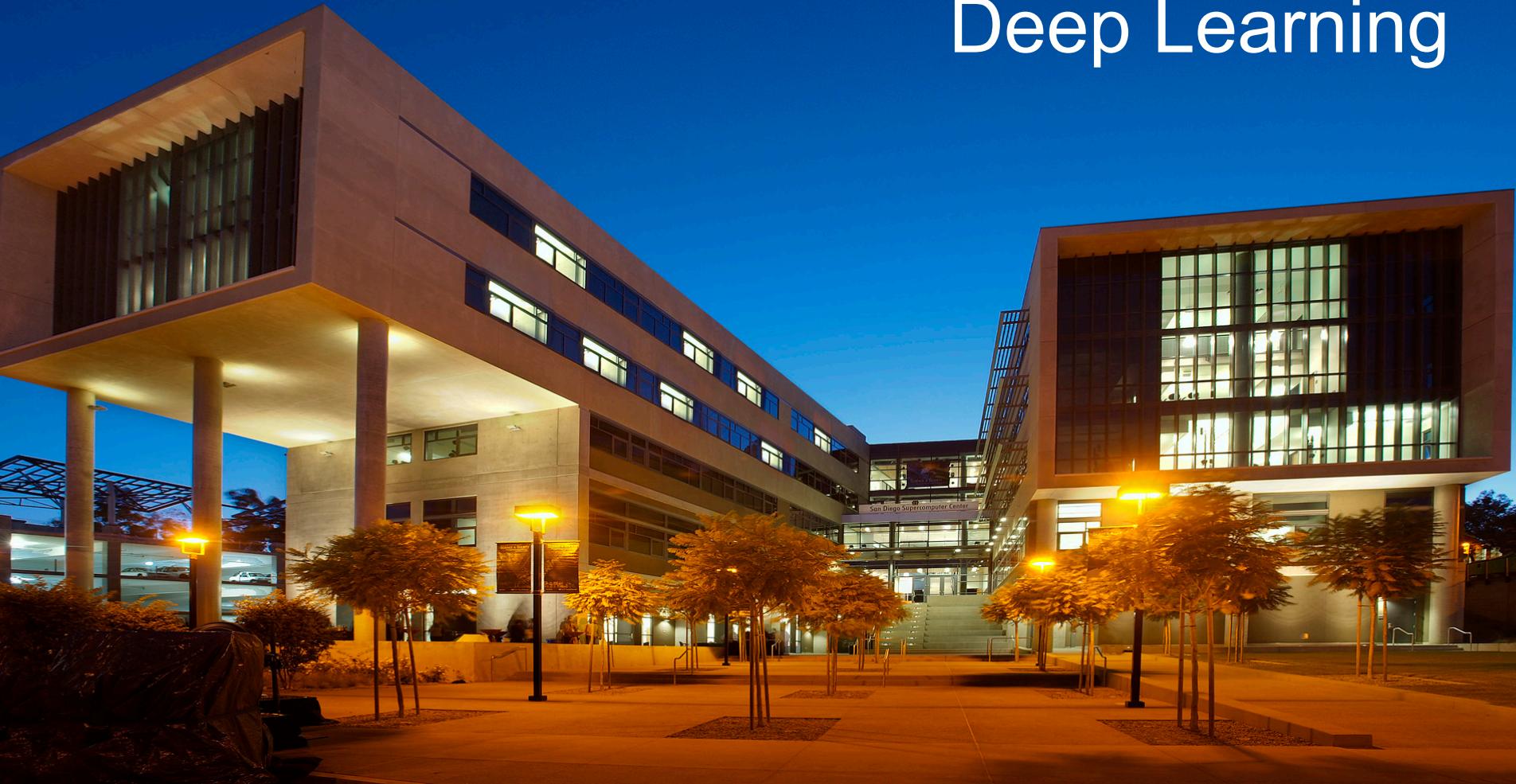


2018 SDSC Summer Institute Deep Learning



CNN Transfer Learning Hands-On

Mai H. Nguyen, Ph.D.

