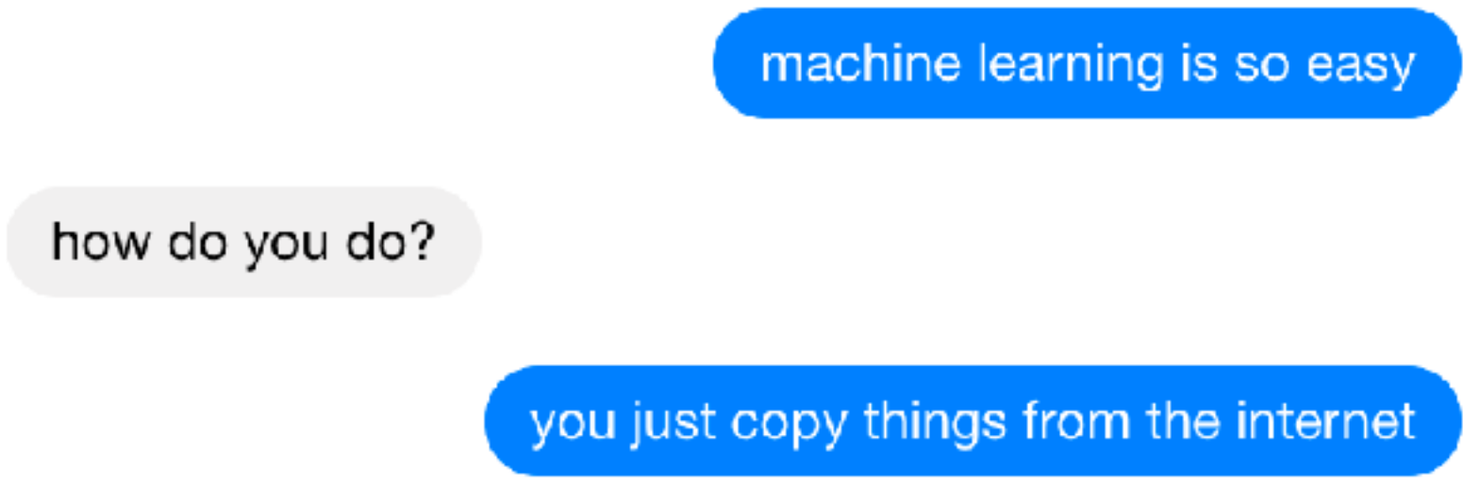
How to do Machine Learning

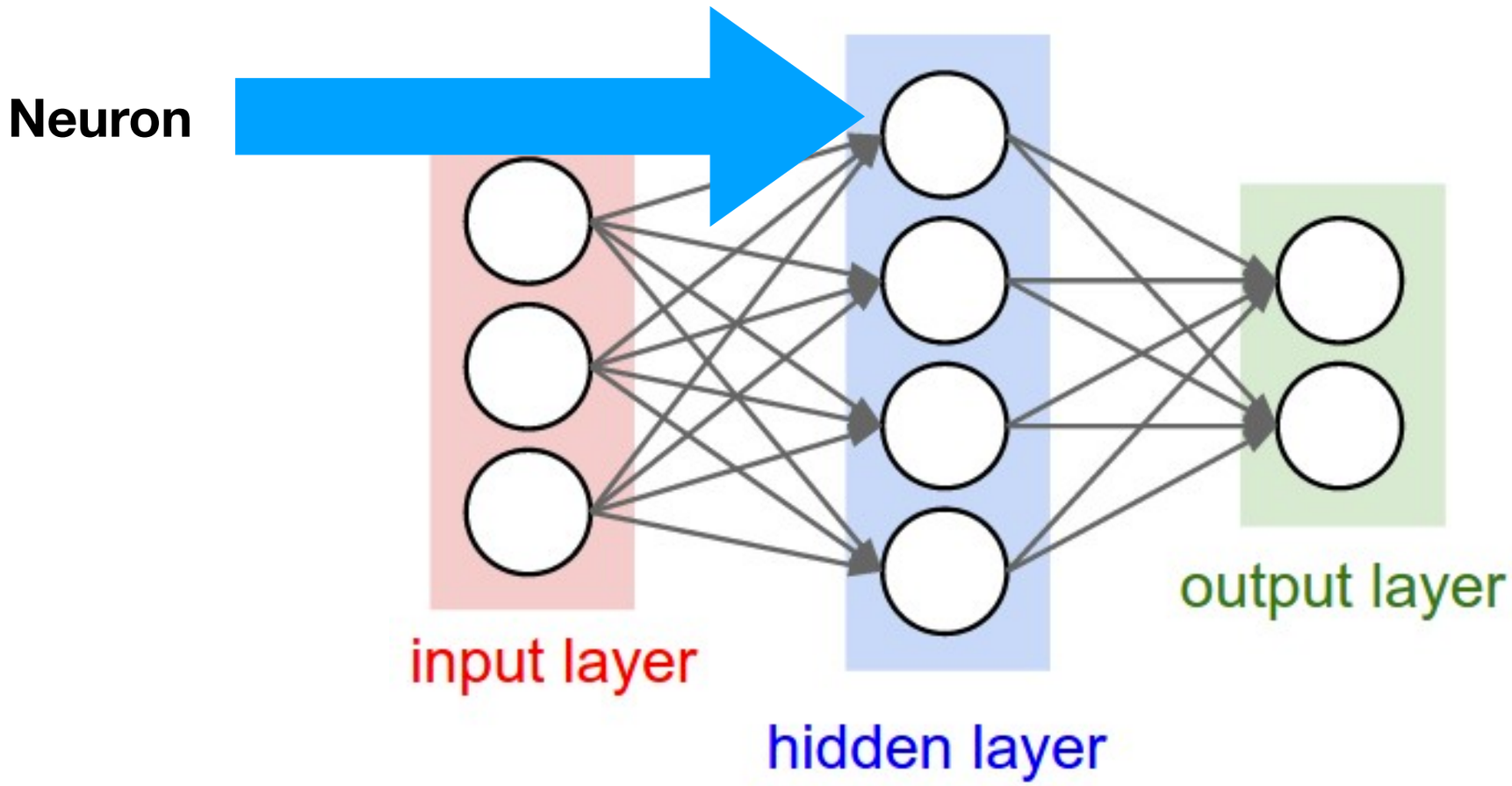
(without knowing Machine Learning)



