LET'S OPEN THE DEEP LEARNING BLACK BOX!

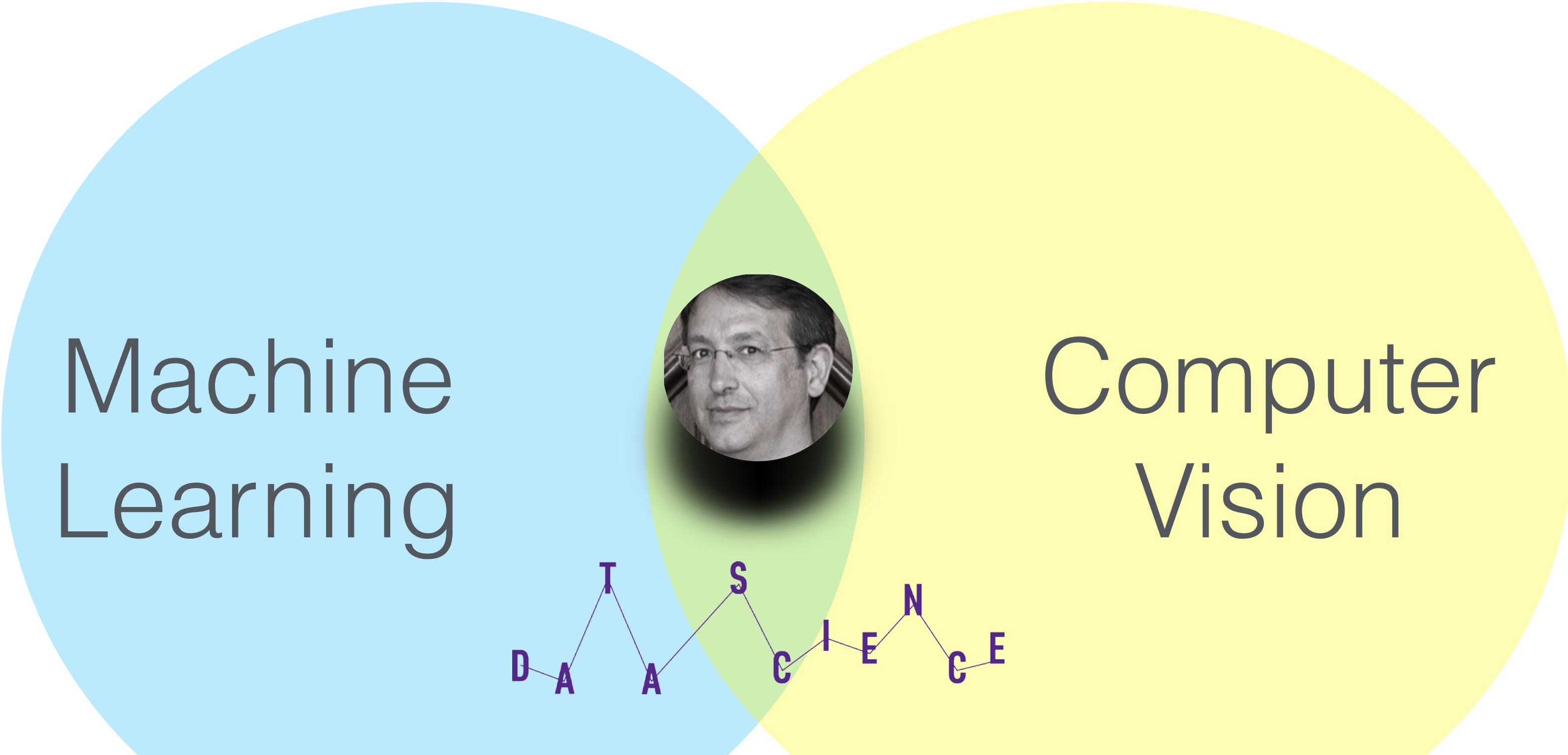
JORDI VITRIÀ

Universitat de Barcelona

jordi.vitria@ub.edu

jordi.vitria@ub.edu

Departament de Matemàtiques i Informàtica
Universitat de Barcelona



Since 2007, I am a Full Professor at the Mathematics & Computer Science Department, **Universitat de Barcelona**. Before that I spent 20 years on the faculty of the CS Department at the **Universitat Autonoma de de Barcelona**. I am the Director of the **Data Science & Big Data Postgraduate Course** and the **Foundations of Data Science Master** at UB. I am the leader of the **DataScience@UB** group, whose objective is to promote technology transfer.

http://datascience.barcelona/
http://www.ub.edu/cvub/jordivitria/

Some examples of our research (that involve **deep learning** methods)

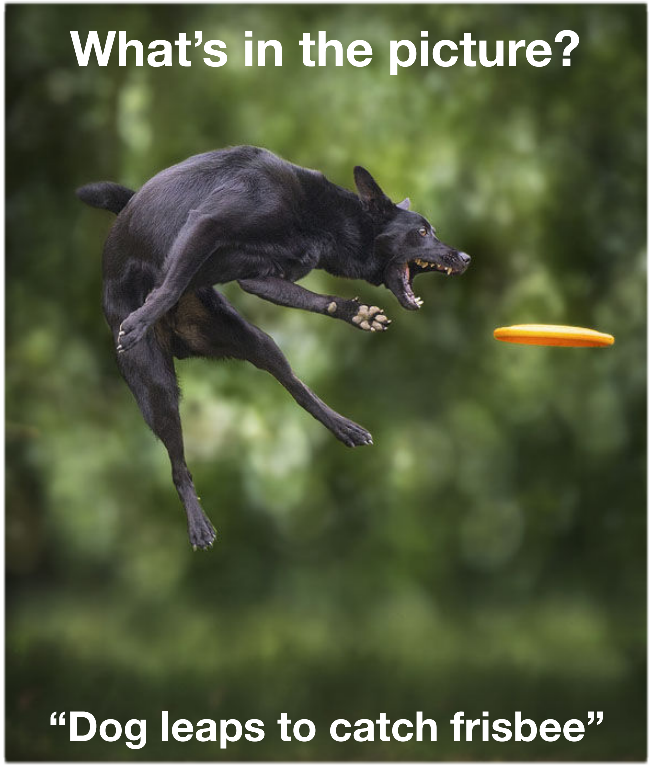
end-to-end learning

deep neural networks

"black box" learning…

Extracting non visual attributes from images using CNN.

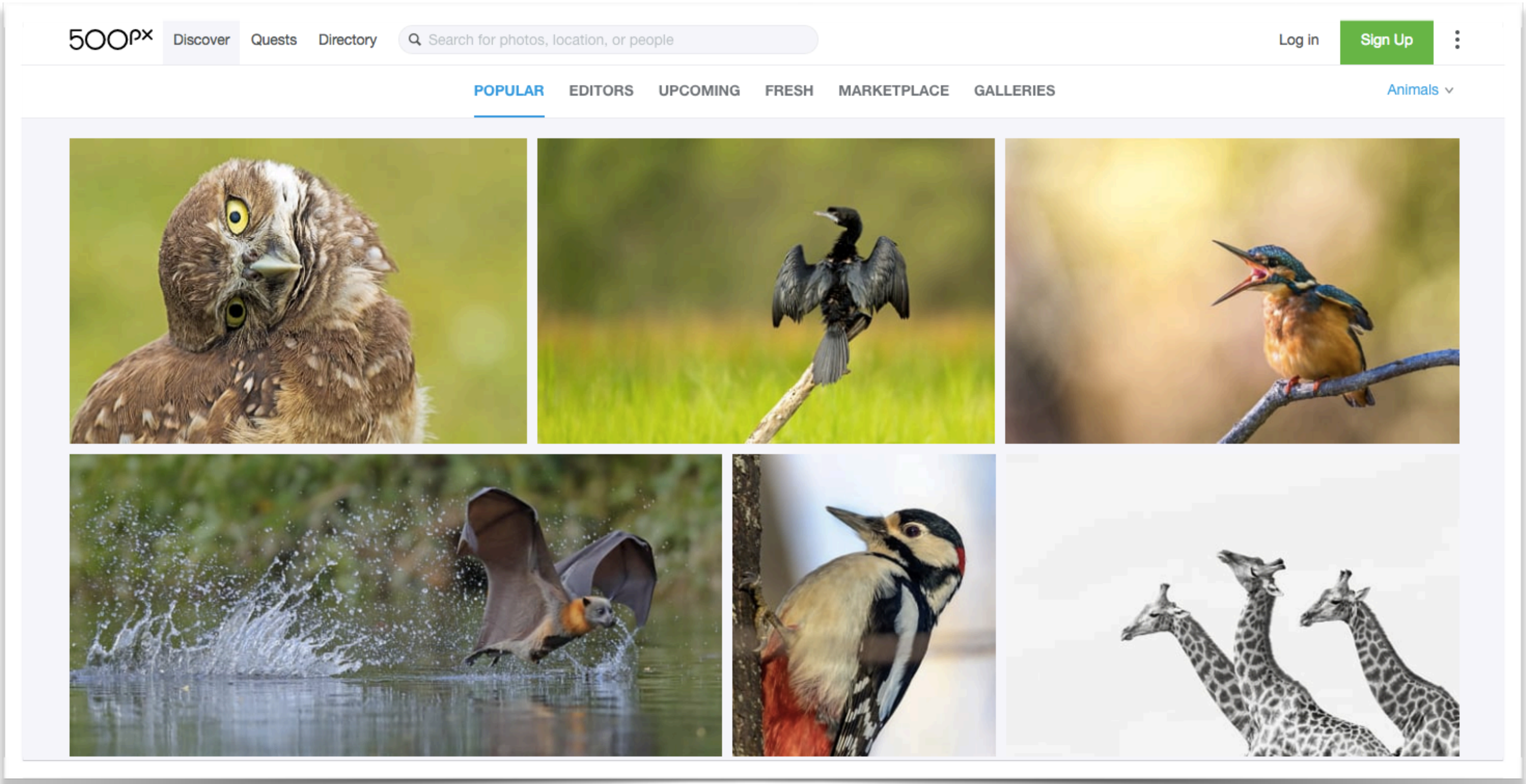
Non-visual attributes are those attributes of an image that can be inferred from visual information but do not have a clear correspondence on the image.



UNIVERSITAT DE BARCELONA

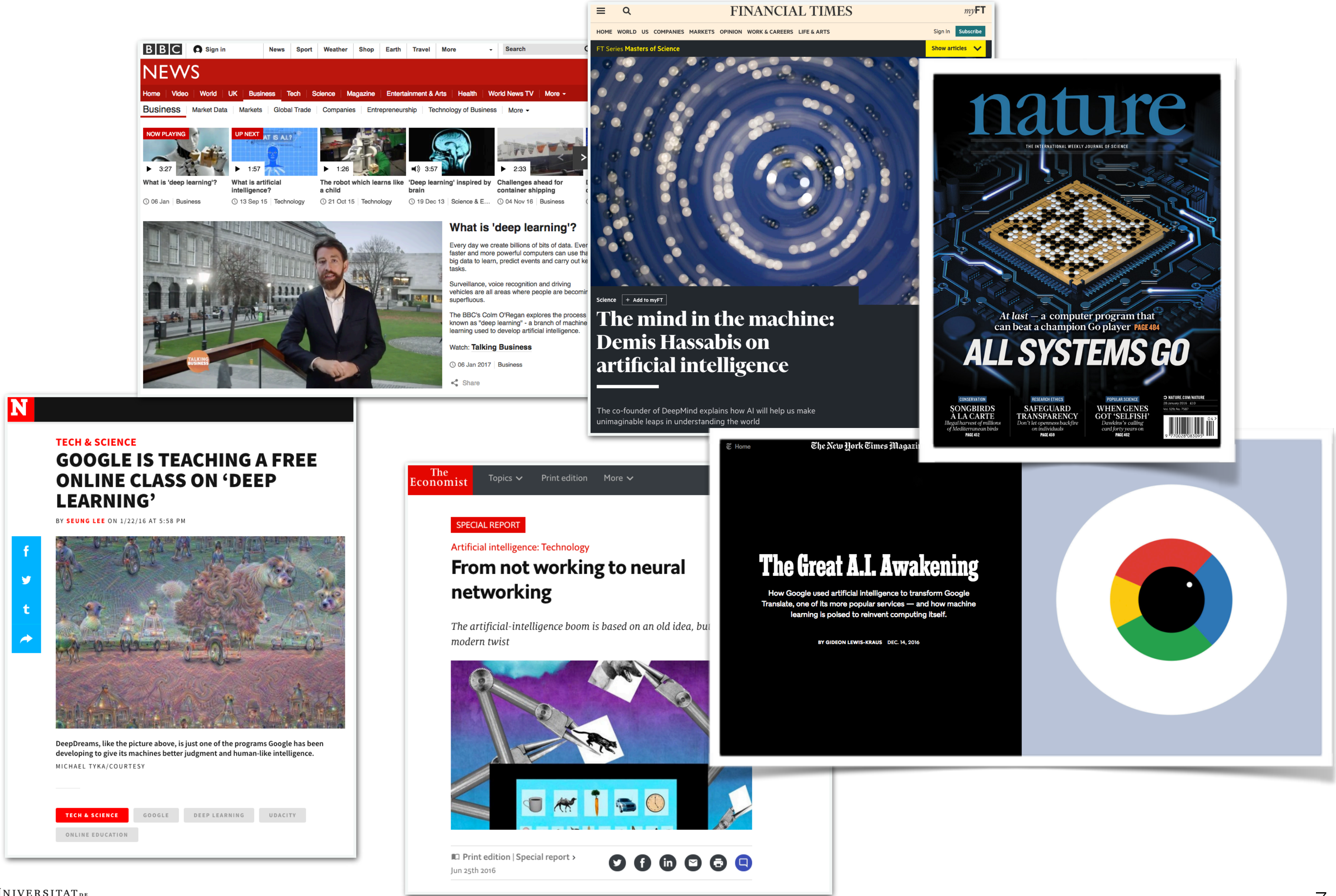
Which apartment is, a priori, more successful on Airbnb?

