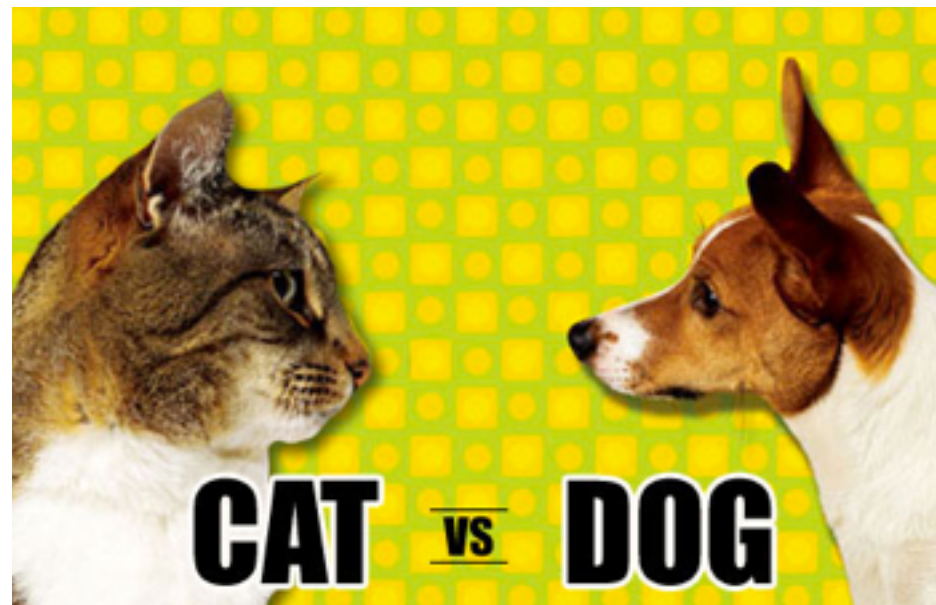


Deep Learning: and Deep Data-Science



ROYAL INSTITUTE
OF TECHNOLOGY

roelof@kth.se

www.csc.kth.se/~roelof/

12 May 2015

Graph Technologies R&D

roelof@graph-technologies.com

slides online at:

<https://www.slideshare.net/roelofp/deep-learning-as-a-catdog-detector>

BUT FIRST...

are you a...

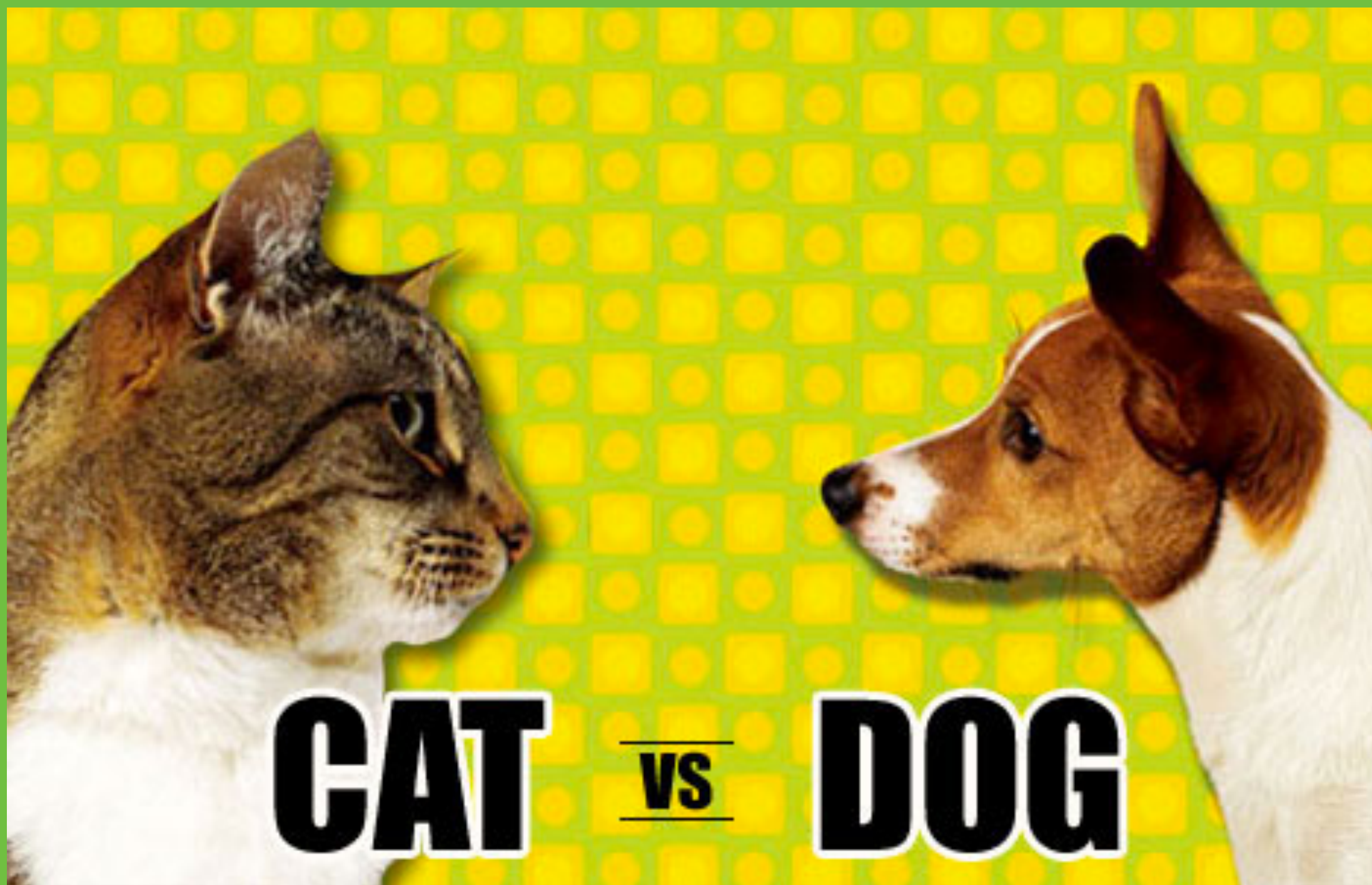
CAT PERSON?



DOG PERSON?



in the next few minutes
we'll be making a



DETECTOR

main Libraries



- scikit-learn (machine learning)
<http://scikit-learn.org>
- caffe (deep learning) – for training deep neural nets
(for today: loading a pre-trained one)
<http://caffe.berkeleyvision.org>
- theano (efficient gpu-powered math)
<http://www.deeplearning.net/software/theano/>
- ipython notebook
<http://ipython.org/notebook.html>



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: *SVM, nearest neighbors, random forest, ...* — [Examples](#)

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: *SVR, ridge regression, Lasso, ...* — [Examples](#)

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: *k-Means, spectral clustering, mean-shift, ...* — [Examples](#)

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: *PCA, feature selection, non-negative matrix factorization.* — [Examples](#)

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: *grid search, cross validation, metrics.* — [Examples](#)

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: *preprocessing, feature extraction.* — [Examples](#)

Caffe

Deep learning framework
by the [BVLC](#)

Created by

[Yangqing Jia](#)

Lead Developer

[Evan Shelhamer](#)

 [View On GitHub](#)

Caffe

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center ([BVLC](#)) and by community contributors.

[Yangqing Jia](#) created the project during his PhD at UC Berkeley. Caffe is released under the [BSD 2-Clause license](#).

Check out our web image classification [demo!](#)

Why Caffe?

Expressive architecture encourages application and innovation. Models and optimization are defined by configuration without hard-coding. Switch between CPU and GPU by setting a single flag to train on a GPU machine then deploy to commodity clusters or mobile devices.

Extensible code fosters active development. In Caffe's first year, it has been forked by over 1,000 developers and had many significant changes contributed back. Thanks to these contributors the framework tracks the state-of-the-art in both code and models.

Speed makes Caffe perfect for research experiments and industry deployment. Caffe can process **over 60M images per day** with a single NVIDIA K40 GPU*. That's 1 ms/image for inference and 4 ms/image for learning. We believe that Caffe is the fastest convnet implementation available.

Community: Caffe already powers academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Join our community of brewers on the [caffe-users group](#) and [Github](#).

* With the ILSVRC2012-winning [SuperVision](#) model and caching IO. Consult performance [details](#).

Welcome

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

- **tight integration with NumPy** – Use `numpy.ndarray` in Theano-compiled functions.
- **transparent use of a GPU** – Perform data-intensive calculations up to 140x faster than with CPU.(float32 only)
- **efficient symbolic differentiation** – Theano does your derivatives for function with one or many inputs.
- **speed and stability optimizations** – Get the right answer for `log(1+x)` even when `x` is really tiny.
- **dynamic C code generation** – Evaluate expressions faster.
- **extensive unit-testing and self-verification** – Detect and diagnose many types of mistake.

Theano has been powering large-scale computationally intensive scientific investigations since 2007. But it is also approachable enough to be used in the classroom (IFT6266 at the University of Montreal).

News

- We support [cuDNN](#) if it is installed by the user.
- Open Machine Learning Workshop 2014 [presentation](#).
- Colin Raffel [tutorial on Theano](#).
- Ian Goodfellow did a [12h class with exercises on Theano](#).

theano

Table Of Contents

[Welcome](#)
[News](#)
[Download](#)
[Status](#)
[Citing Theano](#)
[Documentation](#)
[Community](#)

Next topic

Release Notes

This Page

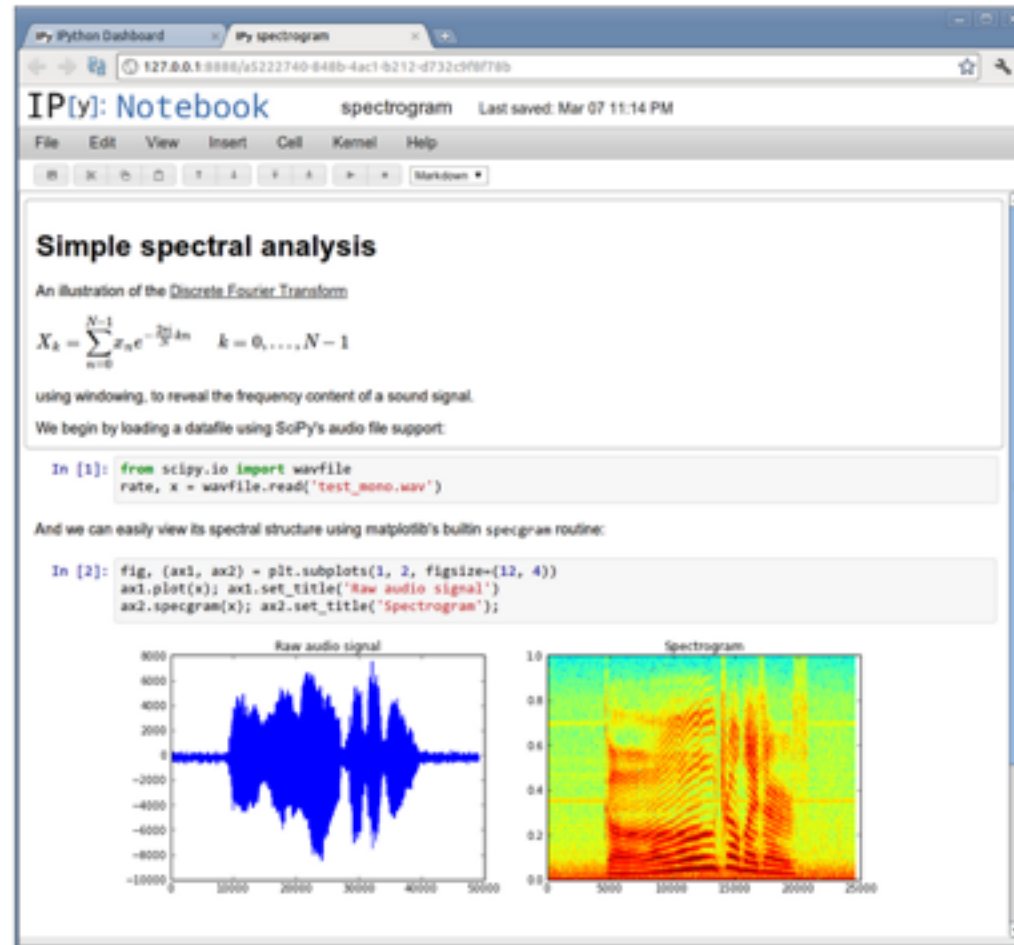
Show Source

Quick search

Enter search terms or a module, class or function name.

The IPython Notebook

The IPython Notebook is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media, as shown in this example session:



It aims to be an agile tool for both exploratory computation and data analysis, and provides a platform to support **reproducible research**, since all inputs and outputs may be stored in a one-to-one way in notebook documents.

VERSIONS

Stable

3.1 – April 2015

[Install](#)

Development

4.0.dev

[GitHub](#)

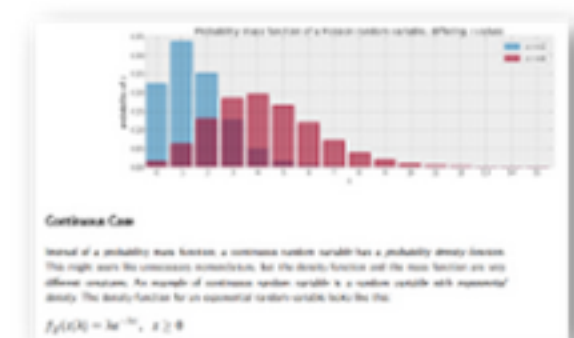
Offline Docs

All Versions

[GitHub](#)

NOTEBOOK VIEWER

Share your notebooks



**Code is ahead, soon...
I promise :)**

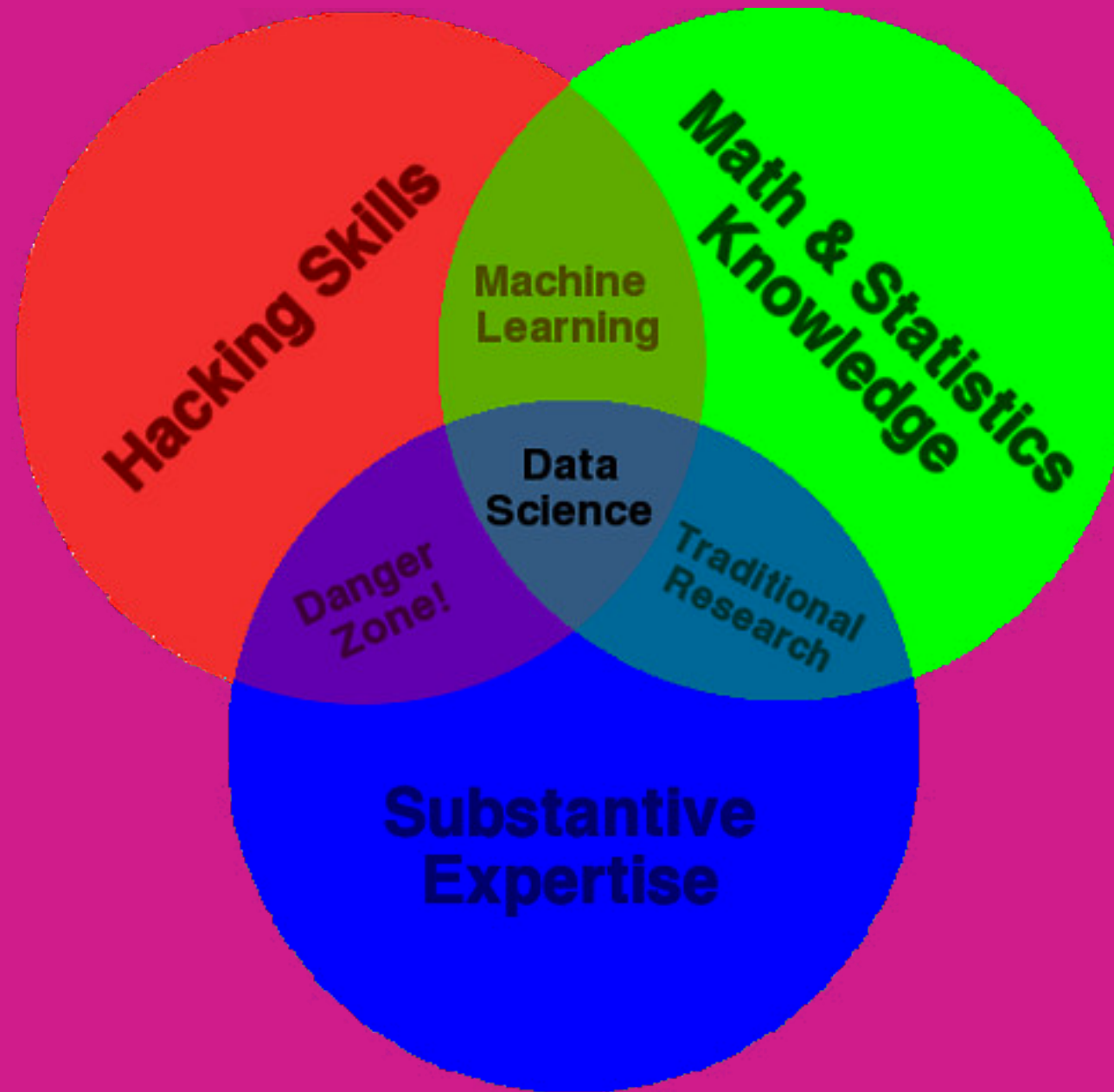
**BEAR
WITH
ME**



Data Science ?