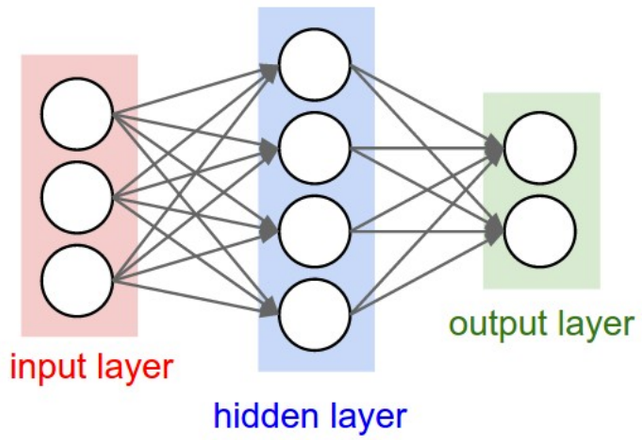
Disclaimers

- Machine learning is super complicated & broad
- This talk will focus specifically on a very simple understanding of neural nets
- I genuinely don’t know machine learning at all. Lots of this is probably wrong.

What is a neural net?

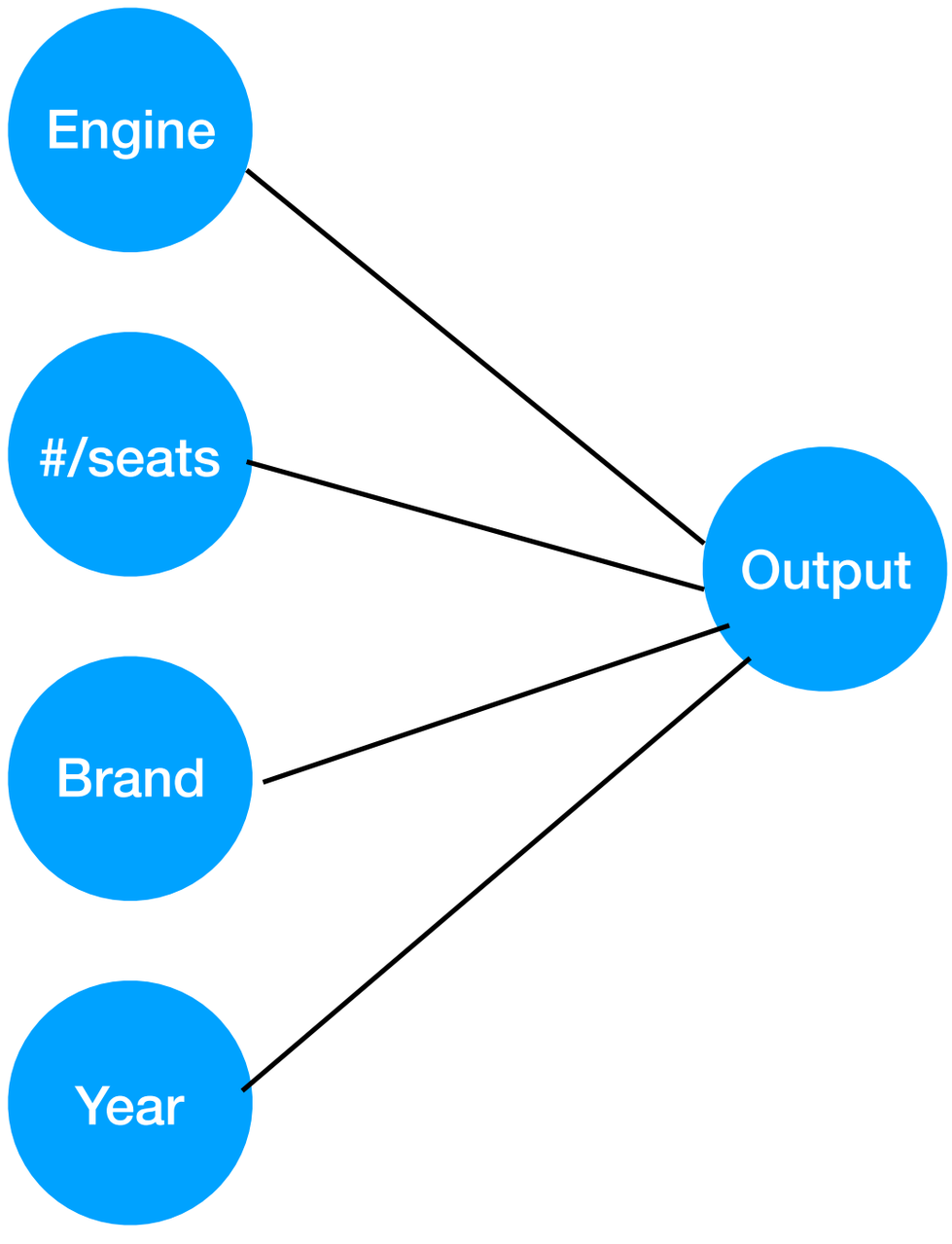


What is a neural net



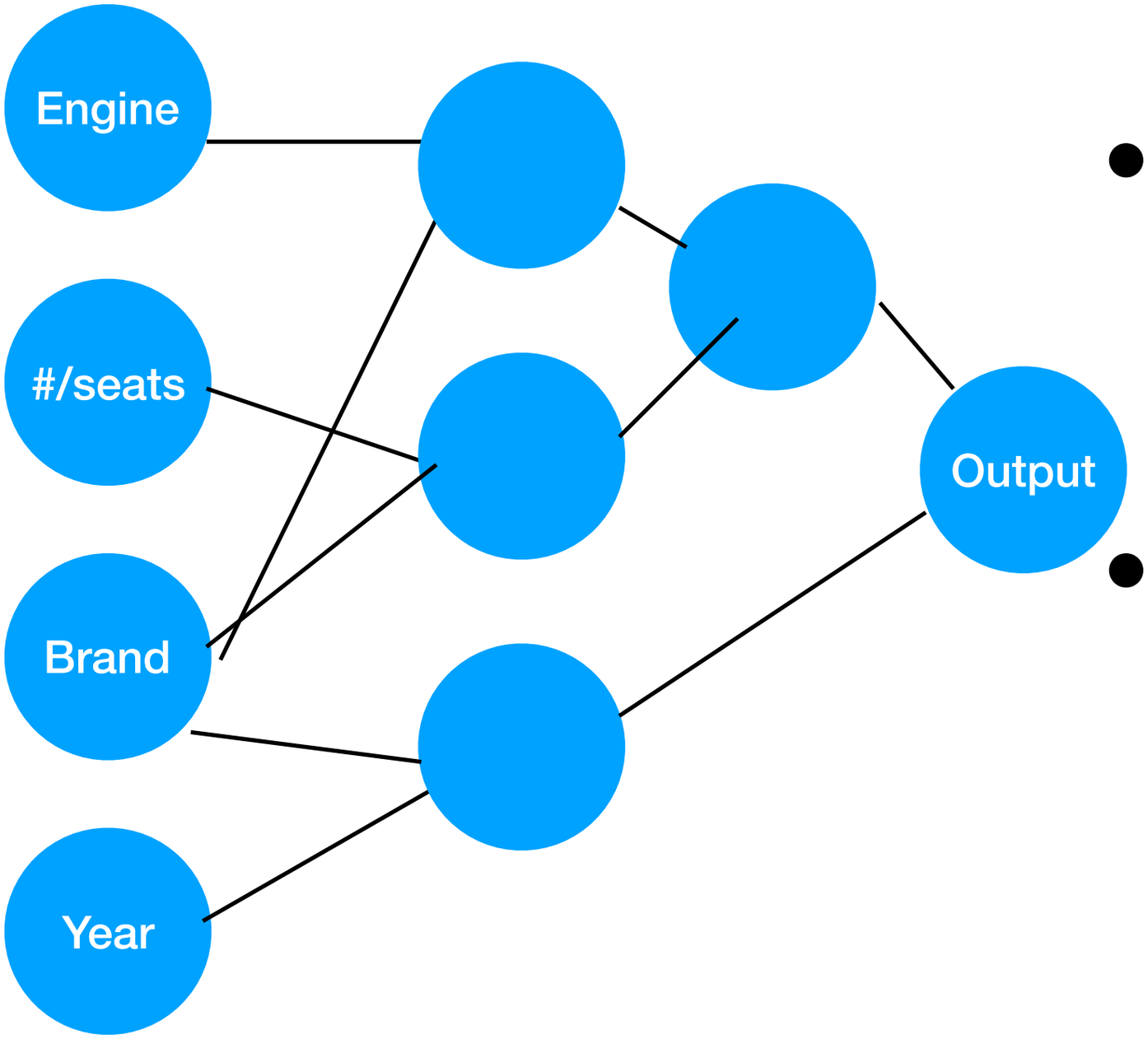
- A connection of inputs and outputs