Online photo marketplace



Which is the expected popularity of these images?

Is Deep Learning Overhyped?



Objectives

1. What is Deep Learning?

or how to train large and highly complex models with deeply cascaded nonlinearities by using automatic differentiation and several tricks.

Deep Learning is not magic.

2. What are the main applications of Deep Learning?

computer vision, natural language, speech, recommenders, time series, etc.

3. What are the main limitations of Deep Learning?

Deep Learning is not the final machine learning method.

4. How to build deep learning models?

Keras, Tensorflow…

Why Deep Learning?

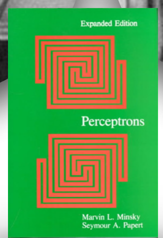
History

- In 1943, neurophysiologist **Warren McCulloch** and mathematician **Walter Pitts** wrote a paper on how neurons might work. In order to describe how neurons in the brain might work, they modeled a simple neural network using **electrical circuits.**

- In 1949, Donald **Hebb** wrote *The Organization of Behavior*, a work which pointed out the fact that neural pathways are strengthened each time they are used, a concept fundamentally essential to the ways in which humans learn. If two nerves fire at the same time, he argued, the connection between them is enhanced.

- In 1957 **Frank Rosenblatt** attempted to build a kind of mechanical brain called the **Perceptron**, which was billed as "a machine which senses, recognizes, remembers, and responds like the human mind".

- In 1962, **Widrow & Hoff** developed a learning procedure that examines the value before the weight adjusts it (i.e. 0 or 1) according to the rule: Weight Change = (Pre-Weight line value) * (Error / (Number of Inputs)). It is based on the idea that while one active perceptron may have a big error, one can adjust the weight values to distribute it across the network, or at least to adjacent perceptrons.



- A critical book written in 1969 by **Marvin Minsky** and his collaborator **Seymour Papert** showed that Rosenblatt's original system was painfully limited, literally blind to some simple logical functions like "exclusive-or" (As in, you can have the cake or the pie, but not both). What had become known as the field of "neural networks" all but disappeared.

First neural network winter is coming

- In 1982, interest in the field was renewed. **John Hopfield** of Caltech presented a paper to the National Academy of Sciences. His approach was to create more useful machines by using bidirectional lines. Previously, the connections between neurons was only one way.

- In 1986, the problem was how to extend the Widrow-Hoff rule to multiple layers. Three independent groups of researchers, which included **David E. Rumelhart, Geoffrey E. Hinton** and **Ronald J. Williams**, came up with similar ideas which are now called **back-propagation networks** because it distributes pattern recognition errors throughout the network.

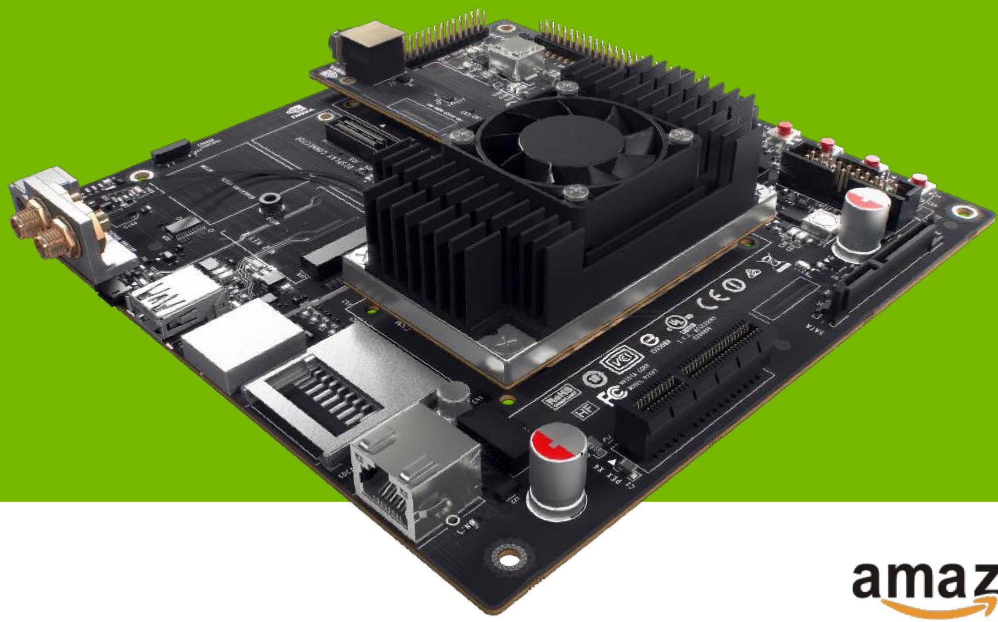
- From 1986 to mid 90's new developments arised: convolutional neural networks (**Y.LeCun**), unsupervised learning (**Y.Bengio**), RBM (**G.Hinton**), etc. But, by this point **new machine learning methods** had begun to also emerge, and people were again beginning to be skeptical of neural nets since they seemed so intuition-based and since computers were still barely able to meet their computational needs.

Second neural network winter is coming

- With the ascent of **Support Vector Machines** and the failure of backpropagation, the early 2000s were a dark time for neural net research.

- Then, what every researcher must dream of actually happened: G.Hinton, S.Osindero, and Y.W.Teh published a paper in 2006 that was seen as a breakthrough, a breakthrough significant enough to rekindle interest in neural nets: *A fast learning algorithm for **deep** belief nets.*

- After that, following Moore's law, computers got dozens of times faster (GPUs) since the slow days of the 90s, making learning with large datasets and many layers much more tractable.



Jetson TX1 Developer Kit

$599 retail

$299 edu

Pre-order Nov 12

Shipping Nov 16 (US)

Intl to follow

amazon newegg.com MICRO CENTER computers & electronics NVIDIA

GPU democratization

NVIDIA ACCELERATED COMPUTING Downloads Training Ecosystem Forums Log in

GPU Grant Program

Home > ComputeWorks > Academic Collaboration > GPU Grant Program

NVIDIA's Academic Programs Team is dedicated to empowering and collaborating with professors and researchers at universities worldwide. We aim to inspire cutting-edge technological innovation and to find new ways of enhancing faculty research as well as the teaching and learning experience. We achieve this through a variety of initiatives and programs including:

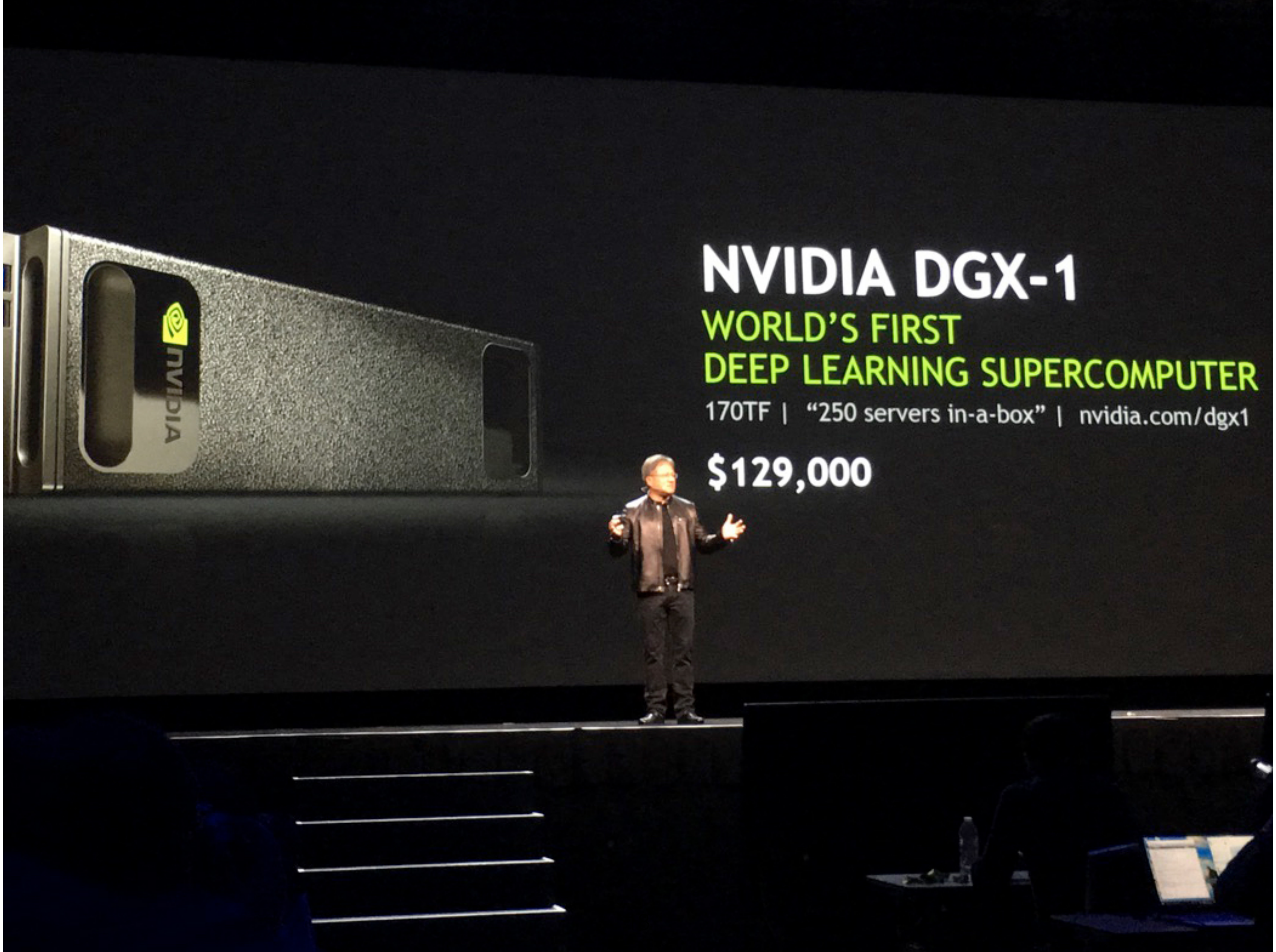
- Small scale GPU grants
- Graduate Fellowships
- Providing free teaching materials and GPU cloud resources through our Deep Learning Institute (DLI) Teaching Kits
- Providing access to developer forums, pre-released tools and drivers through the Accelerated Computing Developer Program

GPU Grant Requests

Professors, Researchers and Advisors should complete the online GPU Grant Request Form to request a GPU for research purposes for themselves and/or their teams. In order to review your request, we require a statement of research and CV. Additional materials should not be used to attach a full proposal in lieu of a statement of research. Your statement of research must include the following:

- contact information
- a short description of your research project(s)
- how you and/or your team will use the GPU
- list of recent publications

