

Deep Learning: Introduction to Deep Learning

```
$ echo "Data Sciences Institute"
```

**Warning: This lecture is more theoretical
compared to the other lectures.**

Goal of the class

Overview

- When and where to use DL
- "How" it works
- Frontiers of DL

Using DL

- Implement using Numpy , and Tensorflow (Keras)
- Engineering knowledge for building and training DL

What is Deep Learning

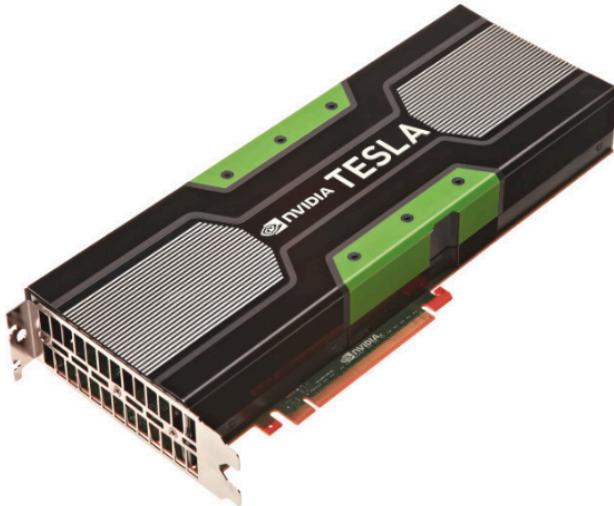
- Good old Neural Networks, with more layers/modules
- Non-linear, hierarchical, abstract representations of data
- Flexible models with any input/output type and size
- Differentiable Functional Programming

Why Deep Learning Now?

- Better algorithms & understanding

Why Deep Learning Now?

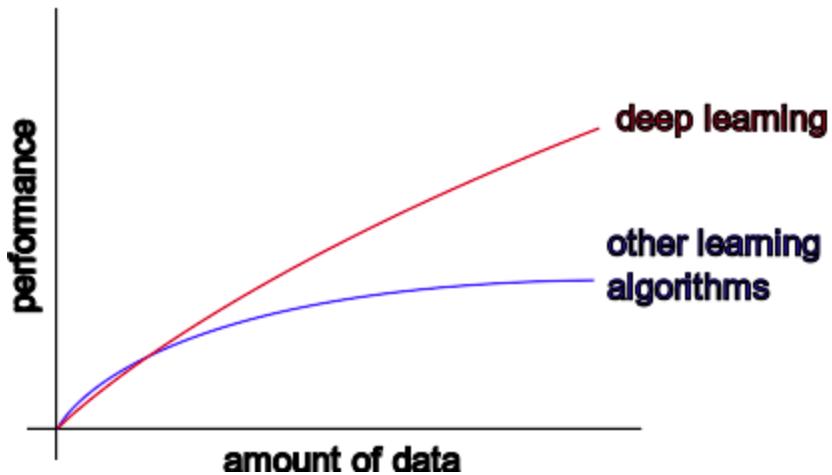
- Computing power (GPUs, TPUs, ...)



GPU and TPU

Why Deep Learning Now?

- Data with labels



Adapted from Andrew Ng

Why Deep Learning Now?