- Inputs are different characteristics
- Outputs are numerical strengths representing the likelihood that the input belongs in a certain output category

Example: Value of a car



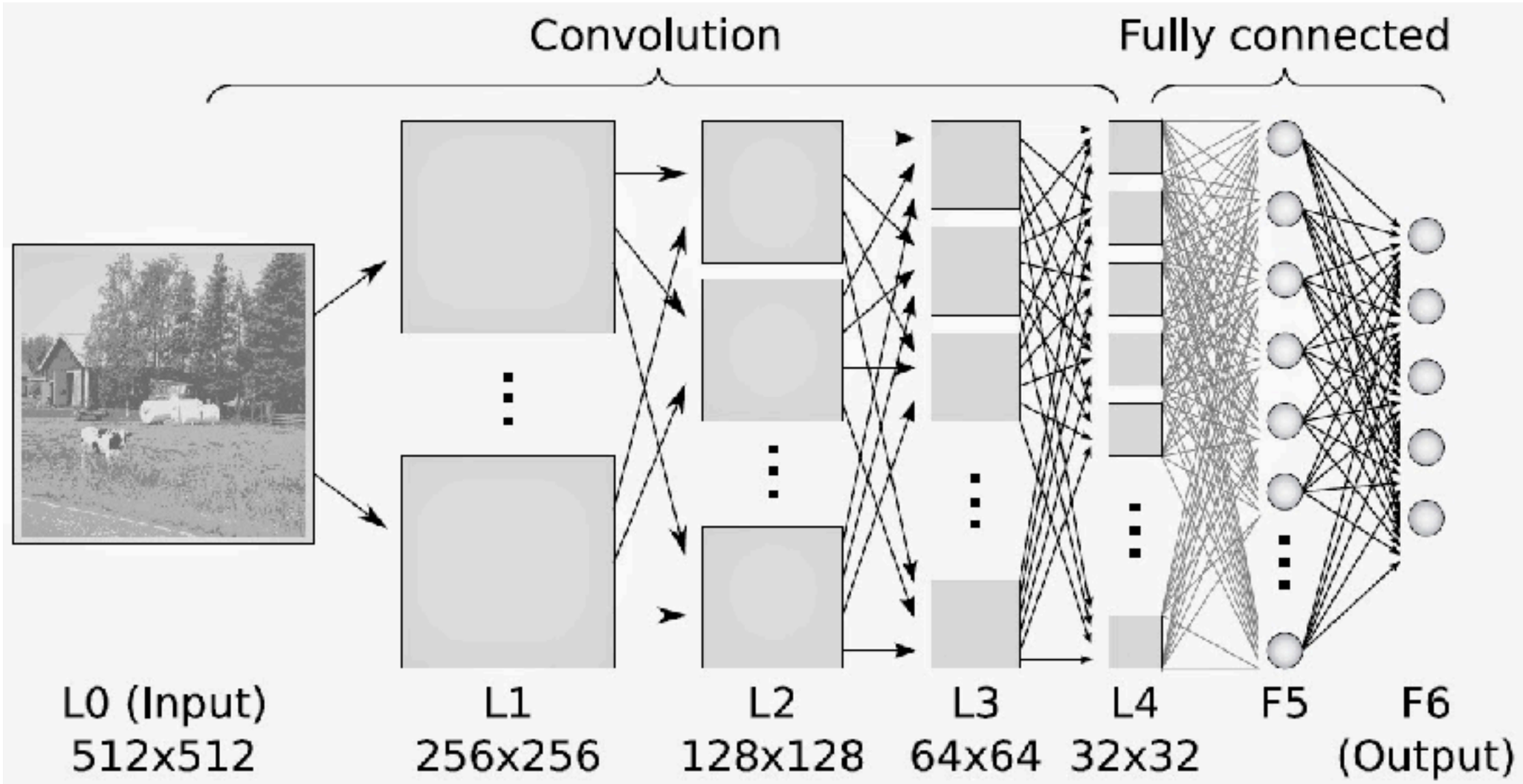
- Each edge is weighted
- Train this by randomly initializing weights, inputting values, and nudging the waits to better fit the training data
- With a single layer, can only get simple ax + bx +cx… formulas