“Data science is clearly a blend of the **hackers' art, statistics** and **machine learning...**”

—Hilary Mason & Chris Wiggins, 2010



(Drew Connoway 2010)

> Features = Awesomeness

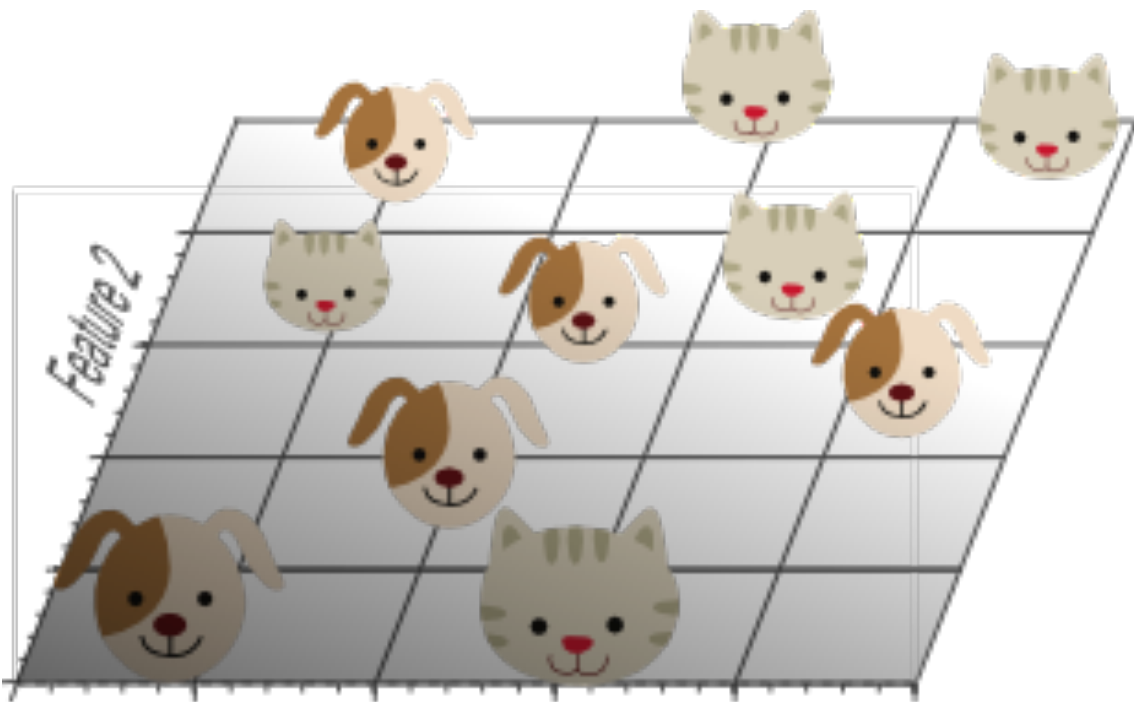


1 feature

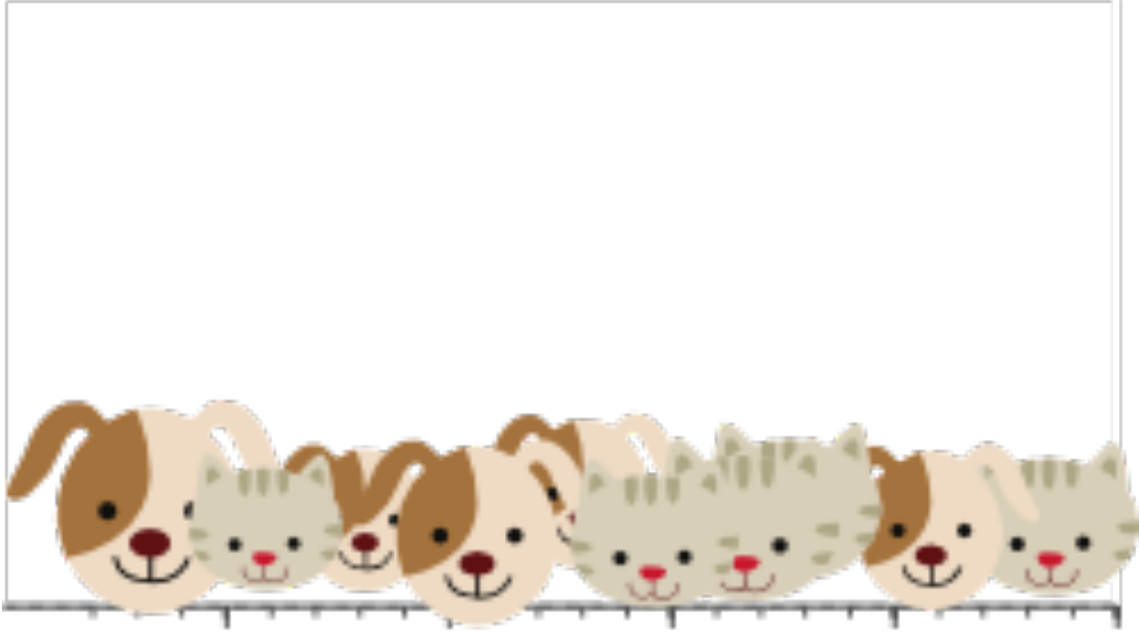
> Features = Awesomeness



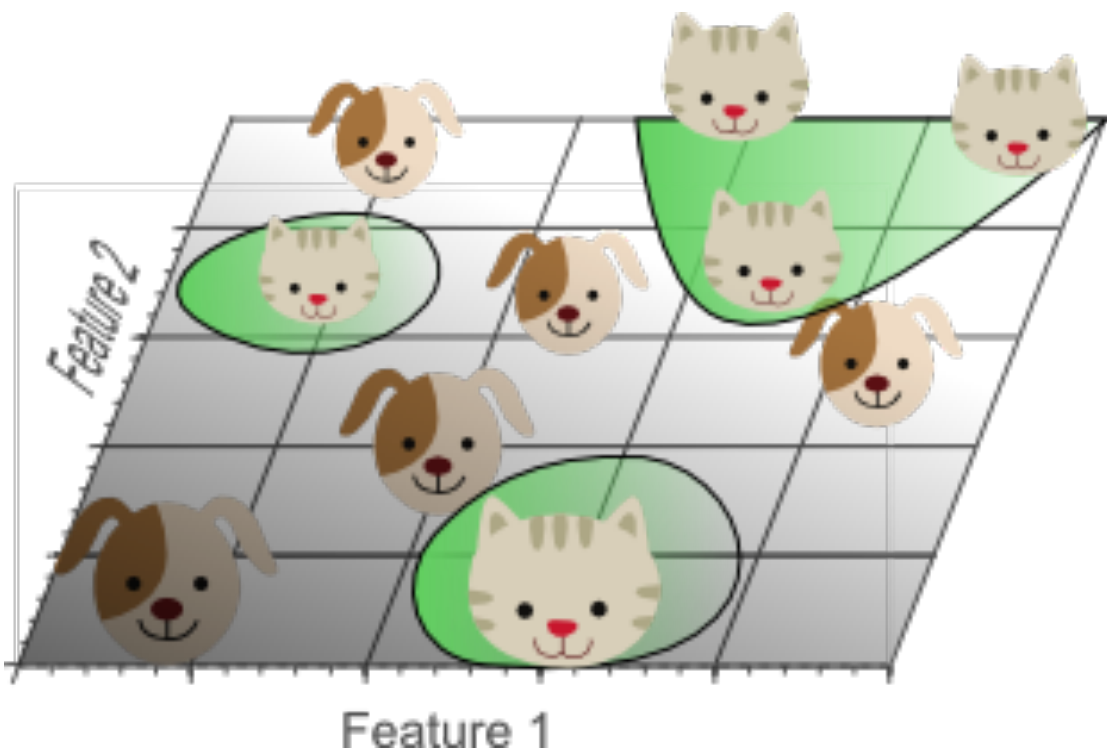
1 feature



2 features



1 feature



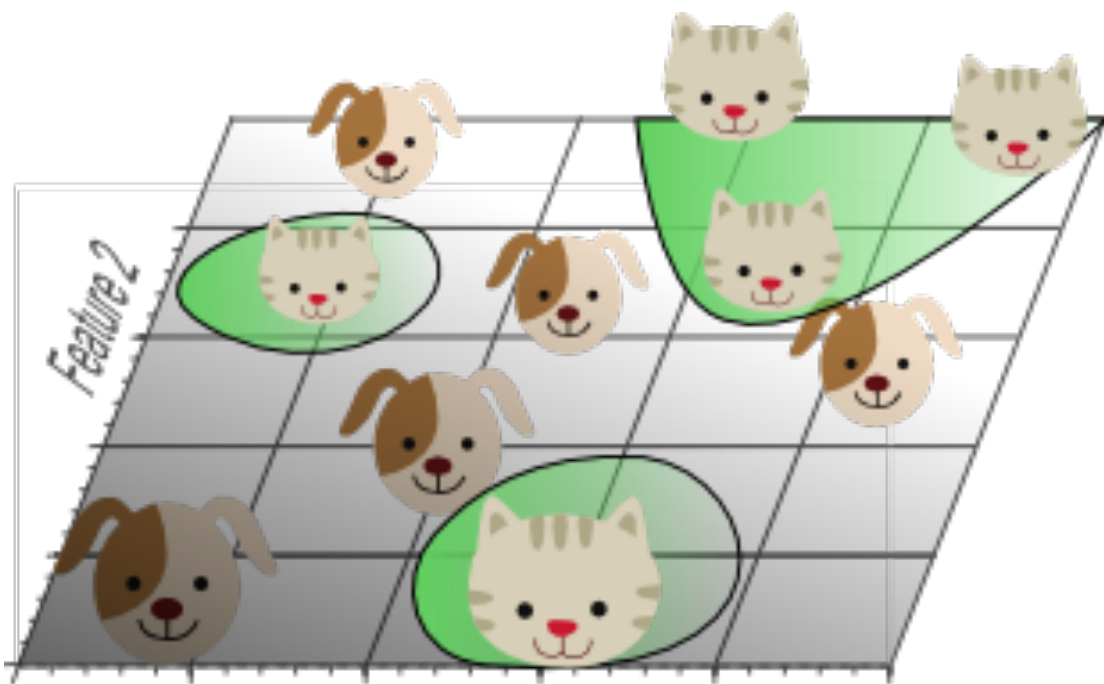
2 features

too few features/dimensions = overfitting

> Features = Awesomeness

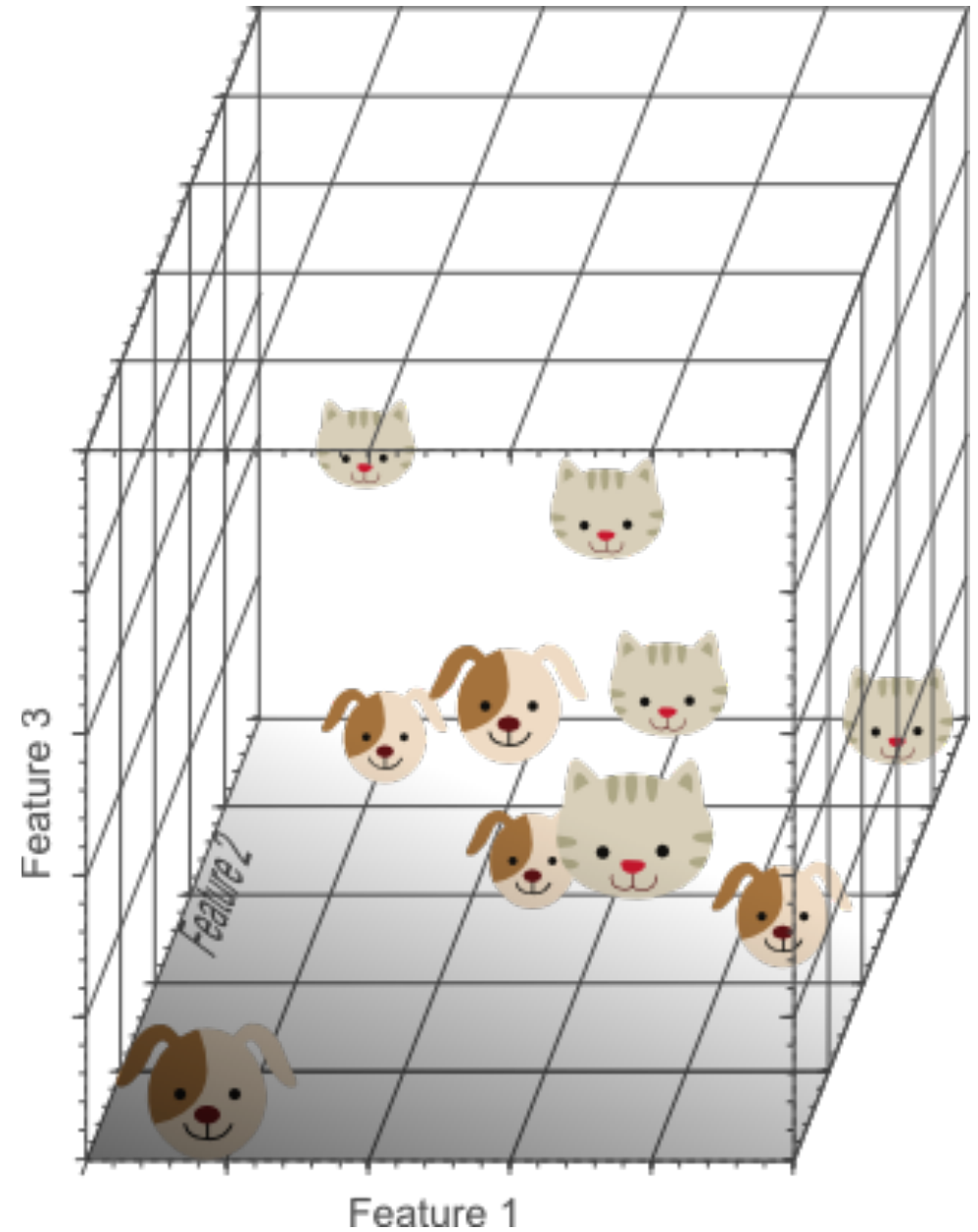


1 feature



2 features

too few features/dimensions = overfitting

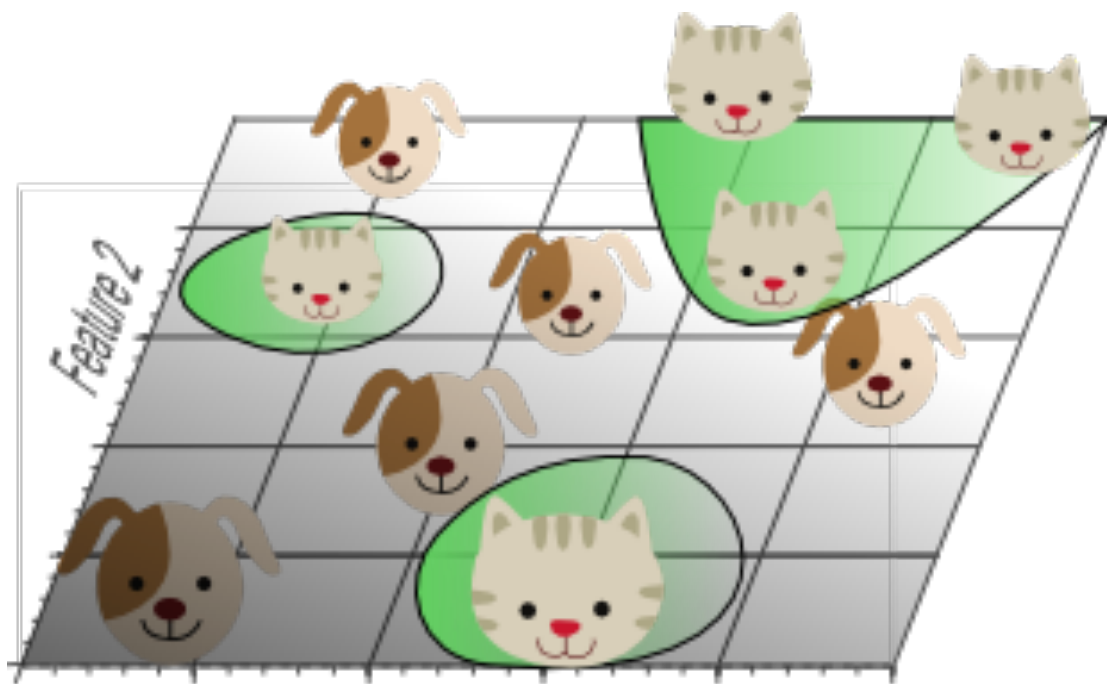


3 features

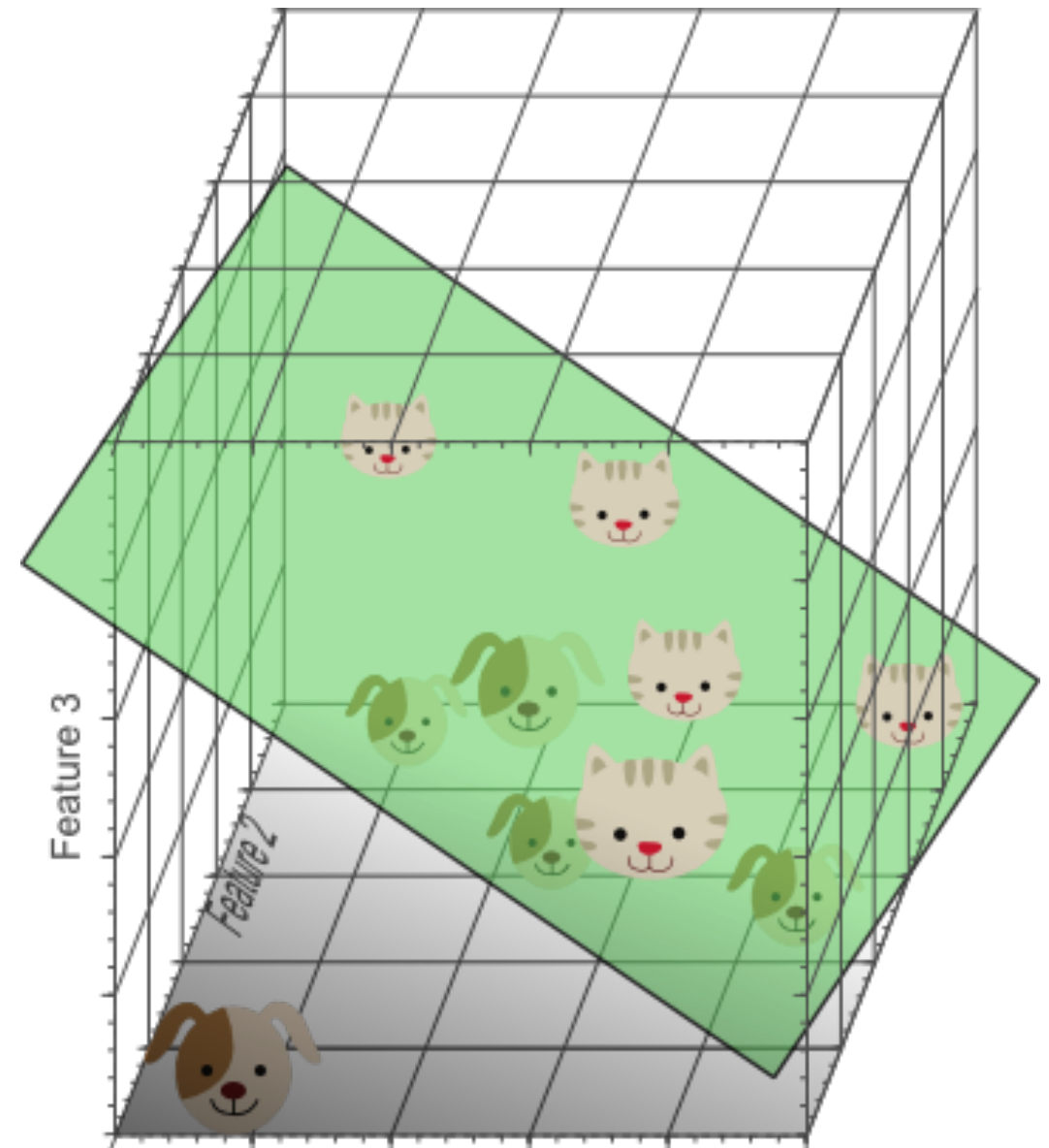
More Features = Awesomeness!