- Open source tools and models



PYTORCH



Microsoft
CNTK

Caffe2

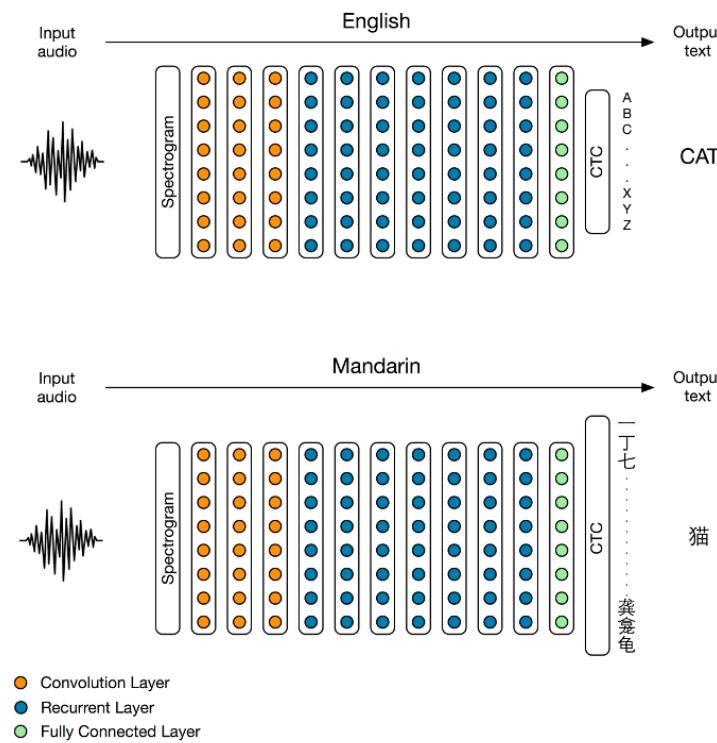
dmlc
mxnet

gensim

spaCy

theano

DL Today: Speech-to-Text



[Baidu 2014]

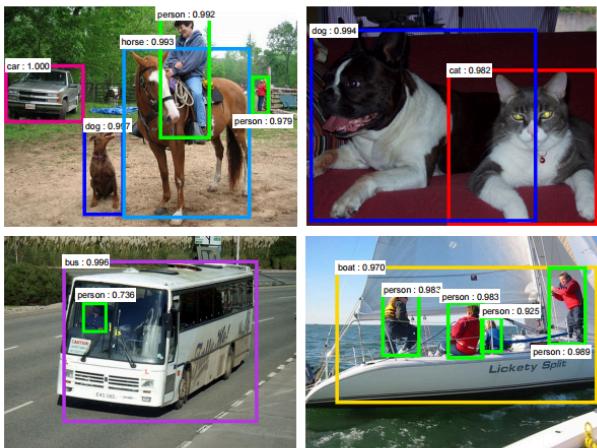
DL Today: Vision



[Krizhevsky 2012]



[Ciresan et al. 2013]



[Faster R-CNN - Ren 2015]

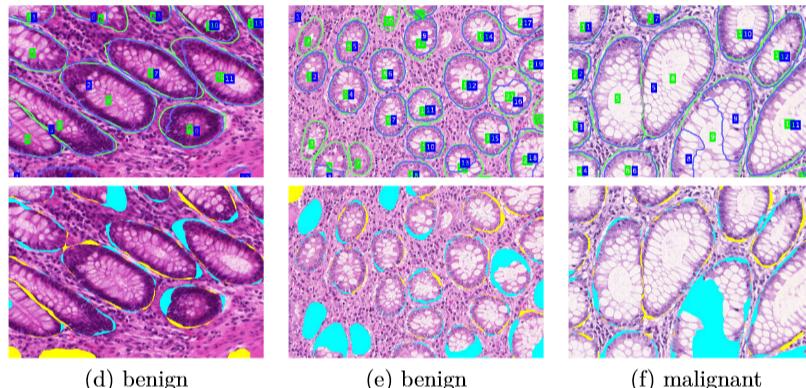


[NVIDIA dev blog]

DL Today: Vision



[Stanford 2017]



[Nvidia Dev Blog 2017]