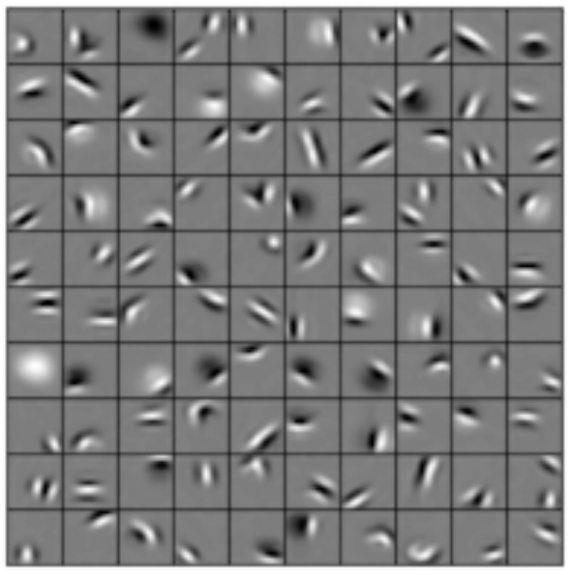
Example: Value of a car

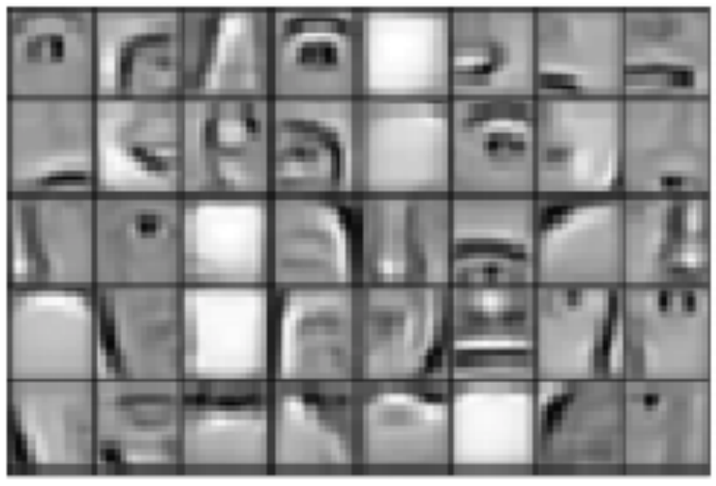


- Adding multiple layers before the output = *Deep* neural net
- Can represent important patterns amongst different inputs

Convolutional Neural Nets can detect the “inputs” and internal layers







Neural nets in practice

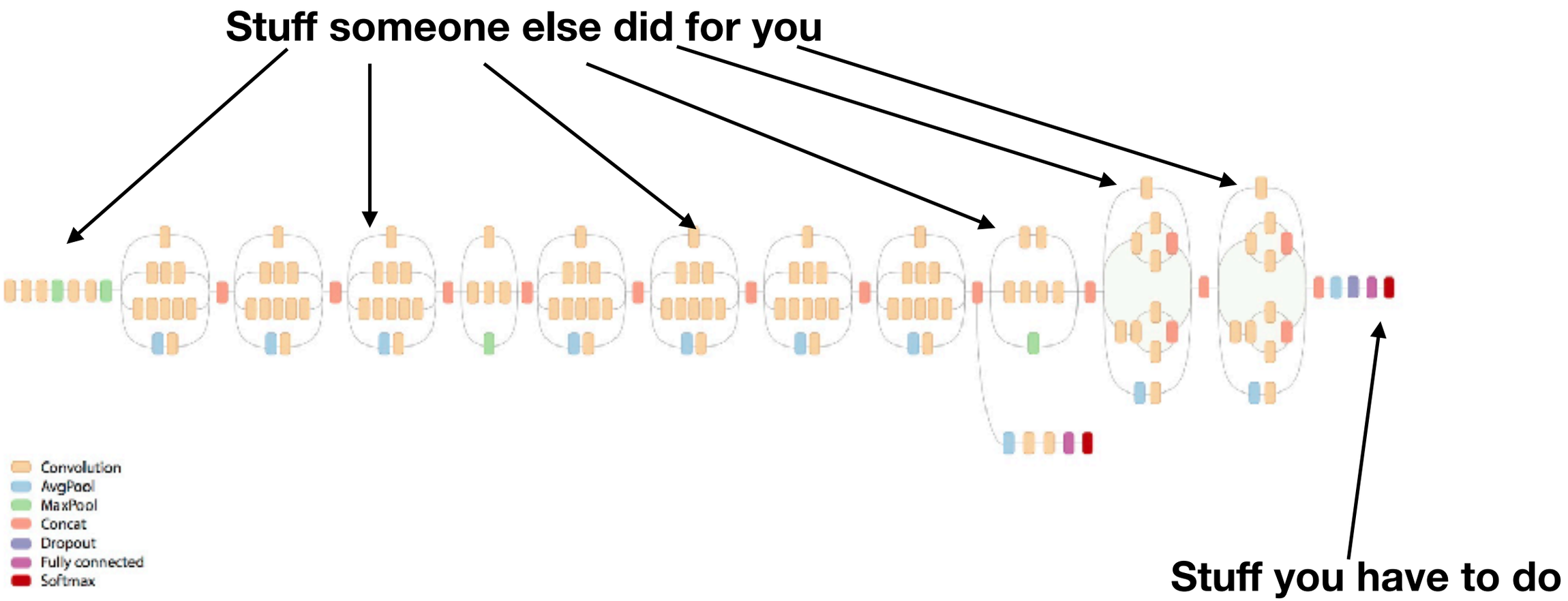
- Conceptualized in the early 1900s, just not enough computing power