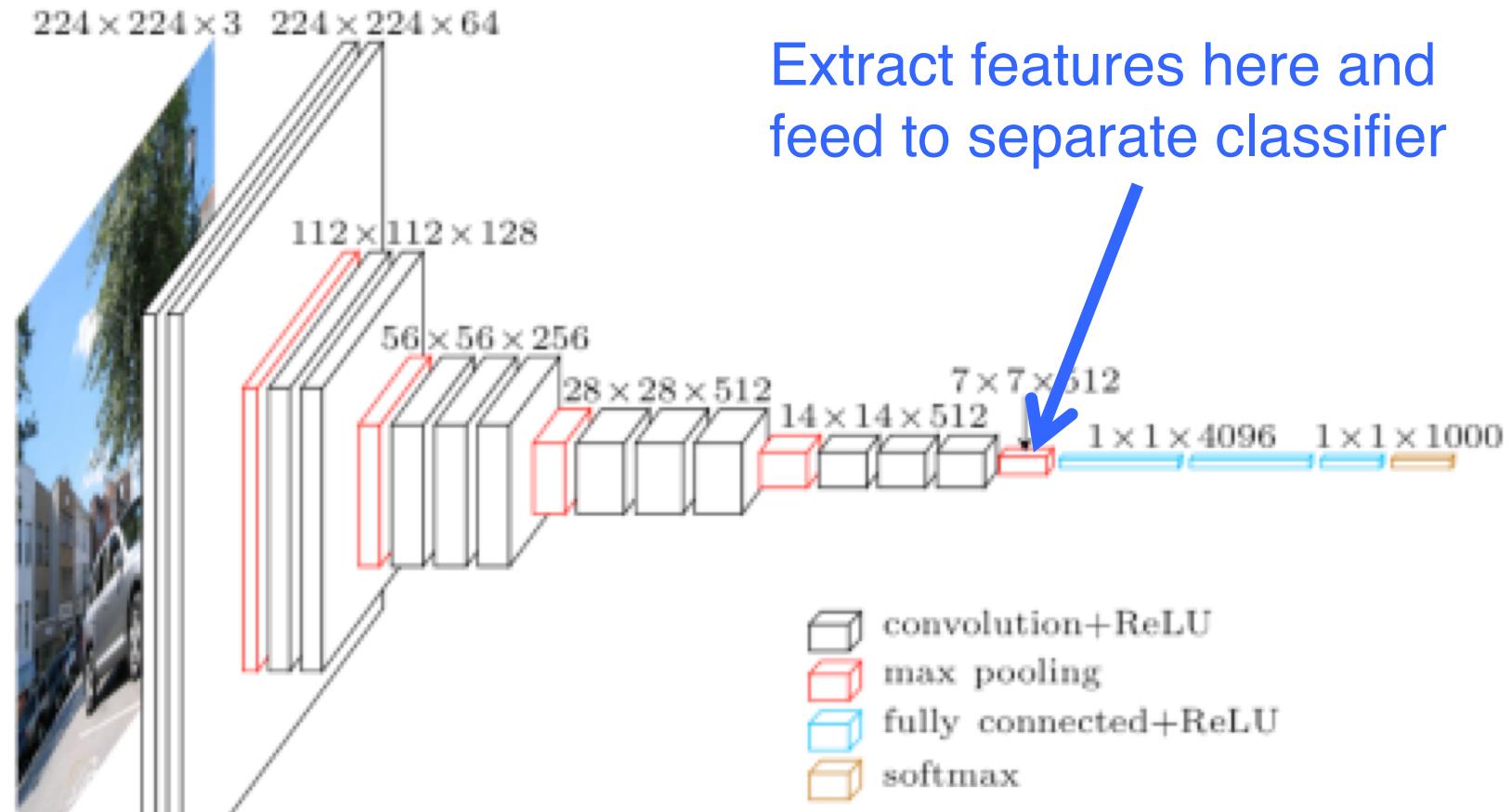
Transfer Learning Hands-On

- **Data**
 - Cats and dogs images from Kaggle
- **Exercises**
 - Feature extraction
 - Use pre-trained CNN to extract features from images
 - Train neural network to classify cats/dogs using extract features
 - Fine tune
 - Adjust weights of last few layers of pre-trained CNN through training

Feature Extraction

- **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

Transfer Learning – Feature Extraction



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

Server Setup

- **Go to keras directory**
 - *cd <SI2018_dir>/datasci2_deep_learning/keras*
- **Request GPU node and start jupyter**
 - *sbatch --res=SI2018DAY2 keras.slrn*

Set Link to Data

- Create soft link to data
 - ln -s ~/data 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 (cont.)

