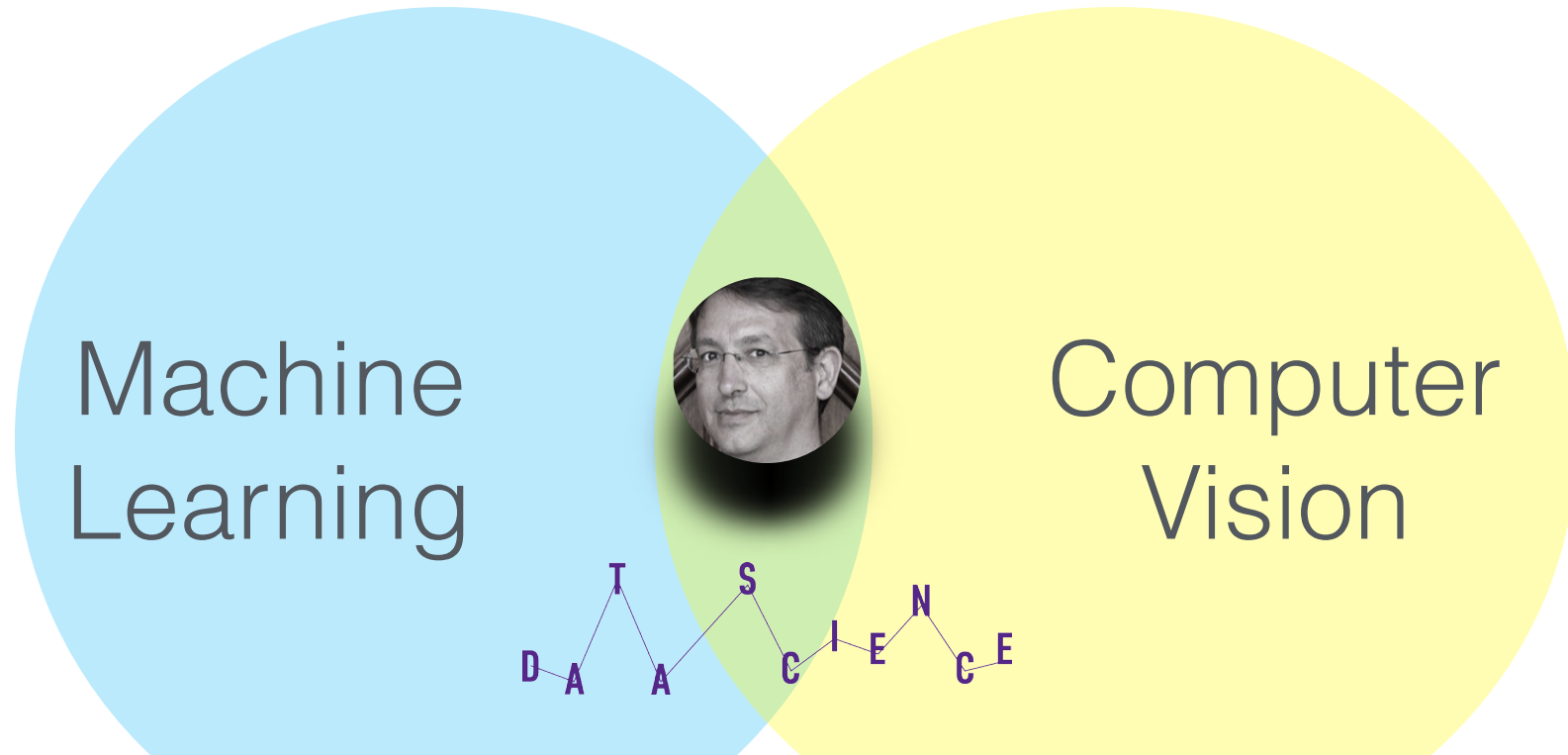


**LET'S OPEN THE
DEEP LEARNING
BLACK BOX!**

JORDI VITRIÀ

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 Autònoma 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.

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

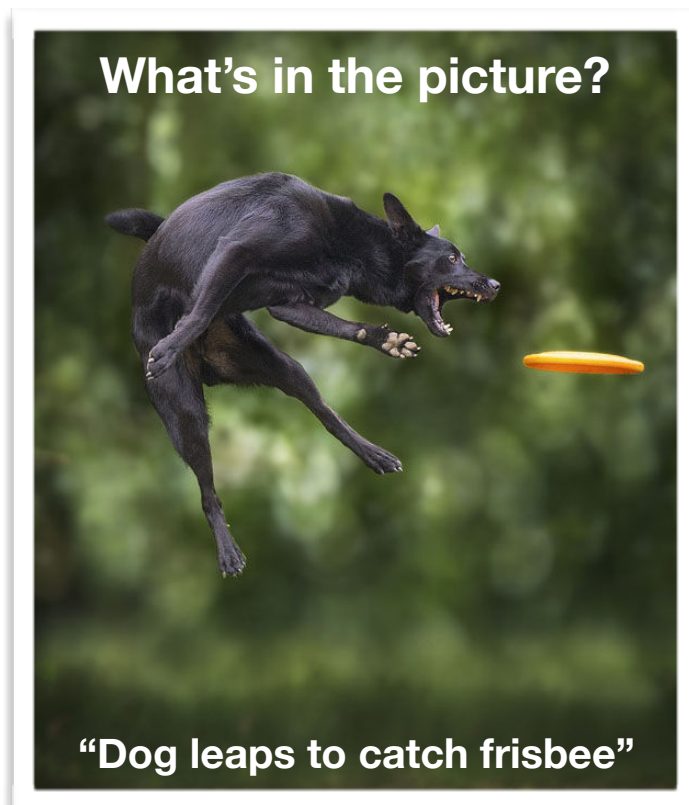
end-to-end learning

deep neural networks

“black box” learning...

Extracting non visual attributes from images using CNN.

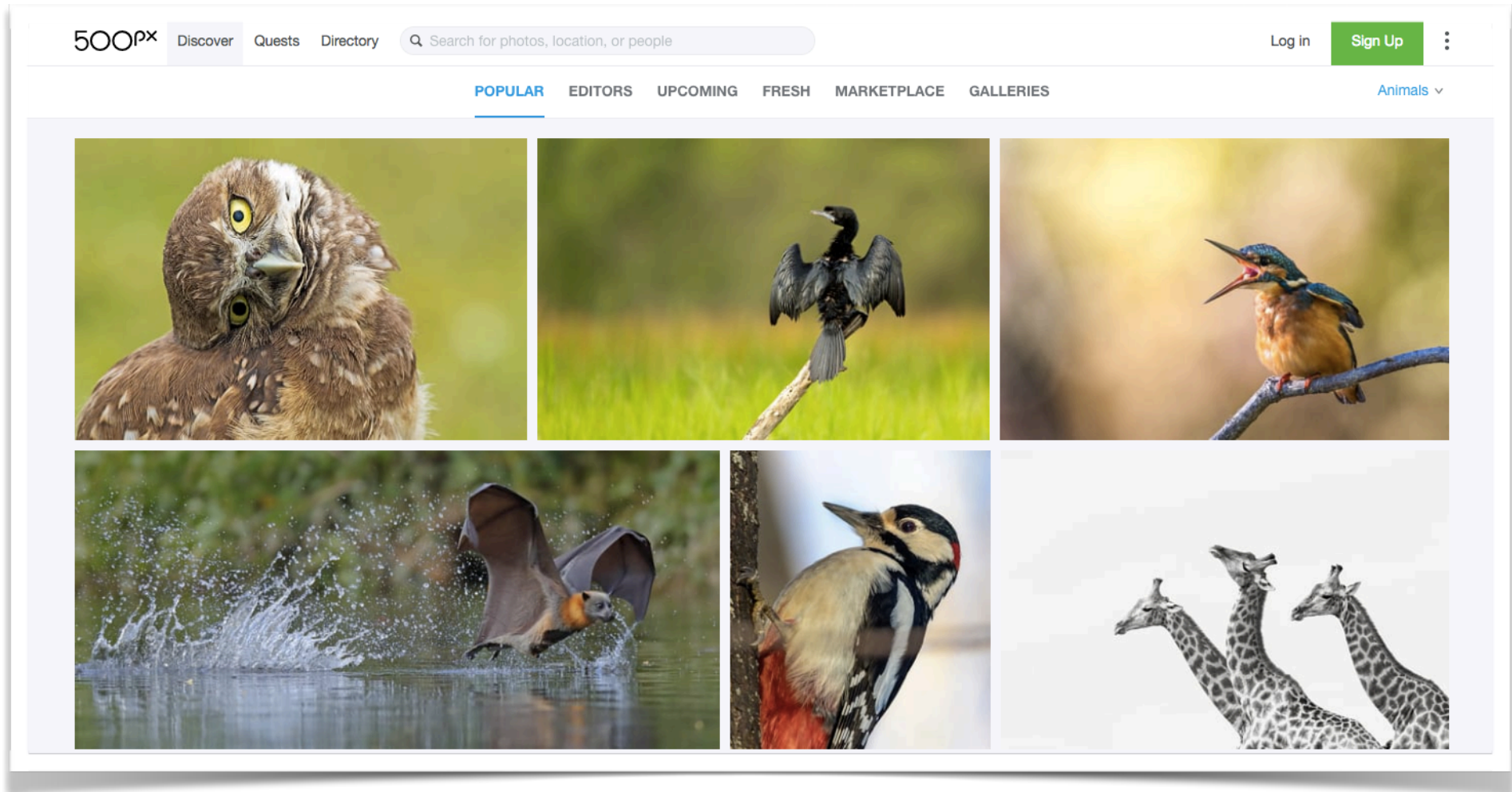
Non-visual attributes are those attributes of an image that can be inferred from visual information but do not have a clear correspondence on the image.