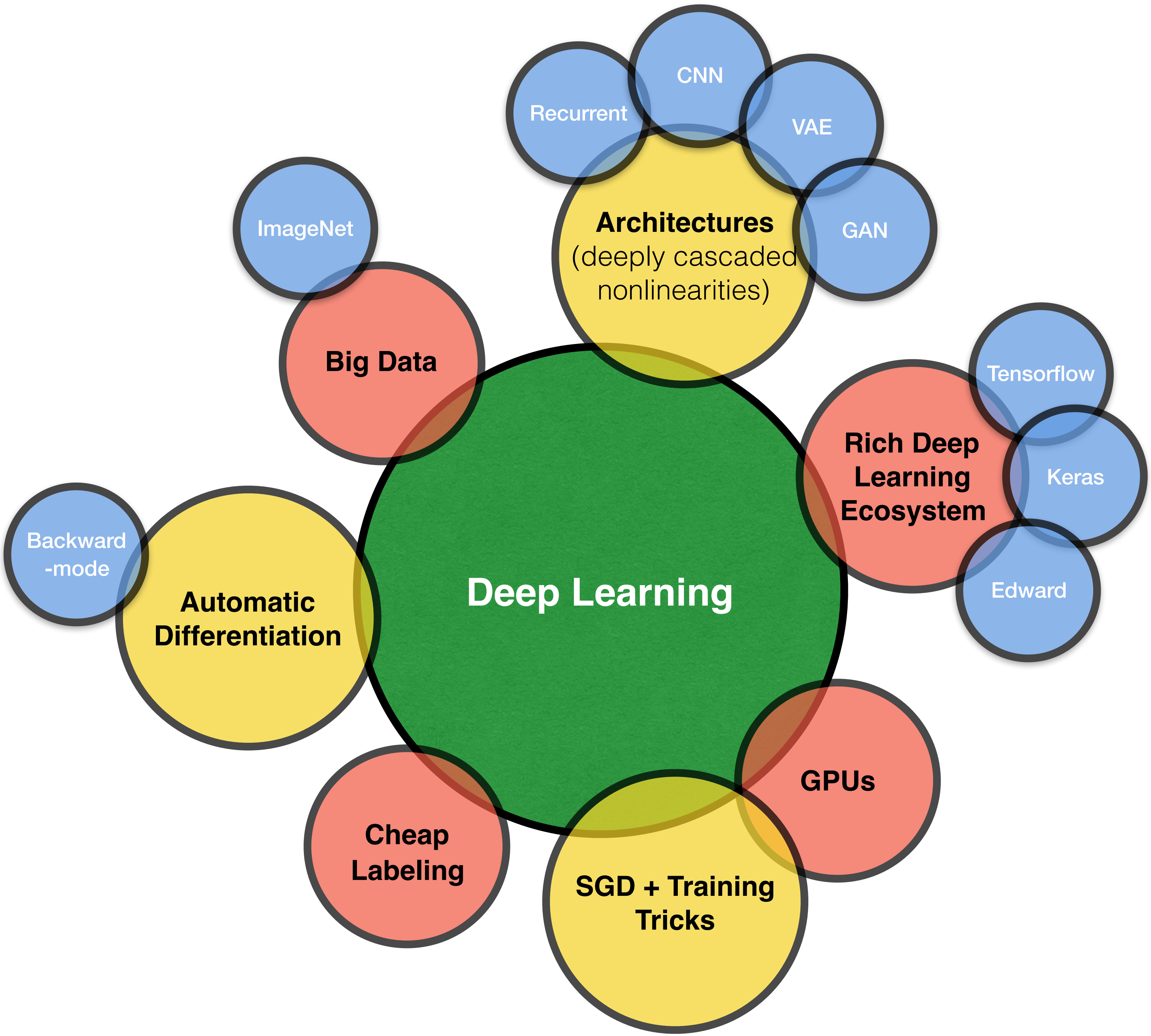
Thank you NVIDIA!


NVIDIA DGX-1
WORLD'S FIRST
DEEP LEARNING SUPERCOMPUTER
170TF | "250 servers in-a-box" | nvidia.com/dgx1
$129,000
NVIDIA

- **Neural Networks (NN)** is a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data.

- **Deep Learning (DL)** is a powerful set of techniques (and tricks) for learning in deep neural networks.

- NN and DL currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing.

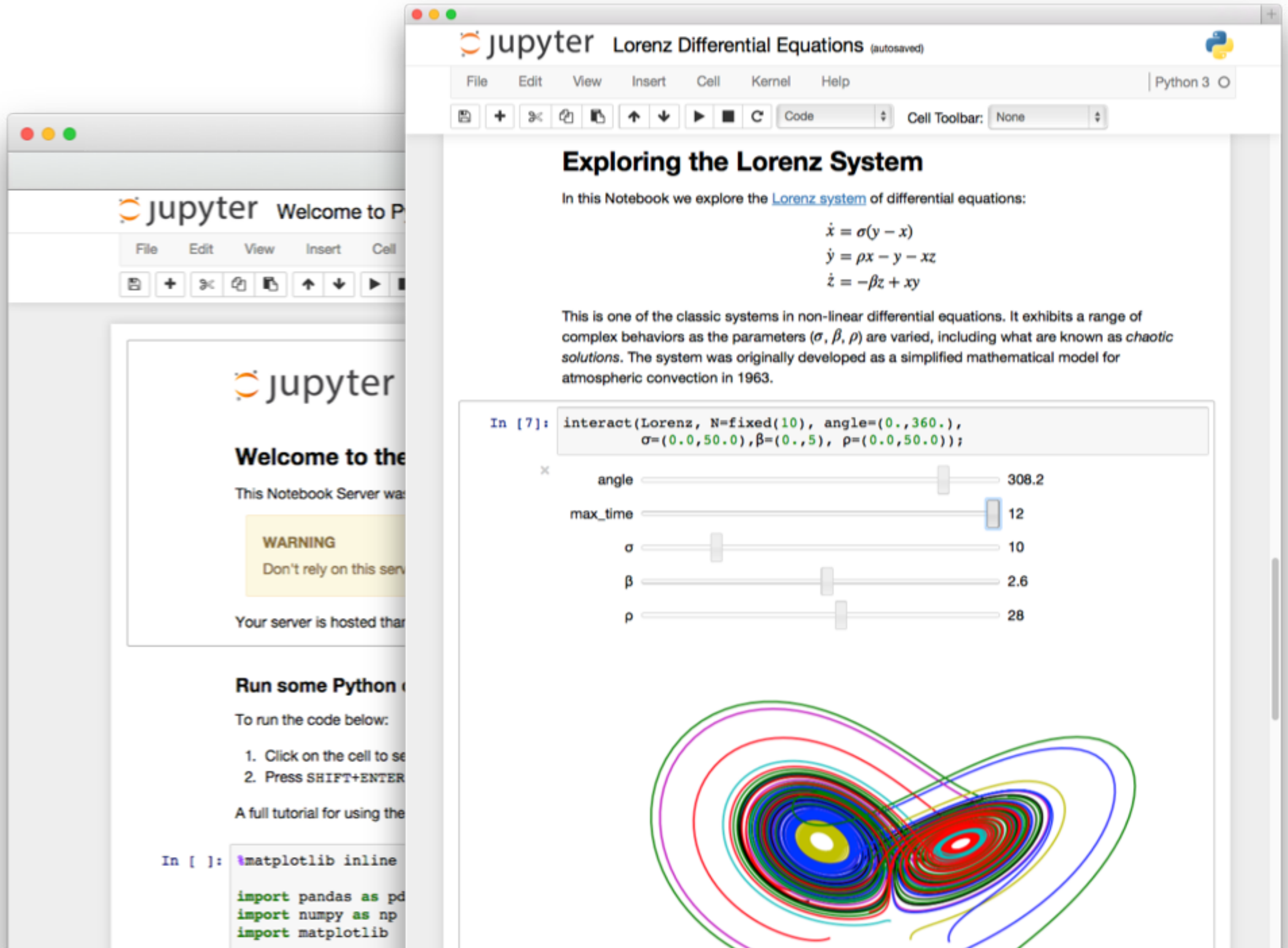

Recurrent
CNN
VAE
GAN
Architectures
(deeply cascaded
nonlinearities)
ImageNet
Big Data
Tensorflow
Keras
Edward
Rich Deep
Learning
Ecosystem
Backward
-mode
Automatic
Differentiation
Deep Learning
GPUs
Cheap
Labeling
SGD + Training
Tricks

Our objectives

- Optimization and Automatic Differentiation
- Programming a Neural Network
- Design and Train a Deep Model

Approach

We will illustrate all contents with Jupyter notebooks, a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.



Approach

We will use a **Docker Container**.

Docker provides the ability to build a runtime environment that not only remains isolated from other running containers, but also can be deployed to multiple locations in a repeatable way. Docker also uses a text document – a Dockerfile – that contains all the commands to assemble an image, which will meet our need to document the build environment. Finally, Docker's runtime options enable us to attach GPU devices when deploying on remote servers.

The problem: **machine learning**

Training data: a set of $(x^{(m)}, y^{(m)})$ pairs.
Learn a function $f_w : x \to y$ to predict on new inputs x.

1. Choose a model function family f_w.

2. Optimize parameters w.

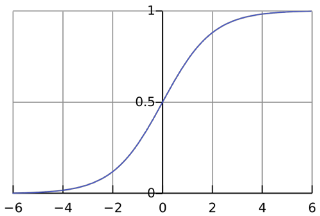
1-layer neural net model

Parameters

Weights Bias

$$f(x) = \sigma(w^T \cdot x + b)$$

Dot Product



Sigmoid Function

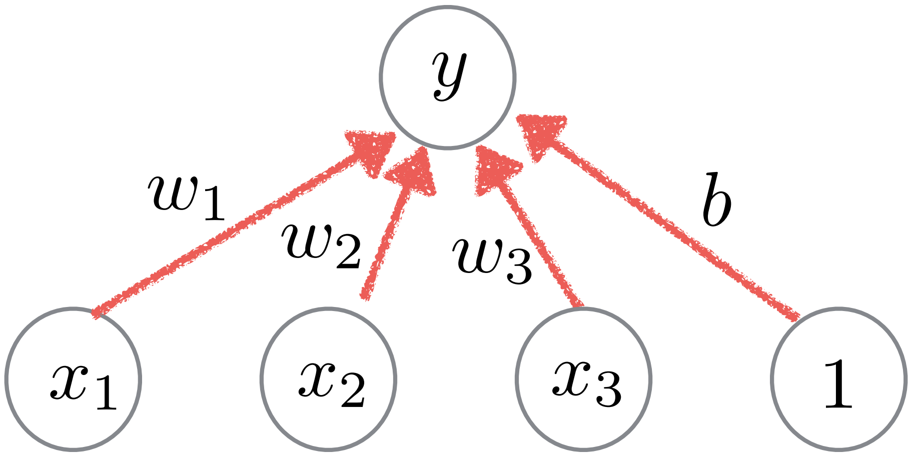
$$\sigma(x) = \frac{1}{(1+e^{-x})}$$

ReLU Function

$$\sigma(x) = max(0, x)$$

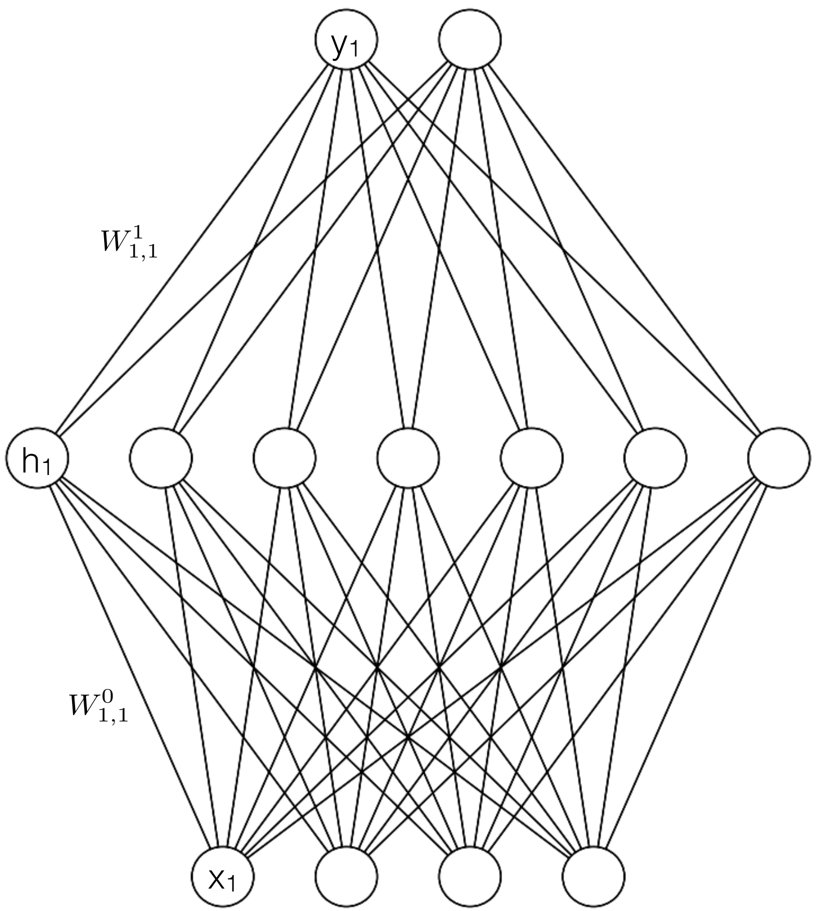
1-layer neural net model

$$f(x) = \sigma(w^T \cdot x + b)$$



Graphical Representation

2-layer neural net model



Output Layer
(y1, y2)