1 feature

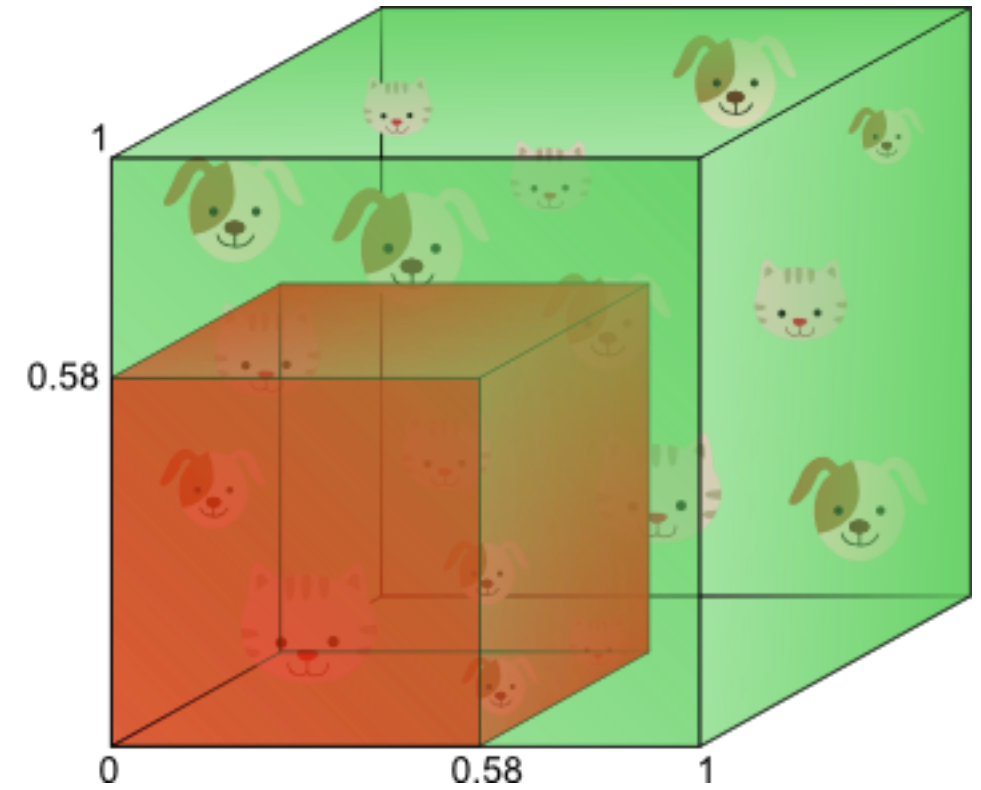
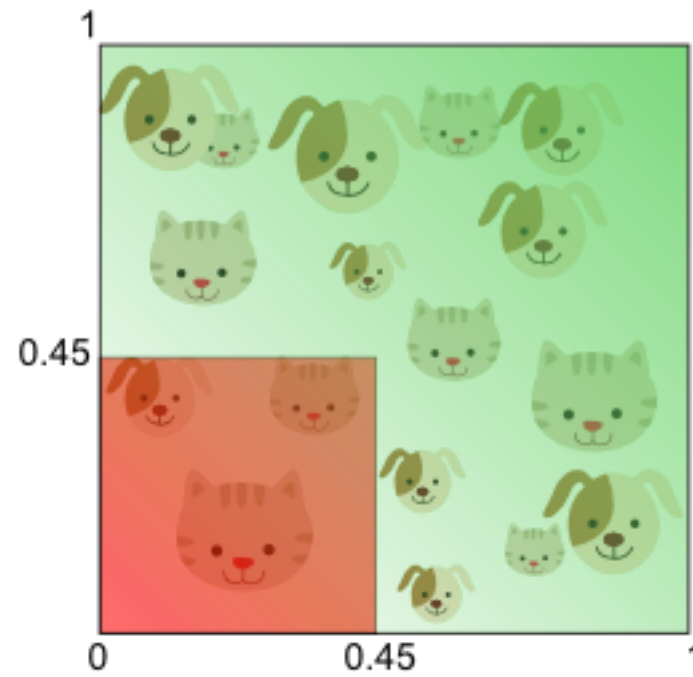


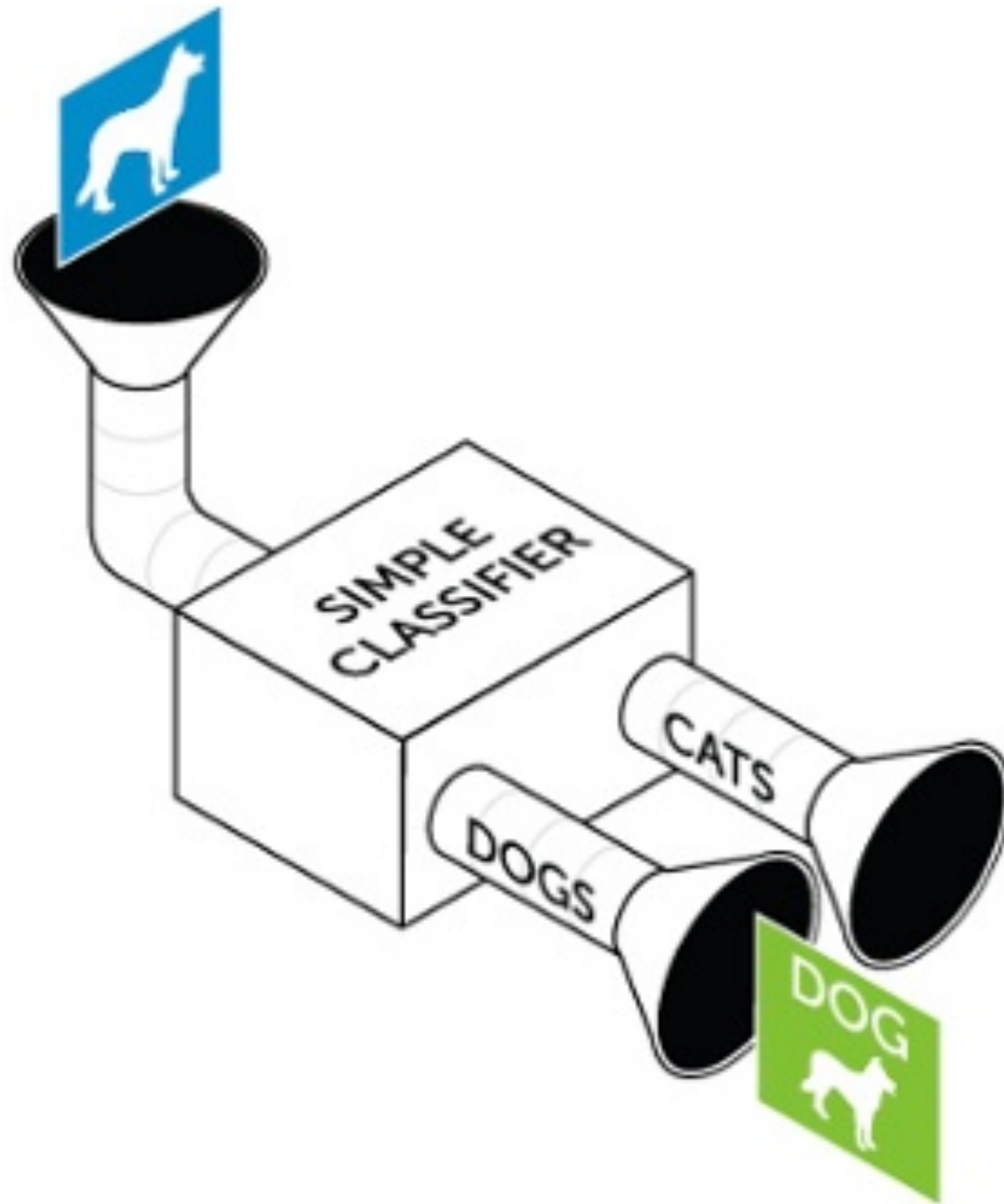
2 features



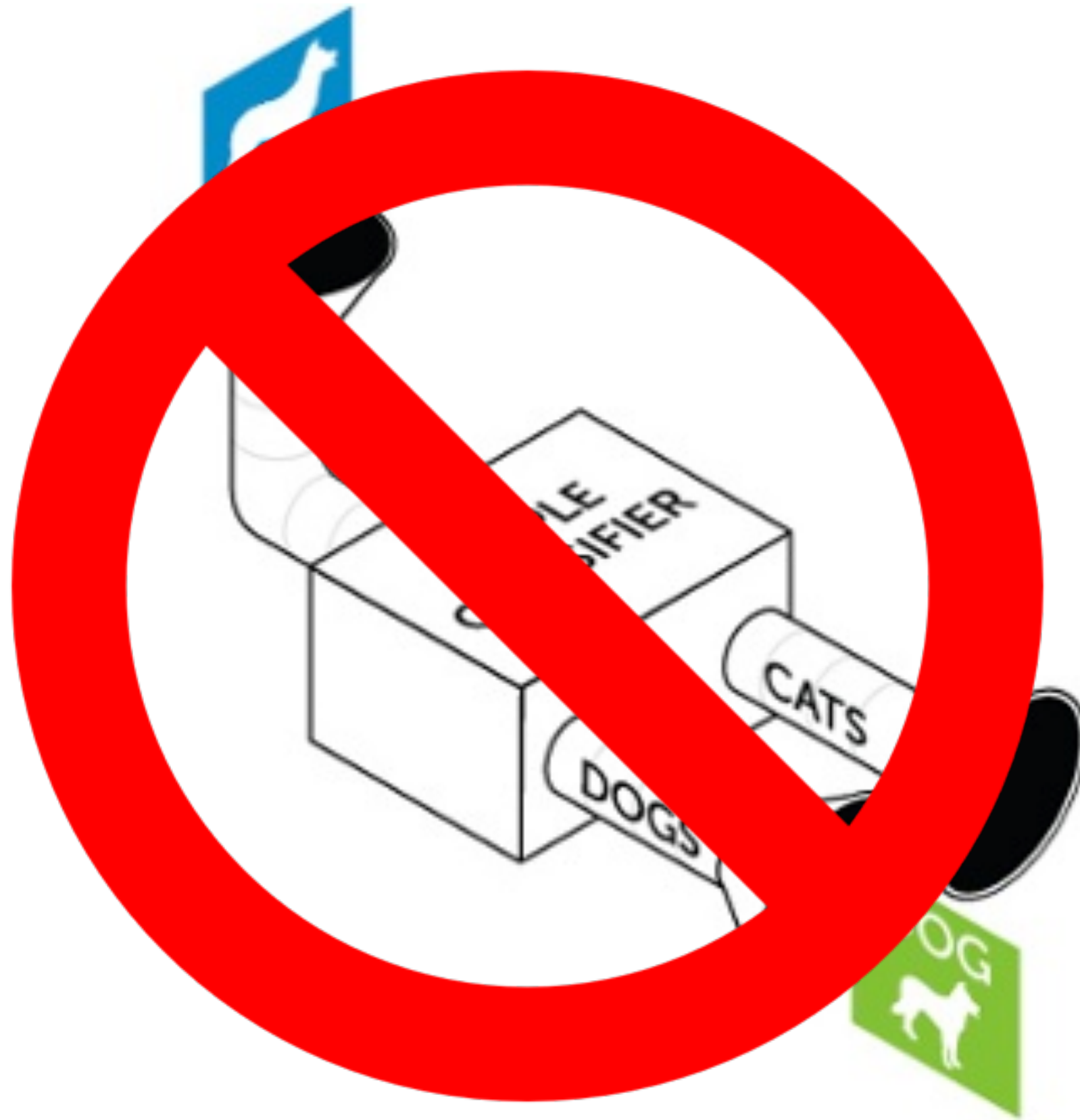
3 features

++ Data Needs also grow!





(picture by Dato)

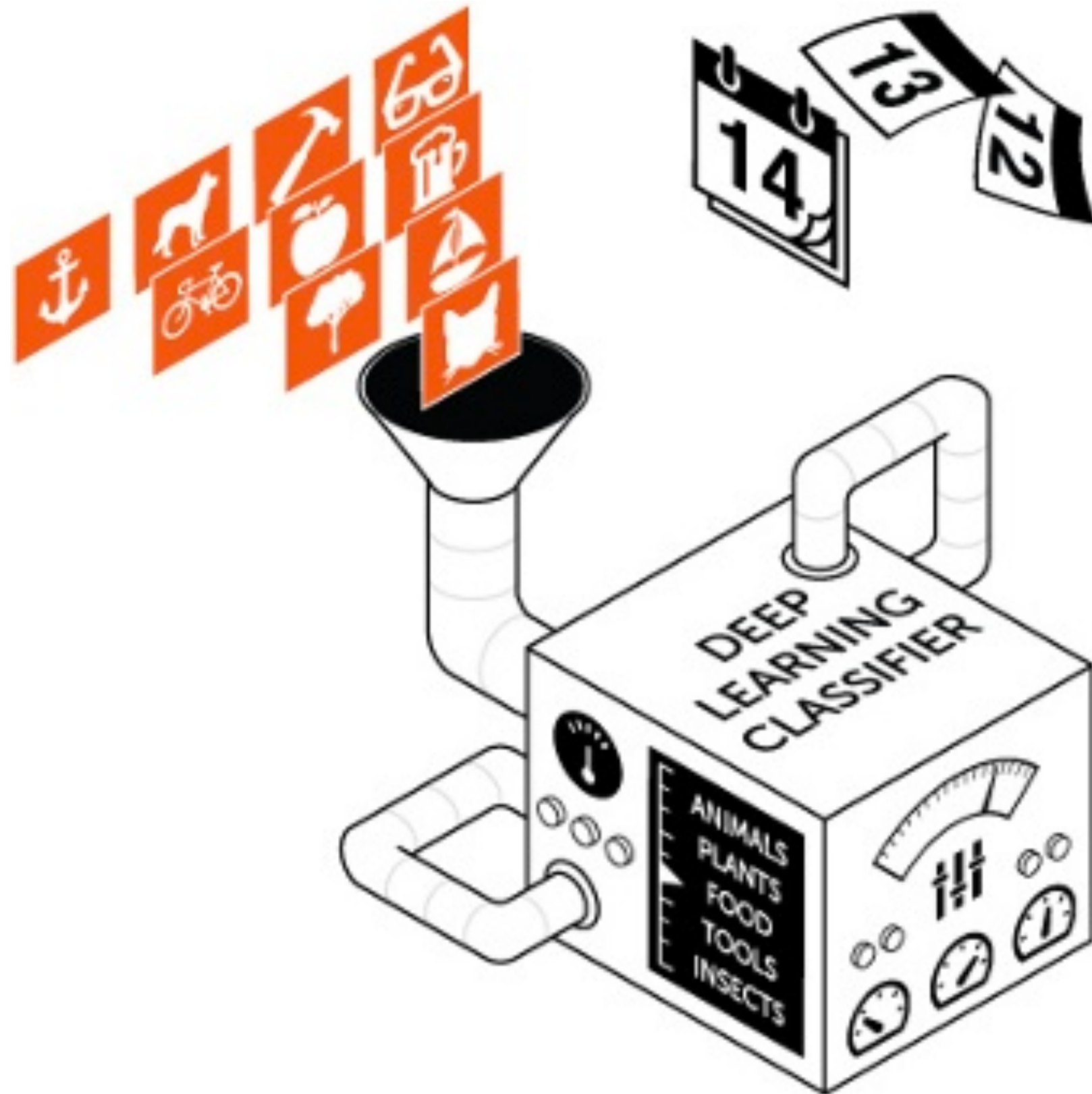


(picture by Dato)

Deep Learning?

- A host of statistical machine learning techniques
- Enables the automatic learning of feature hierarchies
- Generally based on artificial neural networks

Deep Learning



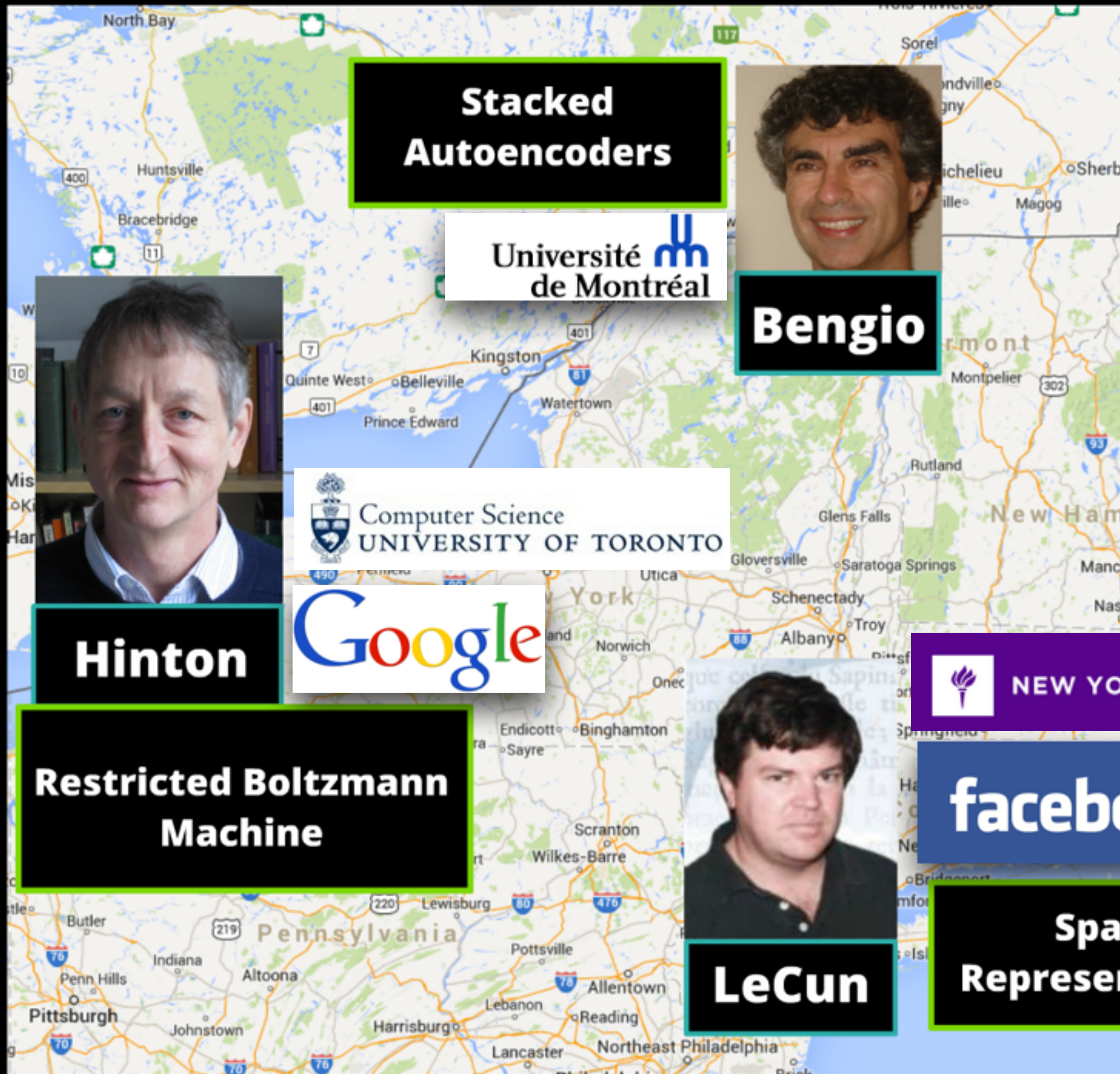
(picture by Dato)

Deep Learning?