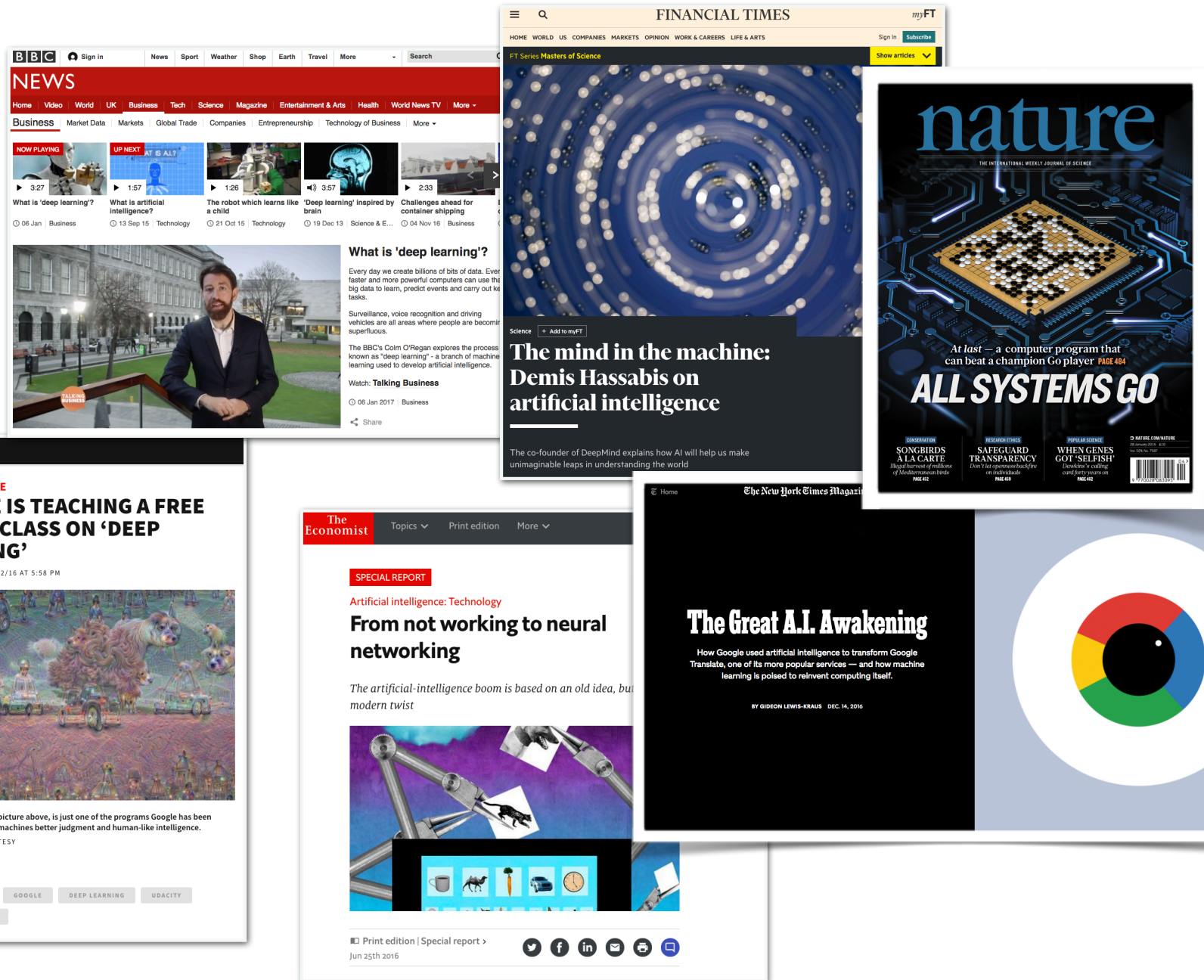
Which apartment is, a priori, more successful on Airbnb?

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...

THE REVENANT

INSPIRED BY TRUE EVENTS

JANUARY 8

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.

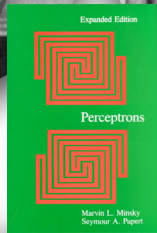


- In 1957 **Frank Rosenblatt** attempted to build a kind of mechanical brain called the **Perceptron**, which was billed as “a machine which senses, recognizes, remembers, and responds like the human mind”.





- 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: $\text{Weight Change} = (\text{Pre-Weight line value}) * (\text{Error} / (\text{Number of Inputs}))$. 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.



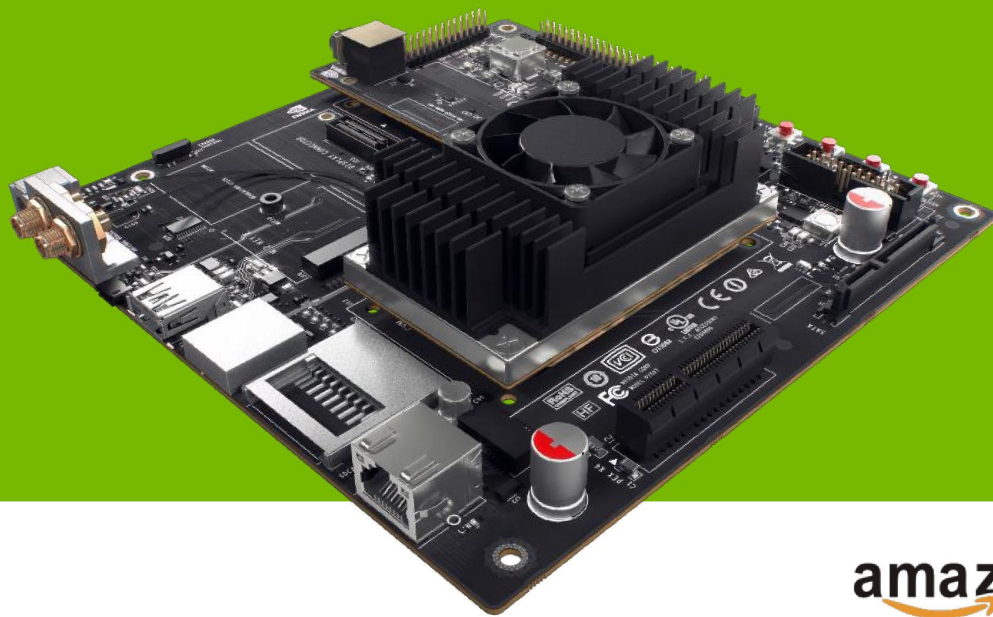
- From 1986 to mid 90's new developments arised: convolucional neural networks (**Y.LeCun**), unsupervised learning (**Y.Bengio**), RBM (**G.Hinton**), etc. But, by this point **new machine learning methods** had begun to also emerge, and people were again beginning to be skeptical of neural nets since 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

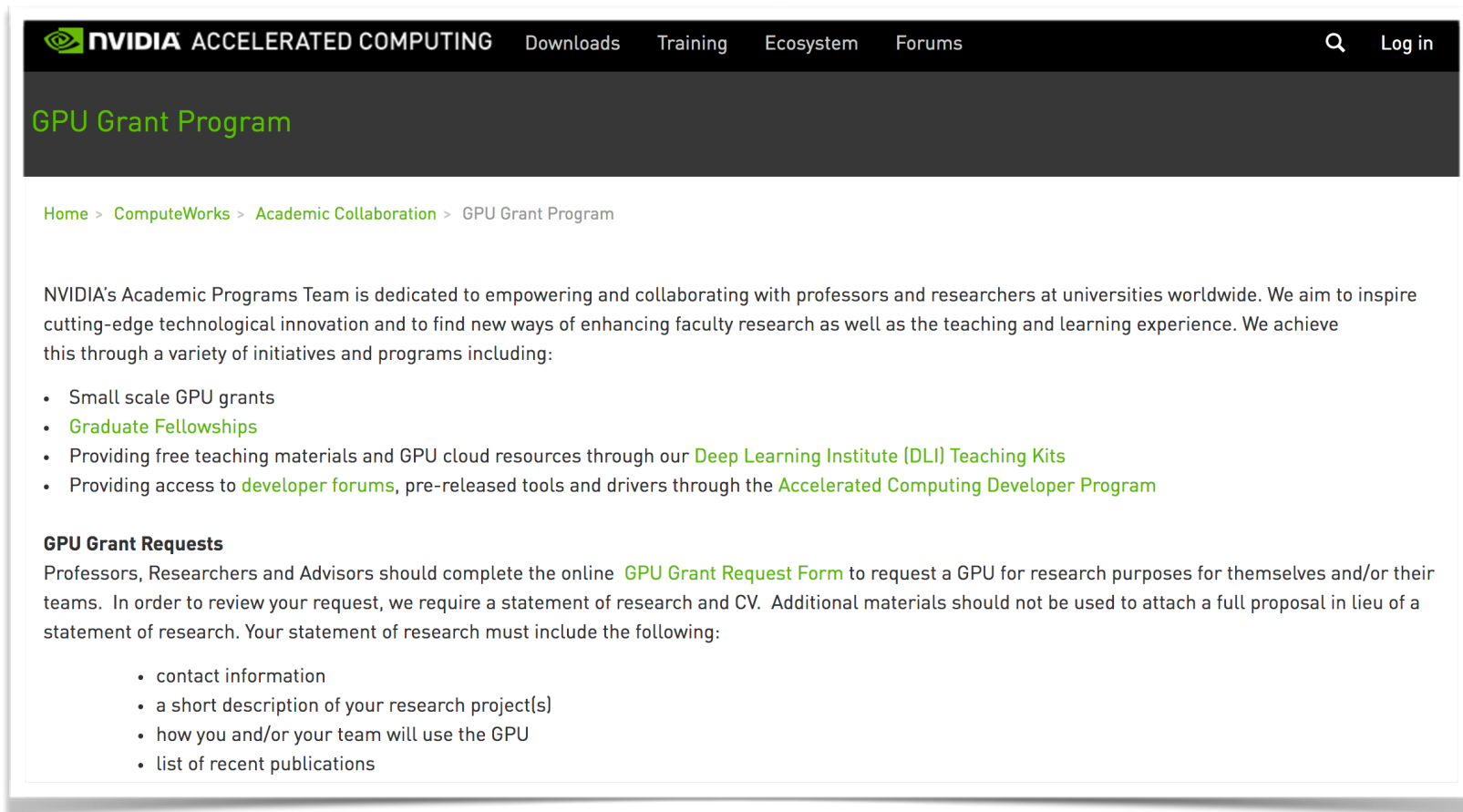
Pre-order Nov 12

Shipping Nov 16 (US)

Intl to follow



GPU democratization



The screenshot shows the NVIDIA Accelerated Computing website. The top navigation bar includes links for Downloads, Training, Ecosystem, and Forums, along with a search icon and a Log in button. The main heading is "GPU Grant Program". Below this, a breadcrumb trail reads: Home > ComputeWorks > Academic Collaboration > GPU Grant Program. The main text states: "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:" followed by a bulleted list: Small scale GPU grants, Graduate Fellowships, Providing free teaching materials and GPU cloud resources through our Deep Learning Institute (DLI) Teaching Kits, and Providing access to developer forums, pre-released tools and drivers through the Accelerated Computing Developer Program. A section titled "GPU Grant Requests" follows, stating that professors, researchers, and advisors should complete the online GPU Grant Request Form to request a GPU for research purposes. It also notes that a statement of research and CV is required, and that a full proposal should be attached in lieu of a statement of research. The required information for the statement of research is listed in a bulleted format: contact information, a short description of the research project(s), how the GPU will be used, and a list of recent publications.

Thank you NVIDIA!