Weights W1

$$y_i = \sigma(\textstyle\sum_{j=1}^{7} h_i * W^1_{i,j})$$

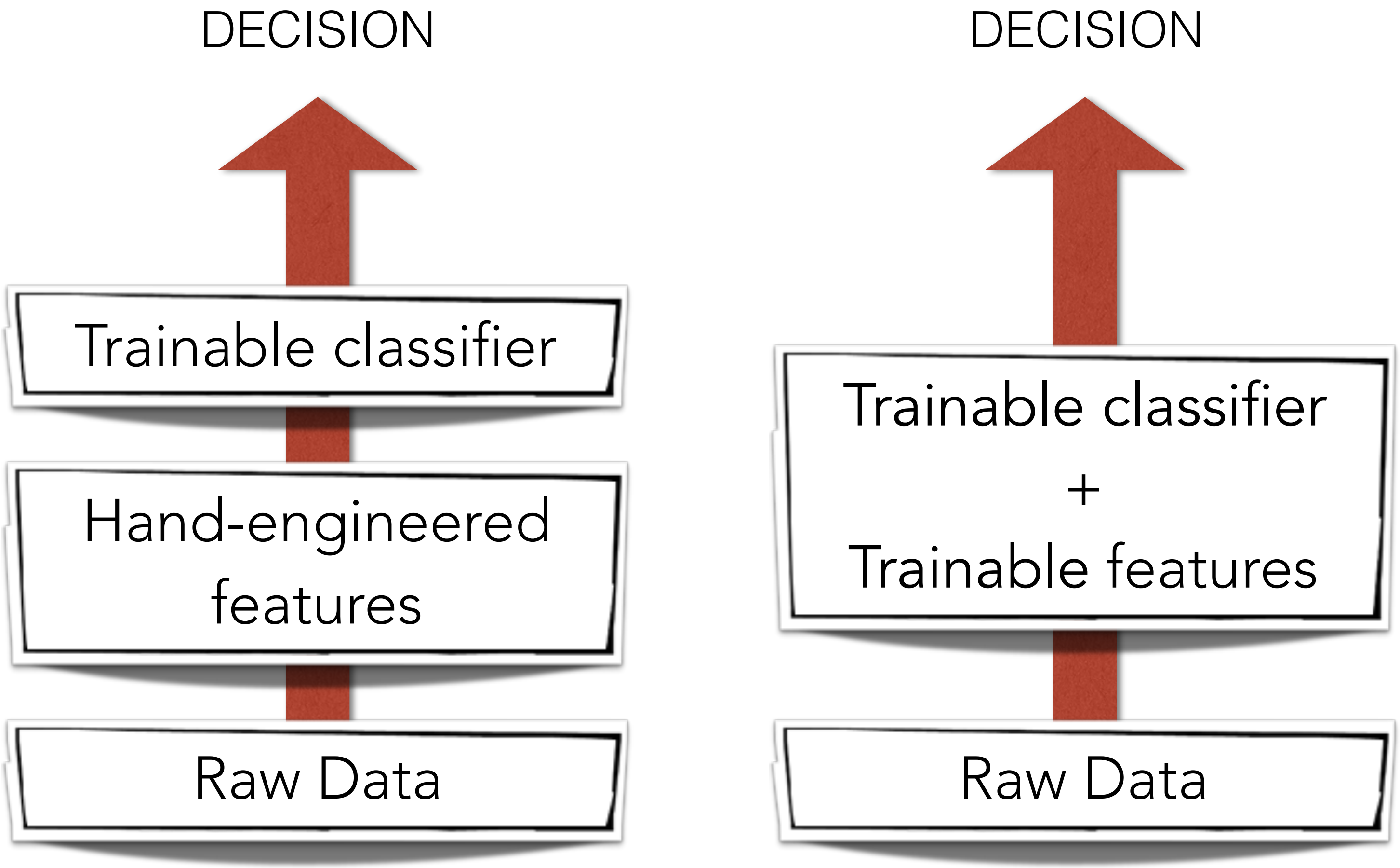
Hidden Layer
(h1, h2, h3, h4, h5, h6 h7)

Weights W0

$$h_i = \sigma(\textstyle\sum_{j=1}^{7} x_i * W^0_{i,j})$$

Input Layer
(x1, x2, x3, x4)

What's new?



STANDARD MACHINE LEARNING

DEEP LEARNING

28

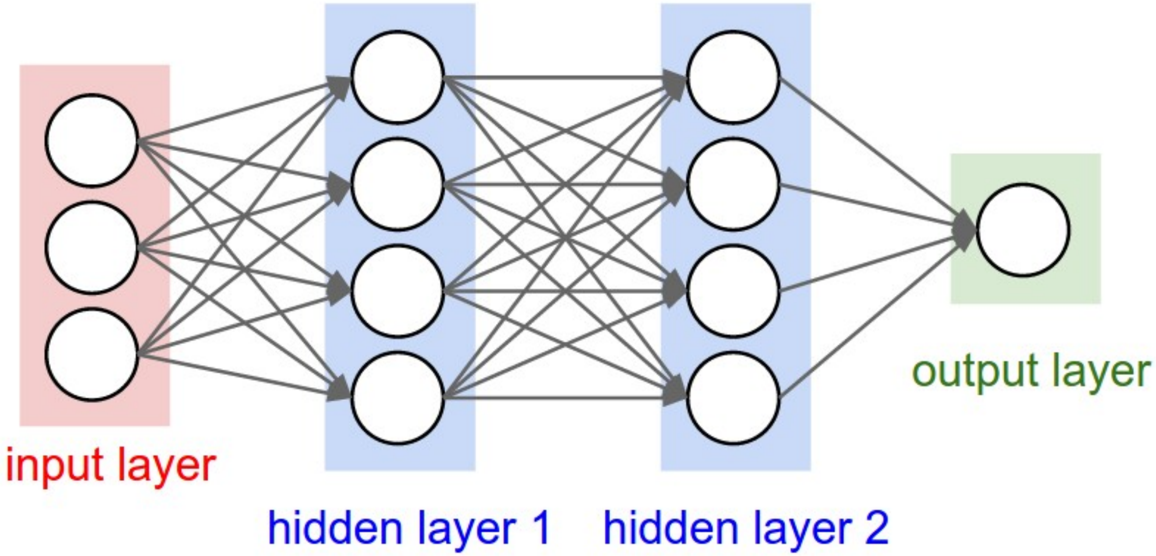
Automatic Differentiation

```
import autograd.numpy as np   # Thinly-wrapped version of Numpy
from autograd import grad

def taylor_sine(x):  # Taylor approximation to sine function
    ans = currterm = x
    i = 0
    while np.abs(currterm) > 0.001:
        currterm = -currterm * x**2 / ((2 * i + 3) * (2 * i + 2))
        ans = ans + currterm
        i += 1
    return ans

grad_sine = grad(taylor_sine)
print "Gradient of sin(pi) is", grad_sine(np.pi)
```

SGD-based logistic regression

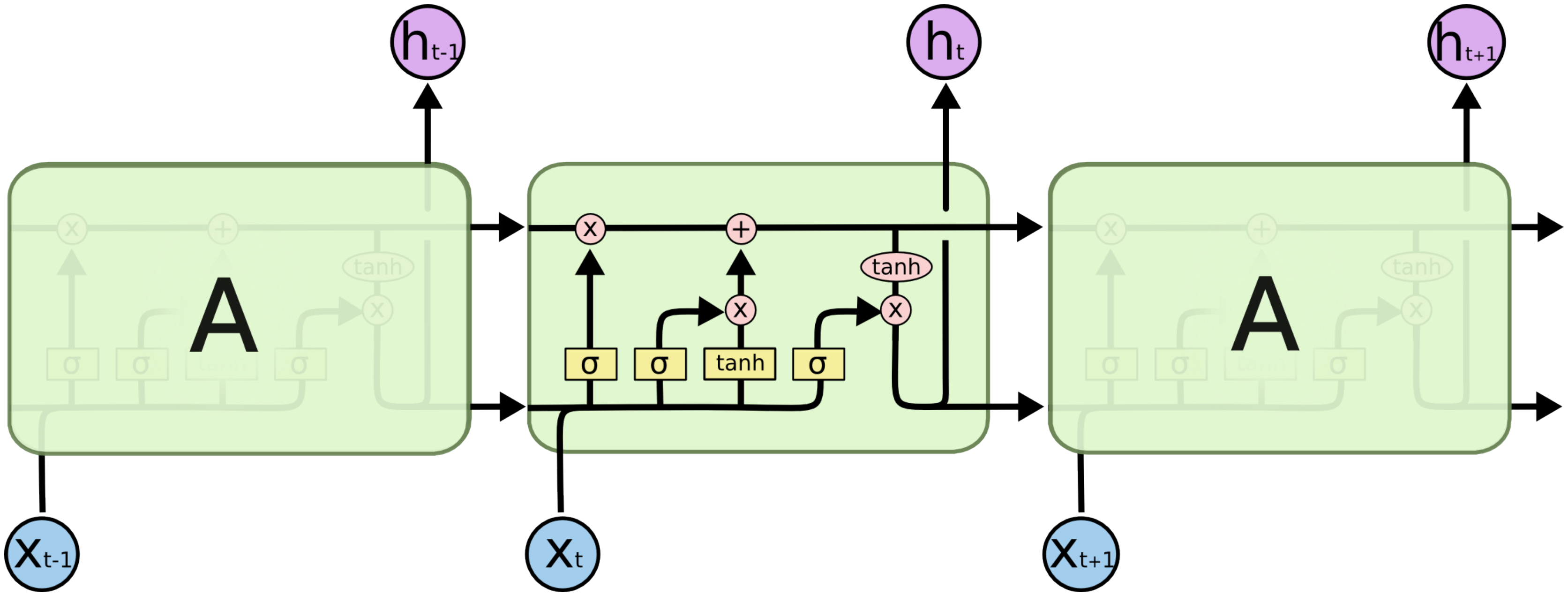
```python
import autograd.numpy as np
from autograd import grad

def sigmoid(x):
    return 0.5*(np.tanh(x) + 1)

def logistic_predictions(weights, inputs):
    # Outputs probability of a label being true according to logistic model.
    return sigmoid(np.dot(inputs, weights))

def training_loss(weights):
    # Training loss is the negative log-likelihood of the training labels.
    preds = logistic_predictions(weights, inputs)
    label_probabilities = preds * targets + (1 - preds) * (1 - targets)
    return -np.sum(np.log(label_probabilities))

# Build a toy dataset.
inputs = np.array([[0.52, 1.12,  0.77],
                   [0.88, -1.08, 0.15],
                   [0.52, 0.06, -1.30],
                   [0.74, -2.49, 1.39]])
targets = np.array([True, True, False, True])

# Define a function that returns gradients of training loss using autograd.
training_gradient_fun = grad(training_loss)

# Optimize weights using gradient descent.
weights = np.array([0.0, 0.0, 0.0])
print "Initial loss:", training_loss(weights)
for i in xrange(100):
    weights -= training_gradient_fun(weights) * 0.01

print  "Trained loss:", training_loss(weights)
```