- Manually designed features are often over-specified, incomplete and take a long time to design and validate
- Learned Features are easy to adapt, fast to learn
- Deep learning provides a very flexible, (almost?) universal, learnable framework for representing world, visual and linguistic information.
- Deep learning can learn unsupervised (from raw text/audio/images/whatever content) and supervised (with specific labels like positive/negative)

(as summarised by Richard Socher 2014)

2006+ : The Deep Learning Conspirators



**Stacked
Autoencoders**

Université 
de Montréal

Bengio

 Computer Science
UNIVERSITY OF TORONTO

Hinton





NEW YORK UNIVERSITY

**Restricted Boltzmann
Machine**

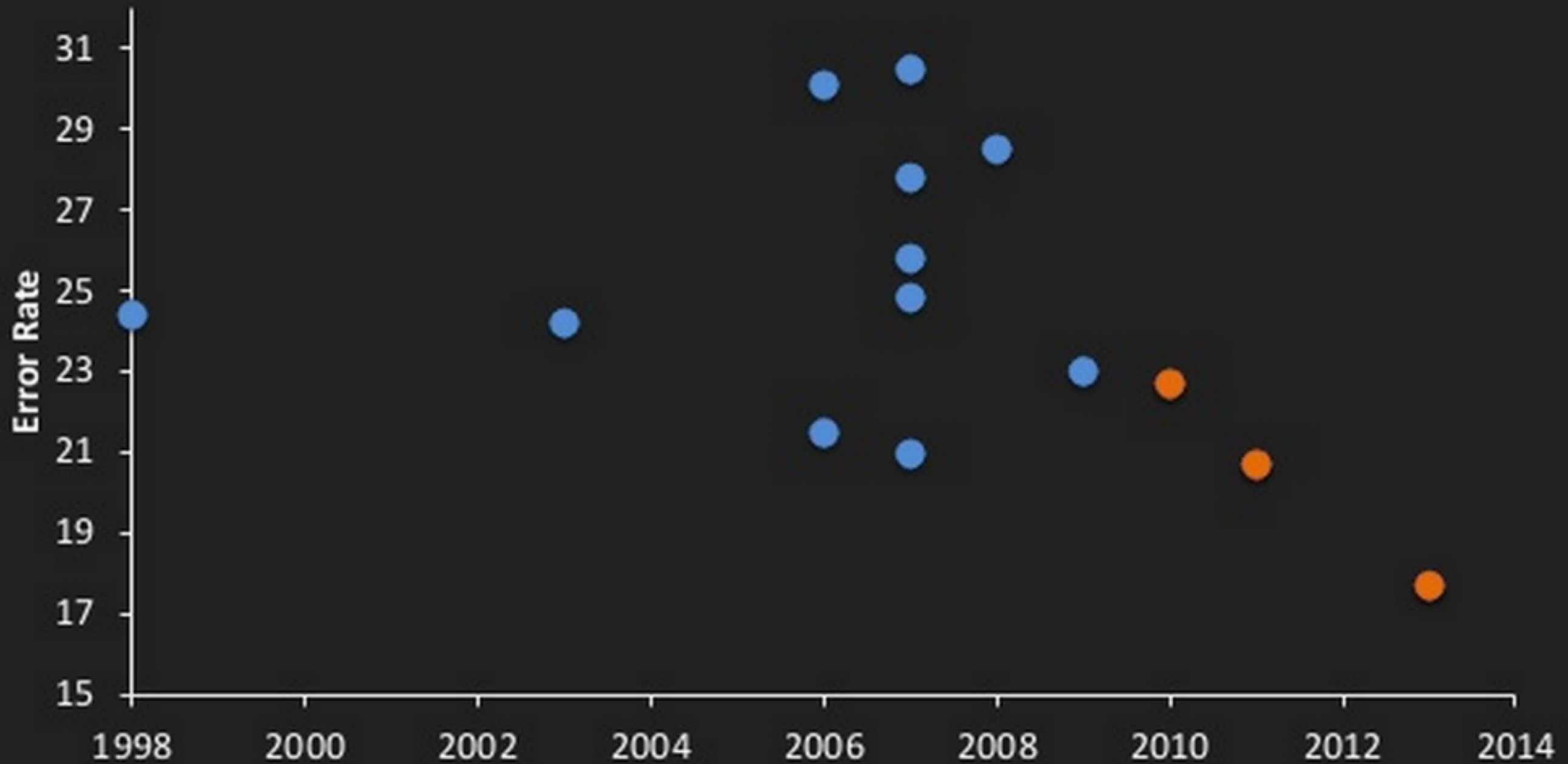
facebook

LeCun

**Sparse
Representations**

Audio Recognition

● Traditional ● Deep Learning



(chart by Clarifai)

Image Recognition



(chart by Clarifai)

Natural Language Processing

...

Natural Language Processing



DL? How ?

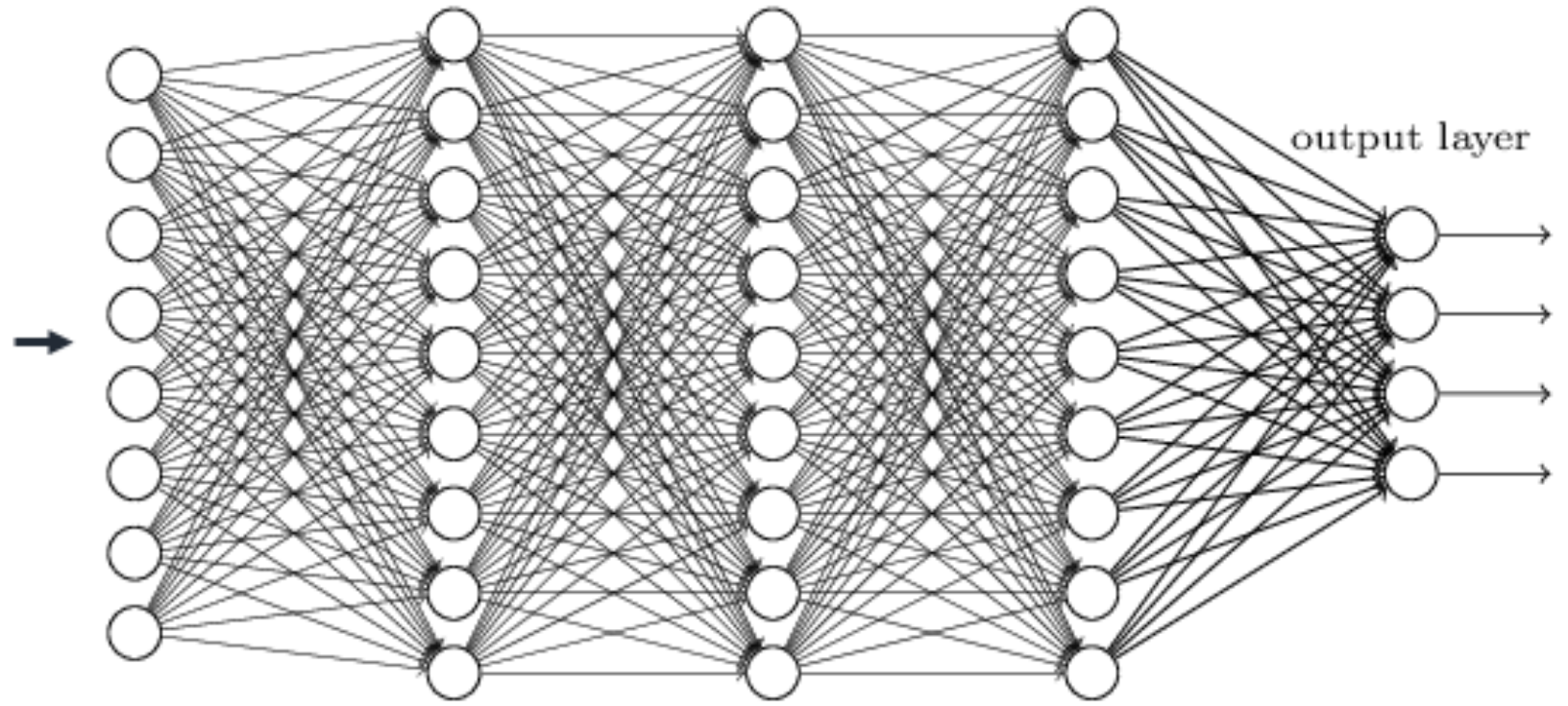


almost at the code...