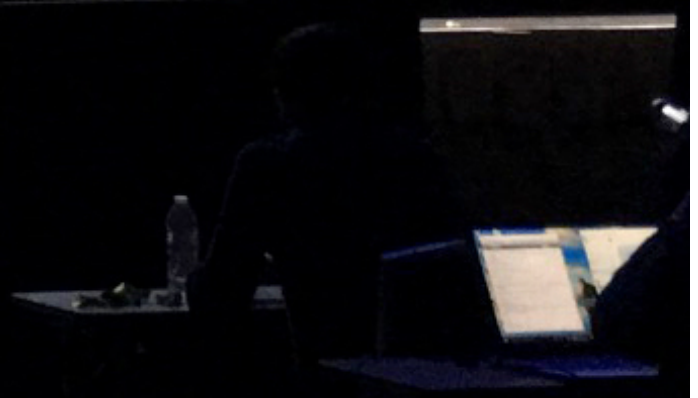
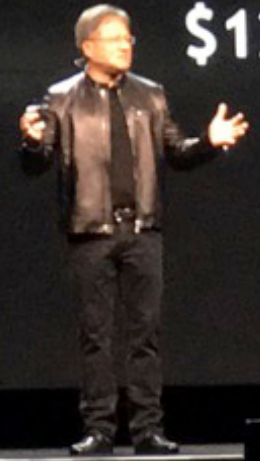


NVIDIA DGX-1

WORLD'S FIRST
DEEP LEARNING SUPERCOMPUTER

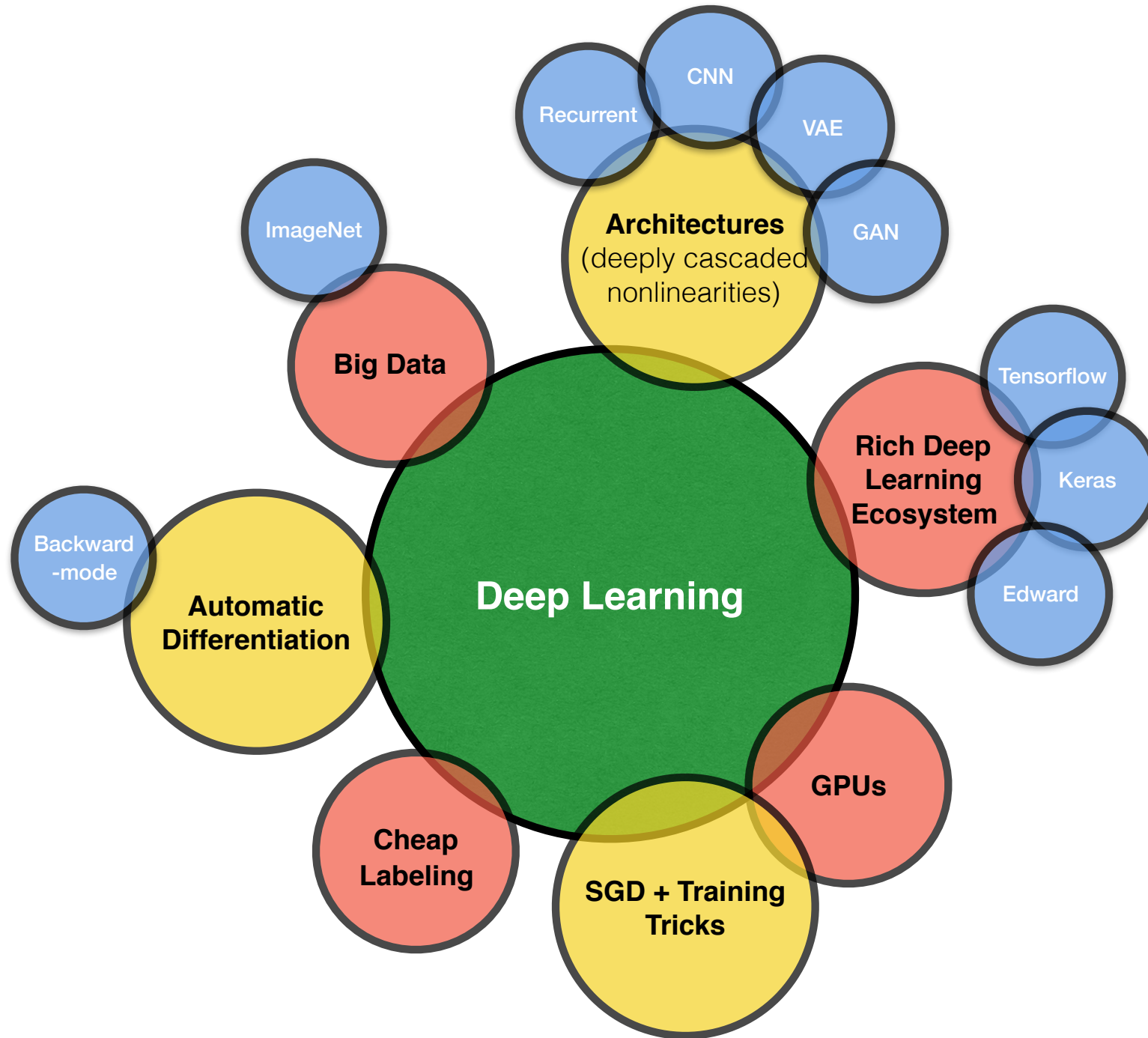
170TF | "250 servers in-a-box" | nvidia.com/dgx1

\$129,000



Definitions

- **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.



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.

The image displays two overlapping Jupyter Notebook windows. The foreground window, titled "Lorenz Differential Equations (autosaved)", shows a notebook with the following content:

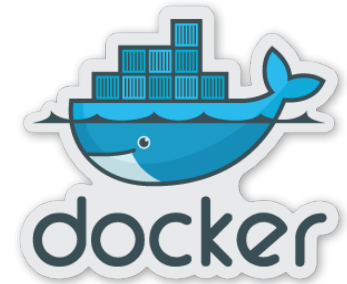
- Title:** Exploring the Lorenz System
- Text:** In this Notebook we explore the [Lorenz system](#) of differential equations:
- Equations:**
$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$
- Text:** This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ , β , ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.
- Code Cell:**