Architectures

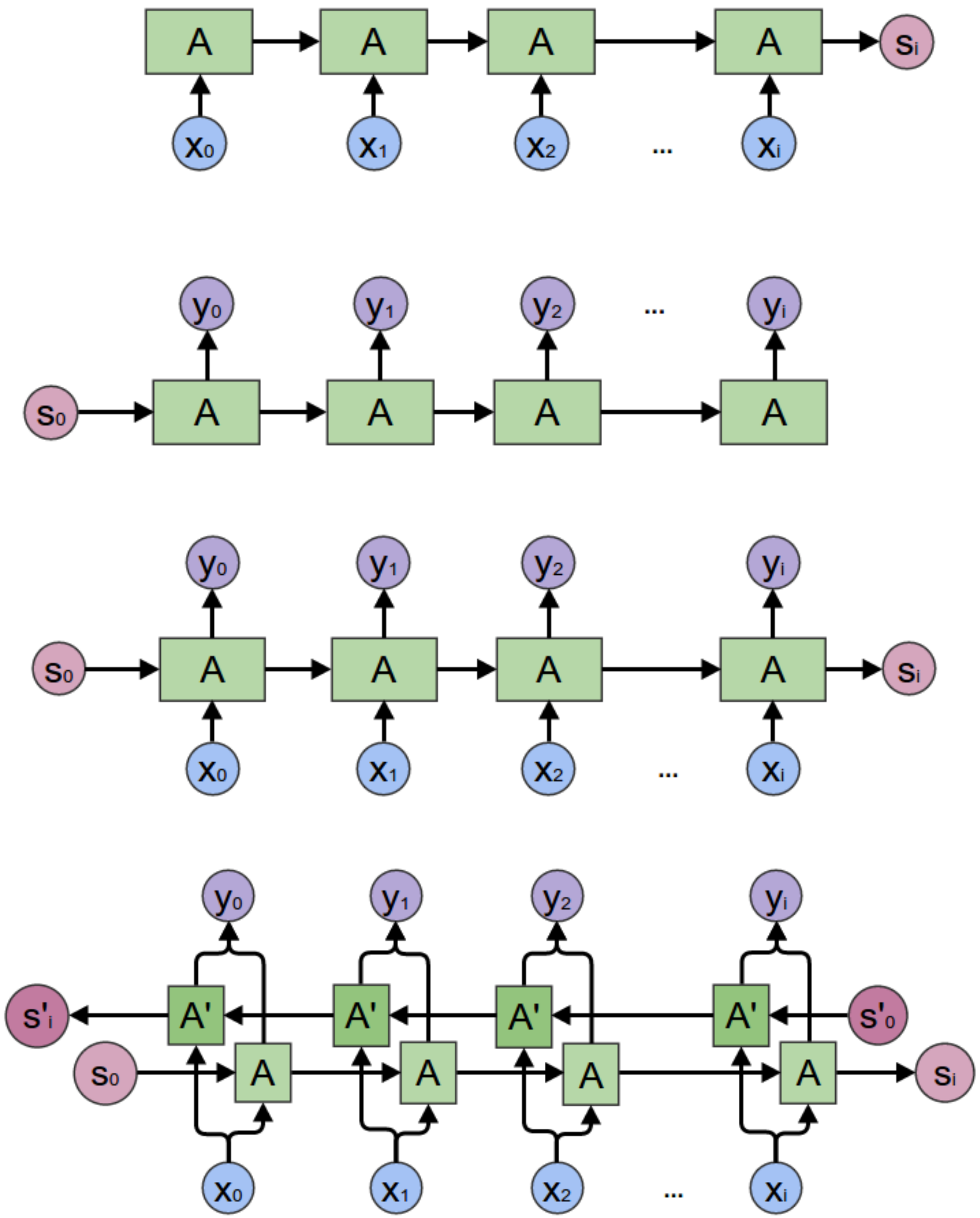


Each layer is a function, acting on the output of a previous layer. As a whole, the network is a chain of composed functions. This chain of composed functions is optimized to perform a task.