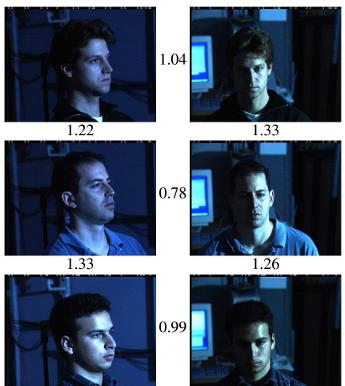
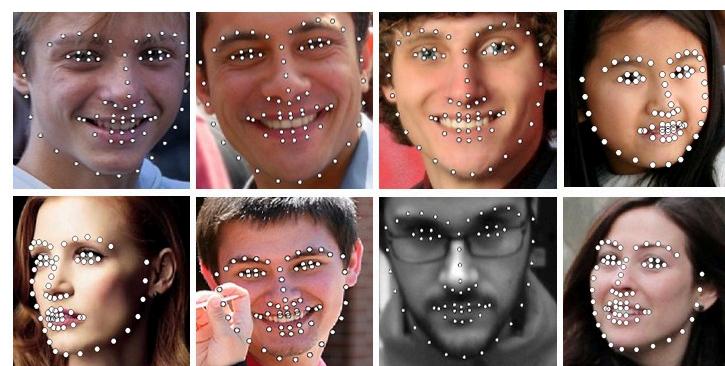


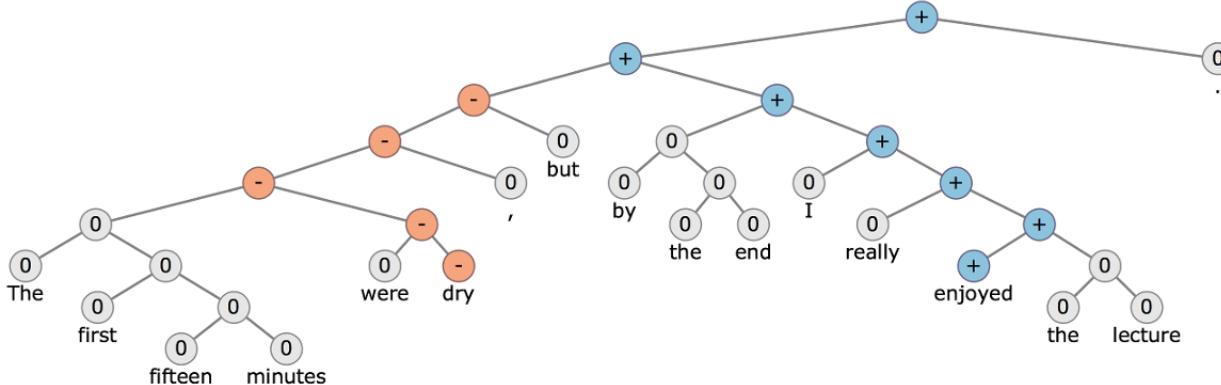
Figure 1. Illumination and Pose invariance.

[FaceNet - Google 2015]



[Facial landmark detection CUHK 2014]

DL Today: NLP



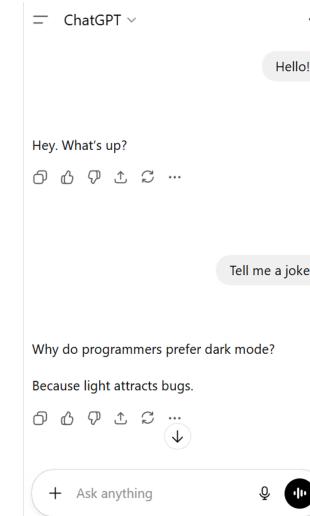
DL Today: NLP



[Google Inbox Smart Reply]