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
               sigma=(0.0,50.0), beta=(0.,5), rho=(0.0,50.0));
```
- Interactive Plot:** A plot of the Lorenz attractor with sliders for parameters: angle (308.2), max_time (12), σ (10), β (2.6), and ρ (28).
- Visualization:** A 3D plot of the Lorenz attractor, showing its characteristic butterfly shape with multiple colored trajectories.

The background window shows a "Welcome to the Jupyter Notebook Server" page with a warning message: "WARNING: Don't rely on this server. Your server is hosted that..."

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.

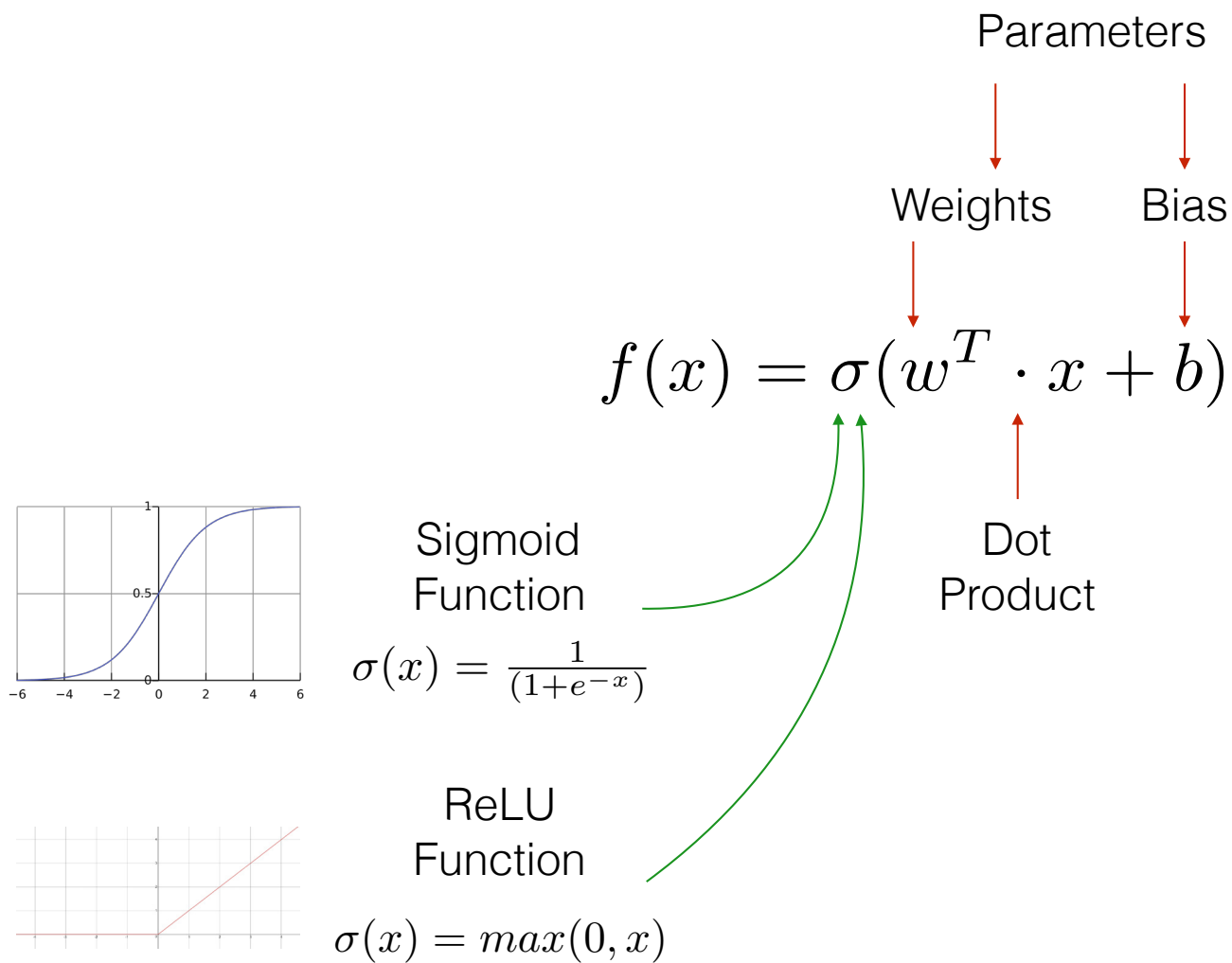
The problem: **machine learning**

Training data: a set of $(x^{(m)}, y^{(m)})$ pairs.

Learn a function $f_w : x \rightarrow y$ to predict on new inputs x .

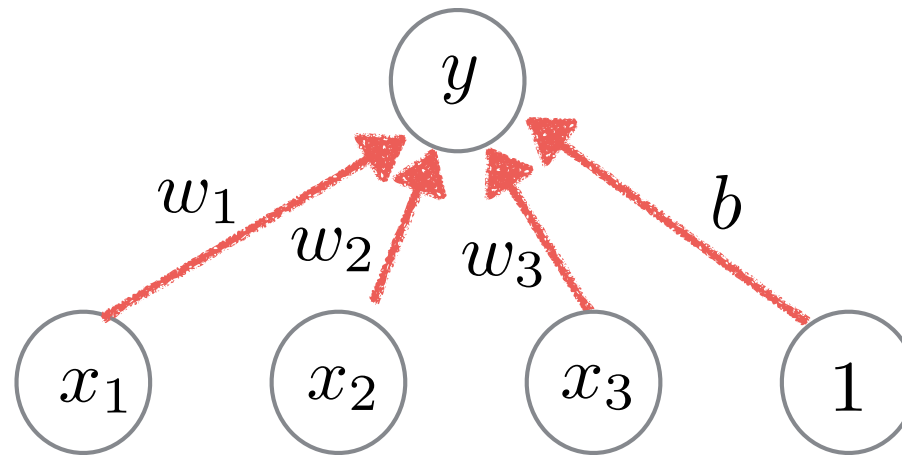
1. Choose a model function family f_w .
2. Optimize parameters w .

1-layer neural net model



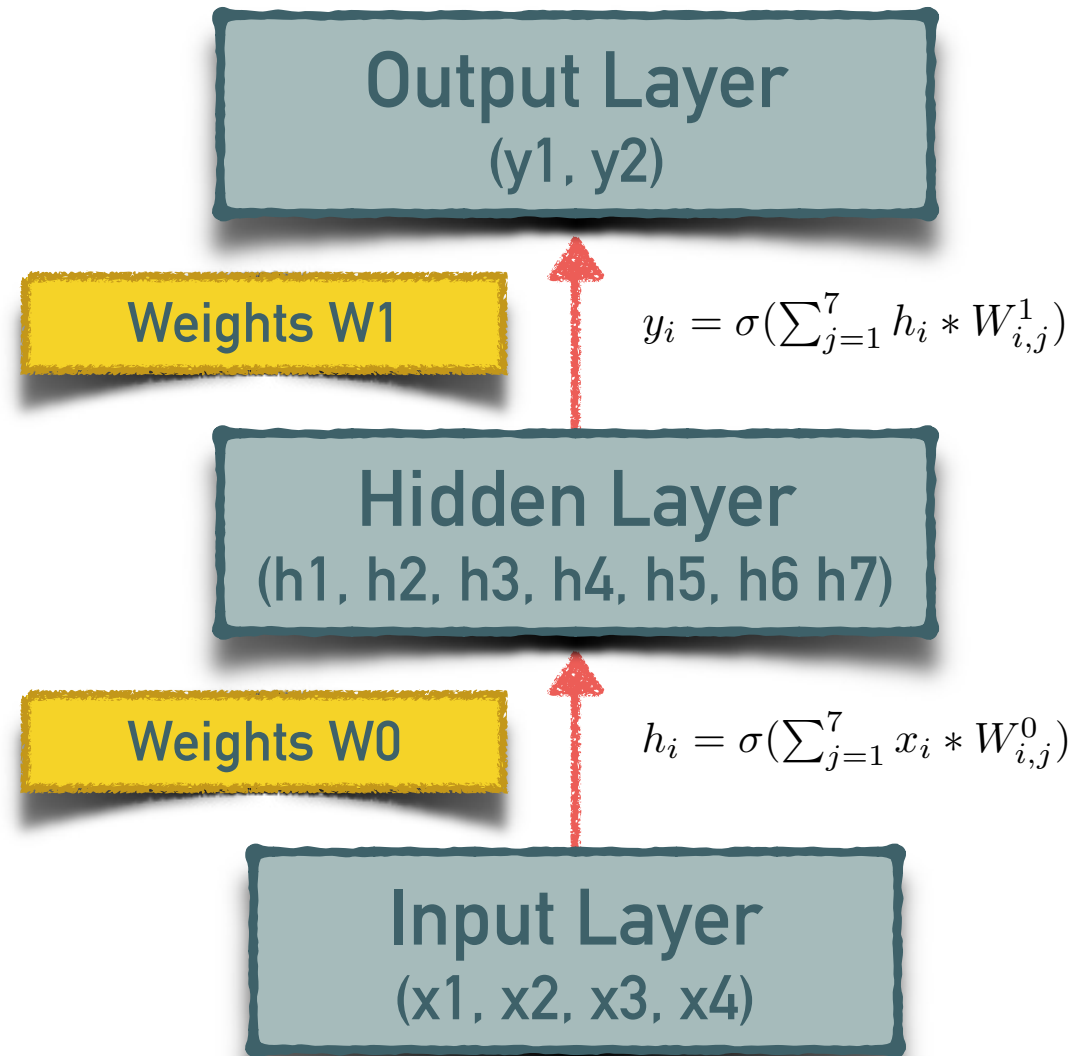
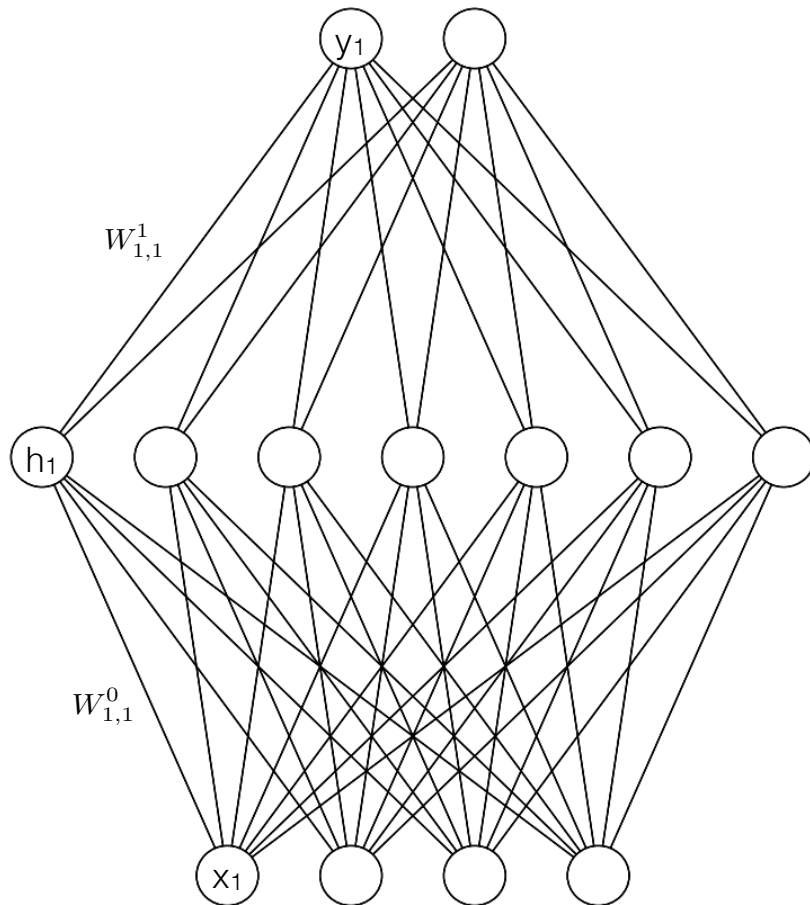
1-layer neural net model

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

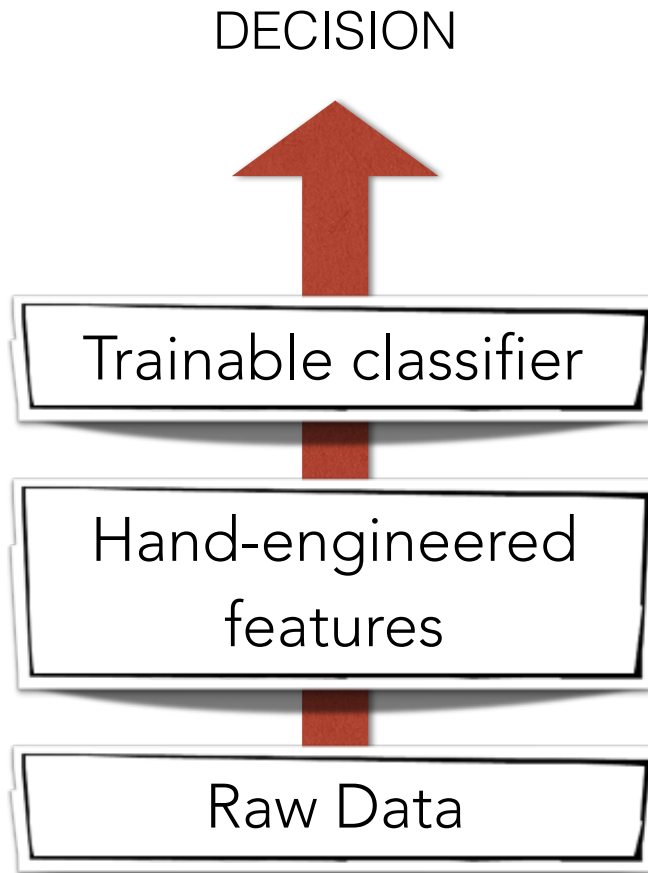


Graphical Representation

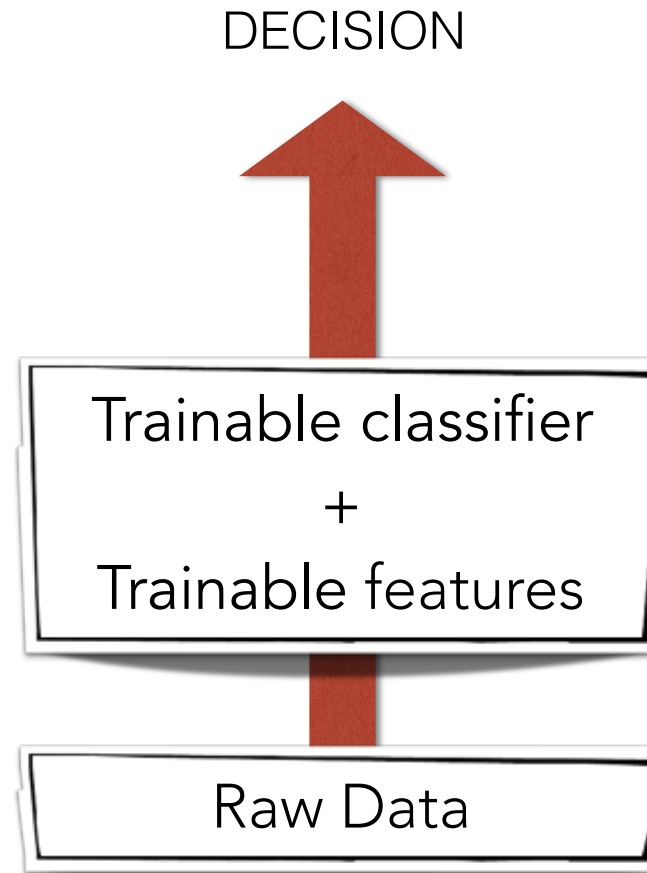
2-layer neural net model



What's new?



STANDARD MACHINE
LEARNING



DEEP LEARNING

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

```
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))