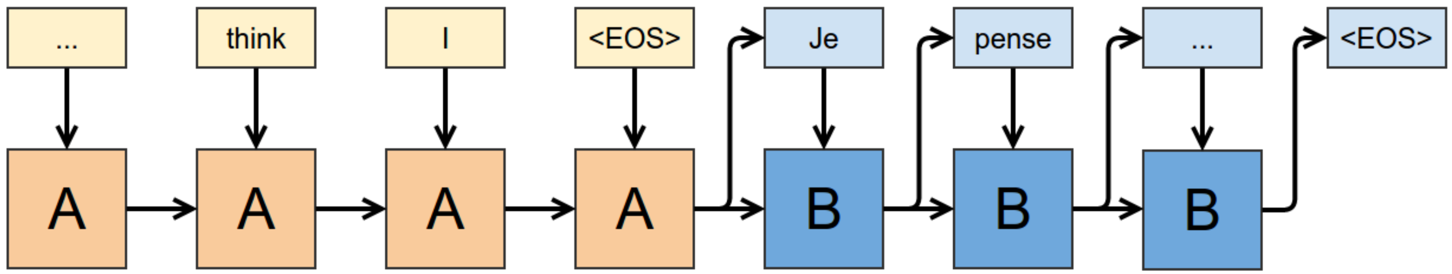
Recurrent neural layer model



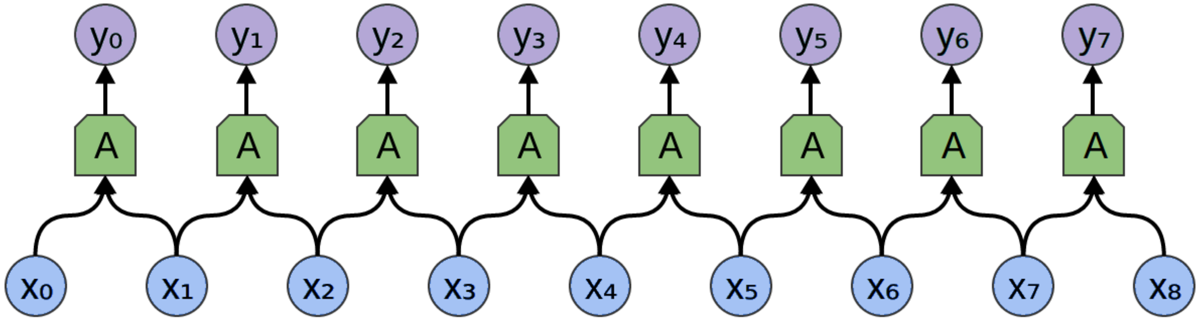
Recurrent neural layer model

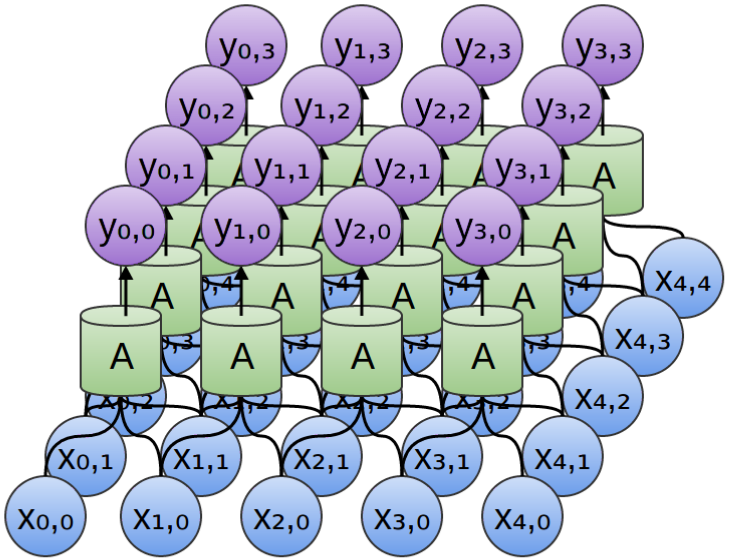


Recurrent neural layer model

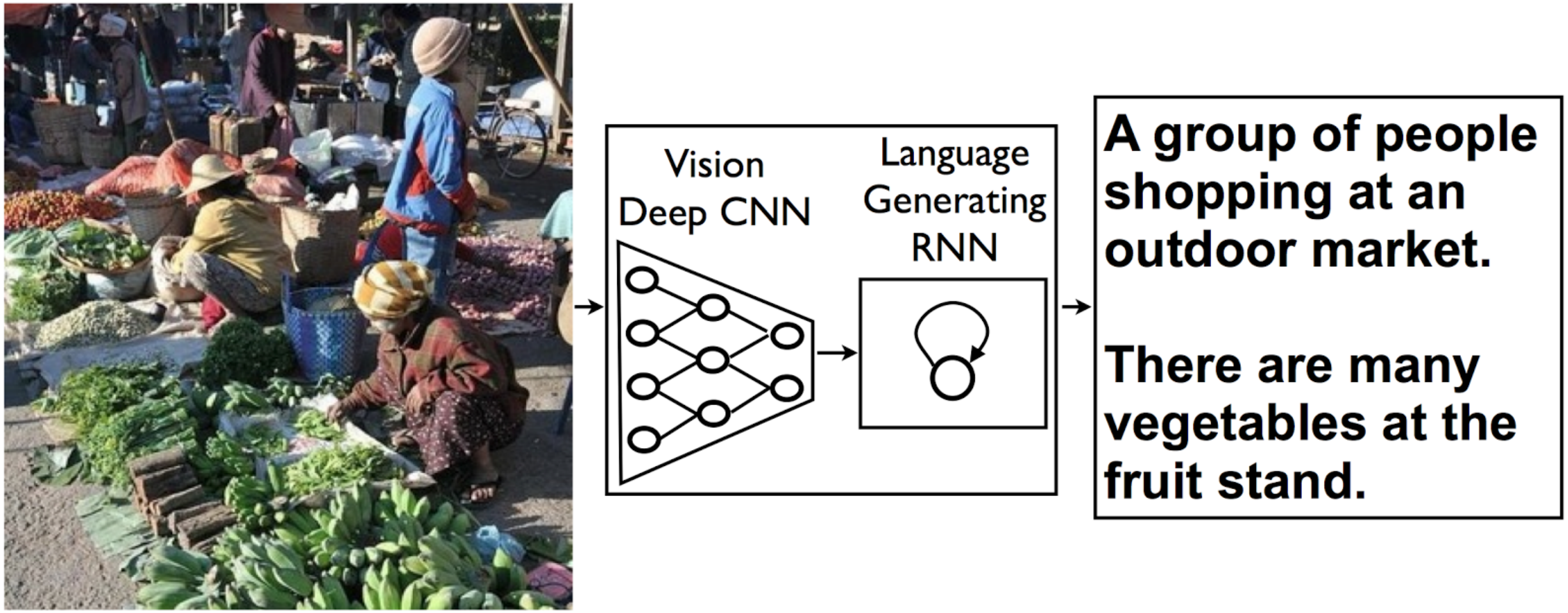


Convolutional neural layer model

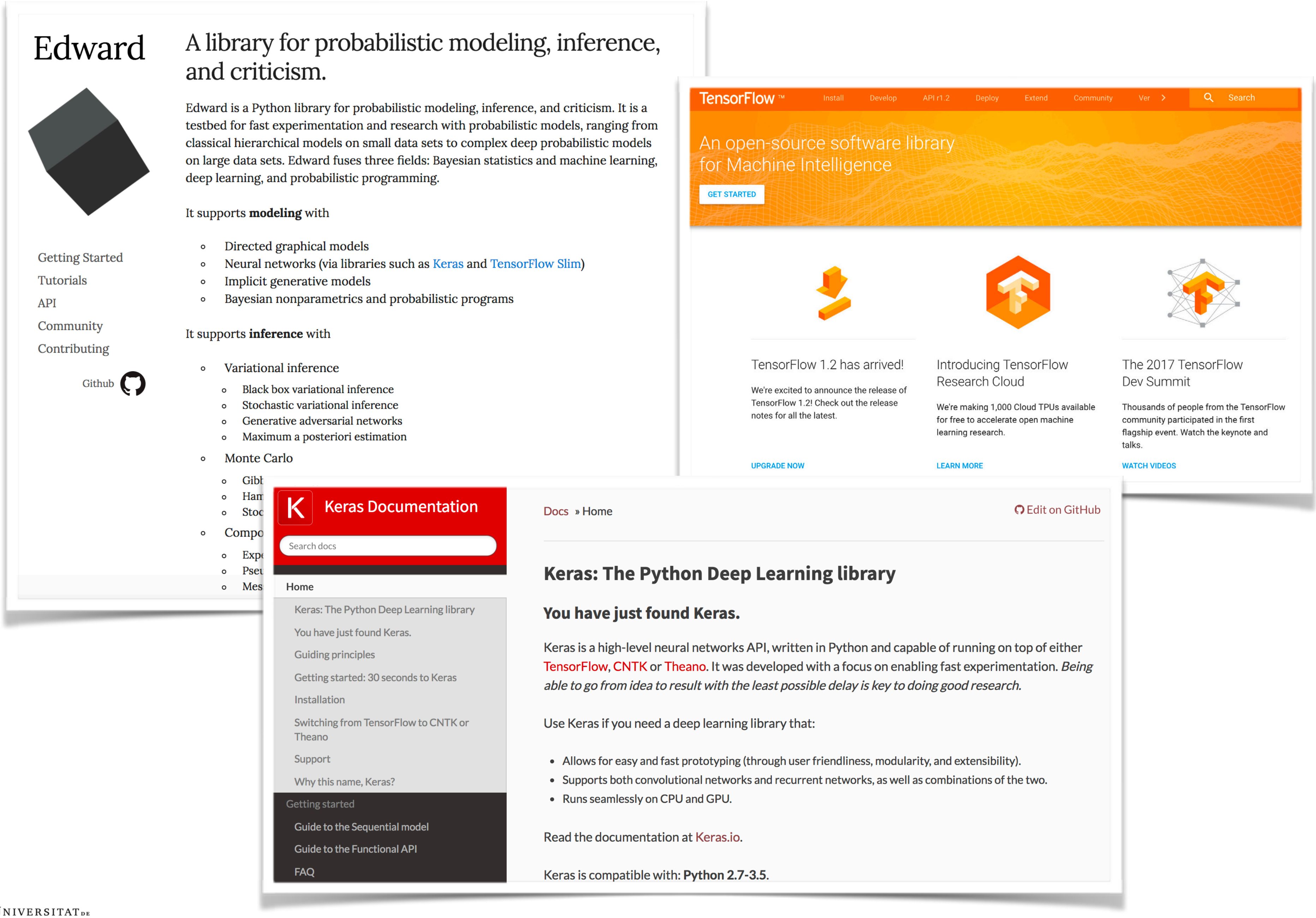




CNN + RNN

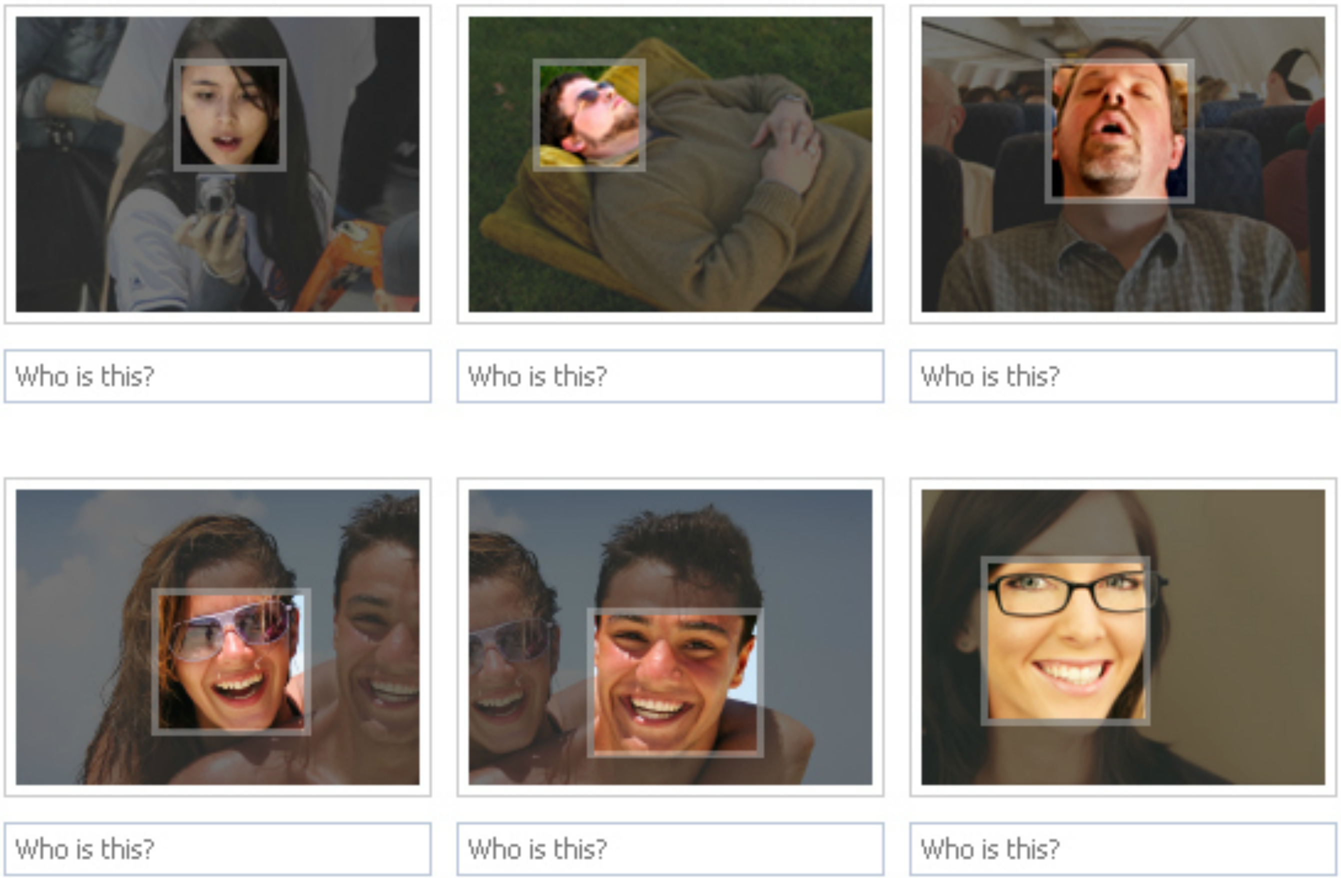

Vision
Deep CNN
Language
Generating
RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

Deep Learning Ecosystem



“Classical” applications:
object classification, detection and segmentation.

Face recognition.



DeepFace (Facebook): Accuracy of 97.35%

New applications: navigation and mapping.

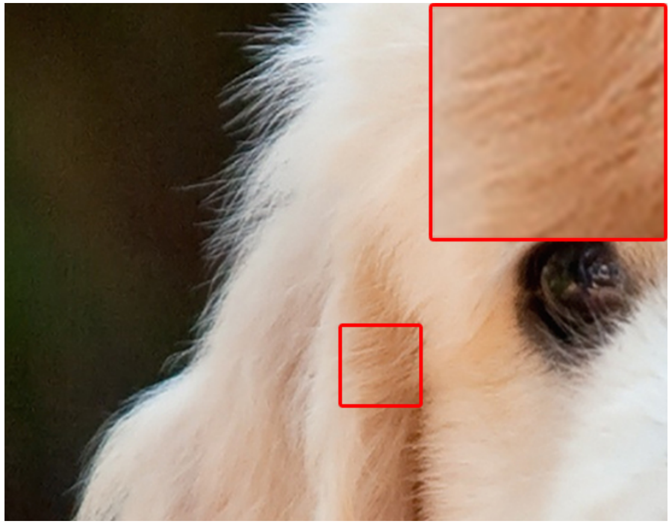
New applications: Image Upscaling (Flipboard)

http://engineering.flipboard.com/2015/05/scaling-convnets/

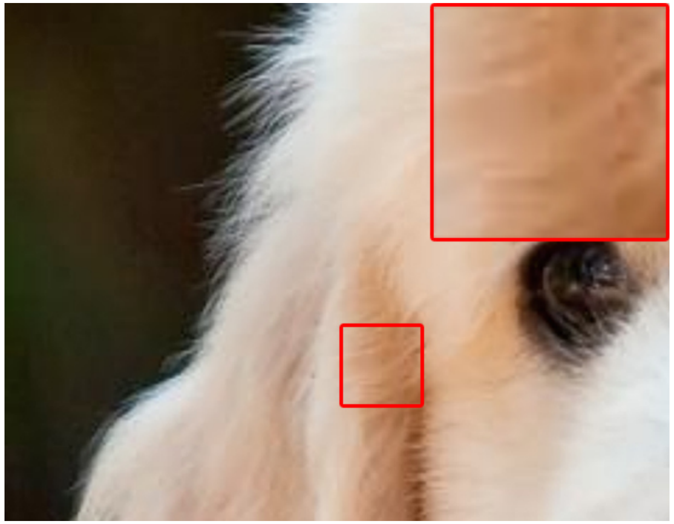
New applications: Image Upscaling (Flipboard)

http://engineering.flipboard.com/2015/05/scaling-convnets/

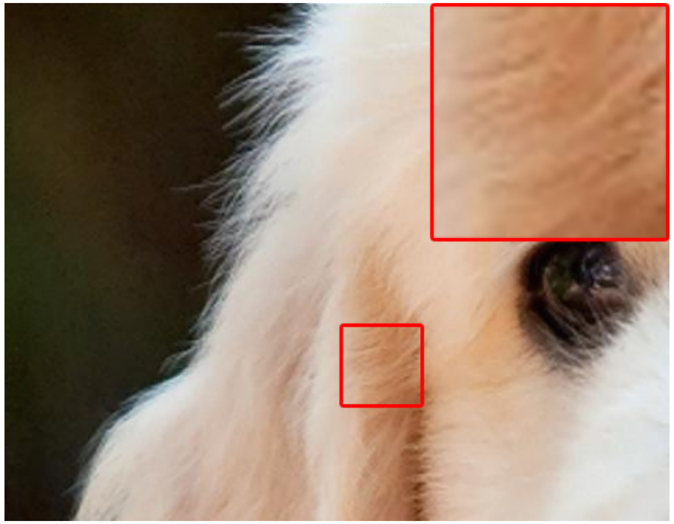
New applications: Image Upscaling (Flipboard)