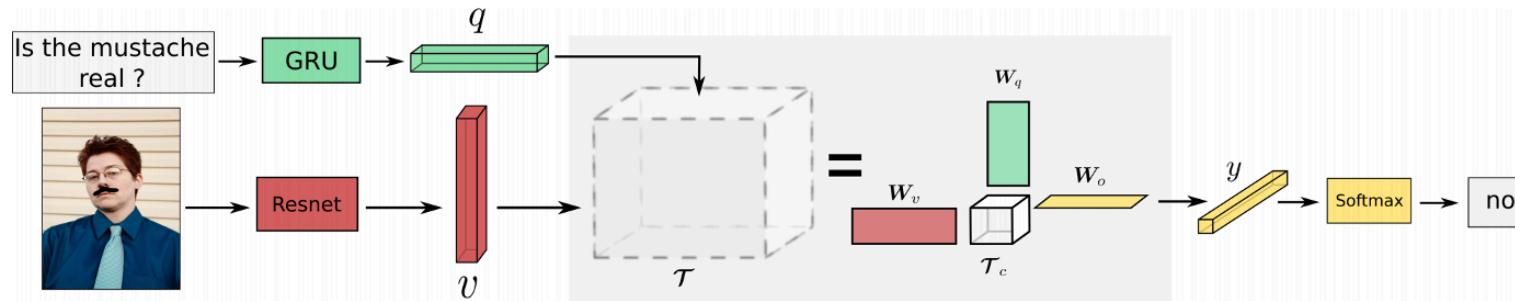


[Amazon Echo/Alexa]



[Chatbots]

DL Today: Vision + NLP



[VQA - Mutan 2017]



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."

[Karpathy 2015]

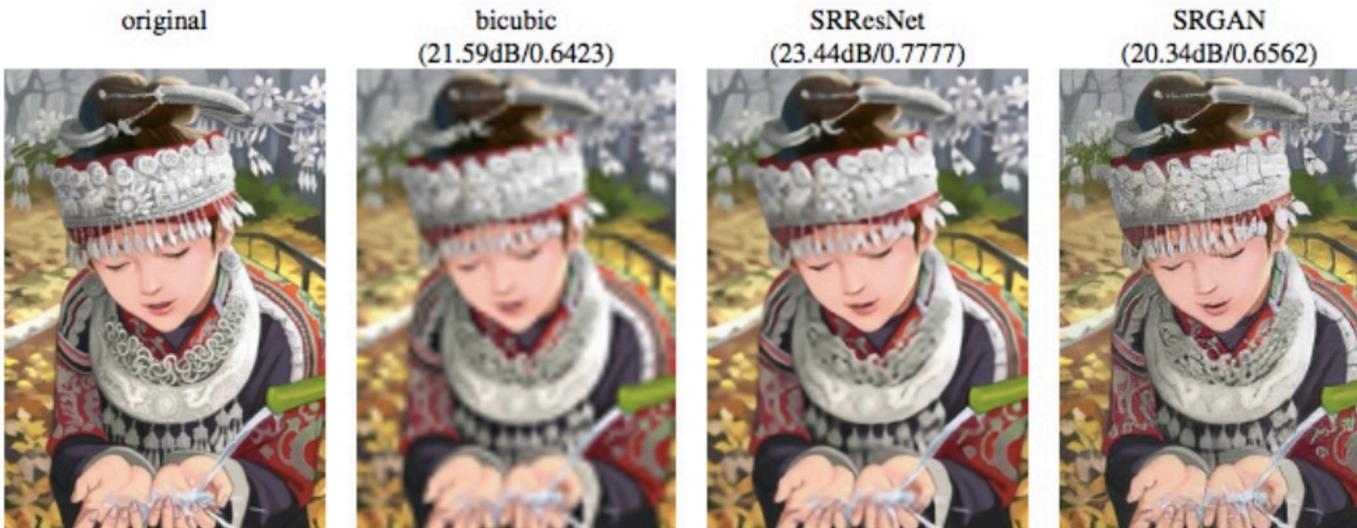
DL Today: Image translation



[DeepDream 2015]



[Gatys 2015]



DL Today: Generative models

Text description	This bird is blue with white and has a very short beak	This bird has wings that are brown and has a yellow belly	A white bird with a black crown and yellow beak	This bird is white, black, and brown in color, with a brown beak	The bird has small beak, with reddish brown crown and gray belly	This is a small, black bird with a white breast and white on the wingbars.	This bird is white black and yellow in color, with a short black beak
Stage-I images							
Stage-II images							

StackGAN v2 [Zhang 2017]

DL Today: Generative models

Guess which one is generated?



DL Today: Generative models

can you modify this code to track both the value and anytime value in the iterations? my goal is to evaluate the anytime performance of branches.