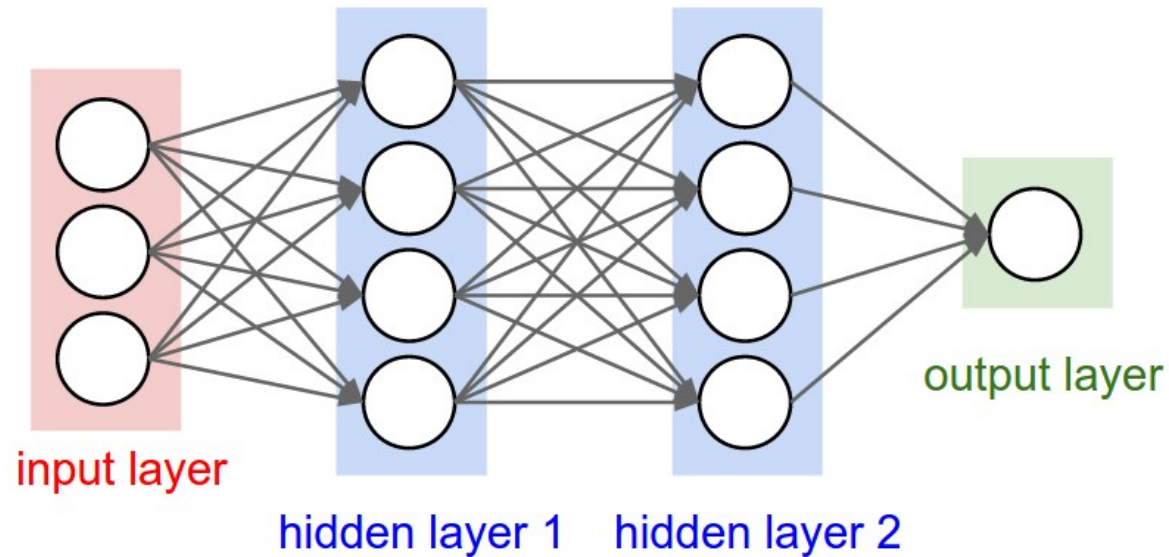
# Build a toy dataset.
inputs = np.array([[0.52, 1.12, 0.77],
                  [0.88, -1.08, 0.15],
                  [0.52, 0.06, -1.30],
                  [0.74, -2.49, 1.39]])
targets = np.array([True, True, False, True])