Original



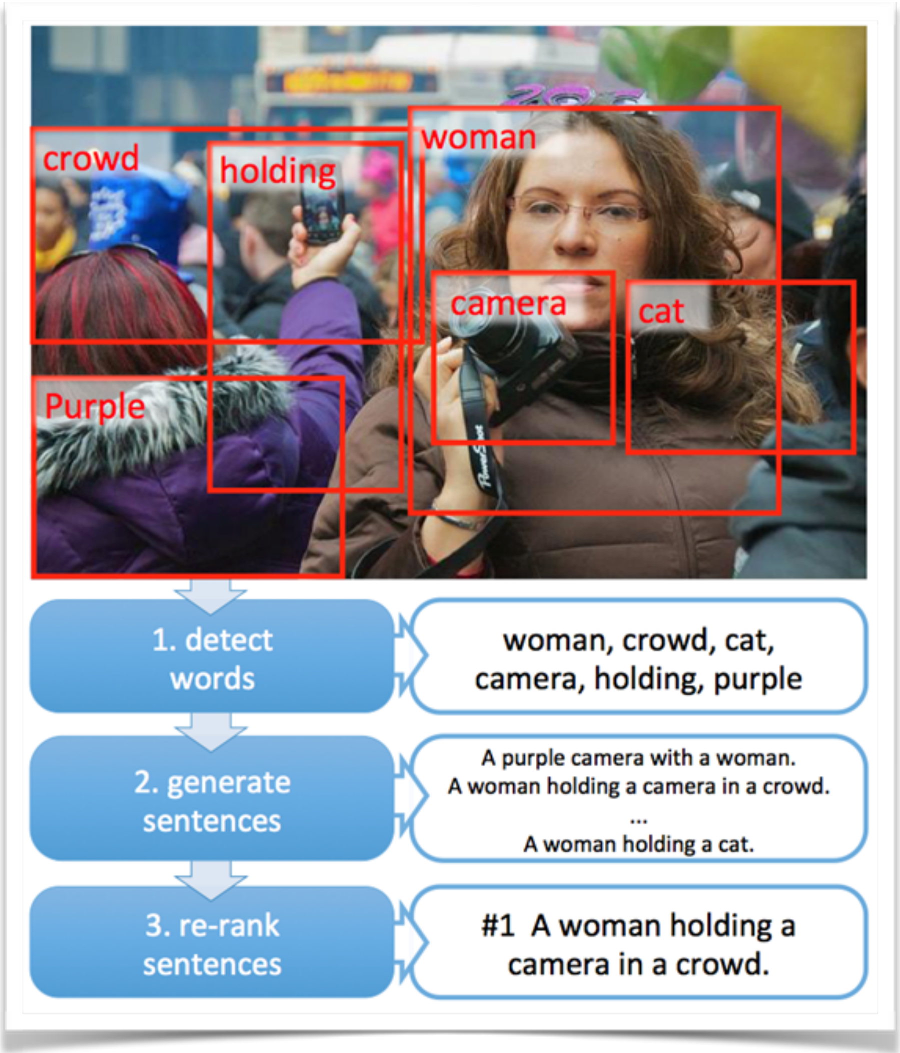
Bicubic



Model

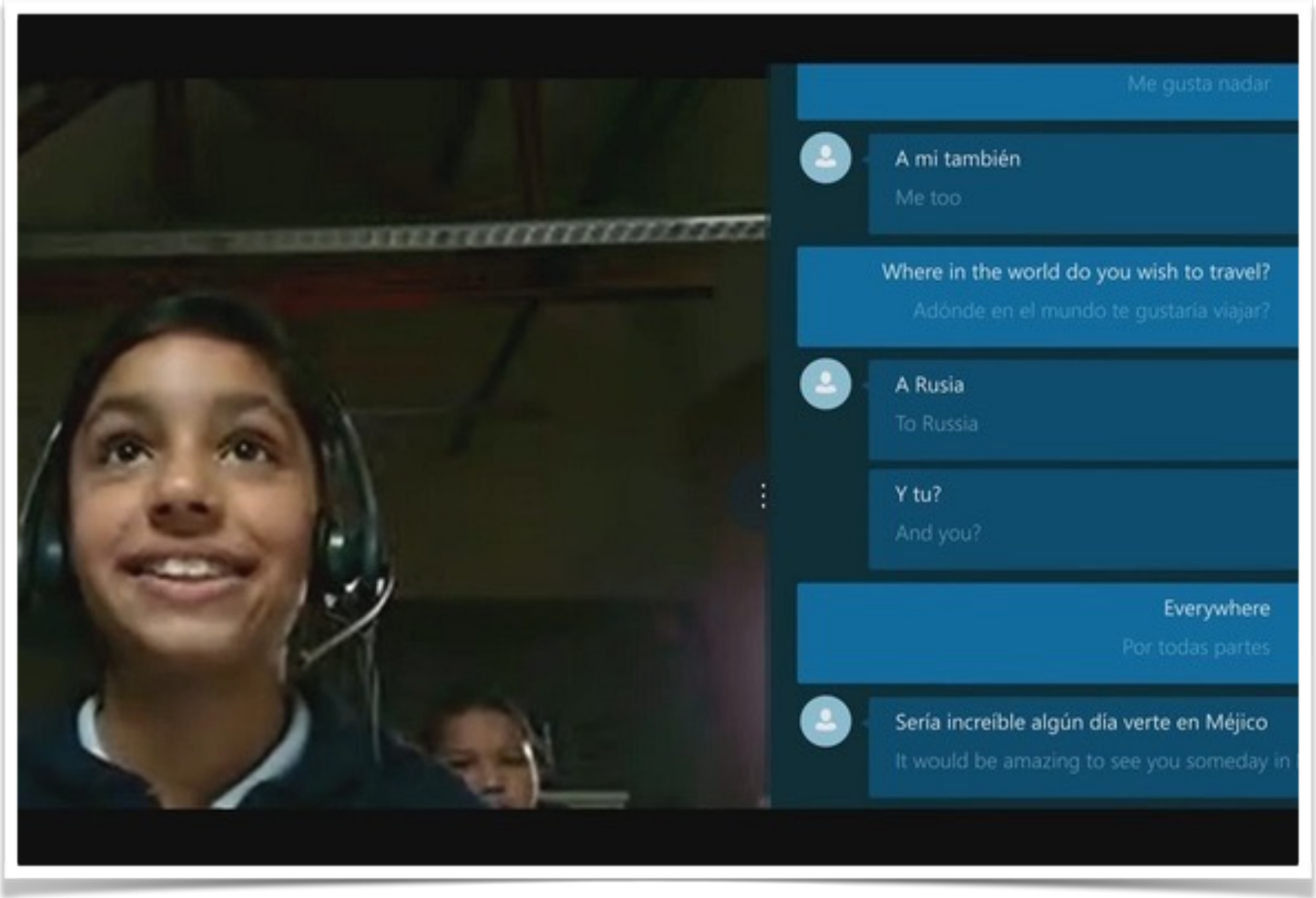
http://engineering.flipboard.com/2015/05/scaling-convnets/

New applications: Automatic Image Captioning

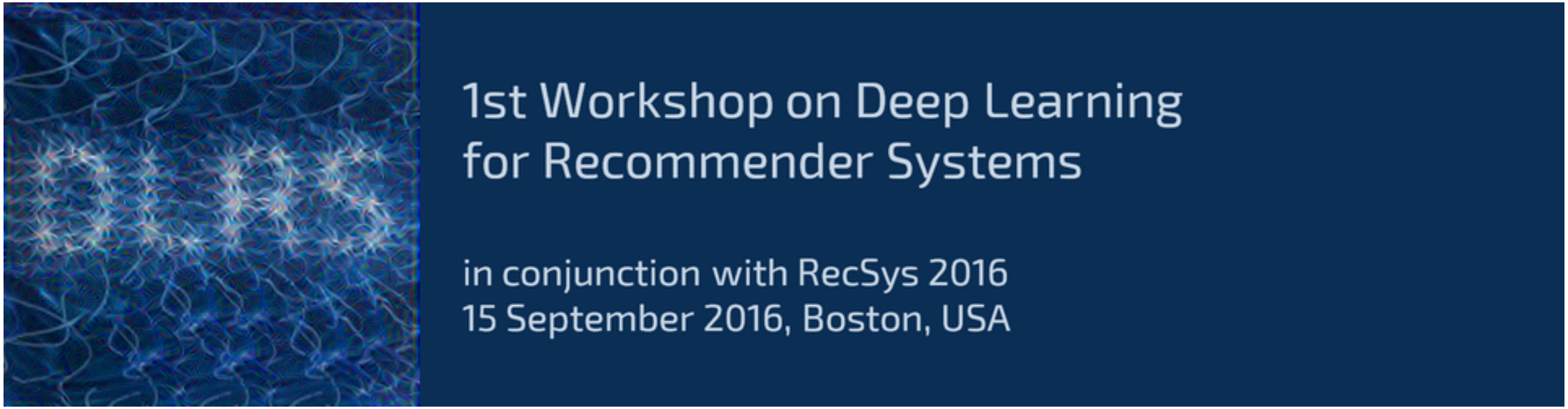


http://blogs.technet.com/b/machinelearning/archive/2014/11/18/rapid-progress-in-automatic-image-captioning.aspx

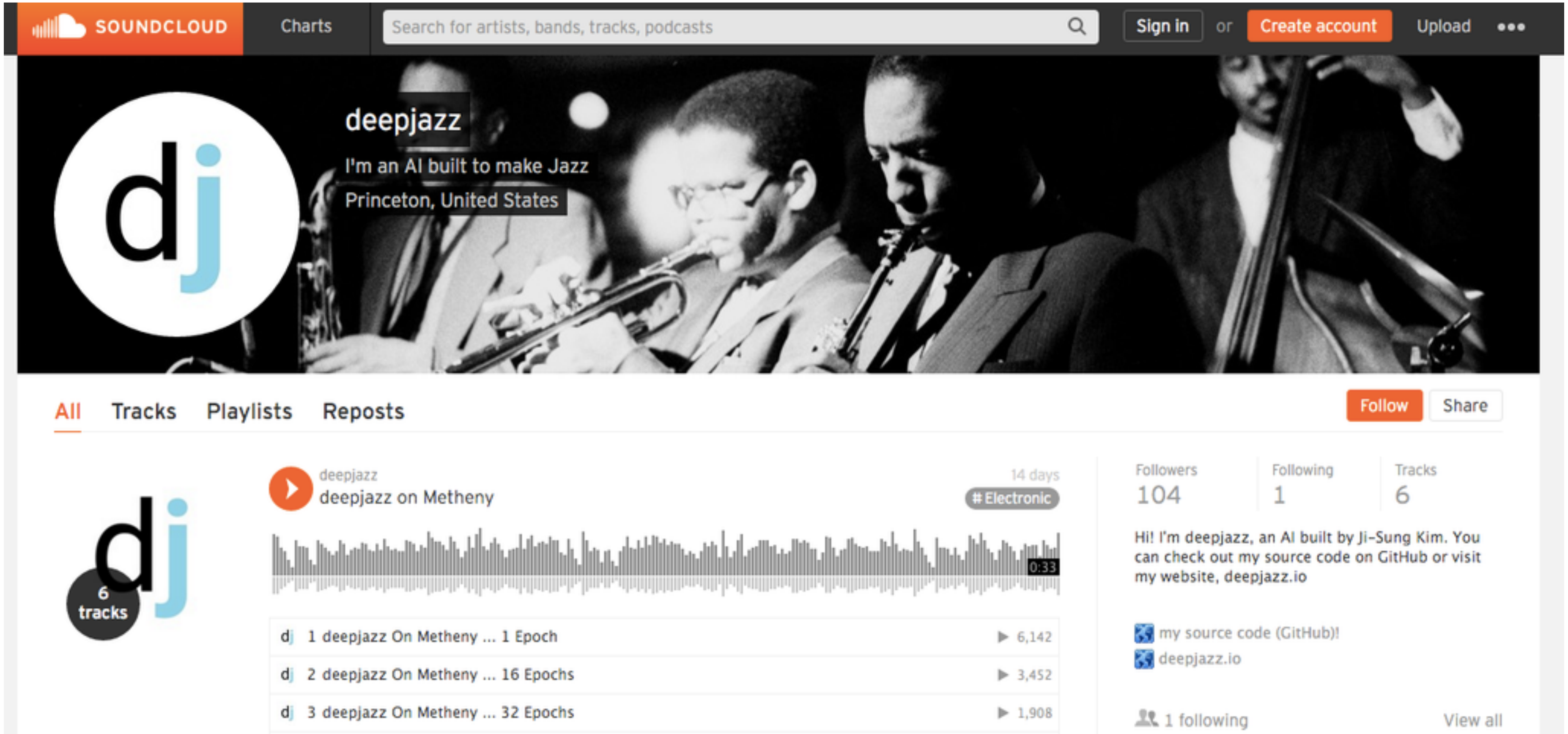
Speech translation



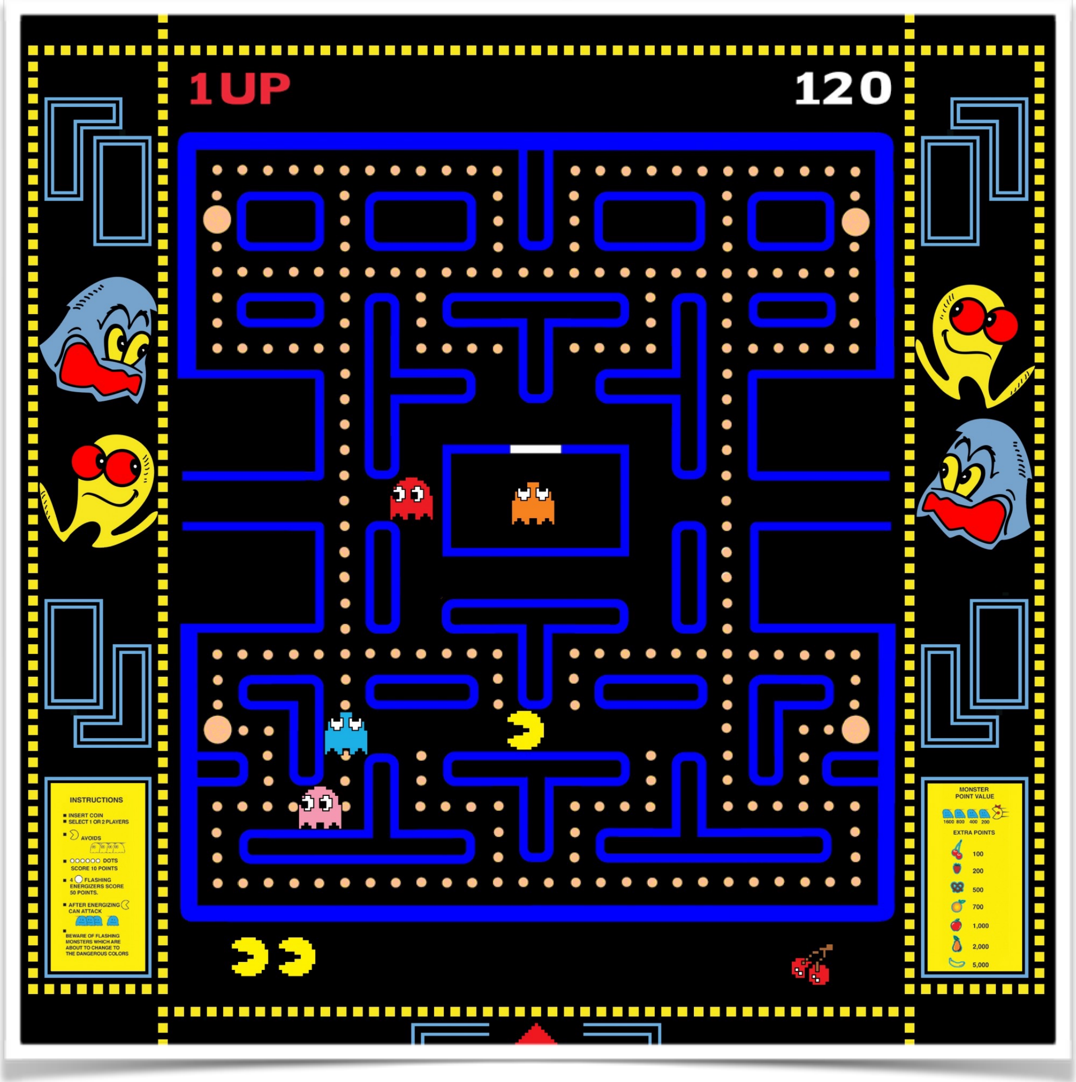
Recommenders



Music Generation



Reinforcement learning.