- I'll help you modify the code to track both the value and anytime value in the iterations. Let me first read the file to understand the current implementation.

Read branches.py

- Now I can see the code structure. To track both the value and anytime value for evaluating anytime performance, I need to modify the `solve_ordinal` and `solve_binary` methods to record the current best value at each iteration. Let me make these modifications:

Edit branches.py

Added 11 lines

```
def solve_binary(self, lambd, n=1000, print_iter=100, time_limit=600):
    """
    Description
    -----
    Search for the optimal Decision Tree.

    Parameters
    -----
    lambd : Float in [0, 1], the complexity parameter.
    n : Int, maximum number of iterations.
    print_iter : Int, number of iterations between two prints.
    time_limit : Int, Time limit in number of seconds.
    """

    # Access the tracking data
    iterations = self.model.iteration_history
    values = self.model.value_history
    anytime_values = self.model.anytime_value_history
```

Changes Made:

1. Added tracking attributes in both `solve_ordinal` (lines 100-103) and `solve_binary` (lines 152-155):

- `self.iteration_history`: List of iteration numbers
- `self.value_history`: List of `self.lattice.root.value` at each iteration
- `self.anytime_value_history`: List of `self.lattice.root.value_complete` at each iteration

2. Recording values after each iteration in both methods:

- In `solve_ordinal` (lines 114-117)
- In `solve_binary` (lines 181-184)

How to Use:

After calling the `fit` method or `solve` method on a `Branches` instance, you can access the anytime performance data through:

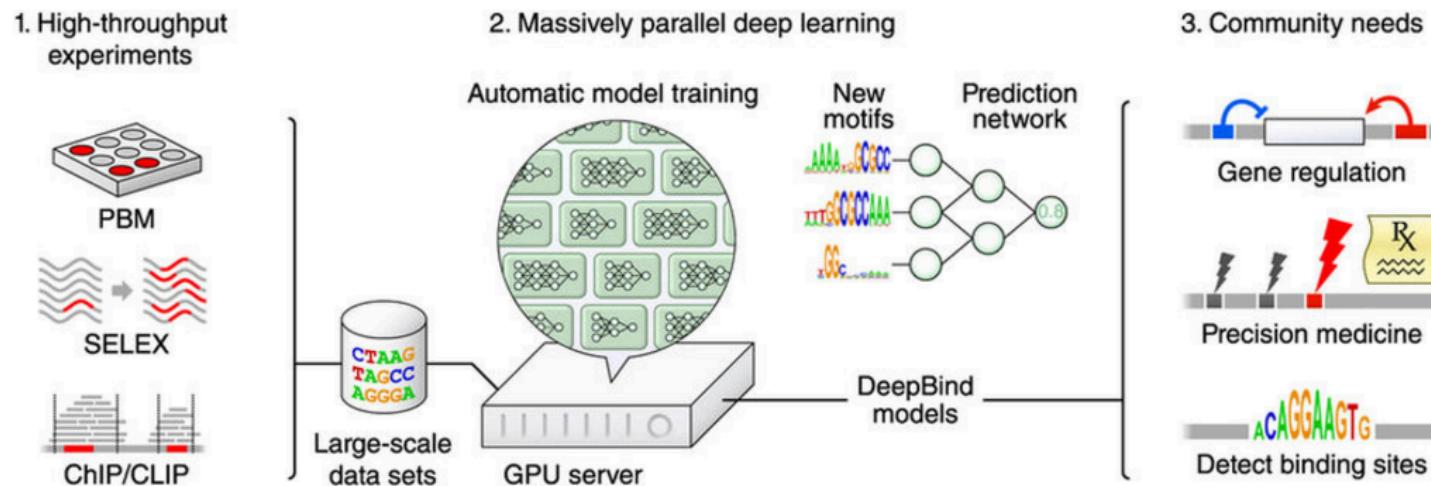
```
branches = Branches(max_depth=4, lambd=0.01, iterations=1000)
branches.fit(X, y)

# Access the tracking data
iterations = branches.model.iteration_history
values = branches.model.value_history
anytime_values = branches.model.anytime_value_history
```

This will allow you to plot and analyze how the value and anytime value evolve over iterations, which is exactly what you need for evaluating the anytime performance of the branches algorithm.