- **Check queue**
 - `squeue -u $USER`
 - Note compute node name (e.g., comet-14-43)
 - 18223060 compute bash <user> R 1:19 1 comet-14-43
- **Check that Jupyter Notebook has started**
 - `ls -l keras*.out`
-rw-r--r-- 1 <user> sds148 840 Aug 4 09:32 keras.18317995.comet-14-43.out
• `tail keras*.out`
• Copy token
 - *Copy/paste this URL into your browser when you connect for the first time, to login with a token:*
`http://localhost:8888/?token=395fe3d456cb951e34ef125adf27e6a62`

Browser Setup

- In browser, type:
 - comet-xx-xx.sdsc.edu:8888
 - Paste token

comet-14-43.sdsc.edu:8888/login?next=%2Ftree%3F

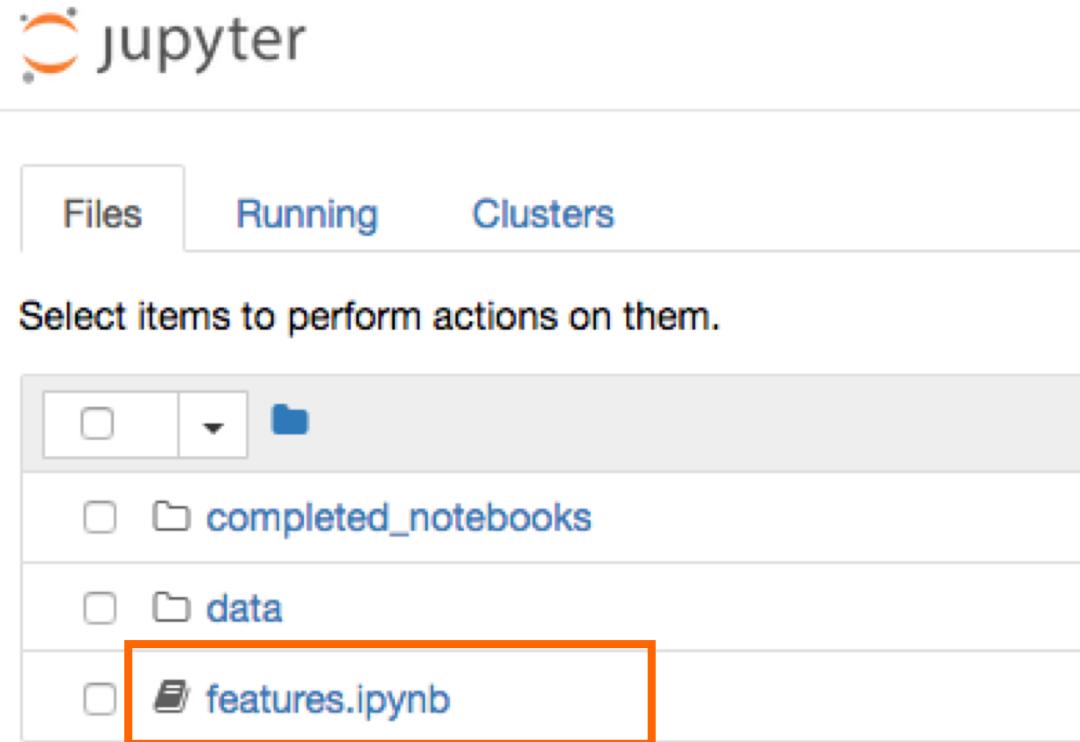


Password or token:

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](#).

Open features.ipynb Notebook



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__)
```

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')
```

(2000,) (800,)

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
- Validation accuracy on CNN trained from scratch
 - ~80%

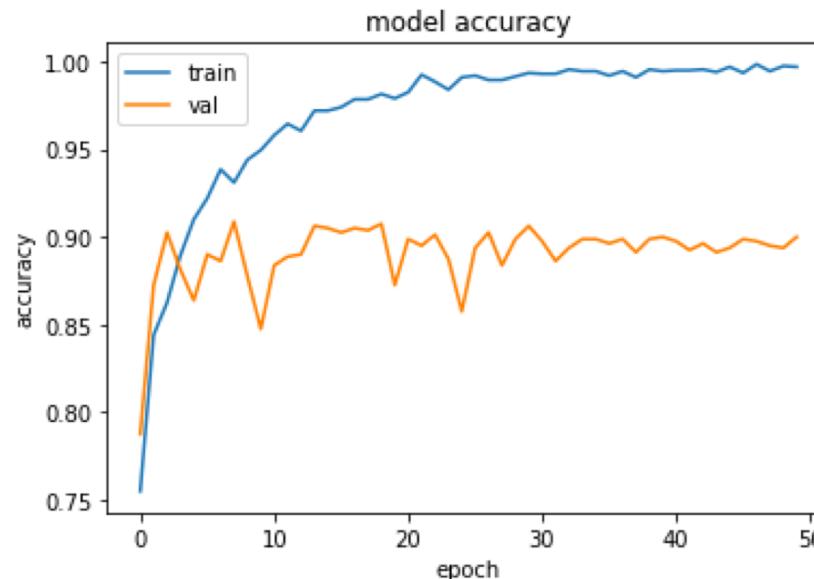
Print History & Plot Performance Measures