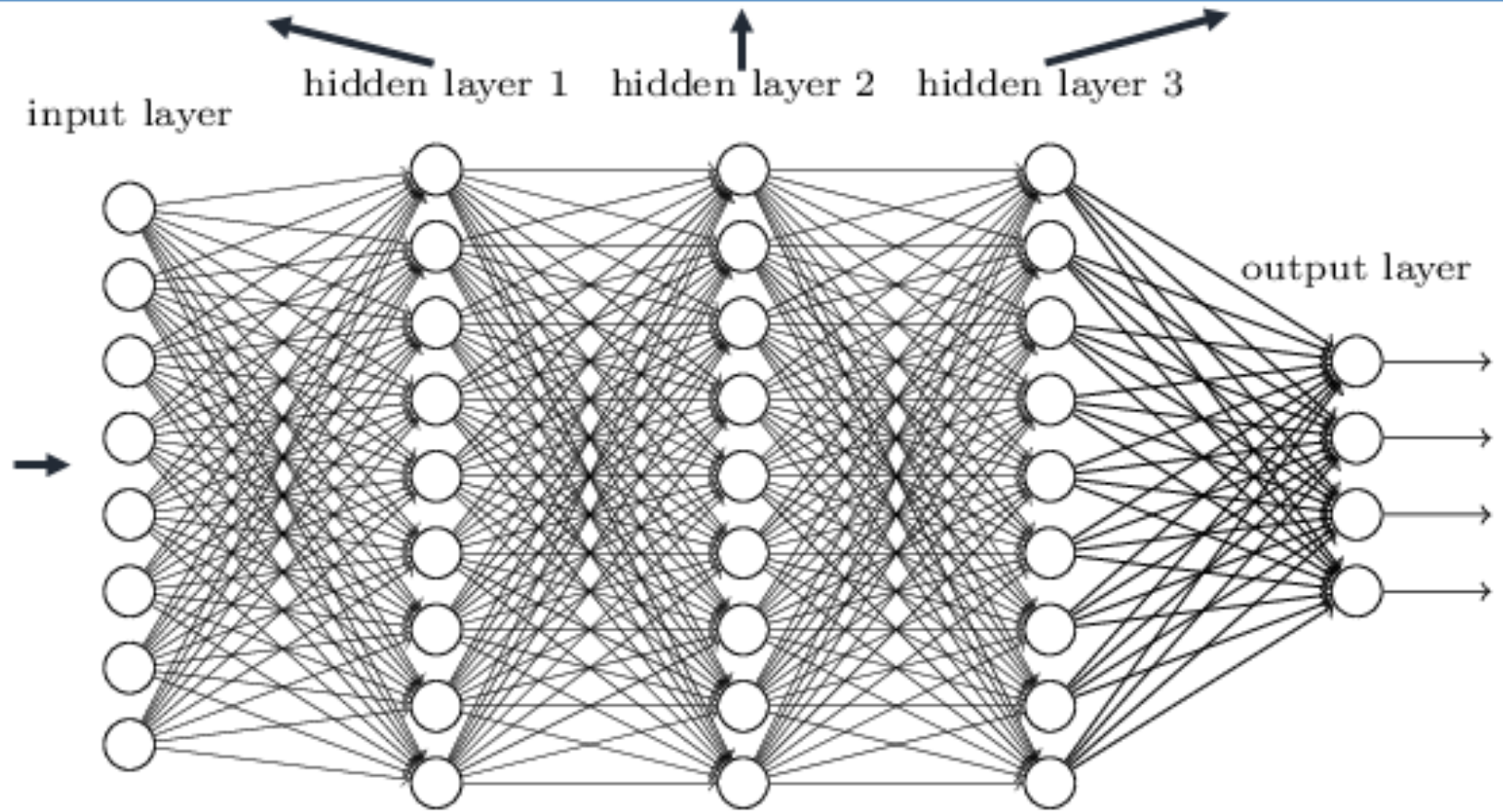


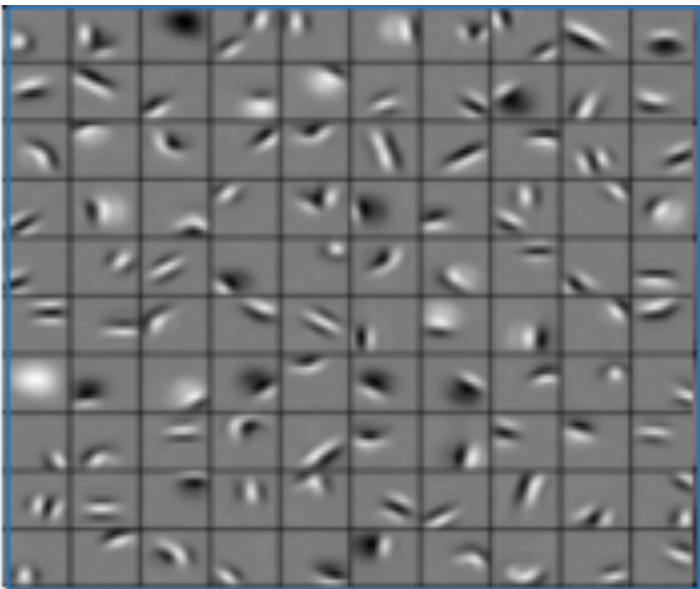


input layer

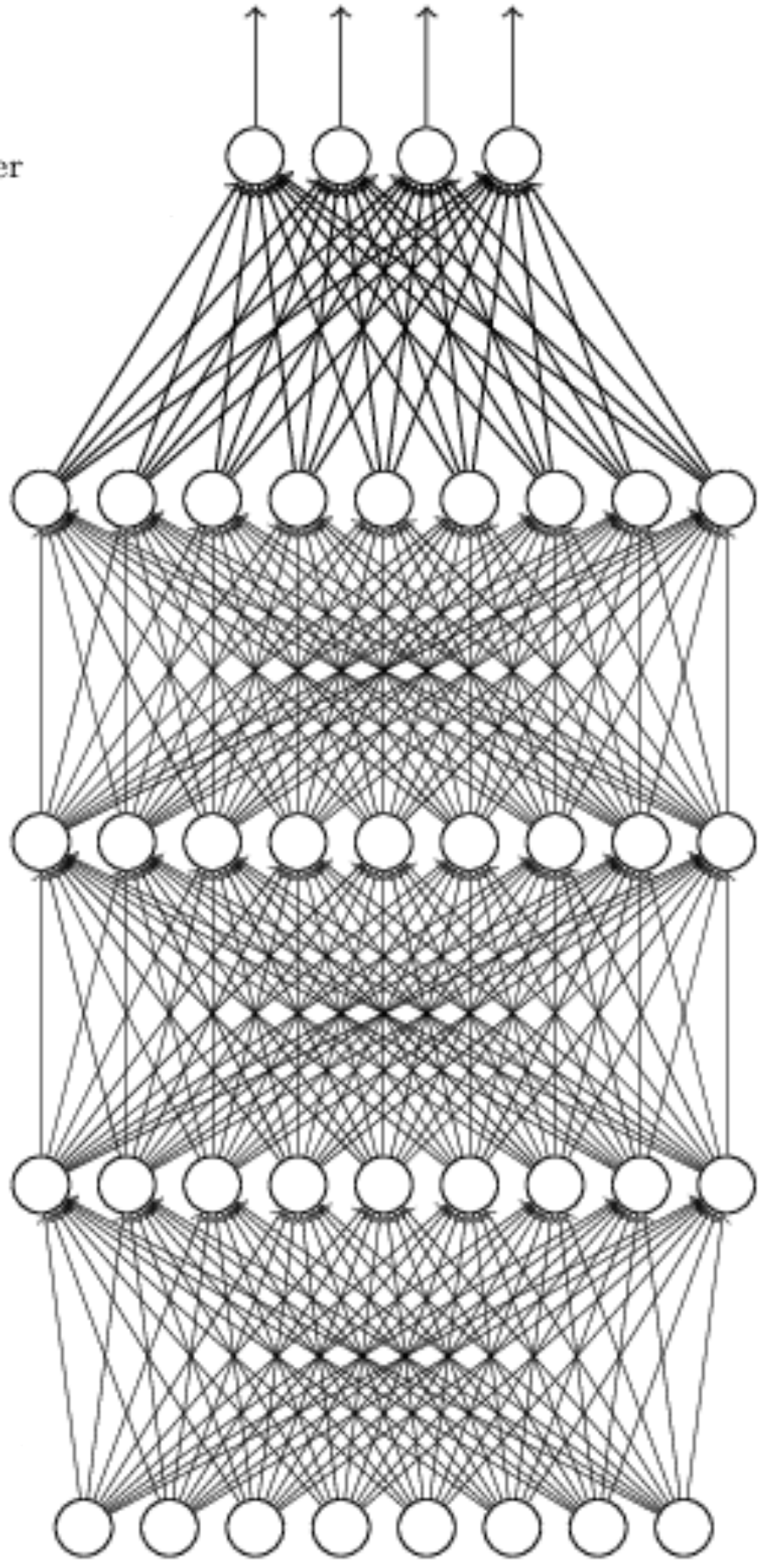


Deep neural networks learn hierarchical feature representations

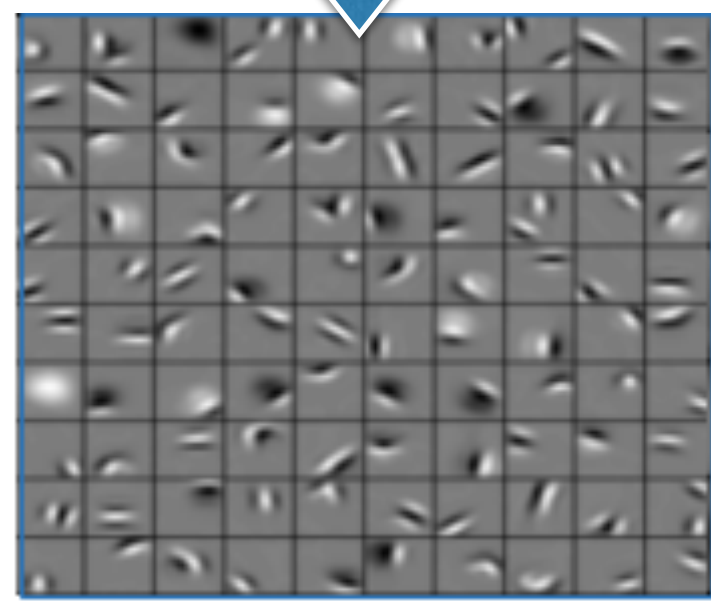




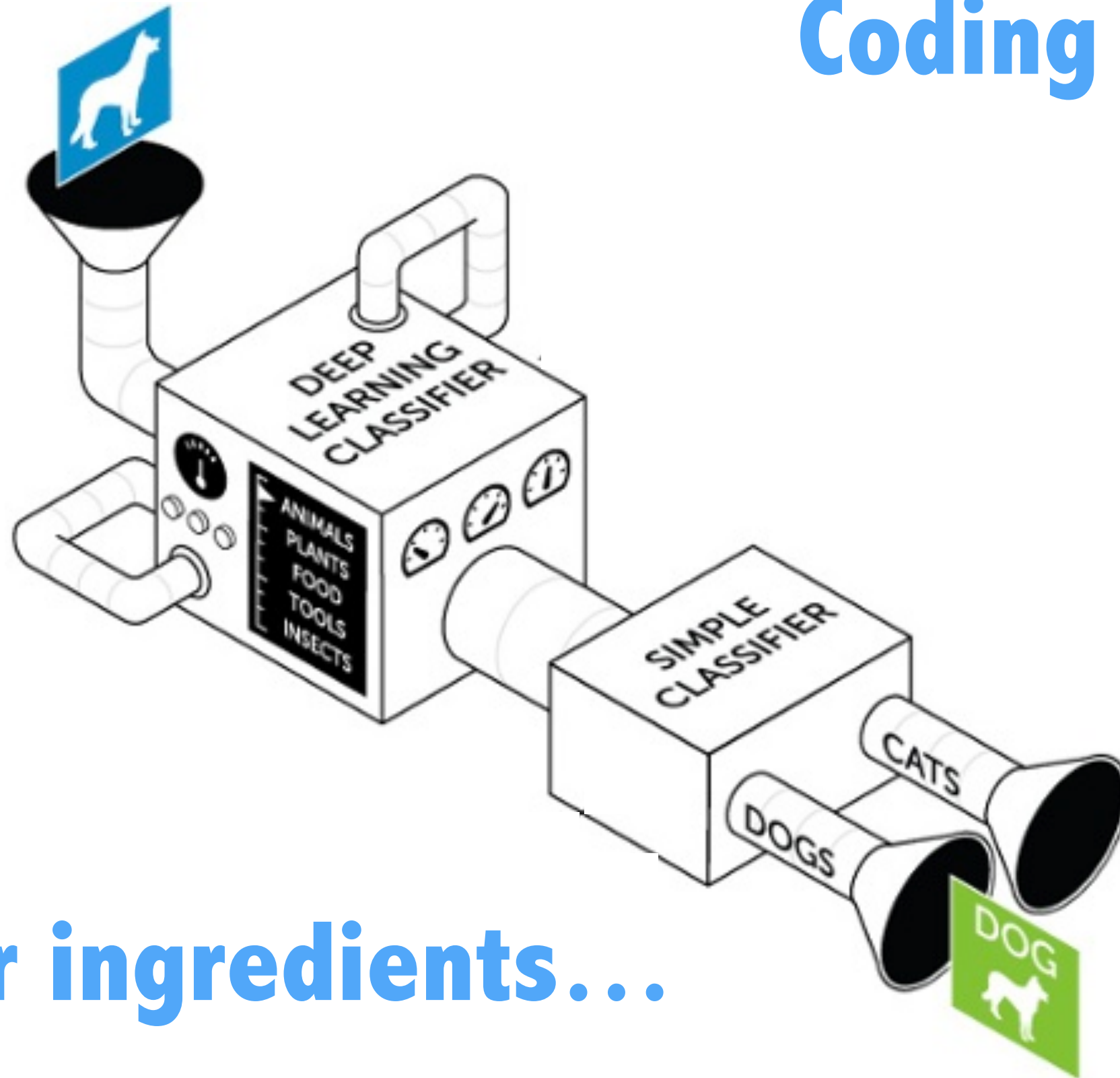
output layer



input layer



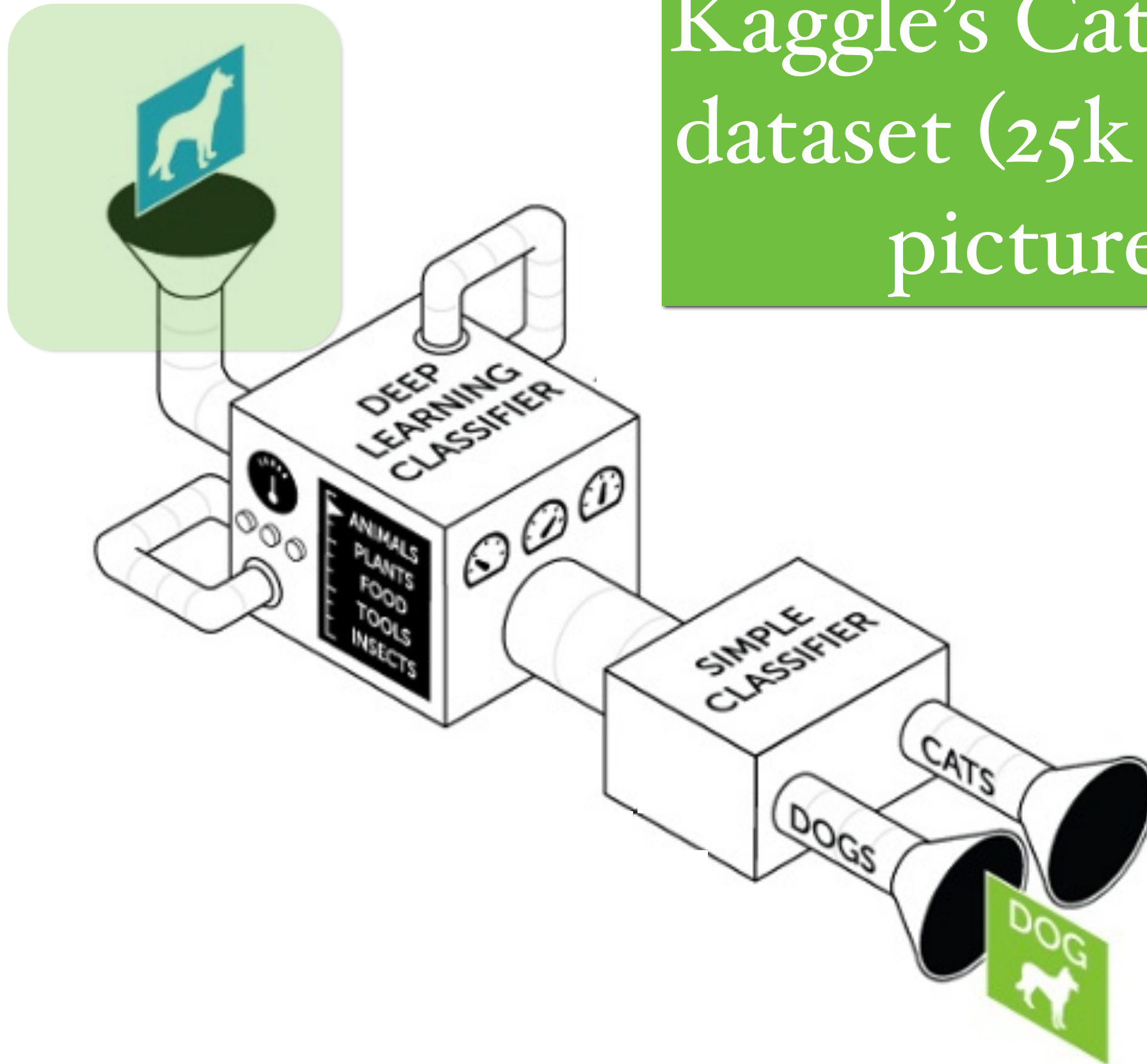
Coding time!



our ingredients...

(picture by Dato)

Kaggle's Cat vs Dog dataset (25k dog/cat pictures)



(picture by Dato)



Completed • Swag • 215 teams

Dogs vs. Cats

Wed 25 Sep 2013 – Sat 1 Feb 2014 (15 months ago)

Dashboard

Home

Data

Make a Submission

Information

Description

Evaluation

Rules

Prizes

Winners

Forum

Leaderboard

Public

Private

Visualization

My Team

GitHub

My Submissions

Leaderboard

1. Pierre Sermanet
2. orchid
3. Owen
4. Paul Covington

Competition Details » [Get the Data](#) » [Make a submission](#)

Create an algorithm to distinguish dogs from cats

In this competition, you'll write an algorithm to classify whether images contain either a dog or a cat. This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.



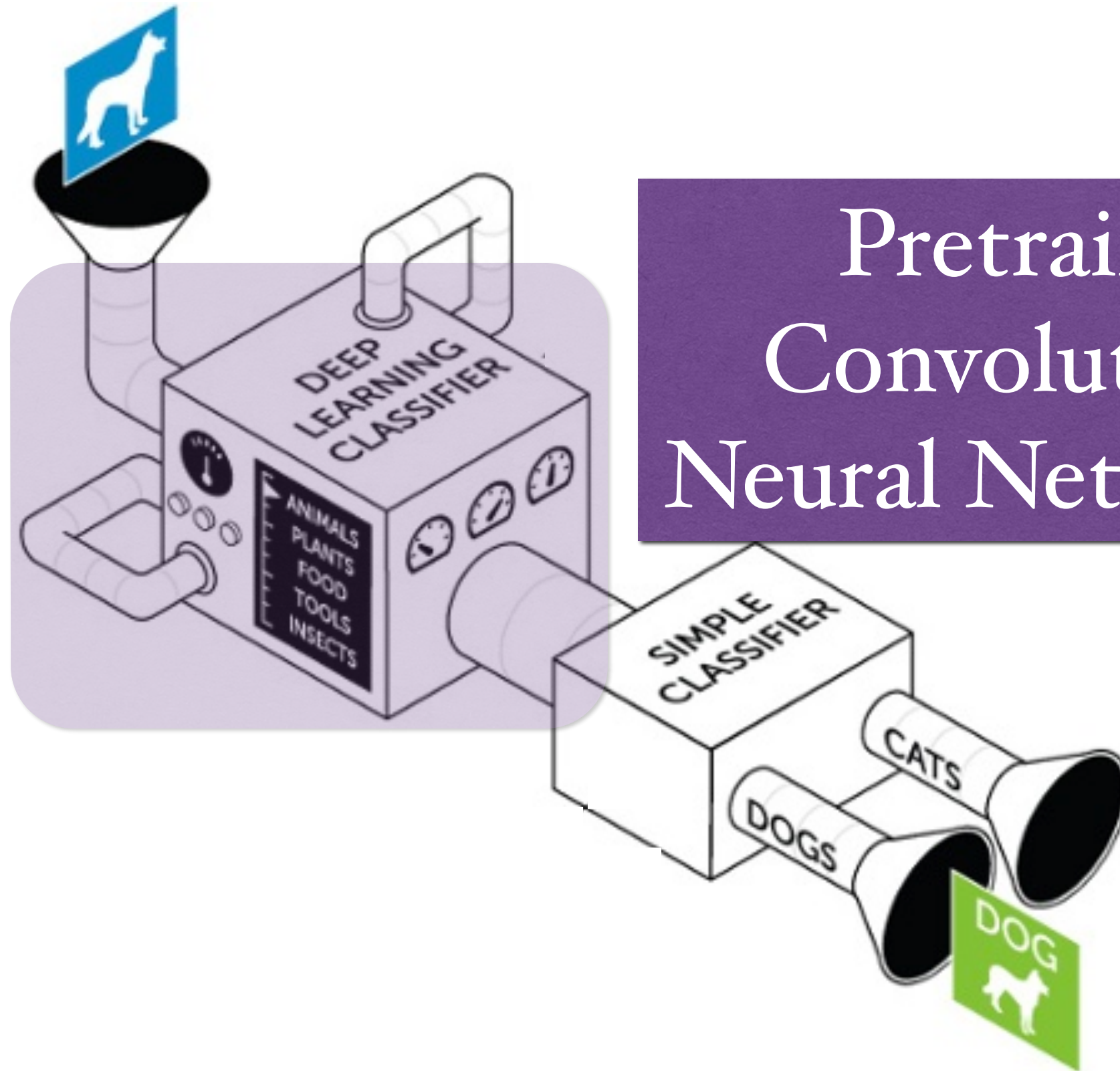
Deep Blue beat
Watson beat the brighte
Can you tell Fi

Data Files

File Name	Available Formats
sampleSubmission	.csv (86.82 kb)
test1	.zip (271.15 mb)
train	.zip (543.16 mb)

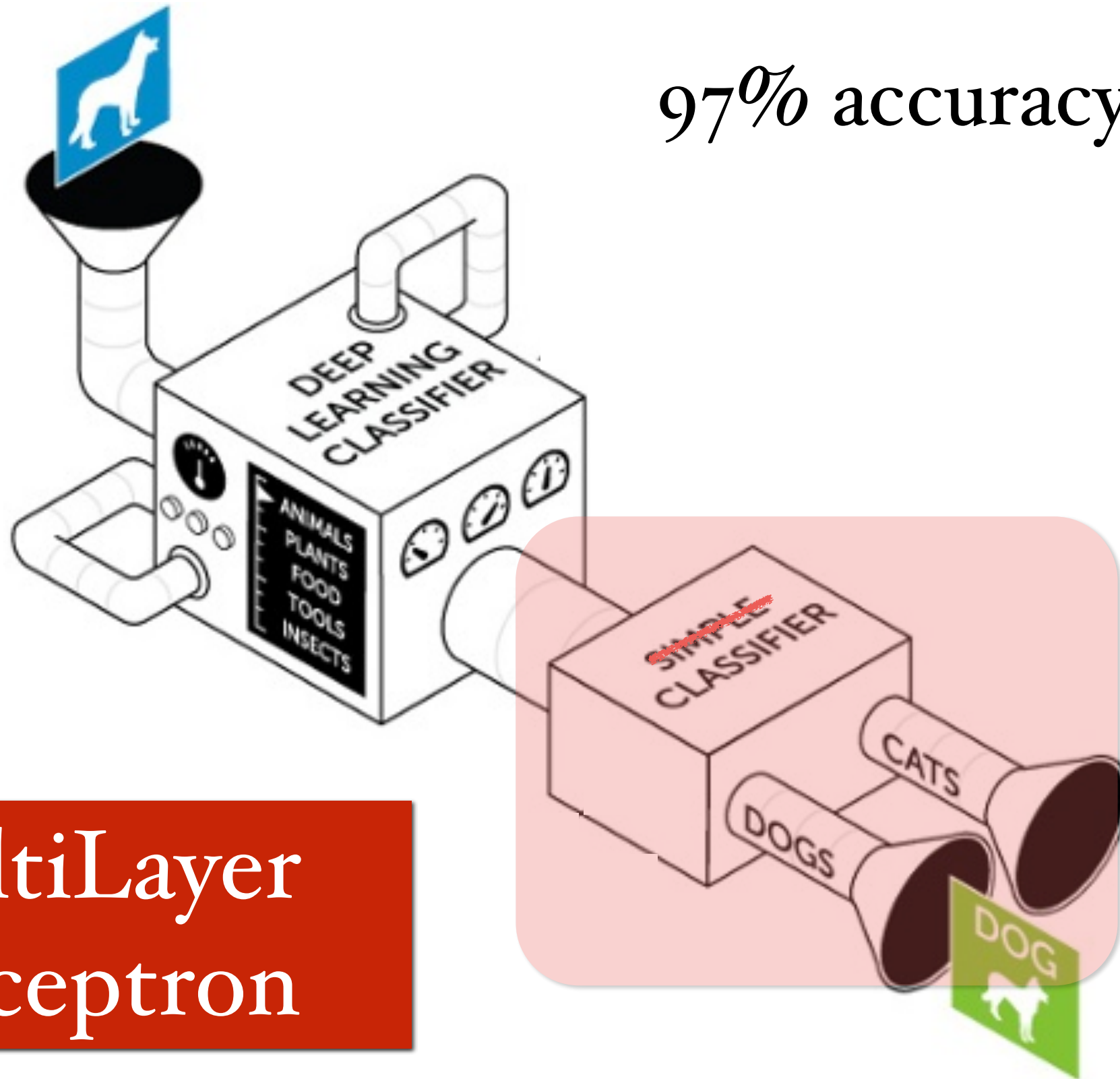
<https://www.kaggle.com/c/dogs-vs-cats/data>

hm on



Pretrained Convolutional Neural Net (CNN)

97% accuracy in < 1h



MultiLayer
Perceptron

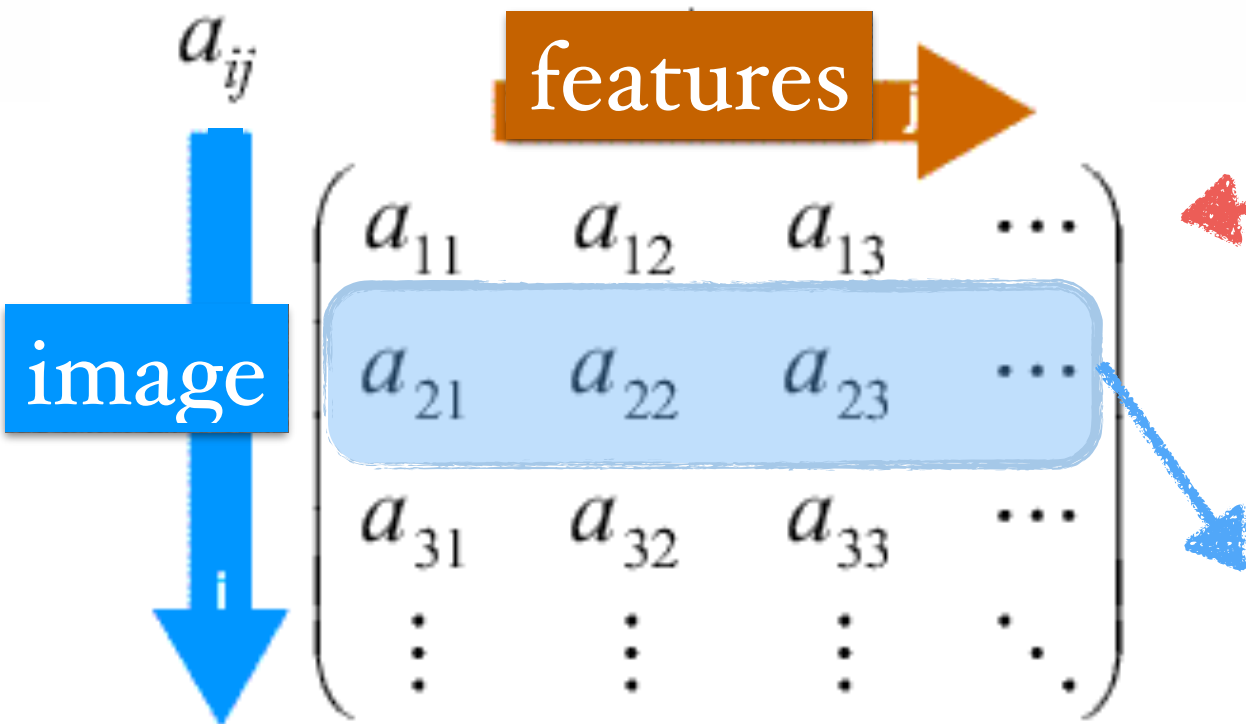
(picture by Dato)