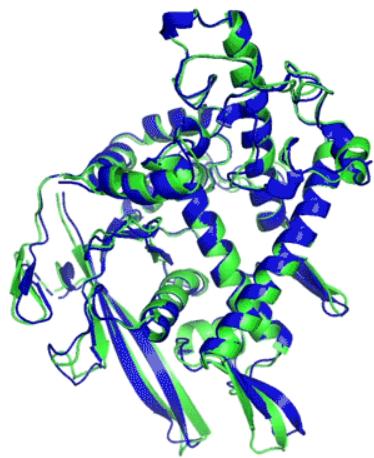
[Claude Code]

DL in Science: Genomics

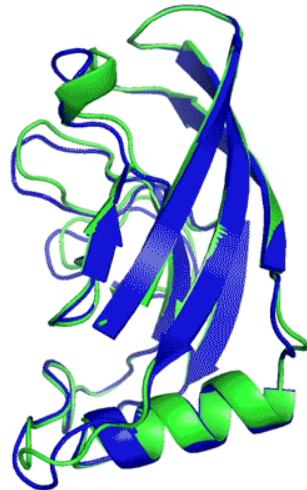


[Deep Genomics 2017]

DL in Science: Genomics



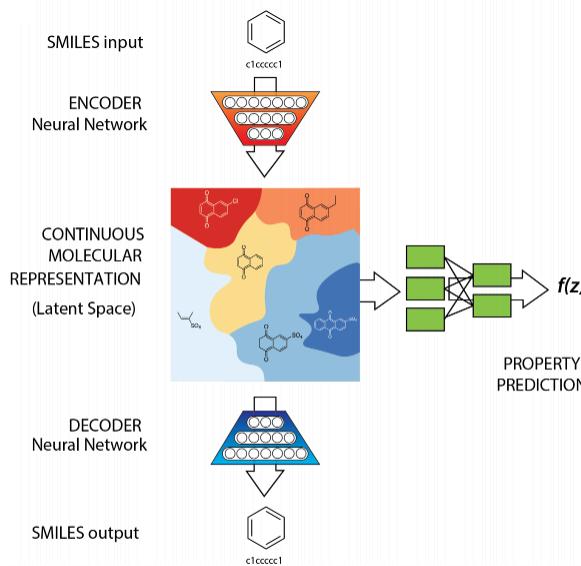
T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



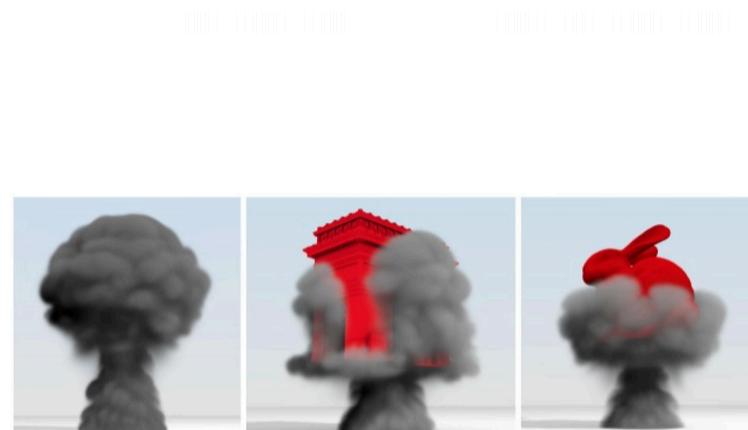
T1049 / 6y4f
93.3 GDT
(adhesin tip)

- Experimental result
- Computational prediction

DL in Science: Chemistry, Physics



[Gómez-Bombarelli 2016]