# 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

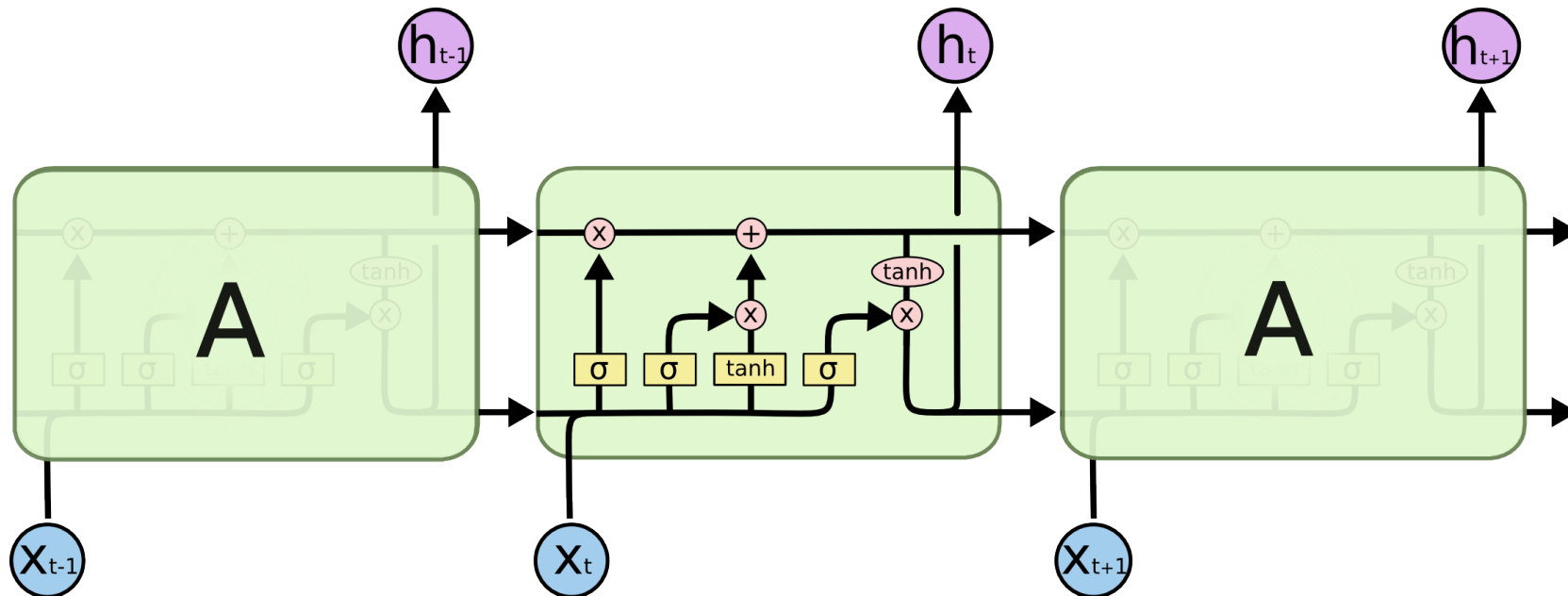
print "Trained loss:", training_loss(weights)
```

Architectures

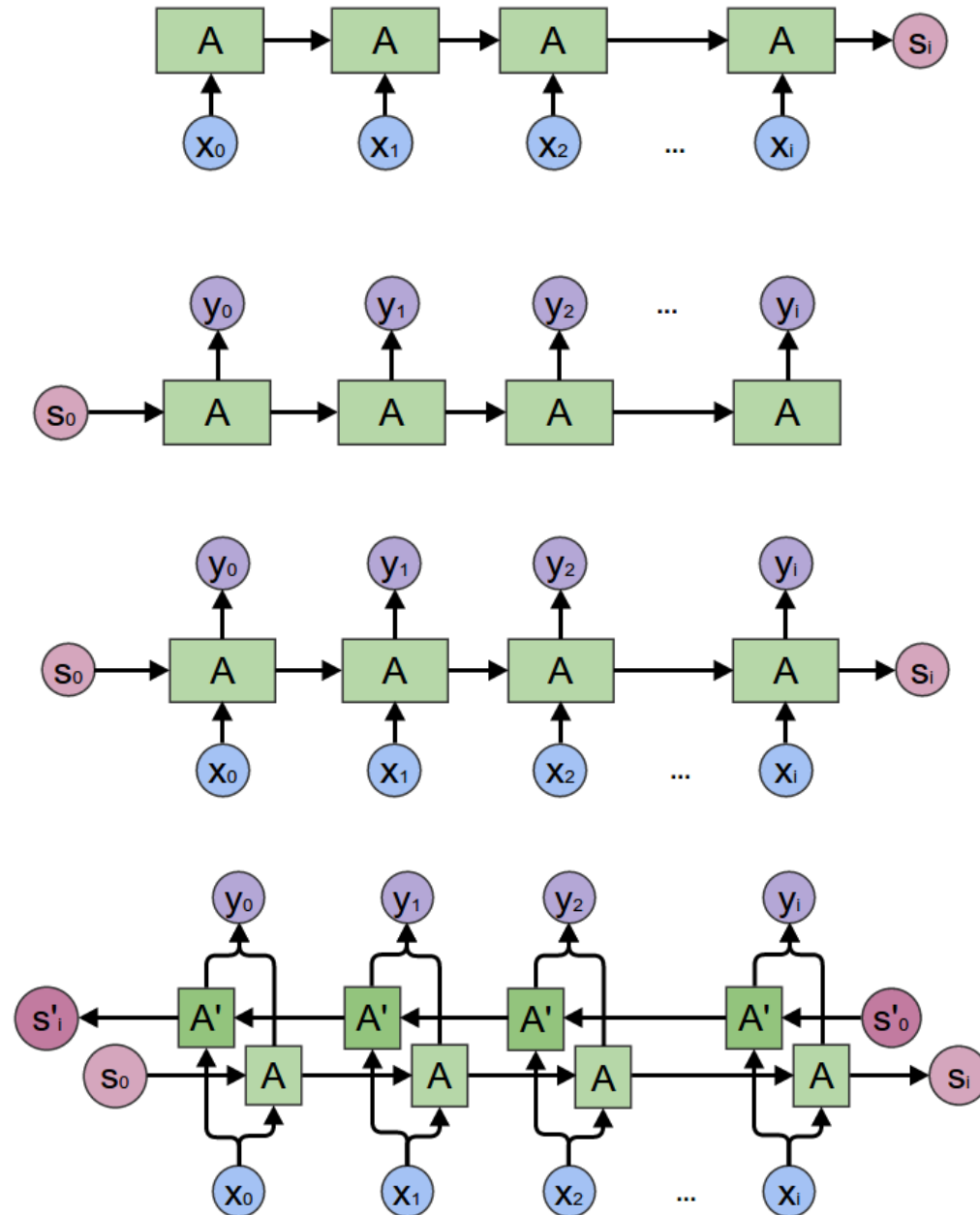


Each layer is a function, acting on the output of a previous layer. As a whole, the network is a chain of composed functions. This chain of composed functions is optimized to perform a task.

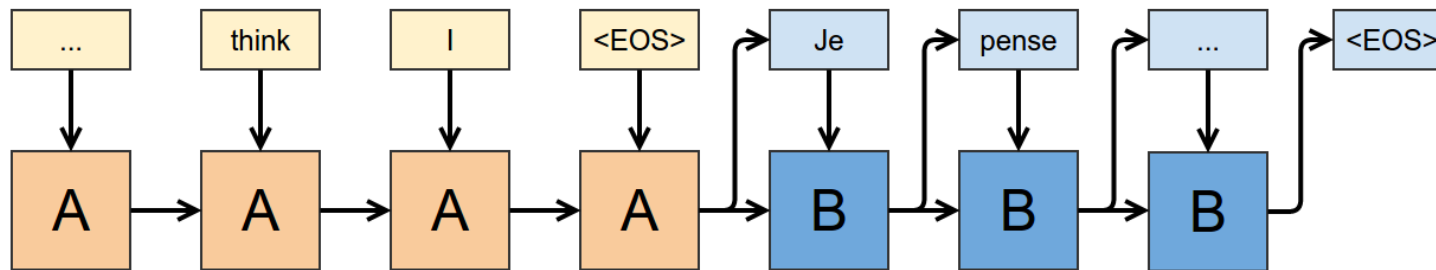
Recurrent neural layer model



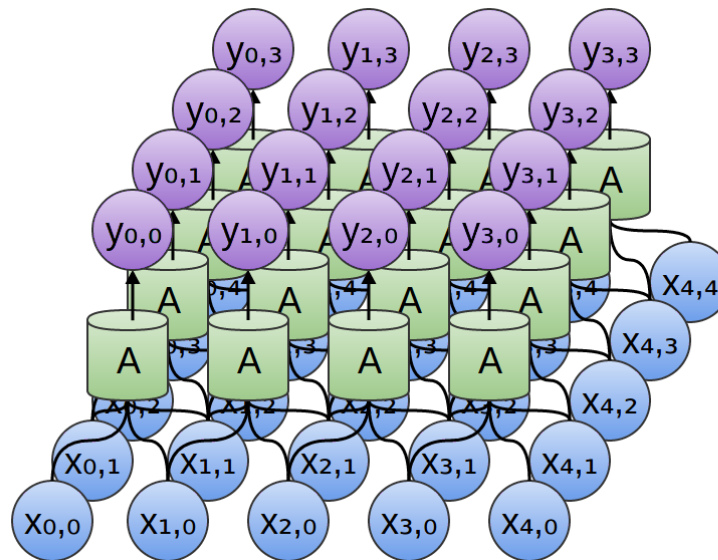
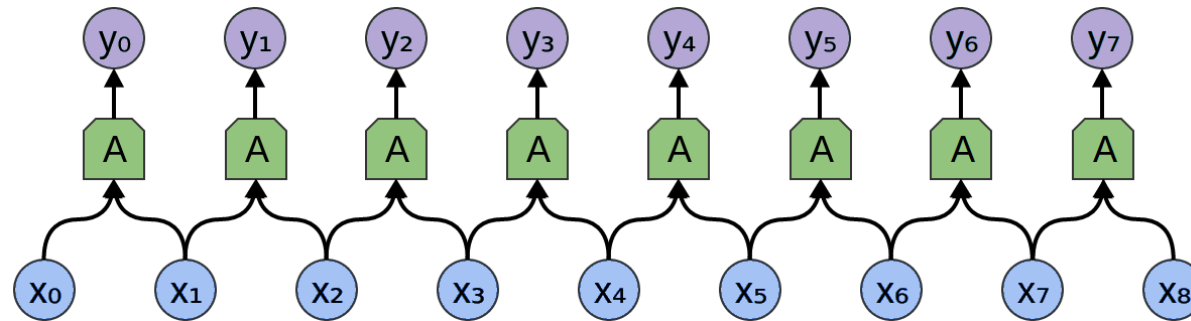
Recurrent neural layer model



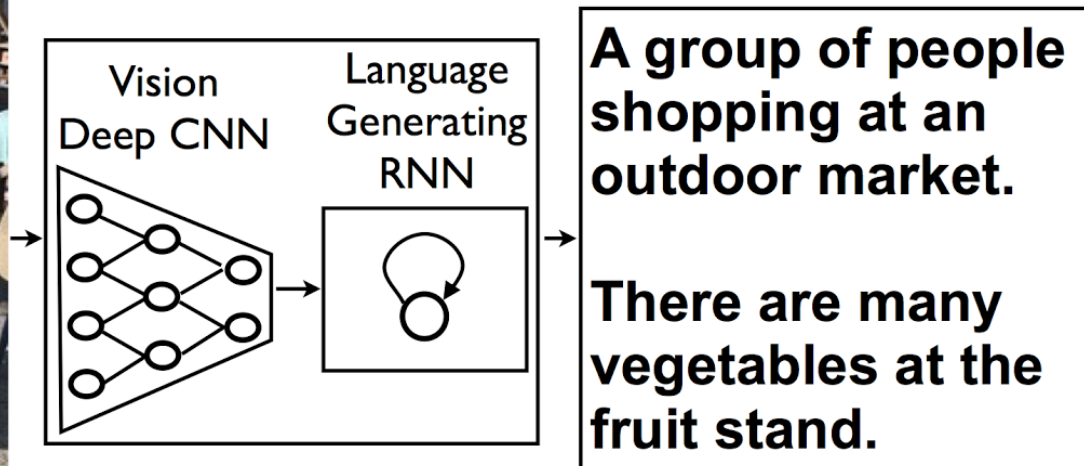
Recurrent neural layer model



Convolutional neural layer model



CNN + RNN



Deep Learning Ecosystem

Edward



- Getting Started
- Tutorials
- API
- Community
- Contributing



A library for probabilistic modeling, inference, and criticism.

Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.

It supports **modeling** with

- Directed graphical models
- Neural networks (via libraries such as [Keras](#) and [TensorFlow Slim](#))
- Implicit generative models
- Bayesian nonparametrics and probabilistic programs

It supports **inference** with

- Variational inference
 - Black box variational inference
 - Stochastic variational inference
 - Generative adversarial networks
 - Maximum a posteriori estimation
- Monte Carlo
 - Gibbs
 - Hamiltonian
 - Stochastic
- Compositional
 - Experimental
 - Pseudo
 - Message passing

TensorFlow™ Install Develop API r1.2 Deploy Extend Community Ver Search

An open-source software library for Machine Intelligence

GET STARTED

TensorFlow 1.2 has arrived!

We're excited to announce the release of TensorFlow 1.2! Check out the release notes for all the latest.

UPGRADE NOW

Introducing TensorFlow Research Cloud

We're making 1,000 Cloud TPUs available for free to accelerate open machine learning research.

LEARN MORE

The 2017 TensorFlow Dev Summit

Thousands of people from the TensorFlow community participated in the first flagship event. Watch the keynote and talks.

WATCH VIDEOS

Keras Documentation

Search docs

- Home
- Keras: The Python Deep Learning library
- You have just found Keras.
- Guiding principles
- Getting started: 30 seconds to Keras
- Installation
- Switching from TensorFlow to CNTK or Theano
- Support
- Why this name, Keras?
- Getting started
- Guide to the Sequential model
- Guide to the Functional API
- FAQ

Docs » Home

Edit on GitHub

Keras: The Python Deep Learning library

You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of either [TensorFlow](#), [CNTK](#) or [Theano](#). It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

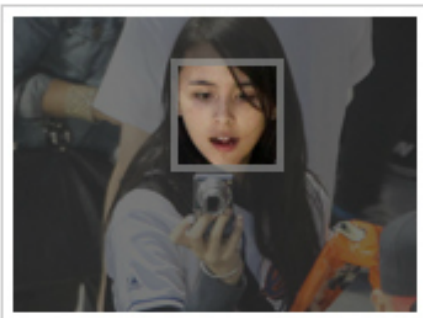
Read the documentation at [Keras.io](#).

Keras is compatible with: **Python 2.7-3.5**.

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



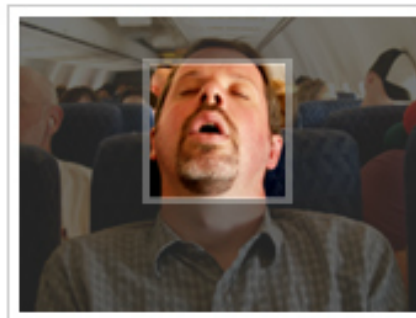
Face recognition.



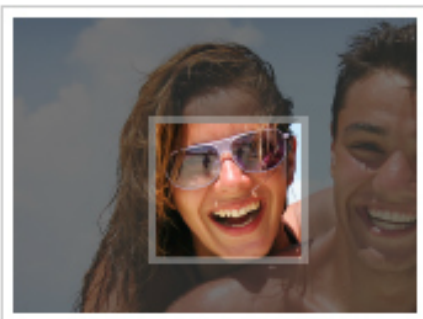
Who is this?



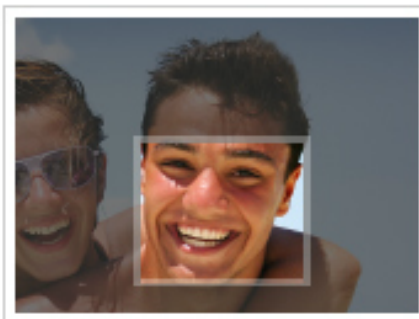
Who is this?



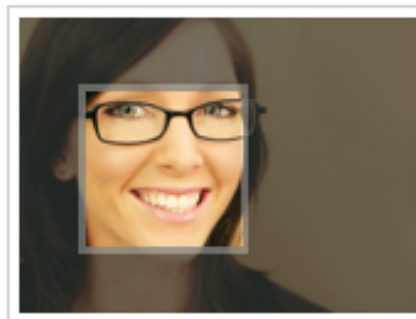
Who is this?



Who is this?



Who is this?



Who is this?

DeepFace (Facebook): Accuracy of 97.35%

New applications: navigation and mapping.

dyson

Tienda Aspiradoras Ventiladores y Calefactores Airblade™ Mi cuenta Soporte

Robot Dyson 360 Eye™

Sea el primero en disfrutarlo

El nuevo robot aspirador de Dyson

Vea a James Dyson presentando el nuevo Dyson 360 Eye™ en Tokio

New applications: Image Upscaling (Flipboard)



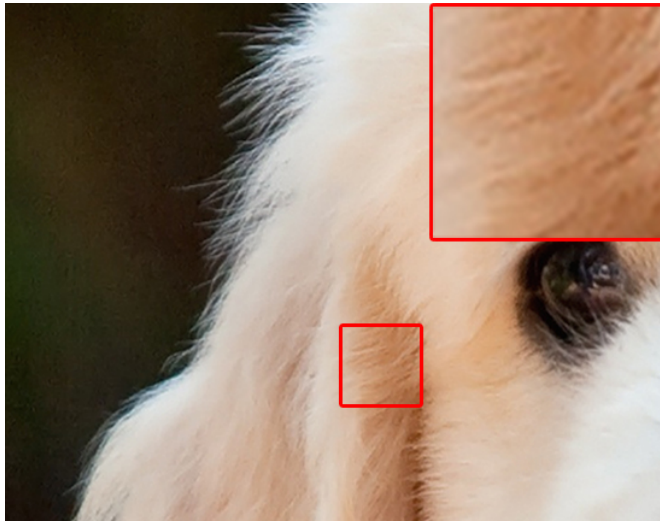
<http://engineering.flipboard.com/2015/05/scaling-convnets/>

New applications: Image Upscaling (Flipboard)

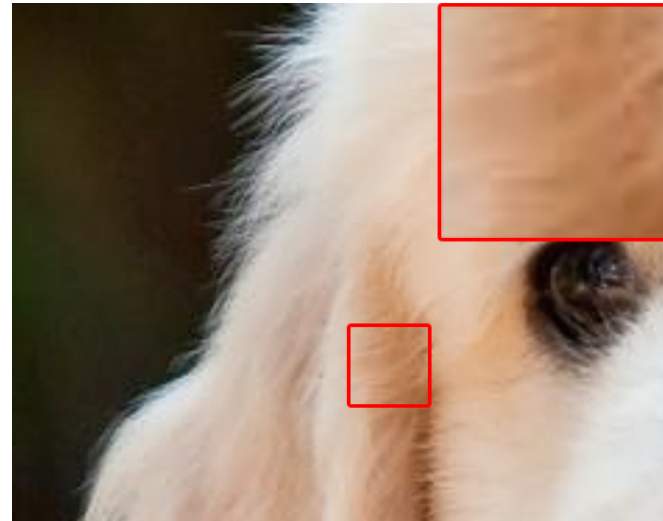


<http://engineering.flipboard.com/2015/05/scaling-convnets/>

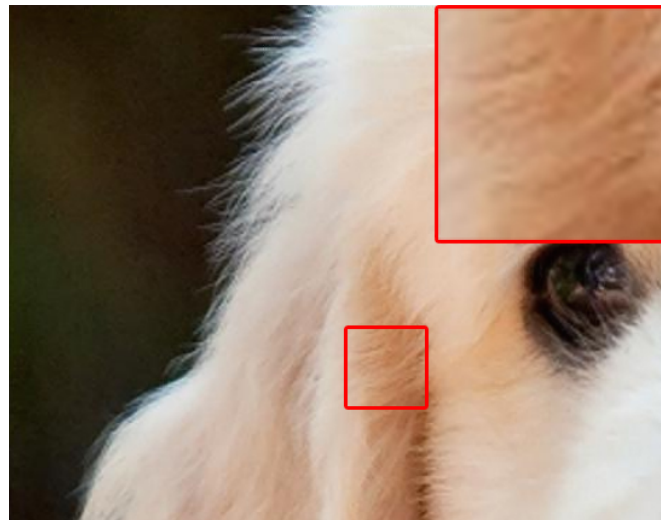
New applications: Image Upscaling (Flipboard)



Original



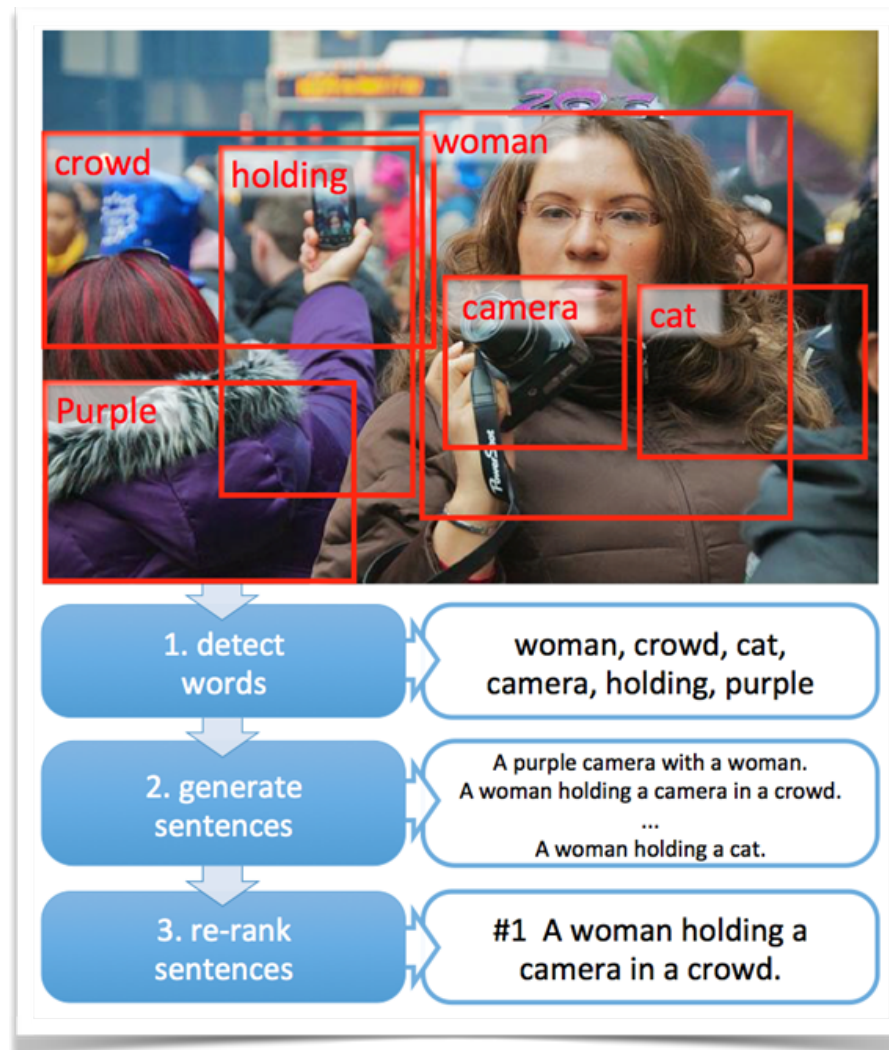
Bicubic



Model

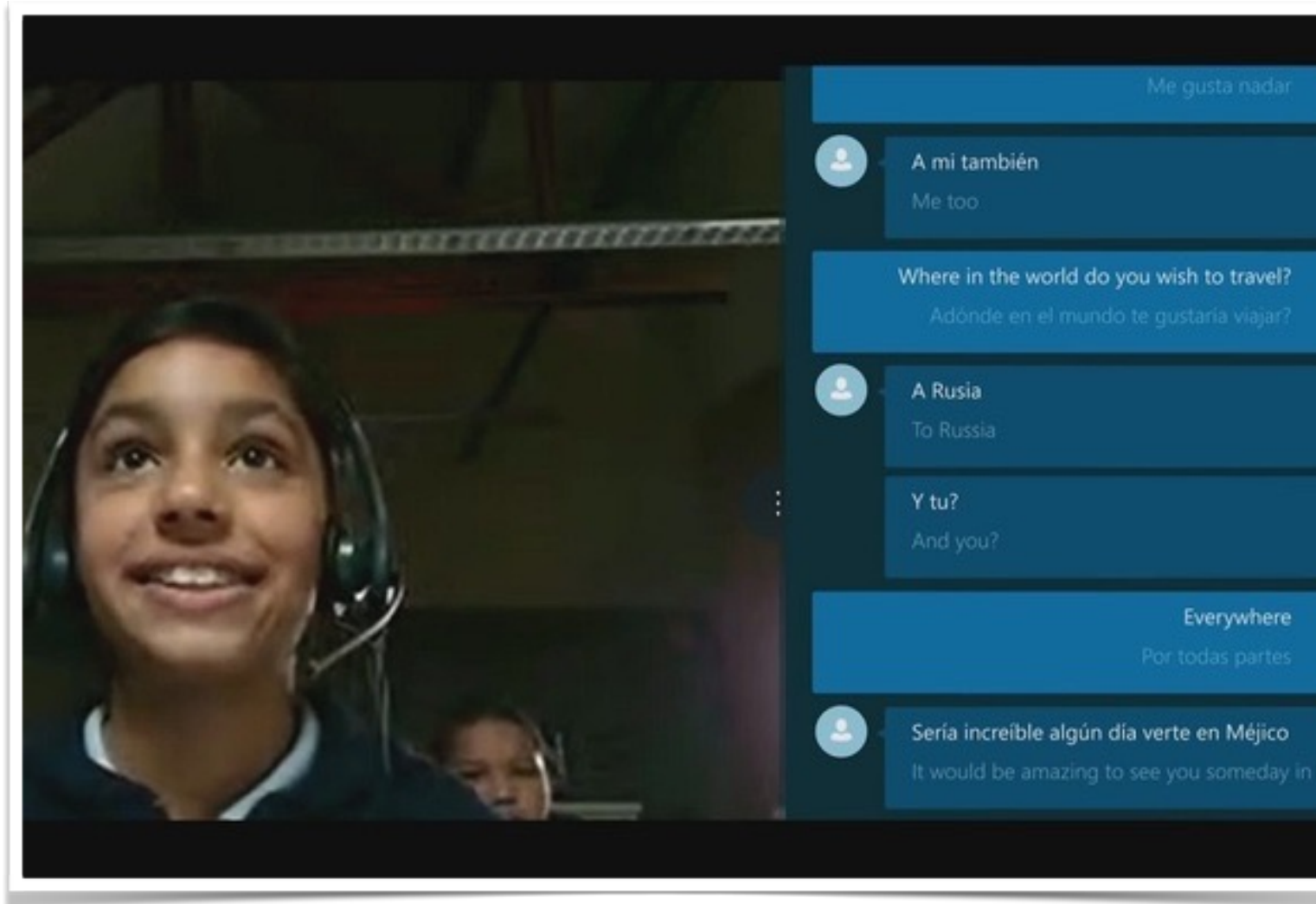
<http://engineering.flipboard.com/2015/05/scaling-convnets/>

New applications: Automatic Image Captioning

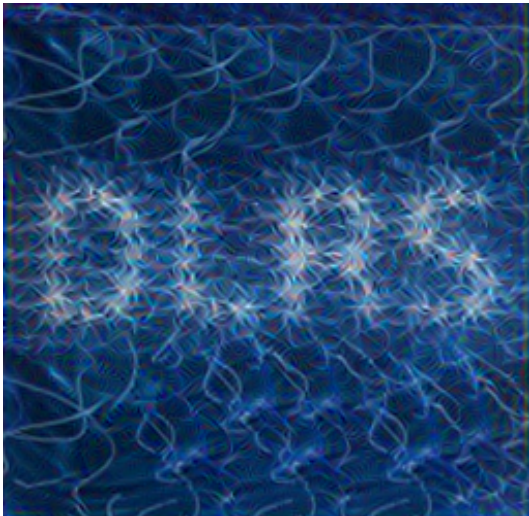


<http://blogs.technet.com/b/machinelearning/archive/2014/11/18/rapid-progress-in-automatic-image-captioning.aspx>

Speech translation



Recommenders



1st Workshop on Deep Learning for Recommender Systems

in conjunction with RecSys 2016
15 September 2016, Boston, USA

Music Generation

The screenshot shows the SoundCloud profile for 'deepjazz'. The profile banner features a black and white photograph of a jazz band. The profile name is 'deepjazz', with a bio that reads 'I'm an AI built to make Jazz' and 'Princeton, United States'. The profile has 104 followers, is following 1 user, and has 6 tracks. A featured track is 'deepjazz on Metheny', which is 14 days old and categorized as '# Electronic'. The track has a waveform and a duration of 0:33. Below the track, there is a list of three tracks, each with a play button and a number of plays: 1 track with 6,142 plays, 2 tracks with 3,452 plays, and 3 tracks with 1,908 plays. The profile also includes links to 'my source code (GitHub)' and 'deepjazz.io', and a '1 following' section.

SOUNDCLLOUD Charts Search for artists, bands, tracks, podcasts Sign in or Create account Upload

deepjazz
I'm an AI built to make Jazz
Princeton, United States

All Tracks Playlists Reposts Follow Share

deepjazz 6 tracks

deepjazz on Metheny 14 days # Electronic

0:33

dj	1 deepjazz On Metheny ... 1 Epoch	▶ 6,142
dj	2 deepjazz On Metheny ... 16 Epochs	▶ 3,452
dj	3 deepjazz On Metheny ... 32 Epochs	▶ 1,908

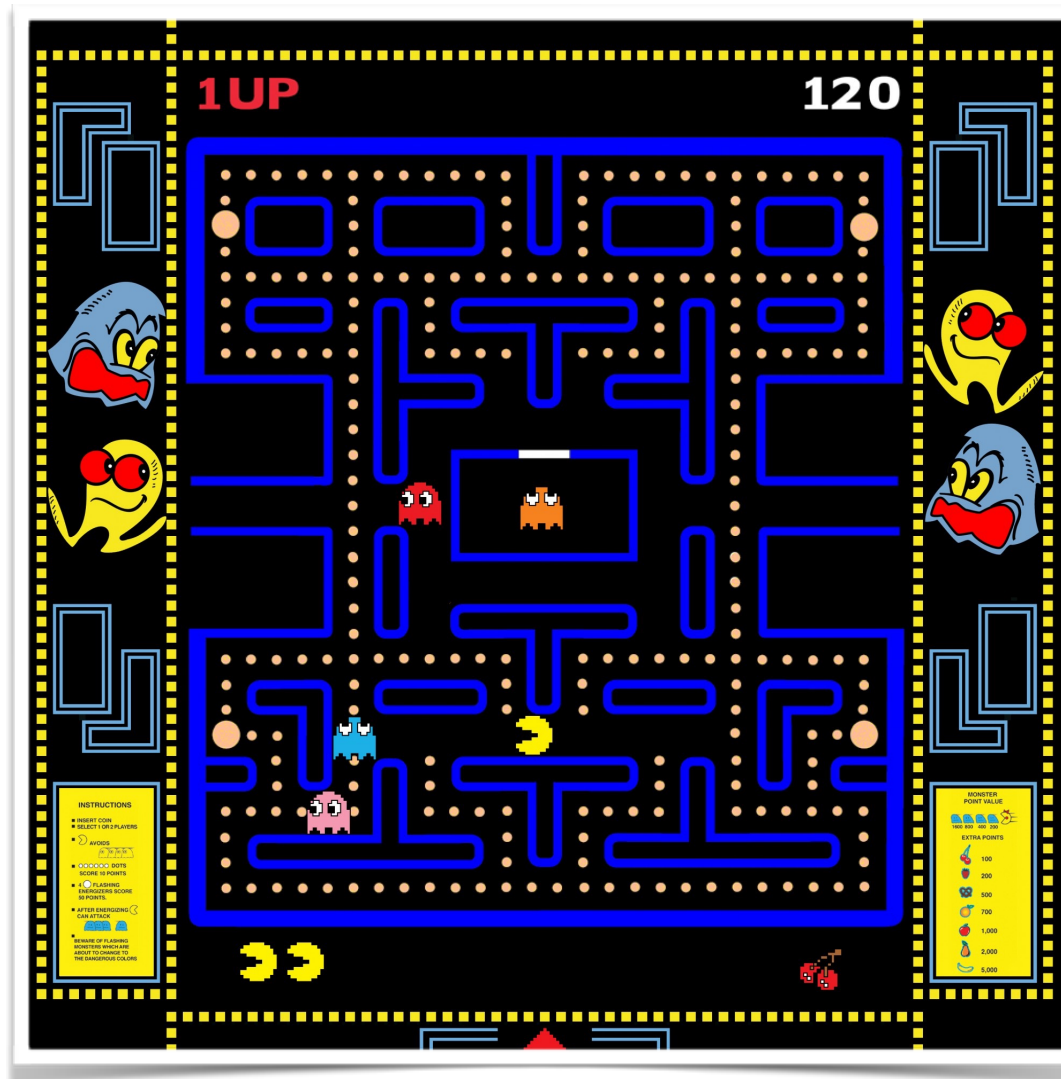
Followers 104 Following 1 Tracks 6

Hi! I'm deepjazz, an AI built by Ji-Sung Kim. You can check out my source code on GitHub or visit my website, deepjazz.io

my source code (GitHub) | deepjazz.io

1 following View all

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



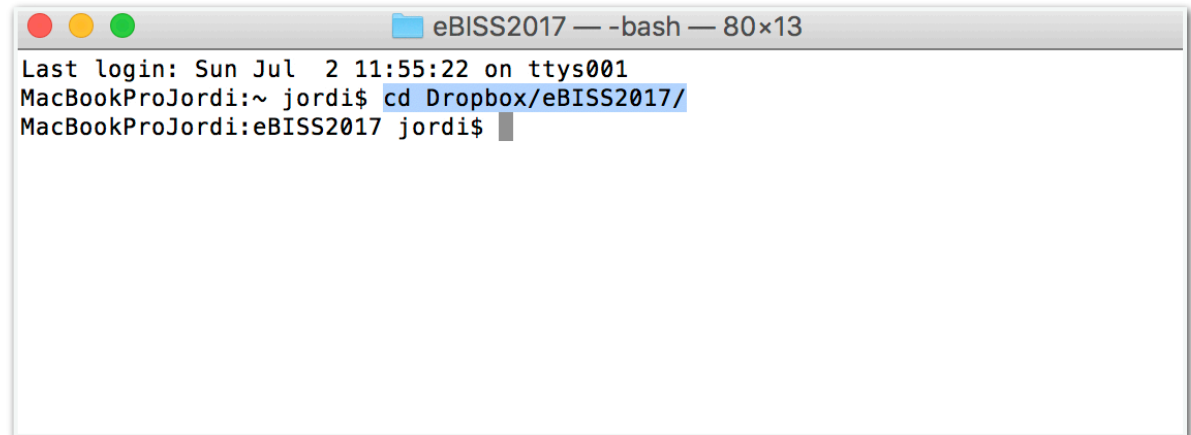
Hands On!

Open a terminal window

A screenshot of a macOS terminal window. The title bar shows a home icon, the name 'jordi', and the shell '-bash' with a window size of '80x13'. The terminal content shows the last login time as 'Sun Jul 2 11:55:22 on ttys001' and the current prompt as 'MacBookProJordi:~ jordi\$' with a cursor.