- Print training history

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

```
{'loss': [0.75620946407318113, 0.38833422219753266,
21934722927771508, 0.17765083876624704, 0.175870591:
7255, 0.10305388736893656, 0.10974937926908024, 0.0'
```

- Plot accuracy



Exit Notebook

The screenshot shows a Jupyter Notebook interface. At the top, there's a logo and the word "jupyter" followed by "features" and "Last Checkpoint: 07/27/2018". Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Widgets". A toolbar below the menu has buttons for "New Notebook", "Open...", "Make a Copy...", "Save as...", "Rename...", "Save and Checkpoint", "Revert to Checkpoint", "Print Preview", "Download as", "Trusted Notebook", and "Close and Halt". The "File" menu is open, and the "Close and Halt" option is highlighted with a red box. To the right of the menu, the main notebook area displays a title "Transfer Learning on" and a snippet of Python code using the Keras library:

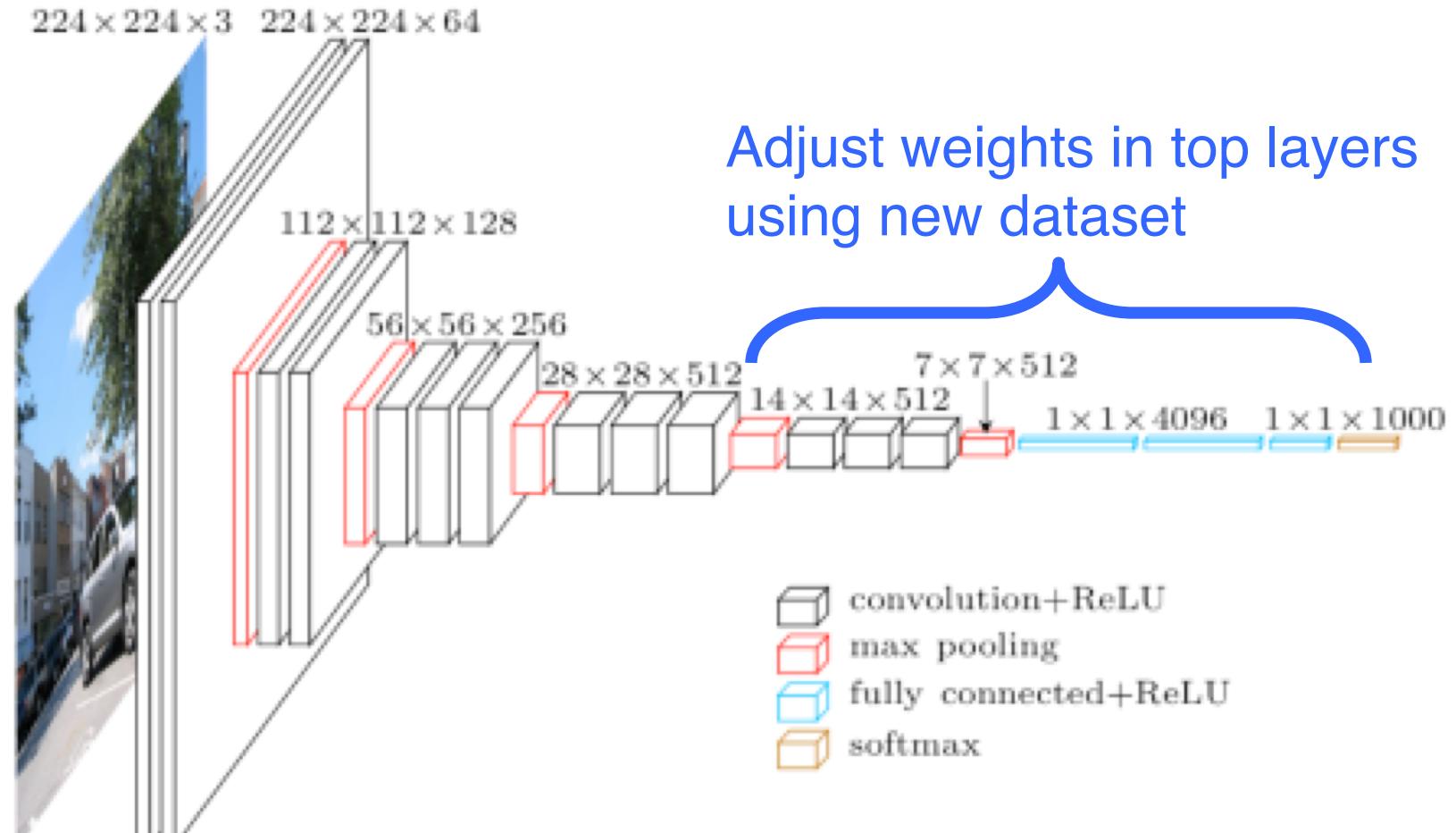
```
from fchollet/classifier_from_little_data
log.keras.io/building-powerful-image-
keras
TensorFlow backend.
```