Cooking Instructions

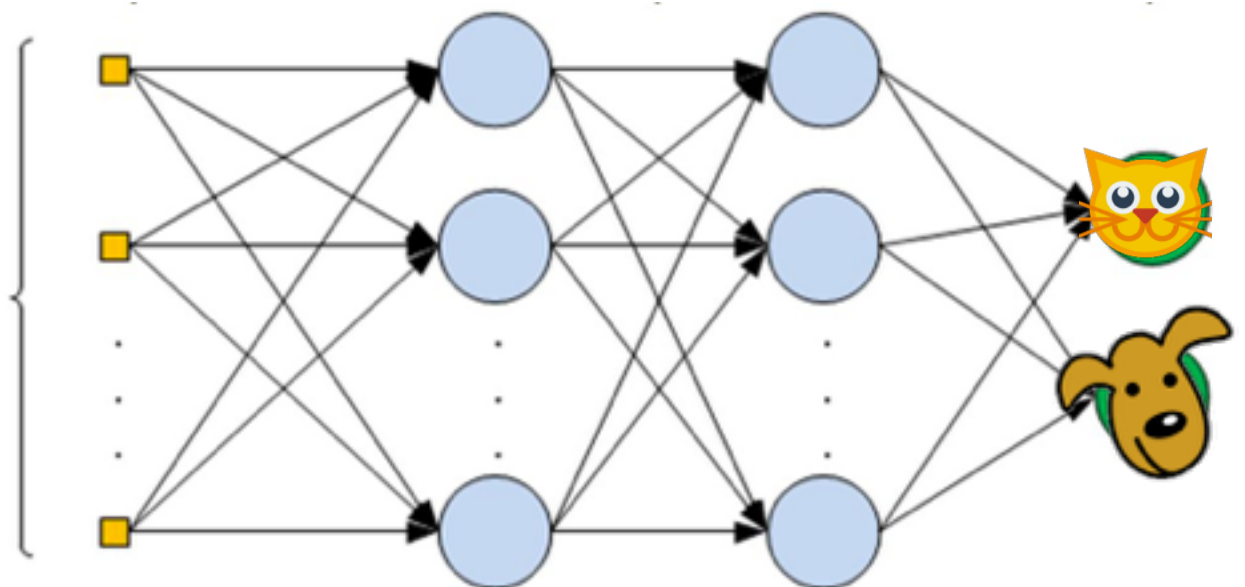
1 Load Pretrained Net



2 Extract features for all training images

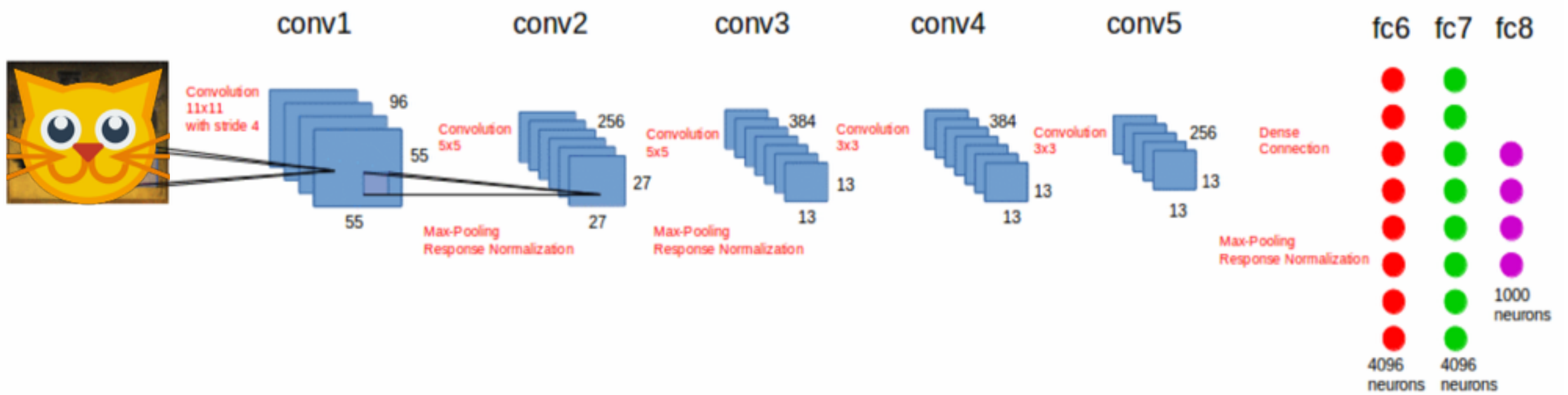


3. train MLP on those features

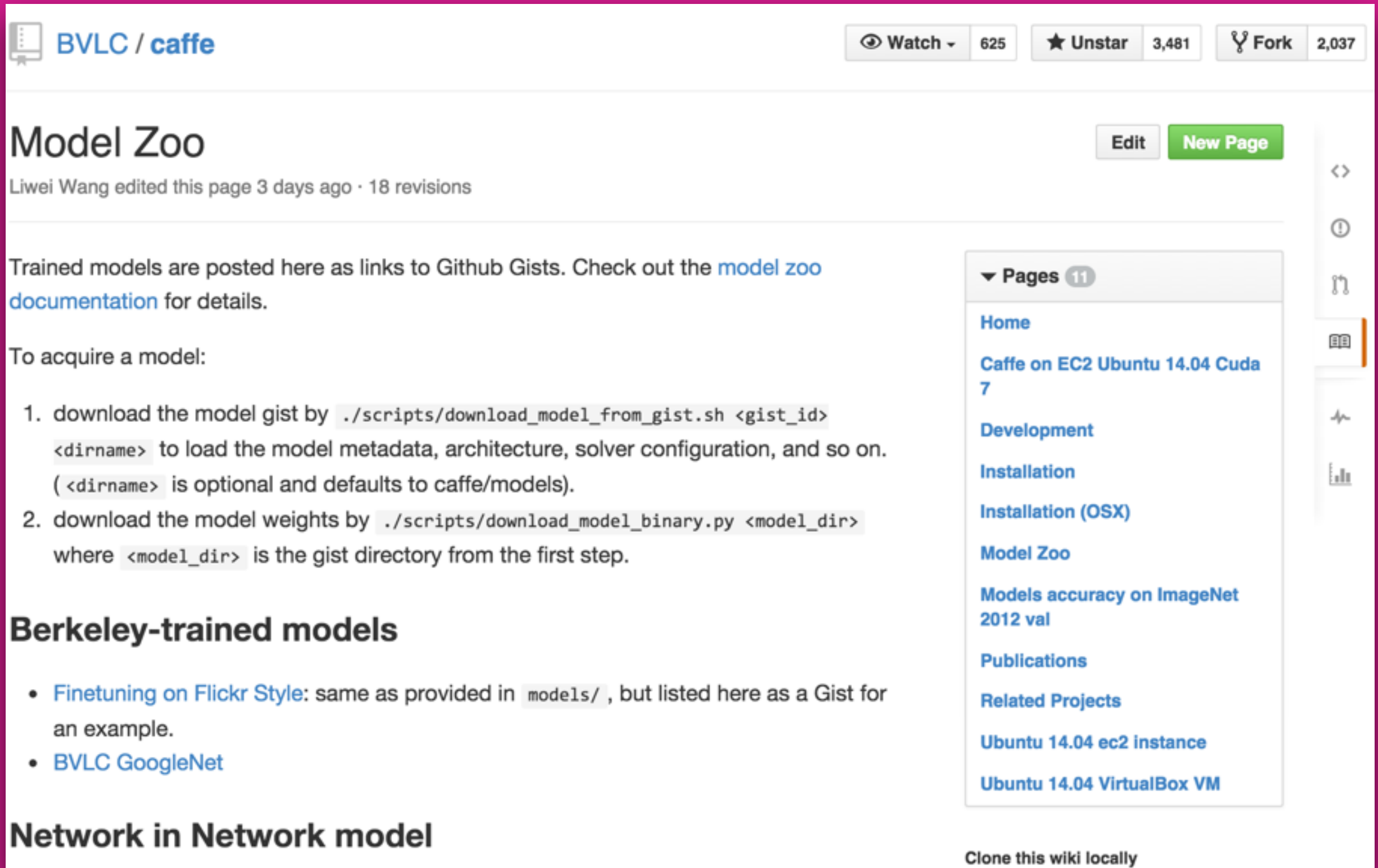


Cooking Instructions

1 Load Pretrained Net



No Free Lunch... But Free Models!



The screenshot shows the GitHub Wiki page for the BVLC / caffe Model Zoo. At the top, the repository name 'BVLC / caffe' is displayed. On the right, there are buttons for 'Watch' (625), 'Unstar' (3,481), and 'Fork' (2,037). The main heading is 'Model Zoo', with 'Edit' and 'New Page' buttons to its right. Below the heading, it says 'Liwei Wang edited this page 3 days ago · 18 revisions'. The main content area contains text about trained models being posted as links to Github Gists, with a link to 'model zoo documentation'. It then provides instructions on how to acquire a model, listing two steps with terminal commands. Below this is a section for 'Berkeley-trained models' with two bullet points. At the bottom, there is a section for 'Network in Network model' and a 'Clone this wiki locally' link. On the right side, there is a 'Pages' sidebar with 11 items, including 'Home', 'Caffe on EC2 Ubuntu 14.04 Cuda 7', 'Development', 'Installation', 'Installation (OSX)', 'Model Zoo', 'Models accuracy on ImageNet 2012 val', 'Publications', 'Related Projects', 'Ubuntu 14.04 ec2 instance', and 'Ubuntu 14.04 VirtualBox VM'.

BVLC / **caffe**

Watch 625 Unstar 3,481 Fork 2,037

Model Zoo

Edit New Page

Liwei Wang edited this page 3 days ago · 18 revisions

Trained models are posted here as links to Github Gists. Check out the [model zoo documentation](#) for details.

To acquire a model:

- download the model gist by `./scripts/download_model_from_gist.sh <gist_id> <dirname>` to load the model metadata, architecture, solver configuration, and so on. (`<dirname>` is optional and defaults to `caffe/models`).
- download the model weights by `./scripts/download_model_binary.py <model_dir>` where `<model_dir>` is the gist directory from the first step.

Berkeley-trained models

- [Finetuning on Flickr Style](#): same as provided in `models/`, but listed here as a Gist for an example.
- [BVLC GoogleNet](#)

Network in Network model

Clone this wiki locally

Pages 11

- Home
- Caffe on EC2 Ubuntu 14.04 Cuda 7
- Development
- Installation
- Installation (OSX)
- Model Zoo
- Models accuracy on ImageNet 2012 val
- Publications
- Related Projects
- Ubuntu 14.04 ec2 instance
- Ubuntu 14.04 VirtualBox VM

<https://github.com/BVLC/caffe/wiki/Model-Zoo>


```
/scripts/download_model_binary.py ../models/bvlc_reference_caffenet
```

jupyter

Files

Running

Clusters

To import a notebook, drag the file onto the listing below or **click**

/ [caffe](#) / [models](#) / [bvlc_reference_caffenet](#)

..

[bvlc_reference_caffenet.caffemodel](#)

[deploy.prototxt](#)

[readme.md](#)

[solver.prototxt](#)

jupyter solver.prototxt ✓ 03/17/2015

Menu

current mode

```
1 net:
  "models/bvlc_reference_caffenet/train_val.prototxt"
2 test_iter: 1000
3 test_interval: 1000
4 base_lr: 0.01
5 lr_policy: "step"
6 gamma: 0.1
7 stepsize: 100000
8 display: 20
9 max_iter: 450000
10 momentum: 0.9
11 weight_decay: 0.0005
12 snapshot: 10000
13 snapshot_prefix:
  "models/bvlc_reference_caffenet/caffenet_train"
14 solver_mode: GPU
15
```

jupyter

deploy.prototxt ✓ 03/17/2015

File

Edit

View

Language

```
1 name: "CaffeNet"
2 input: "data"
3 input_dim: 10
4 input_dim: 3
5 input_dim: 227
6 input_dim: 227
7 layer {
8   name: "conv1"
9   type: "Convolution"
10  bottom: "data"
11  top: "conv1"
12  convolution_param {
13    num_output: 96
14    kernel_size: 11
15    stride: 4
16  }
17 }
18 layer {
19   name: "relu1"
20   type: "ReLU"
21   bottom: "conv1"
22   top: "conv1"
23 }
24 layer {
25   name: "pool1"
26   type: "Pooling"
27   bottom: "conv1"
28   top: "pool1"
29   pooling_param {
30     pool: MAX
31     kernel_size: 3
32     stride: 2
33 }
```

imports

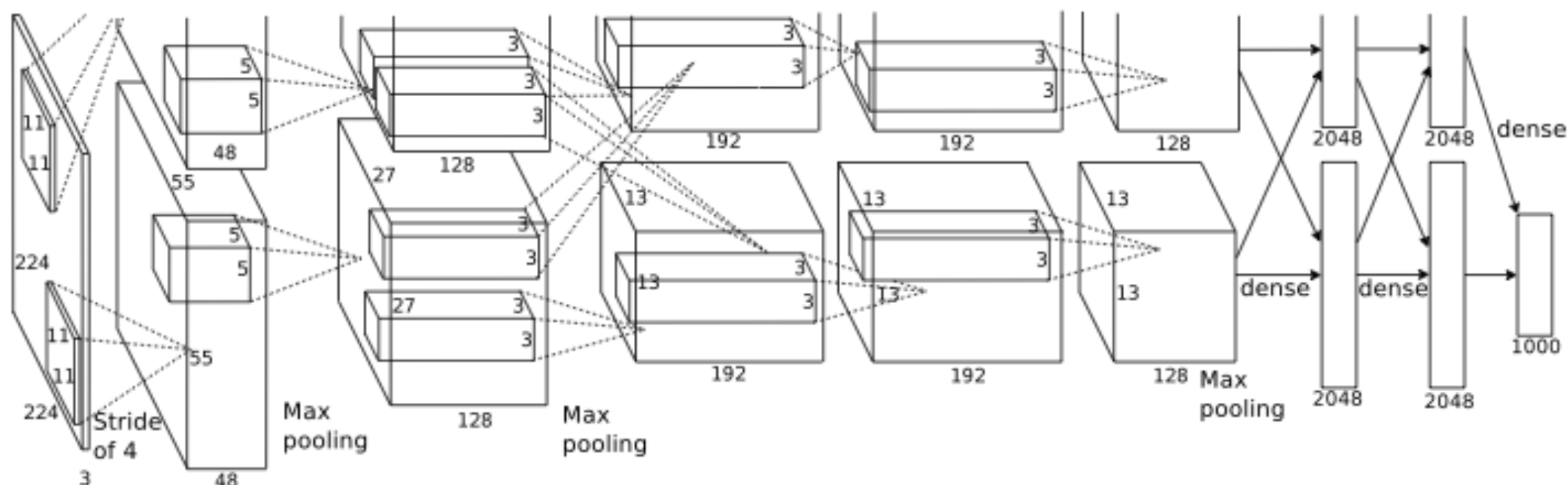
demo

```
1 %matplotlib inline
2 import logging
3 from glob import glob
4 from random import shuffle
5 import pickle
6
7 # Make sure that caffe is on the python path:
8 caffe_root = '../'
9 import sys
10 sys.path.insert(0, caffe_root + 'python')
11 import caffe
12
13 import numpy as np
14 import matplotlib.pyplot as plt
15 import matplotlib.image as mpimg
```

load pretrained deep neural net

demo

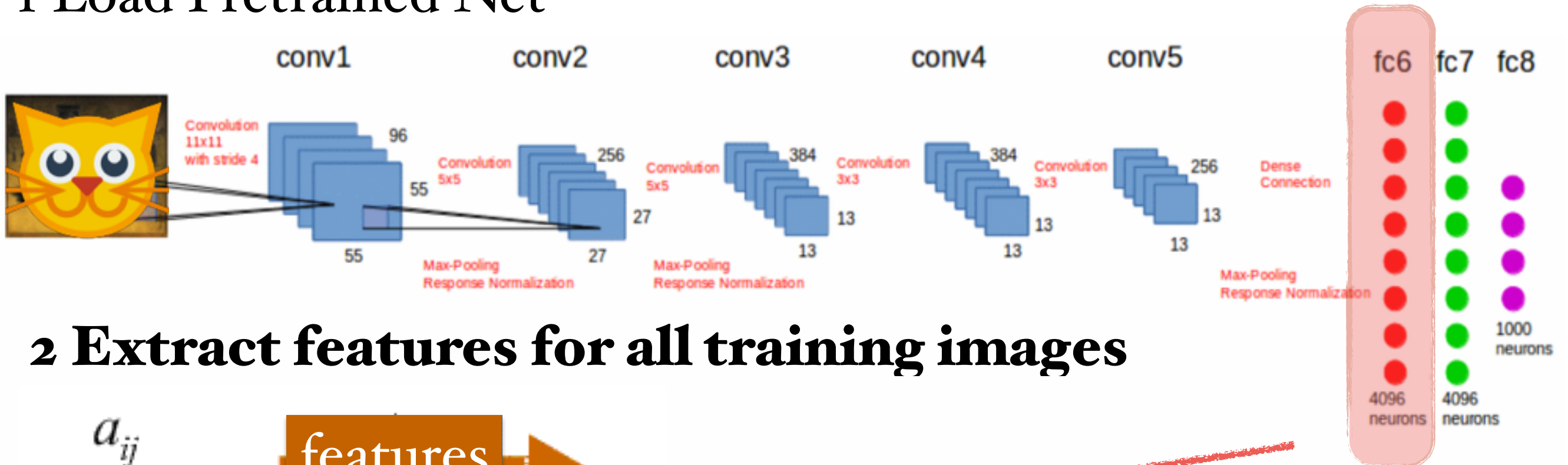
```
1 MODEL_FILE = '../models/bvlc_reference_caffenet/deploy.prototxt'
2 PRETRAINED = '../models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel'
3
4 def png_to_np(basedir, fetch_target=False):
5     logging.getLogger().setLevel(logging.INFO)
6     caffe.set_mode_gpu()
7     net = caffe.Classifier(MODEL_FILE, PRETRAINED,
8                           mean=np.load(caffe_root + 'python/caffe/imagenet/
9                                       ilsvrc_2012_mean.npy').mean(1).mean(1),
10                          channel_swap=(2, 1, 0),
11                          raw_scale=255,
12                          image_dims=(256, 256))
```



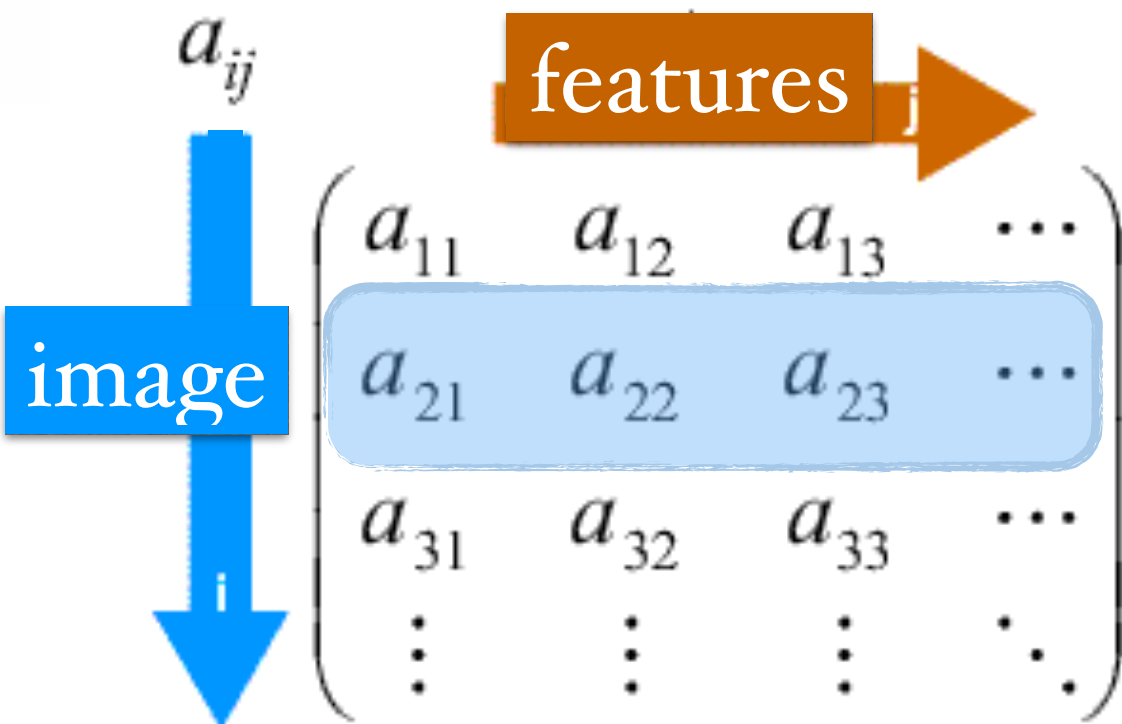
(convnet from Krizhevsky et al.'s NIPS 2012 ImageNet classification paper)

Cooking Instructions

1 Load Pretrained Net



2 Extract features for all training images



feed image into the network and return internal feature representation of layer fc6

demo

```
1 def activate(net, im):
2     input_image = caffe.io.load_image(im)
3     # Resize the image to the standard (256, 256) and oversample net input
    sized crops.
4     input_oversampled = caffe.io.oversample([caffe.io.resize_image(input_image
        , net.image_dims)], net.crop_dims)
5     # 'data' is the input blob name in the model definition, so we preprocess
    for that input.
6     caffe_input = np.asarray([net.transformer.preprocess('data', in_) for in_
        in input_oversampled])
7     # forward() takes keyword args for the input blobs with preprocessed input
    arrays.
8     predicted = net.forward(data=caffe_input)
9     # Activation of all convolutional layers and first fully connected
10    feat = net.blobs['fc6'].data[0]
11    return feat
```

#extract features from images

demo

```
15     feature_info = activate(net, files[0])
16     feature_count = feature_info.shape[0]
17     feature_dtype = feature_info.dtype
18     data = np.zeros((len(files), feature_count), dtype=feature_dtype)
19     for n, im in enumerate(files):
20         data[n, :] = activate(net, im)
21         if n % 1000 == 0:
22             print 'Reading in image', n
23
24     return data, target, files
```