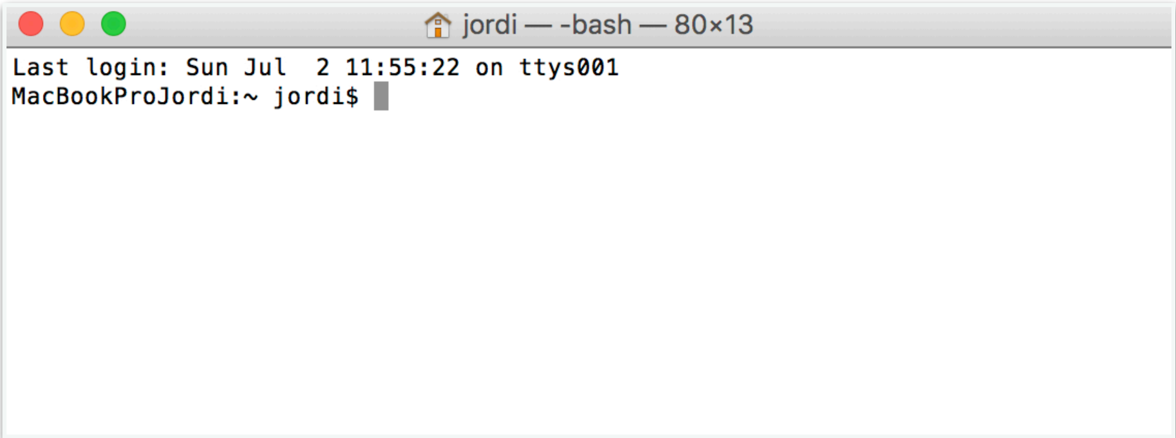


Go

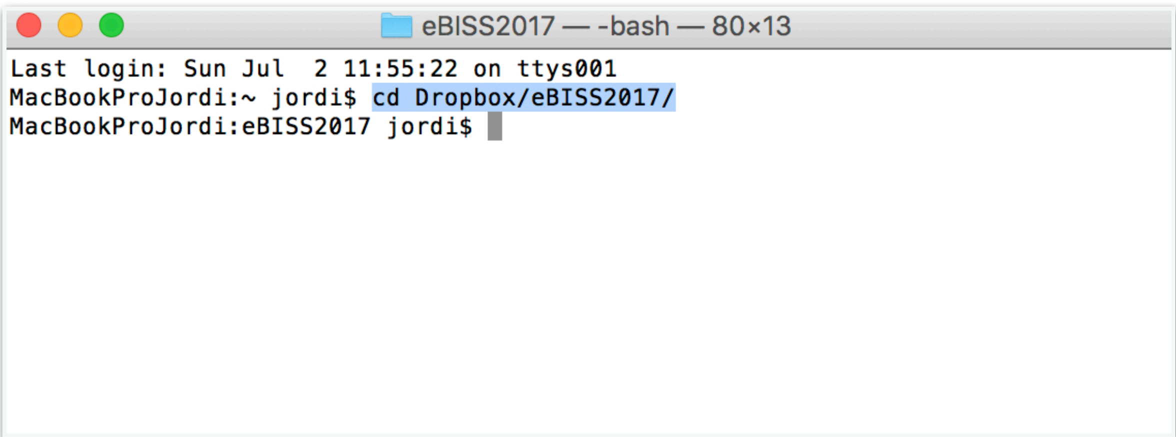

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card forty years on
PAGE 462
NATURE.COM/NATURE
28 January 2016 £10
Vol. 529, No. 7587

Hands On!

Open a terminal window



Go to the working directory of your choice



Hands On!

Start your docker image

```
eBISS2017 — -bash — 82×10
```

```
Last login: Sun Jul  2 12:27:25 on ttys002
MacBookProJordi:~ jordi$ cd Dropbox/eBISS2017/
MacBookProJordi:eBISS2017 jordi$ docker run -it -p 8888:8888 -v /$(pwd):/notebooks
 datascienceub/deepubebiss2017
```

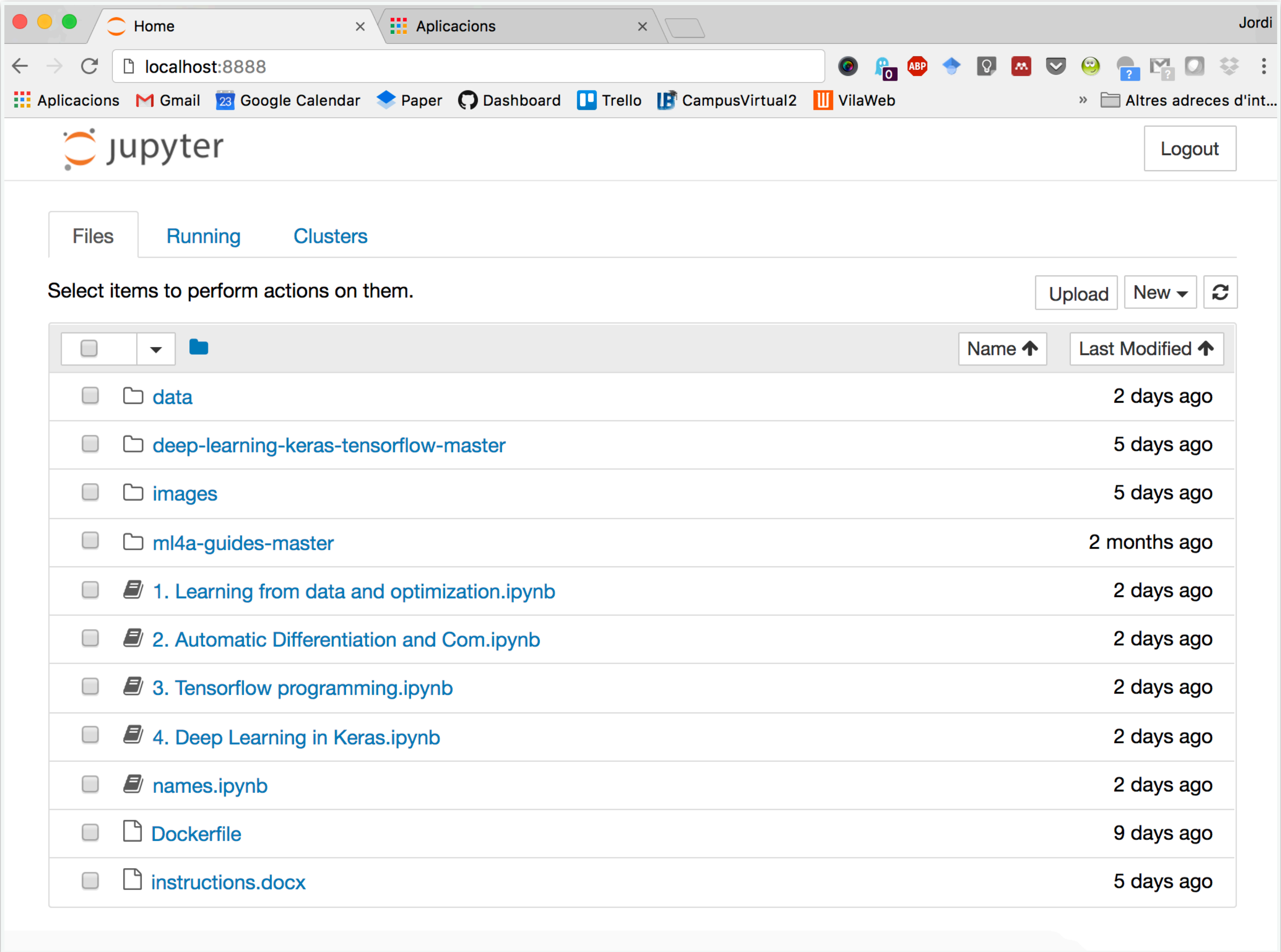
and go with your default browser to

localhost:8888

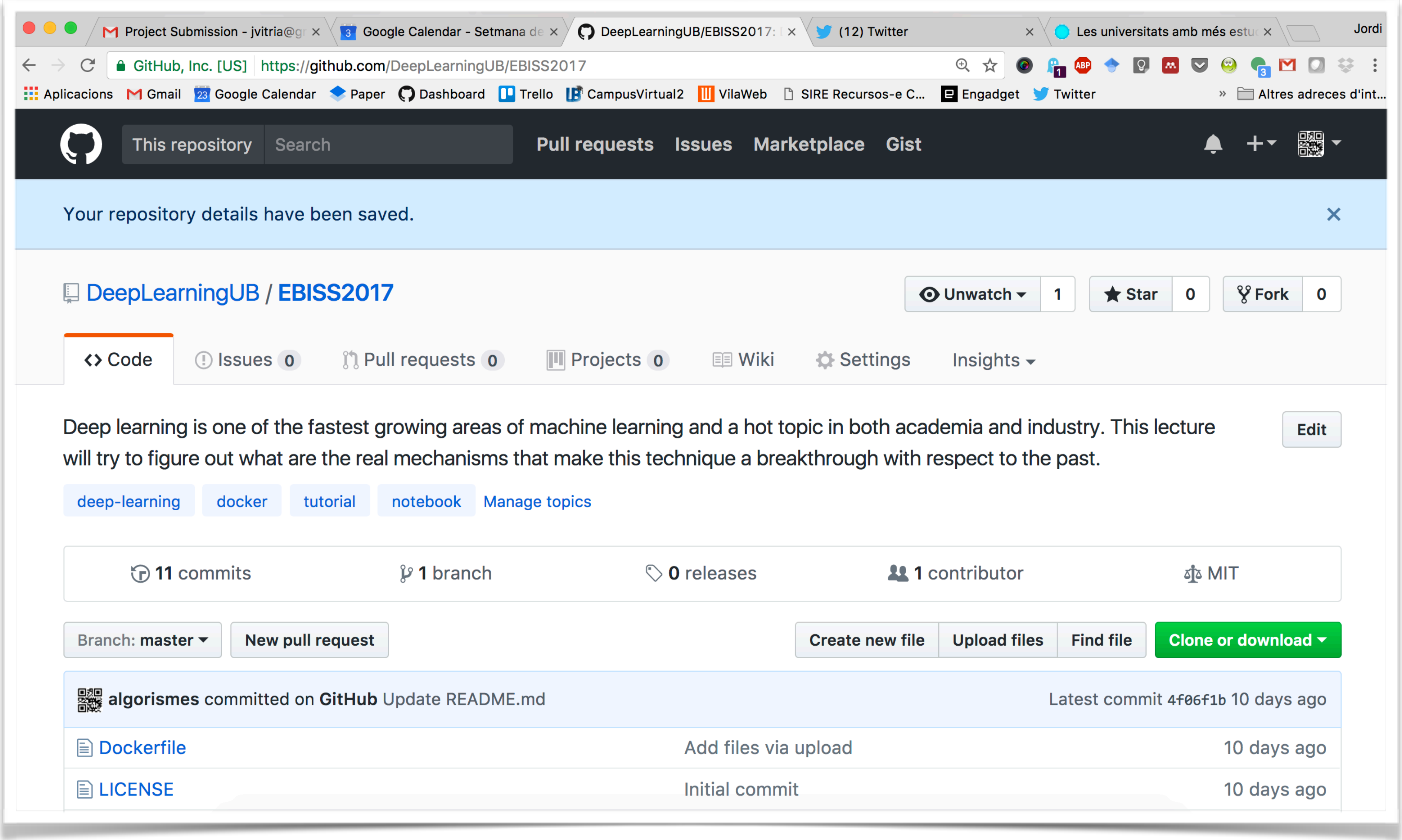
The fist time you connect you will get this message:

```
Copy/paste this URL into your browser when you connect for the first time, to login with a token:
    http://localhost:8888/?token=defbc4266e1de04bde6055ed0c0832c6e803c0efdbf74960
```

We can start to code!



https://github.com/DeepLearningUB/EBISS2017



`git clone https://github.com/DeepLearningUB/EBISS2017`