Below the code, a message says "Keras models import Sequential" and "from keras.preprocessing.image import".

Fine Tuning Hands-On

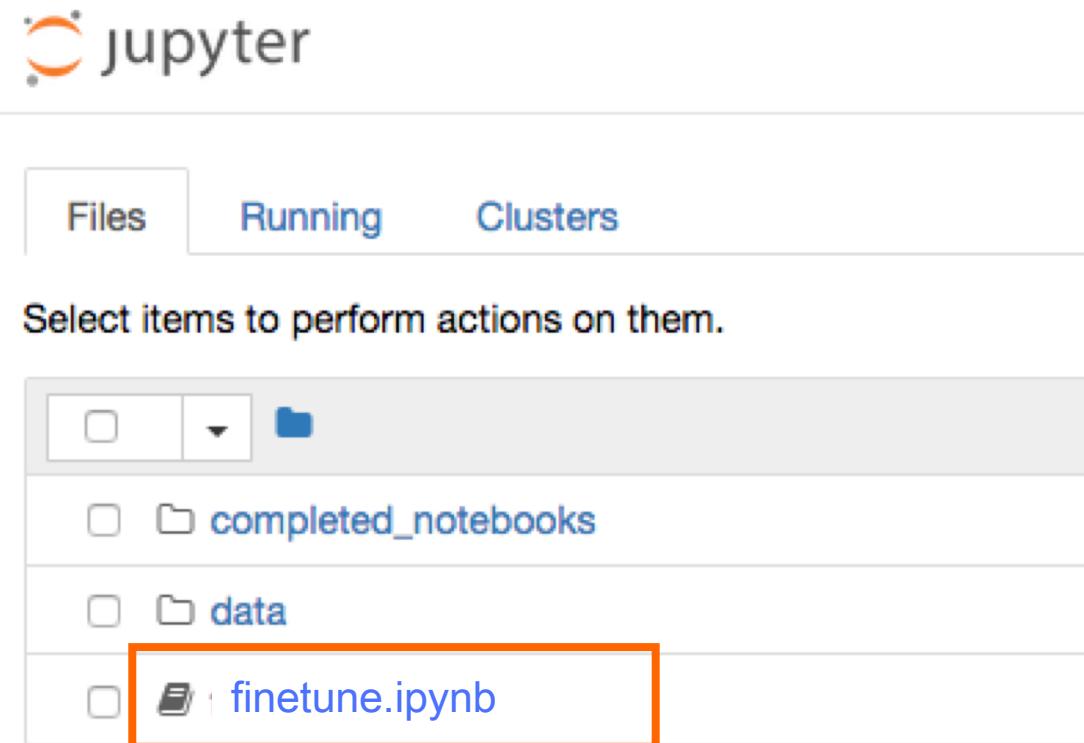
- **Data**
 - Cats and dogs images from Kaggle
- **Method**
 - Use VGG16 trained on ImageNet data as pre-trained model.
 - Replace last fully connected layer with neural network trained from Feature Extraction hands-on.
 - Fine tune last convolution block and fully connected layer.

Transfer Learning – Fine Tuning



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

Open fine-tune.ipynb Notebook



Fine Tuning Results

```
print (model2.metrics_names)
results = model.evaluate_generator(validation_generator,
                                    steps=nb_validation_samples // batch_size)
print (results)
# scores = model2.evaluate_generator(validation_generator)
```

```
['loss', 'acc']
[0.3735525435901945, 0.9125]
```

Clean Up

- **Exit notebook**
 - File -> Close and Halt
- **Exit Jupyter Notebook**
 - Click on ‘Logout’

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?