- Computing power has obvious caught up recently, but implementing this requires incredible complex understanding of Machine Learning

Tensorflow makes it easy

- (Relatively) simple API in lots of language. Easiest in Python.
- Lots of open-sourced ML models (https://github.com/tensorflow/models)
  - Predicting future frames in a video
  - Image -> Text extraction
  - Name recognition
  - etc.

Transfer Learning



- Open-source models that are already designed on generic datasets to recognize characteristics
- You just need to retrain the final layer to work with your categories
- This means it's easy *and* fast

Retraining

- Initial layers have already been trained on large, generic data sets
- You provide a more specific data set
    - Training data: Generate the model
    - Test data: Check accuracy of the model, “nudge” the weights to refit the model

Simple overview

1. Collect data
2. Clean data
3. Retrain model
4. Use model
5. Celebrate! 🎉

Example:

Hotdog or Not Hotdog using Inception v3

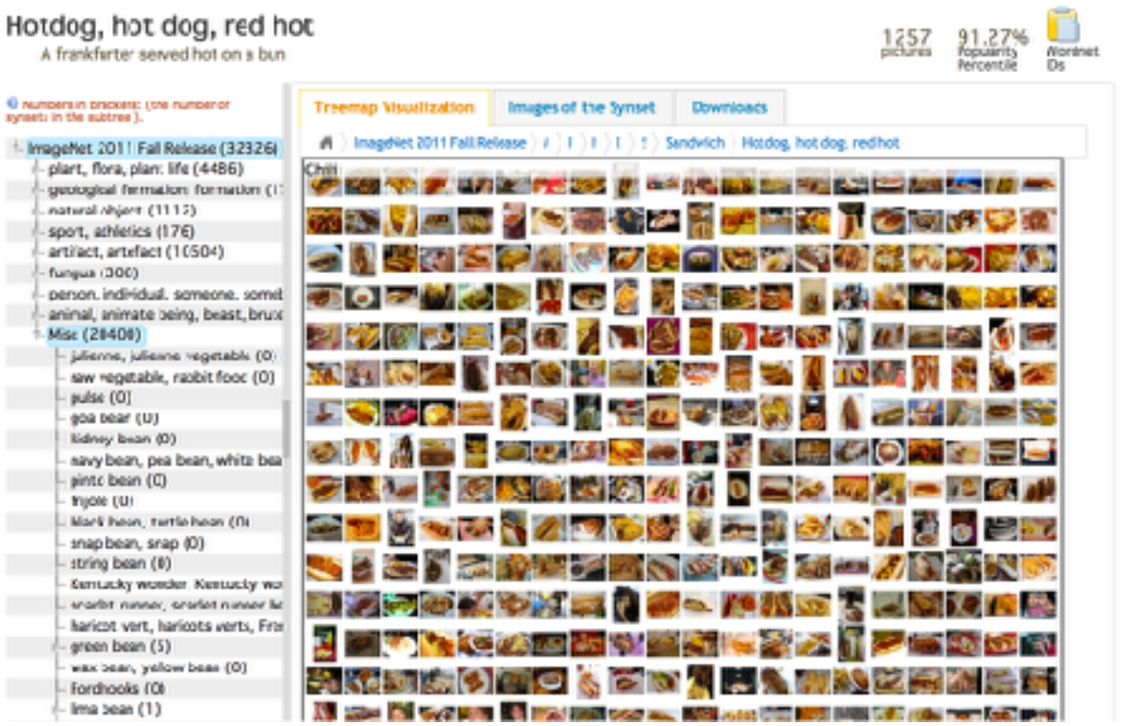
Step 0: Installation/setup

- Python + Pip + Virtualenv
- You can also do this with docker + Google's Bazel build tool, and in other languages, but this is the most robust & easiest way

```
mkvirtualenv hotdog
pip install — upgrade tensorflow
source ~/.virtualenvs/hotdog/bin/activate
```