#dump features as pickle file

demo

```
In [*]: x, y, filenames = png_to_np(
         '/mnt/pet/train/', fetch_target=True)
pickle.dump(x, open('saved_x_v2.pkl', 'wb'))
pickle.dump(y, open('saved_y_v2.pkl', 'wb'))
pickle.dump(filenames, open('saved_filenames_v2.pkl', 'wb'))
```

```
Reading in image 0
Reading in image 1000
Reading in image 2000
Reading in image 3000
Reading in image 4000
Reading in image 5000
Reading in image 6000
Reading in image 7000
Reading in image 8000
```

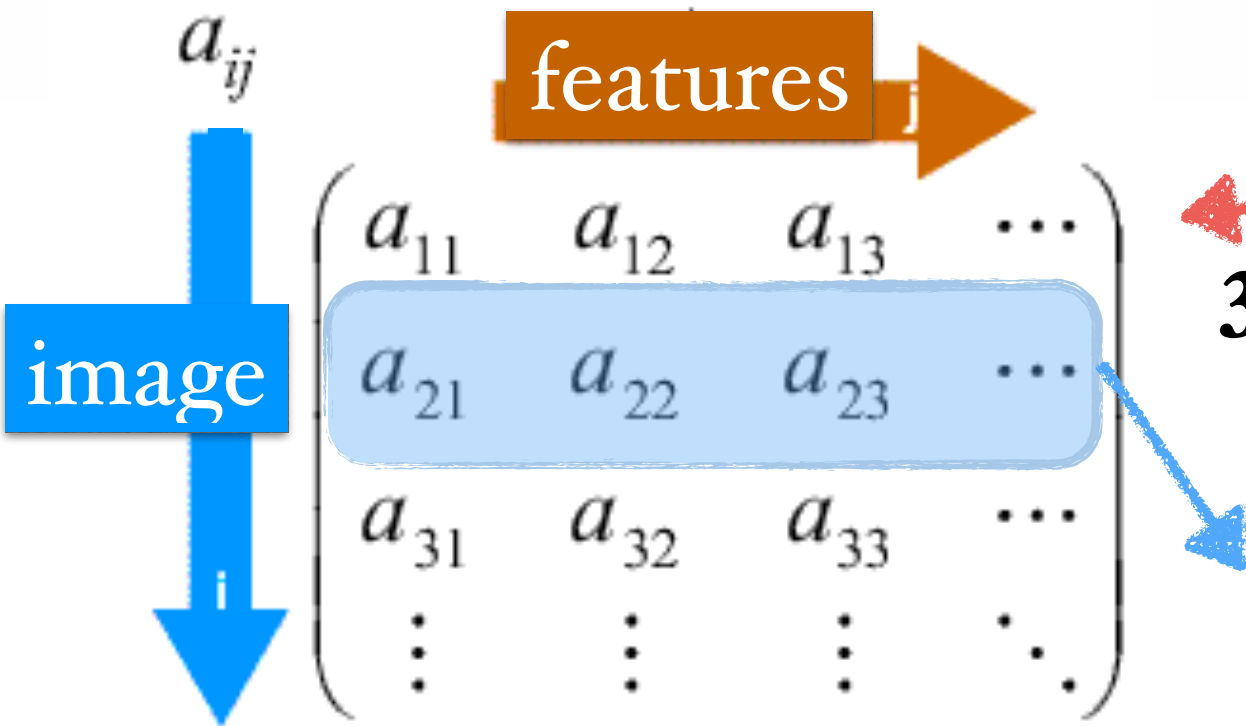
(...)

Cooking Instructions

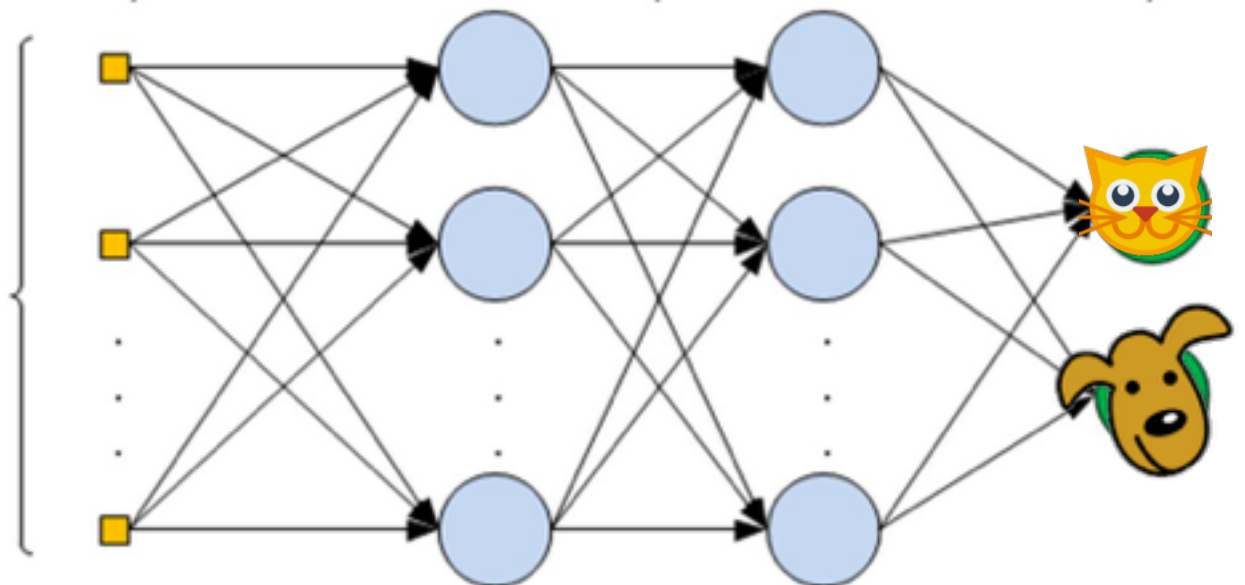
1 Load Pretrained Net



2 Extract features for all training images



3. train MLP on those features



Pylearn2: Multilayer Perceptron (MLP) on top of extracted features

#imports

demo

```
1 from pylearn2.models import mlp
2 from pylearn2.costs.mlp.dropout import Dropout
3 from pylearn2.training_algorithms import sgd, learning_rule
4 from pylearn2.termination_criteria import EpochCounter
5 from pylearn2.datasets import DenseDesignMatrix
6 from pylearn2.train import Train
7 from pylearn2.train_extensions import best_params
8 from pylearn2.space import VectorSpace
9
10 import pickle
11 import numpy as np
```

#load earlier extracted features and labels

#convert to input that pylearn understands

demo

```
1 x = pickle.load(open('saved_x_v2.pkl', 'rb'))
2 y = pickle.load(open('saved_y_v2.pkl', 'rb'))
3 filenames = pickle.load(open('saved_filenames_v2.pkl', 'rb'))
4 y = to_one_hot(y)
5 in_space = VectorSpace(dim=x.shape[1])
6 full = DenseDesignMatrix(X=x, y=y)
```

*# create
layers of
MLP

with
softmax
as final layer*

```
9 l1 = mlp.RectifiedLinear(layer_name='l1',
10 ..... sparse_init=12,
11 ..... dim=5000,
12 ..... max_col_norm=1.)
13
14 l2 = mlp.RectifiedLinear(layer_name='l2',
15 ..... sparse_init=12,
16 ..... dim=5000,
17 ..... max_col_norm=1.)
18
19 l3 = mlp.RectifiedLinear(layer_name='l3',
20 ..... sparse_init=12,
21 ..... dim=5000,
22 ..... max_col_norm=1.)
23
24 output = mlp.Softmax(layer_name='y',
25 ..... n_classes=2,
26 ..... irange=.005)
27
28 layers = [l1, l2, l3, output]
29
30 mdl = mlp.MLP(layers,
31 ..... input_space=in_space)
32
33 lr = .0001
34 epochs = 100
35 trainer = sgd.SGD(learning_rate=lr,
36 ..... batch_size=128,
37 ..... learning_rule=learning_rule.Momentum(.5),
38 ..... # Remember, default dropout is .5
39 ..... cost=Dropout(input_include_probs={'l1': .8},
40 ..... input_scales={'l1': 1.}),
41 ..... termination_criterion=EpochCounter(epochs),
42 ..... monitoring_dataset={'train': full})
```

*# trainer
initialized
with SGD,
momentum,
dropout*

train/test splits

demo

```
1 # no sklearn.cross_validation > train_test_split
2 # own test/train split so we can also link filenames
3 splitter = round(len(x)*0.8)
4 X_train, X_test = x[:splitter],x[splitter:]
5 y_train, y_test = y[:splitter],y[splitter:]
6 filenames_train, filenames_test = filenames[:splitter],filenames[splitter:]
7
8 pickle.dump(X_train, open('saved_feat_x_train_v2.pkl', 'wb'))
9 pickle.dump(X_test, open('saved_feat_x_test_v2.pkl', 'wb'))
10 pickle.dump(y_train, open('saved_feat_y_train_v2.pkl', 'wb'))
11 pickle.dump(y_test, open('saved_feat_y_test_v2.pkl', 'wb'))
12 pickle.dump(filenames_train, open('saved_feat_filenames_train_v2.pkl', 'wb'))
13 pickle.dump(filenames_test, open('saved_feat_filenames_test_v2.pkl', 'wb'))
14
```


#start our MLP (pylearn experiment method)

demo

```
1 #liftoff!  
2 trn = DenseDesignMatrix(X=X_train, y=y_train)  
3 tst = DenseDesignMatrix(X=X_test, y=y_test)  
4 trainer.monitoring_dataset={'valid': tst,  
5                             'train': trn}  
6 experiment.main_loop()
```

```
In [*]: trn = DenseDesignMatrix(X=X_train, y=y_train)  
tst = DenseDesignMatrix(X=X_test, y=y_test)  
trainer.monitoring_dataset={'valid': tst,  
                             'train': trn}  
experiment.main_loop()
```

Parameter and initial learning rate summary:

```
l1_W: 9.99999974738e-05  
l1_b: 9.99999974738e-05  
l2_W: 9.99999974738e-05  
l2_b: 9.99999974738e-05  
l3_W: 9.99999974738e-05  
l3_b: 9.99999974738e-05  
softmax_b: 9.99999974738e-05  
softmax_W: 9.99999974738e-05
```

Compiling sgd_update...

Compiling sgd_update done. Time elapsed: 1.686666 seconds

compiling begin_record_entry...

compiling begin_record_entry done. Time elapsed: 0.548132 seconds

Monitored channels:

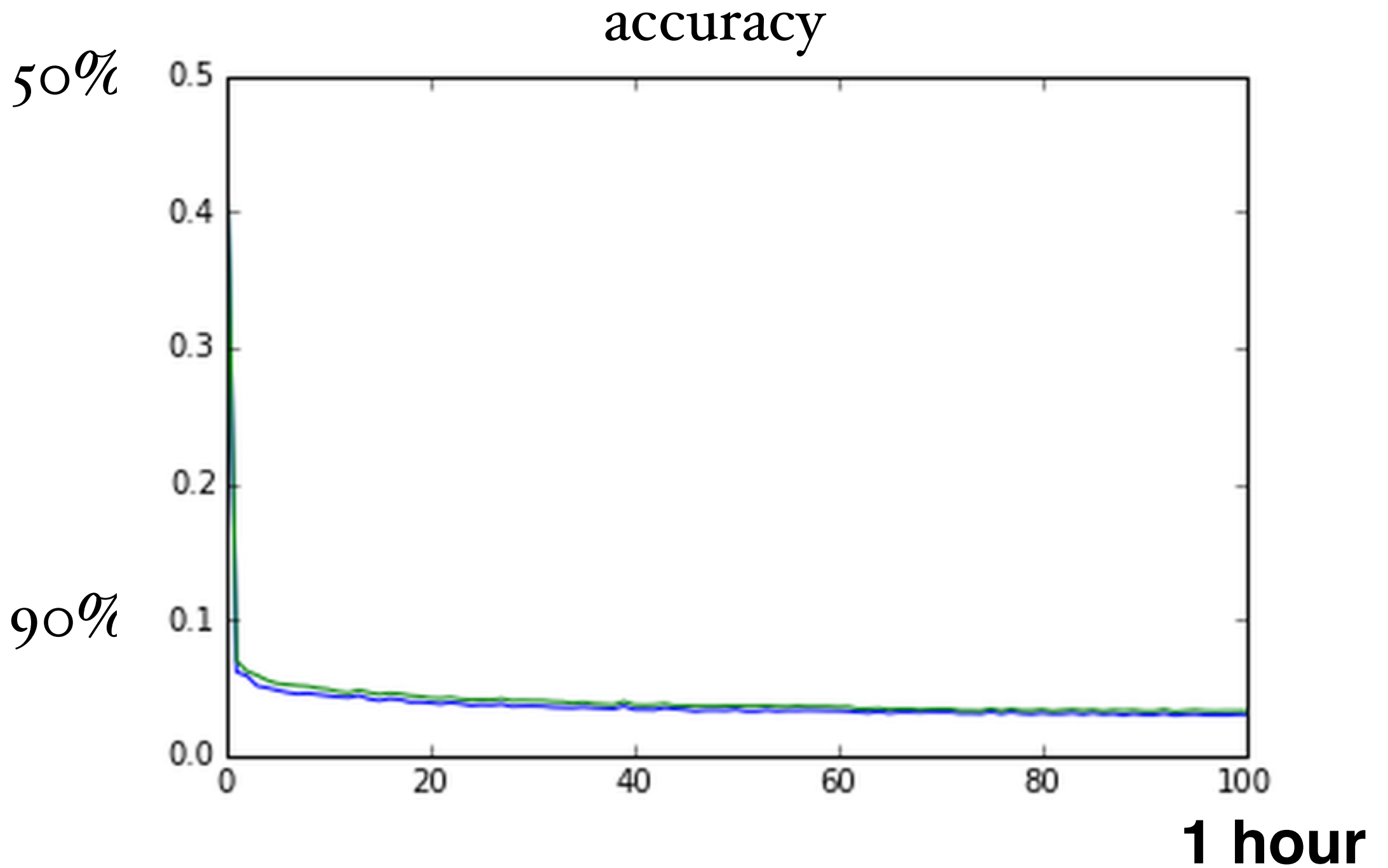
```
learning_rate  
momentum  
total_seconds_last_epoch
```

(...)

already after 5 min: valid_y_misclass: 0.06199999989867

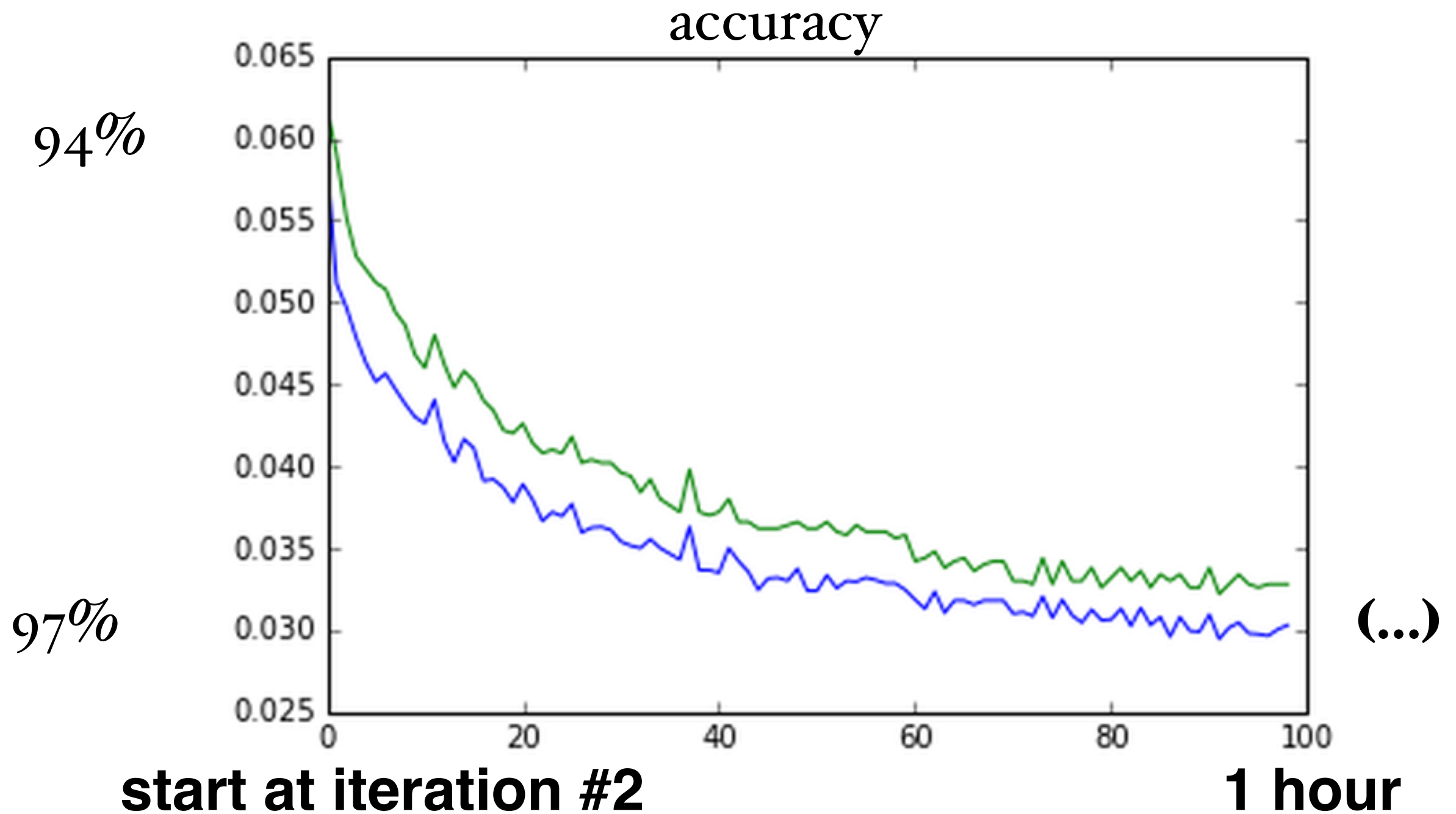
```
In [319]: plt.plot(score_train[2:])  
plt.plot(score_test[2:])
```

```
Out[319]: [<matplotlib.lines.Line2D at 0x7f517c694450>]
```