```
jordi — -bash — 80x13
Last login: Sun Jul 2 11:55:22 on ttys001
MacBookProJordi:~ jordi$
```

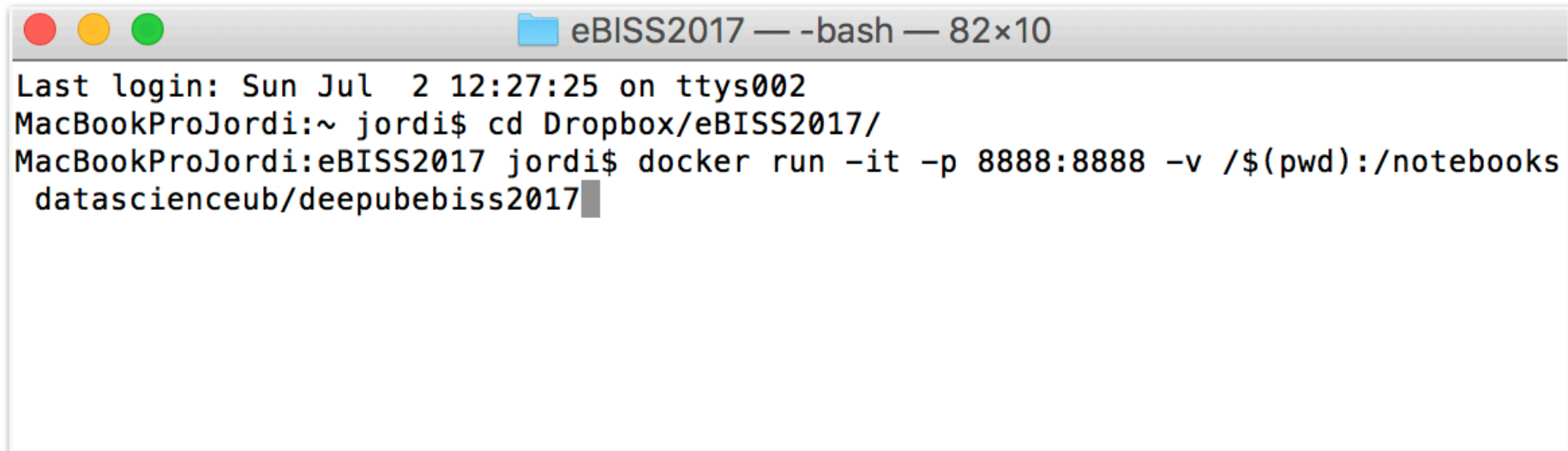
Go to the working directory of your choice

A screenshot of a macOS terminal window. The title bar shows a folder icon, the name 'eBISS2017', and the shell '-bash' with a window size of '80x13'. The terminal content shows the last login time as 'Sun Jul 2 11:55:22 on ttys001' and the command 'cd Dropbox/eBISS2017/' being executed, with the prompt changing to 'MacBookProJordi:eBISS2017 jordi\$' with a cursor.