Step 1: Collect data

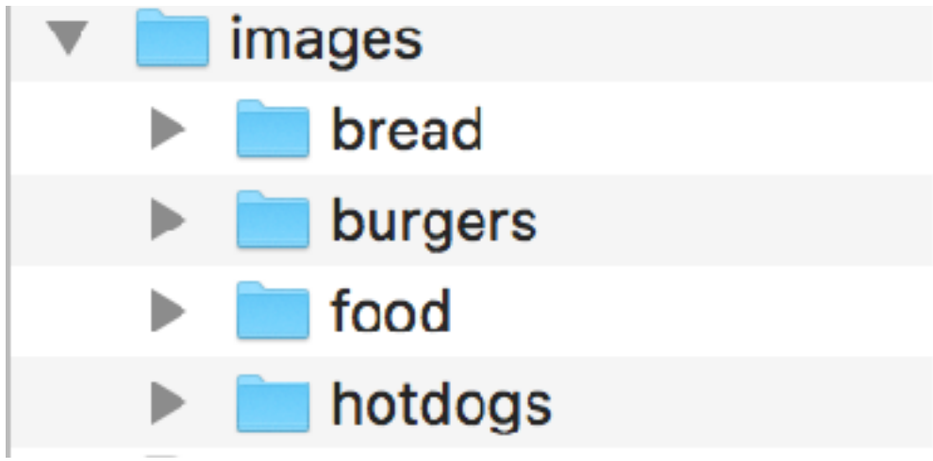
IMAGENET



Download URLs of images in the synset

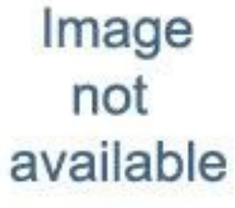
URLs

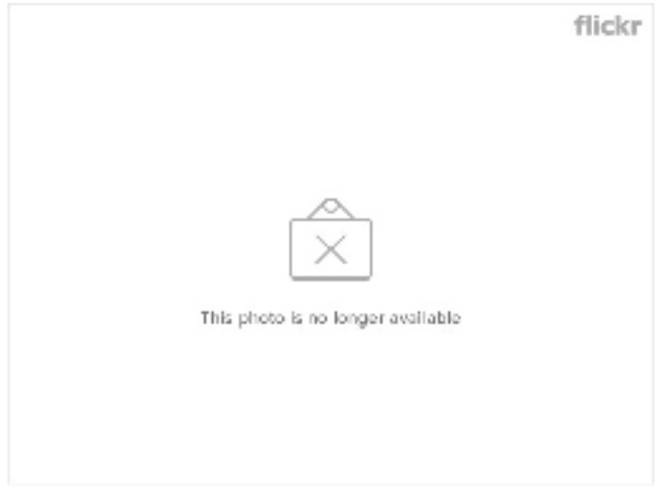
```python
import urllib
import os
import json

PARENT_DIR = './images'
DATA_SOURCE = 'data.json'

def get_data():
    with open(DATA_SOURCE) as data:
        return json.load(data)


def collect_images(data):
    """
    Expected format:
        {
            "subdir name": [
                url1,
                url2,
                ...
            ]
        }
    """
    for subdir in data.keys():
        print 'PROCESSING ' + subdir
        print '='*80

        mkdir_if_not_exists(subdir)

        unique_id = 0

        for image_url in data[subdir]:
            unique_id += 1
            image_name = "%s_%s" % (subdir, unique_id)
            image_dest = os.path.join(PARENT_DIR, subdir, image_name)
            download_image(image_url, image_dest)


def mkdir_if_not_exists(dir_name):
    path = os.path.join(PARENT_DIR, dir_name)
    if not os.path.exists(path):
        os.makedirs(path)


def download_image(image_url, image_dest):
    file_extension = image_url.split('.')[-1]
    image_dest = "%s.%s" % (image_dest, file_extension)

    print "Downloading %s to %s" % (image_url, image_dest)
    try:
        urllib.urlretrieve(image_url, image_dest)
        print "Success!"
    except Exception as e:
        print "Got error!"
        print (str(e))


if __name__ == '__main__':
    collect_images(get_data())
```



Step 2: Clean Data

- Will have lots of “invalid” images
- You can rotate, resize, transform images so that you have a larger data set
- Configure bounding boxes





Step 3: Retrain model

tensorflow / tensorflow

Watch ▾ 5,455 | Star 60,988 | Fork 29,337

<> Code | Issues 551 | Pull requests 65 | Projects 0 | Insights ▾

Branch: master ▾ tensorflow / tensorflow / examples / image_retraining / retrain.py