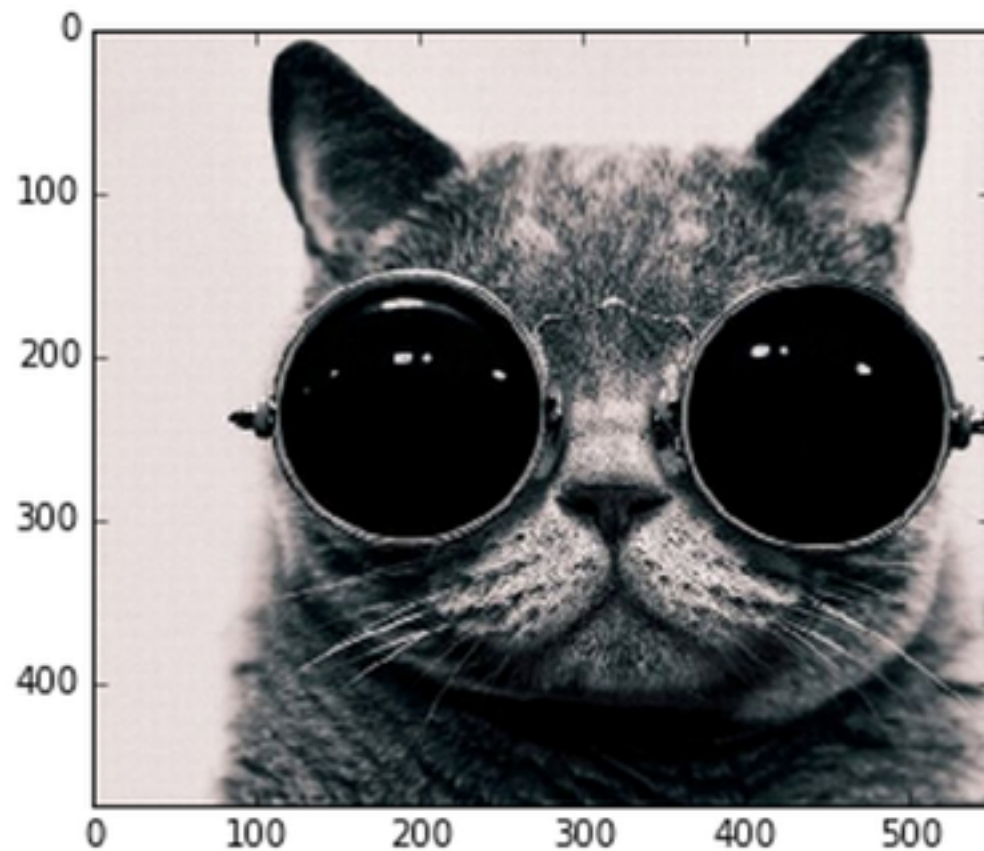
```
In [319]: plt.plot(score_train[2:])  
plt.plot(score_test[2:])
```

```
Out[319]: [<matplotlib.lines.Line2D at 0x7f517c694450>]
```



```
In [380]: input_image = caffe.io.load_image('google-glasses-cat-2.jpg')
plt.imshow(input_image)
```

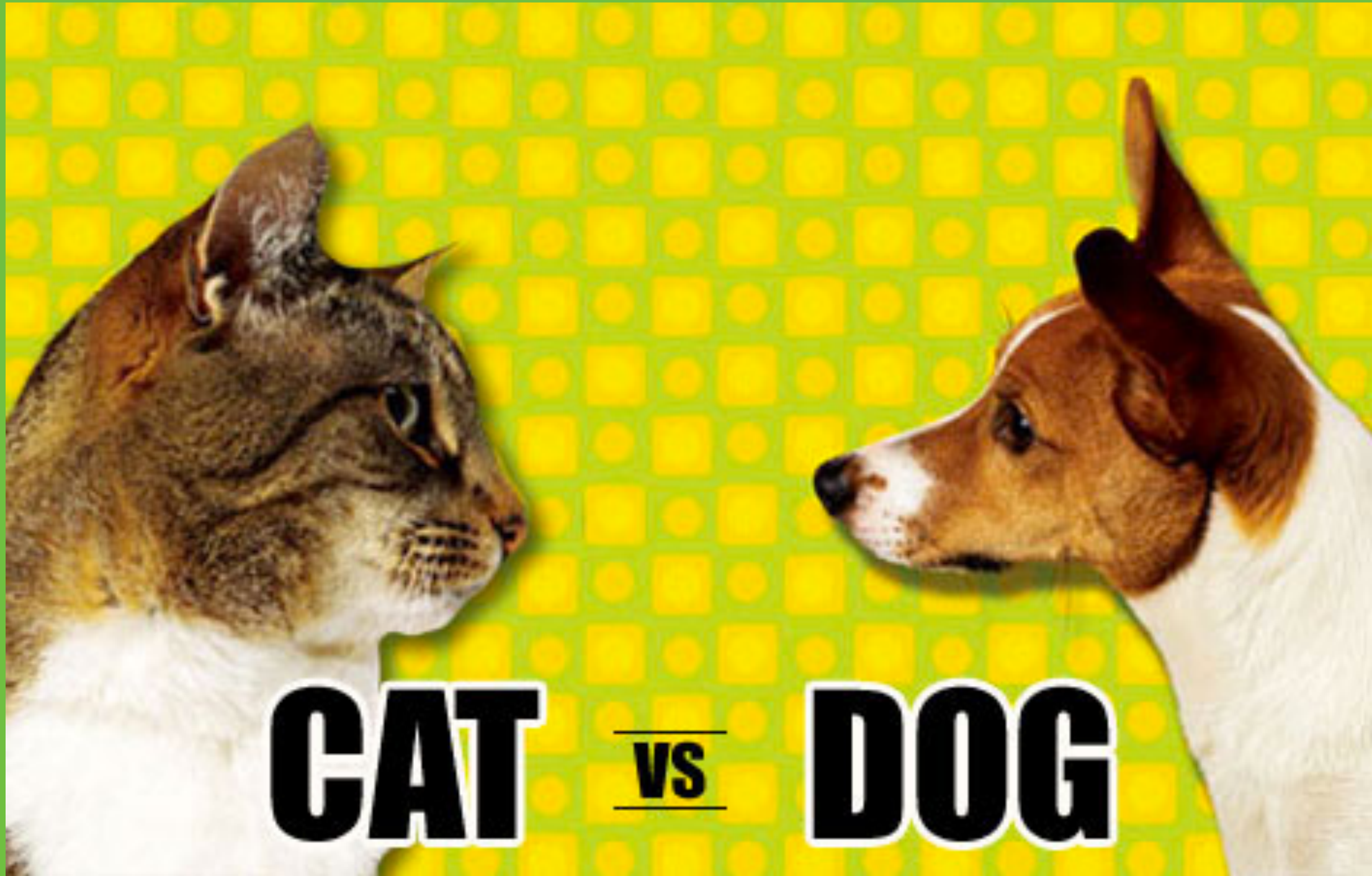
```
Out[380]: <matplotlib.image.AxesImage at 0x7f51a017d410>
```



```
In [381]: feat = getfeat_single_image('google-glasses-cat-2.jpg') #run image through cnn
x = feat
y = f([x]) #run feature through DBN > out: prediction
if y:
    print "WOOF!"
else:
    print "MEOW!"
```

MEOW!

So are YOU more like a Dog or Cat?




CAT vs **DOG**


DETECTOR

**What
about
me?**

← → ↻ pycon:8005/index.html

CAT or DOG, that's the question...



 Share Snapshot

webrtc/webcode adapted from [quizduell](#)

Requires a webcam and a modern browser with WebRTC support such as Firefox or Chrome. Bui

(I might put it up as a Flask site online, if people are interested?)

share



2 seconds ago · 0 views · [stats](#)



[Download full resolution](#) · [Get embed codes](#)

```
In [391]: !wget http://i.imgur.com/oMJyD00.jpg
```

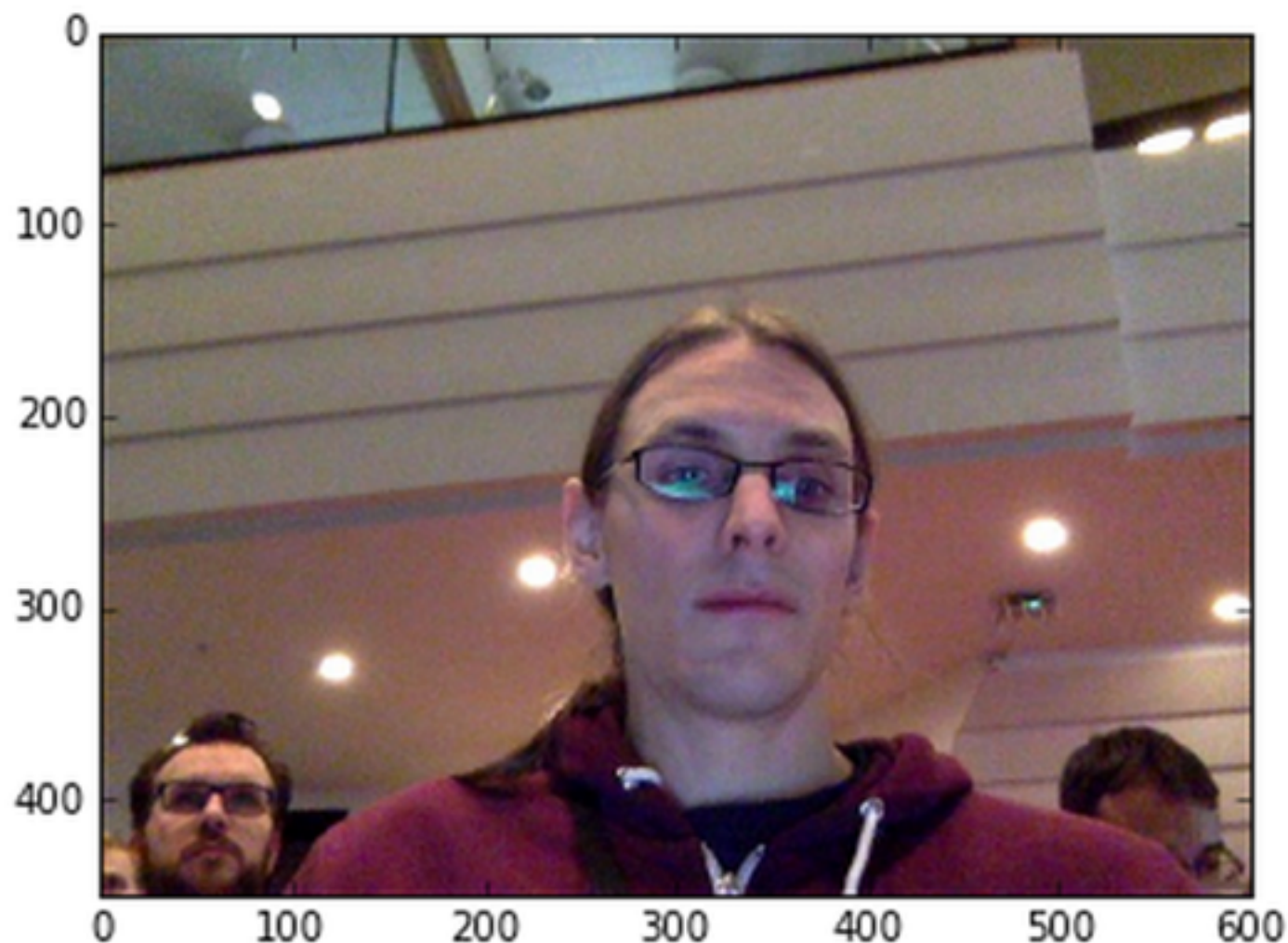
```
--2015-05-12 12:12:00-- http://i.imgur.com/oMJyD00.jpg
Resolving i.imgur.com (i.imgur.com)... 199.27.76.193
Connecting to i.imgur.com (i.imgur.com)|199.27.76.193|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 58456 (57K) [image/jpeg]
Saving to: 'oMJyD00.jpg'

100%[=====>] 58,456      --.-K/s   in 0.02s

2015-05-12 12:12:00 (2.97 MB/s) - 'oMJyD00.jpg' saved [58456/58456]
```

```
In [393]: input_image = caffe.io.load_image('oMJyD00.jpg')
plt.imshow(input_image)
```

```
Out[393]: <matplotlib.image.AxesImage at 0x7f51aefe5e50>
```



In [395]:

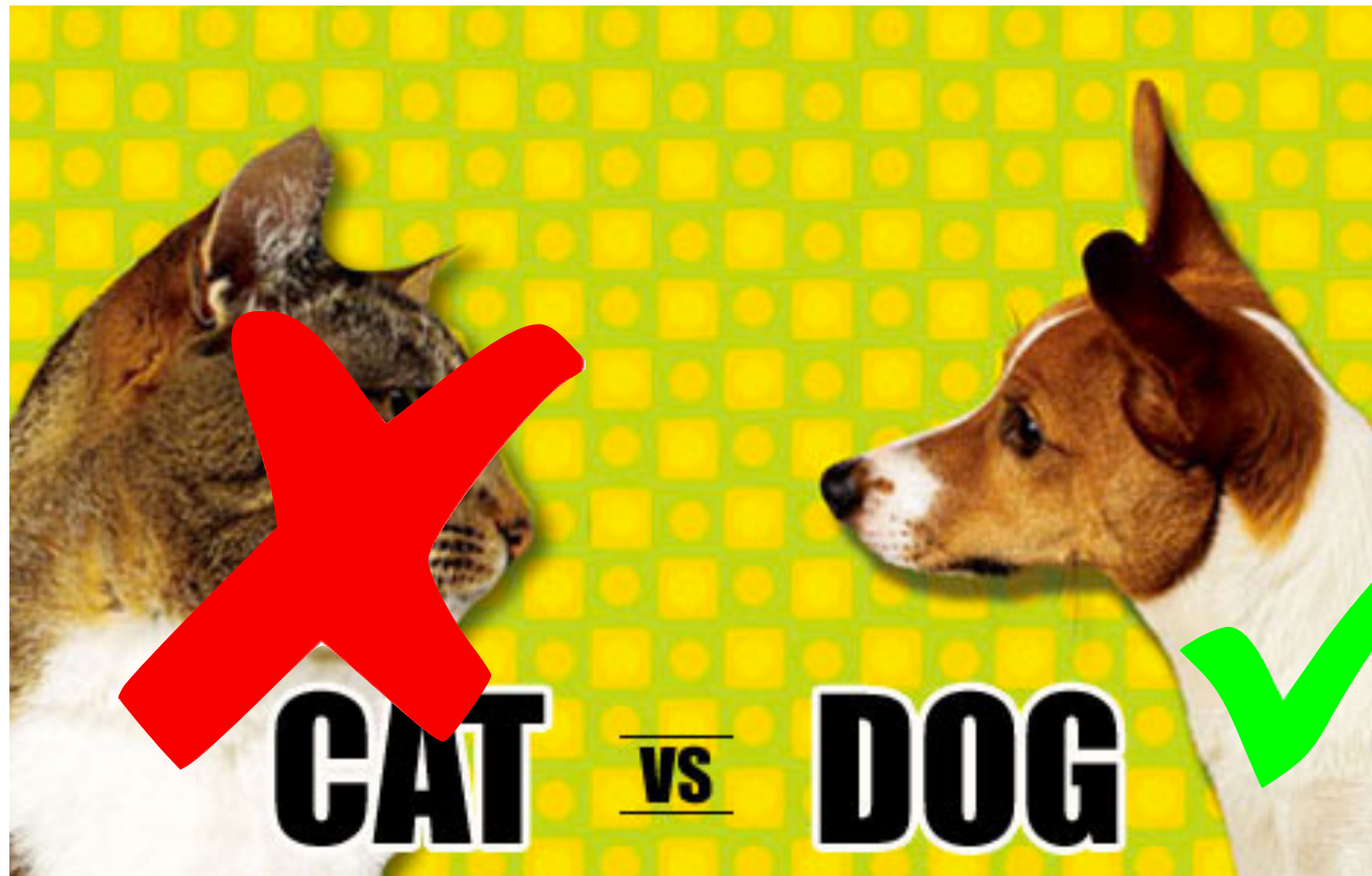
```
feat = getfeat_single_image('rQ4bKra.jpg') #run image t
x = feat
y = f([x]) #run feature through DBN > out: prediction
if y:
    print "WOOF I'm a Dog!"
else:
    print "MEOW I'm a Cat!"
```

WOOF I'm a Dog!

In [395]:

```
feat = getfeat_single_image('rQ4bKra.jpg') #run image t  
x = feat  
y = f([x]) #run feature through DBN > out: prediction  
if y:  
    print "WOOF I'm a Dog!"  
else:  
    print "MEOW I'm a Cat!"
```

WOOF I'm a Dog!



THATS ALL!



EASY PIEZY...

In Touch!

Academic/Research

as PhD candidate KTH/CSC:
“Always interested in discussing
Machine Learning, Deep
Architectures, Graphs, and
Language Technology”



ROYAL INSTITUTE
OF TECHNOLOGY

roelof@kth.se

www.csc.kth.se/~roelof/



Data Science Consultancy

**Gve Systems
Graph Technologies**



roelof@graph-systems.com

www.graph-technologies.com

Wanna Know More?

bit.ly/SthImDL

Stockholm Deep Learning Meetup

Home

Members

Sponsors

Photos

Pages

Discussions

More

Group tools



My profile

Stockholm, Sweden

Founded Feb 21, 2015

About us...

Datamaniacs 295

Group reviews 6

Past Meetups 2

Our calendar

Organizer:

Roelof Pieters



We're about:

Big Data Analytics · Artificial Intelligence · Open Source · Software Development · New Technology · Big Data · Natural Language Processing · Machine Learning · Data

Welcome!

+ SCHEDULE A NEW MEETUP

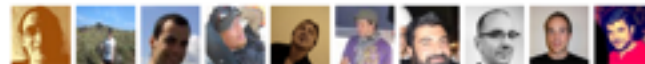
Upcoming

Past

Calendar

April 20 · 6:00 PM

Deep Learning for Bioinformatics



103 Datamaniacs | ★★★★★ | 40 Photos

Agenda: • 18:00 - 18:15 Grab a coffee/beer and get ready to rumble... • 18:15 - 18:30 Welcome • 18:30 - 19:00 Roelof Pieters: Deep Learning, a birds eye view • 19:00 -... [LEARN MORE](#)

March 10 · 6:00 PM

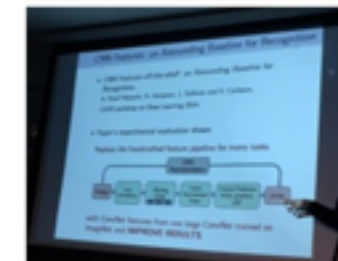
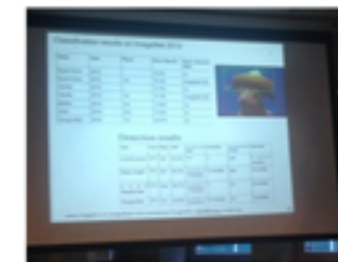
Kickoff! Deep Learning: Revolution or Hype in Data-Science ?



144 Datamaniacs | ★★★★★ | 35 Photos

Deep Learning is kicking off everywhere (see this front page article in the New York Times for example)! There is good reason to be excited about deep learning, as it's... [LEARN MORE](#)

What's new



[MORE](#)

NEW MEMBER

Stefan Avesand joined



4 days ago

NEW MEMBER

Måns Magnusson joined



Wanna Play ? General Deep Learning

- Theano - CPU/GPU symbolic expression compiler in python (from LISA lab at University of Montreal).
<http://deeplearning.net/software/theano/>
- Pylearn2 - library designed to make machine learning research easy. <http://deeplearning.net/software/pylearn2/>
- Torch - Matlab-like environment for state-of-the-art machine learning algorithms in lua (from Ronan Collobert, Clement Farabet and Koray Kavukcuoglu)
<http://torch.ch/>
- more info: <http://deeplearning.net/software/links/>

Wanna Play ? NLP

- RNNLM (Mikolov)
<http://rnnlm.org>
- NB-SVM
<https://github.com/mesnilgr/nbsvm>
- Word2Vec (skipgrams/cbow)
<https://code.google.com/p/word2vec/> (original)
<http://radimrehurek.com/gensim/models/word2vec.html> (python)
- GloVe
<http://nlp.stanford.edu/projects/glove/> (original)
<https://github.com/maciejkula/glove-python> (python)
- Socher et al / Stanford RNN Sentiment code:
<http://nlp.stanford.edu/sentiment/code.html>
- Deep Learning without Magic Tutorial:
<http://nlp.stanford.edu/courses/NAACL2013/>

Wanna Play ? Computer Vision

- cuda-convnet2 (Alex Krizhevsky, Toronto) (c++/
CUDA, optimized for GTX 580)
<https://code.google.com/p/cuda-convnet2/>
- Caffe (Berkeley) (Cuda/OpenCL, Theano, Python)
<http://caffe.berkeleyvision.org/>
- OverFeat (NYU)
<http://cilvr.nyu.edu/doku.php?id=code:start>