```
eBISS2017 — -bash — 80x13
Last login: Sun Jul 2 11:55:22 on ttys001
MacBookProJordi:~ jordi$ cd Dropbox/eBISS2017/
MacBookProJordi:eBISS2017 jordi$
```

Hands On!

Start your docker image

A terminal window titled "eBISS2017 — -bash — 82x10" with three colored window control buttons (red, yellow, green) in the top-left corner. The terminal text shows the user navigating to a directory and running a Docker command to start a container.

```
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 first 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!

The screenshot shows a web browser window with the JupyterLab interface. The browser's address bar displays 'localhost:8888'. The JupyterLab header includes the 'jupyter' logo and a 'Logout' button. Below the header, there are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' with 'Upload', 'New', and refresh icons. The main area is a file browser showing a list of files and folders:

	Name ↑	Last Modified ↑
<input type="checkbox"/>	data	2 days ago
<input type="checkbox"/>	deep-learning-keras-tensorflow-master	5 days ago
<input type="checkbox"/>	images	5 days ago
<input type="checkbox"/>	ml4a-guides-master	2 months ago
<input type="checkbox"/>	1. Learning from data and optimization.ipynb	2 days ago
<input type="checkbox"/>	2. Automatic Differentiation and Com.ipynb	2 days ago
<input type="checkbox"/>	3. Tensorflow programming.ipynb	2 days ago
<input type="checkbox"/>	4. Deep Learning in Keras.ipynb	2 days ago
<input type="checkbox"/>	names.ipynb	2 days ago
<input type="checkbox"/>	Dockerfile	9 days ago
<input type="checkbox"/>	instructions.docx	5 days ago

https://github.com/DeepLearningUB/EBISS2017

The screenshot shows the GitHub repository page for `DeepLearningUB/EBISS2017`. The repository is owned by `DeepLearningUB` and contains 1 commit, 1 branch, and 0 releases. The repository is licensed under MIT and has 1 contributor. The README file is the latest commit, dated 10 days ago. The commit history table shows the following entries:

File	Commit Message	Commit Hash	Time
Dockerfile	Add files via upload	4f06f1b	10 days ago
LICENSE	Initial commit		10 days ago

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