Find file | Copy path

AstrerAI fixed type in intermediate word 5d8a1af 5 days ago

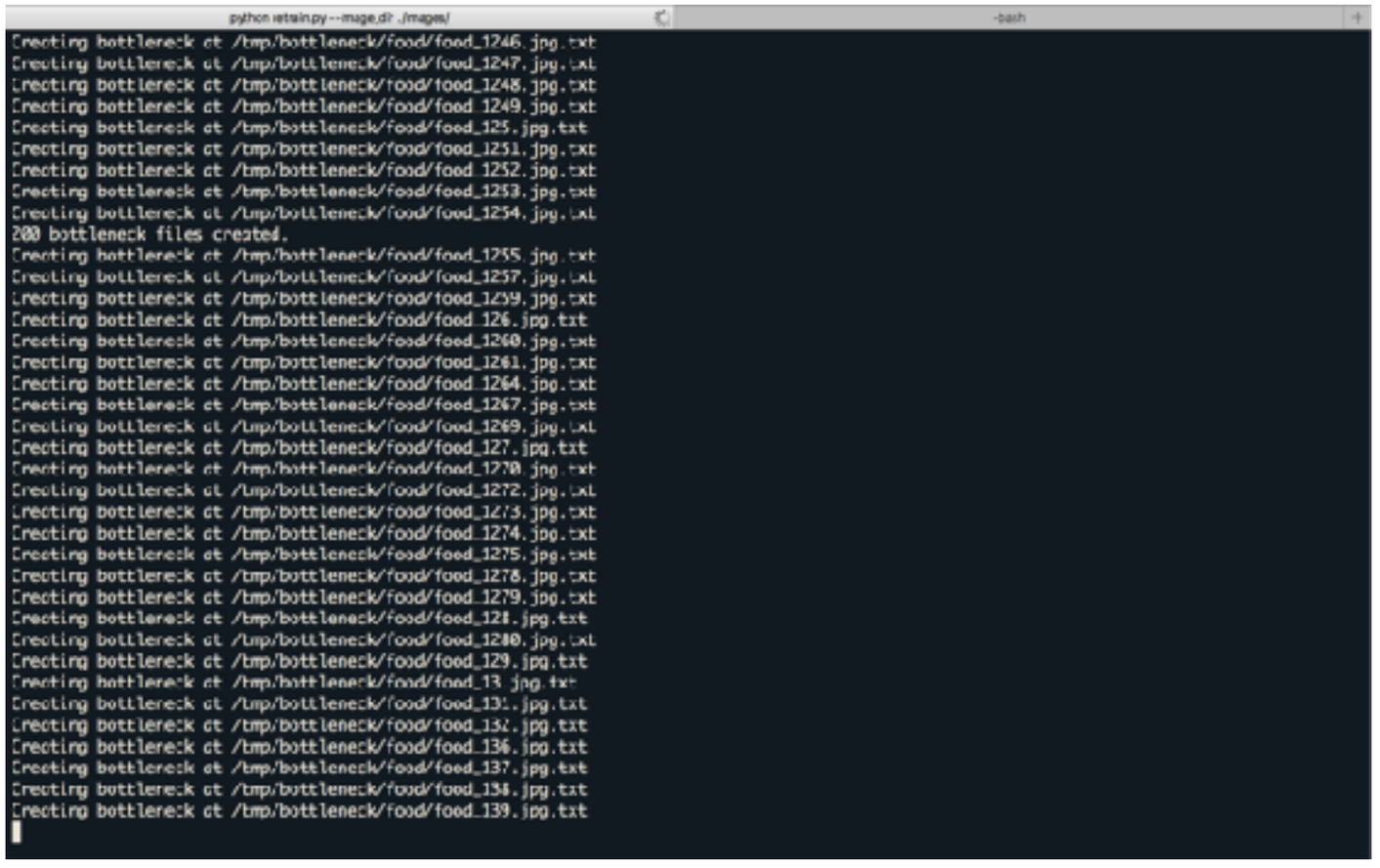
15 contributors

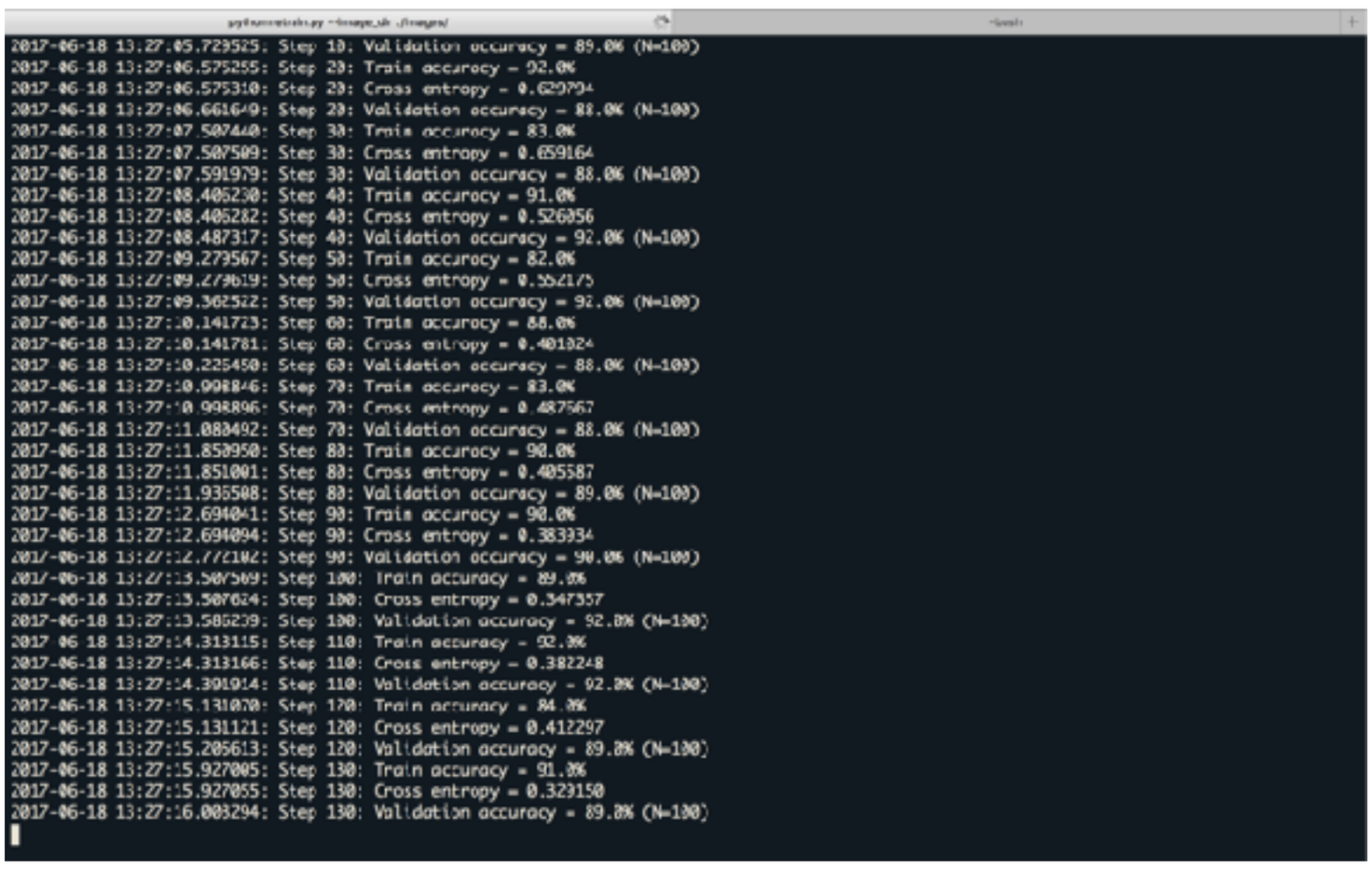
1139 lines (989 sloc) 44.2 KB

Raw | Blame | History

```
1  # Copyright 2015 The TensorFlow Authors. All Rights Reserved.
2  #
3  # Licensed under the Apache License, Version 2.0 (the "License");
4  # you may not use this file except in compliance with the License.
5  # You may obtain a copy of the License at
6  #
7  #     http://www.apache.org/licenses/LICENSE-2.0
8  #
9  # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14 # ==============================================================================
15 """Simple transfer learning with an Inception v3 architecture model.
16
17 With support for TensorBoard.
18
19 This example shows how to take a Inception v3 architecture model trained on
20 ImageNet images, and train a new top layer that can recognize other classes of
21 images.
22
23 The top layer receives as input a 2048-dimensional vector for each image. We
24 train a softmax layer on top of this representation. Assuming the softmax layer
```

```
python retrain.py ——image_dir ./images
```


python retrain.py --image_dir ./images/
-bash
Creating bottleneck at /tmp/bottleneck/food/food_1246.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1247.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1248.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1249.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_125.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1251.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1252.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1253.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1254.jpg.txt
200 bottleneck files created.
Creating bottleneck at /tmp/bottleneck/food/food_1255.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1257.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1259.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_126.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1260.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1261.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1264.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1267.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1269.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_127.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1270.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1272.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1273.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1274.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1275.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1278.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1279.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_128.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_1280.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_129.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_13.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_131.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_132.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_136.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_137.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_138.jpg.txt
Creating bottleneck at /tmp/bottleneck/food/food_139.jpg.txt


2017-06-18 13:27:05.723525: Step 10: Validation accuracy = 89.0% (N=100)
2017-06-18 13:27:06.575255: Step 20: Train accuracy = 92.0%
2017-06-18 13:27:06.575310: Step 20: Cross entropy = 0.629794
2017-06-18 13:27:06.661649: Step 20: Validation accuracy = 88.0% (N=100)
2017-06-18 13:27:07.507440: Step 30: Train accuracy = 83.0%
2017-06-18 13:27:07.507509: Step 30: Cross entropy = 0.659164
2017-06-18 13:27:07.591979: Step 30: Validation accuracy = 88.0% (N=100)
2017-06-18 13:27:08.405230: Step 40: Train accuracy = 91.0%
2017-06-18 13:27:08.405282: Step 40: Cross entropy = 0.526056
2017-06-18 13:27:08.487317: Step 40: Validation accuracy = 92.0% (N=100)
2017-06-18 13:27:09.279567: Step 50: Train accuracy = 82.0%
2017-06-18 13:27:09.279619: Step 50: Cross entropy = 0.554175
2017-06-18 13:27:09.362522: Step 50: Validation accuracy = 92.0% (N=100)
2017-06-18 13:27:10.141723: Step 60: Train accuracy = 88.0%
2017-06-18 13:27:10.141781: Step 60: Cross entropy = 0.401024
2017-06-18 13:27:10.225450: Step 60: Validation accuracy = 88.0% (N=100)
2017-06-18 13:27:10.998846: Step 70: Train accuracy = 83.0%
2017-06-18 13:27:10.998896: Step 70: Cross entropy = 0.487567
2017-06-18 13:27:11.083492: Step 70: Validation accuracy = 88.0% (N=100)
2017-06-18 13:27:11.850950: Step 80: Train accuracy = 90.0%
2017-06-18 13:27:11.851001: Step 80: Cross entropy = 0.405587
2017-06-18 13:27:11.935508: Step 80: Validation accuracy = 89.0% (N=100)
2017-06-18 13:27:12.694041: Step 90: Train accuracy = 90.0%
2017-06-18 13:27:12.694094: Step 90: Cross entropy = 0.383934
2017-06-18 13:27:12.777102: Step 90: Validation accuracy = 90.0% (N=100)
2017-06-18 13:27:13.507569: Step 100: Train accuracy = 89.0%
2017-06-18 13:27:13.507624: Step 100: Cross entropy = 0.347357
2017-06-18 13:27:13.585239: Step 100: Validation accuracy = 92.0% (N=100)
2017-06-18 13:27:14.313115: Step 110: Train accuracy = 92.0%
2017-06-18 13:27:14.313166: Step 110: Cross entropy = 0.382248
2017-06-18 13:27:14.391014: Step 110: Validation accuracy = 92.0% (N=100)
2017-06-18 13:27:15.131070: Step 120: Train accuracy = 84.0%
2017-06-18 13:27:15.131121: Step 120: Cross entropy = 0.412297
2017-06-18 13:27:15.205613: Step 120: Validation accuracy = 89.0% (N=100)
2017-06-18 13:27:15.927005: Step 130: Train accuracy = 91.0%
2017-06-18 13:27:15.927055: Step 130: Cross entropy = 0.323150
2017-06-18 13:27:16.003294: Step 130: Validation accuracy = 89.0% (N=100)

Step 4: Use model

```
import tensorflow as tf, sys

# constants
LABELS_FILE = './output_labels.txt'
GRAPH_FILE = './output_graph.pb'

# load image
image_path = sys.argv[1]
image_data = tf.gfile.FastGFile(image_path, 'rb').read()

# parse labels file
label_lines = [ line.rstrip() for line in tf.gfile.GFile(LABELS_FILE) ]

# load the trained model from the output graph
with tf.gfile.FastGFile(GRAPH_FILE, 'rb') as f:
    graph = tf.GraphDef()
    graph.ParseFromString(f.read())
    _ = tf.import_graph_def(graph, name='')

with tf.Session() as sess:
    # pass the image into the graph as input
    tensor = sess.graph.get_tensor_by_name('final_result:0')

    predictions = sess.run(tensor, { 'DecodeJpeg/contents:0': image_data })
    sorted_predictions = predictions[0].argsort()[-len(predictions[0]):][::-1]

    for node in sorted_predictions:
        label = label_lines[node]
        score = predictions[0][node]
        print "%s: %.5f" % (label, score)
```

Step 5: Celebrate 🎉

(and turn it into a service)

```
import os
import time

from flask import Flask, request, jsonify

from download_images import download_image
from label_image import get_predictions

app = Flask(__name__)

@app.route('/is_hotdog', methods=['POST'])
def is_hotdog():
    image_url = request.args.get('image')
    image_location = os.path.join('tmp', "download_%s" % int(time.time()))
    image_location = download_image(image_url, image_location)
    if image_location:
        try:
            predictions = get_predictions(image_location)
            is_hotdog = make_prediction(predictions)
            return jsonify(is_hotdog=is_hotdog, scores=predictions)
        except Exception as e:
            print(str(e))
            return jsonify({ 'error': 'Error processing image' }), 500
    else:
        return jsonify({ 'error': 'Couldn\'t download image' }), 500


def make_prediction(predictions):
    return predictions[0][0] == 'hotdogs' and predictions[0][1] >= 0.8 \
            and predictions[0][1] - predictions[1][1] > 0.2


if __name__ == '__main__':
    app.run()
```

Code & Slides:

benmusch/honhaas