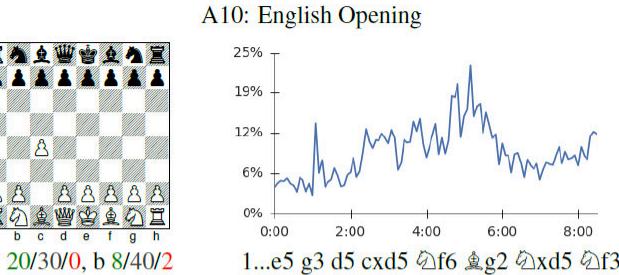
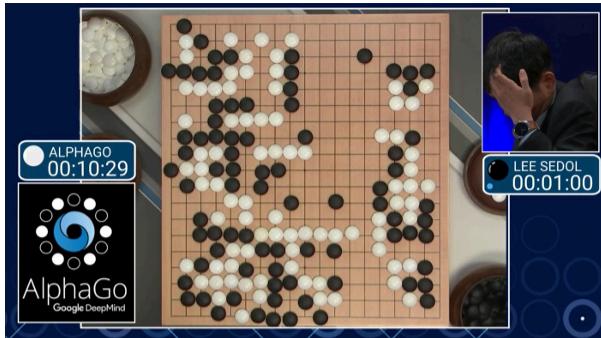


[Tompson 2016]

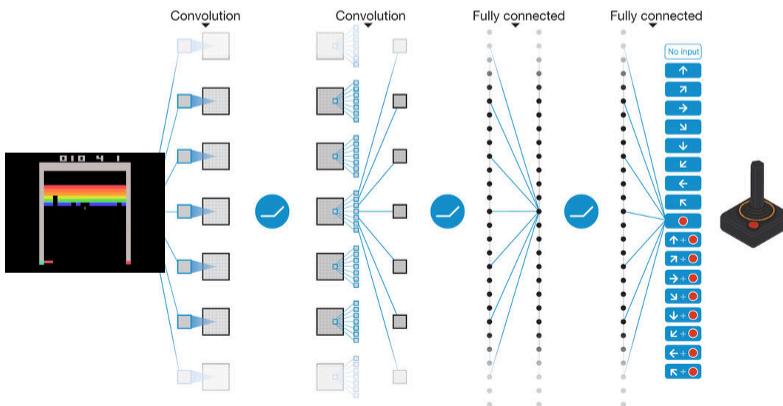
DL in Science: Chemistry, Physics

Finite element simulator accelerated (~100 fold) by a 3D convolutional network

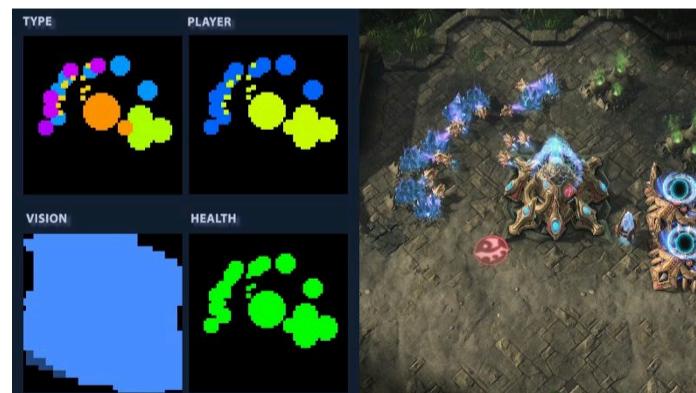
DL for AI in games



[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]



[Starcraft 2 for AI research]

Outline of the class

- Backpropagation
- Computer Vision
- Recommender Systems
- Natural Language Processing
- Optimization: theory, methods and tricks
- Generative models & unsupervised learning

How this course works works

- Lectures ~1 hour
- Break ~15 minutes
- Practical session ~1 hour
 - Work in breakout groups and discuss!
 - Homework: complete the lab
- Two assignments
 - One due at the end of week 1, one at the end of week 2

Frameworks and Computation Graphs

Libraries & Frameworks



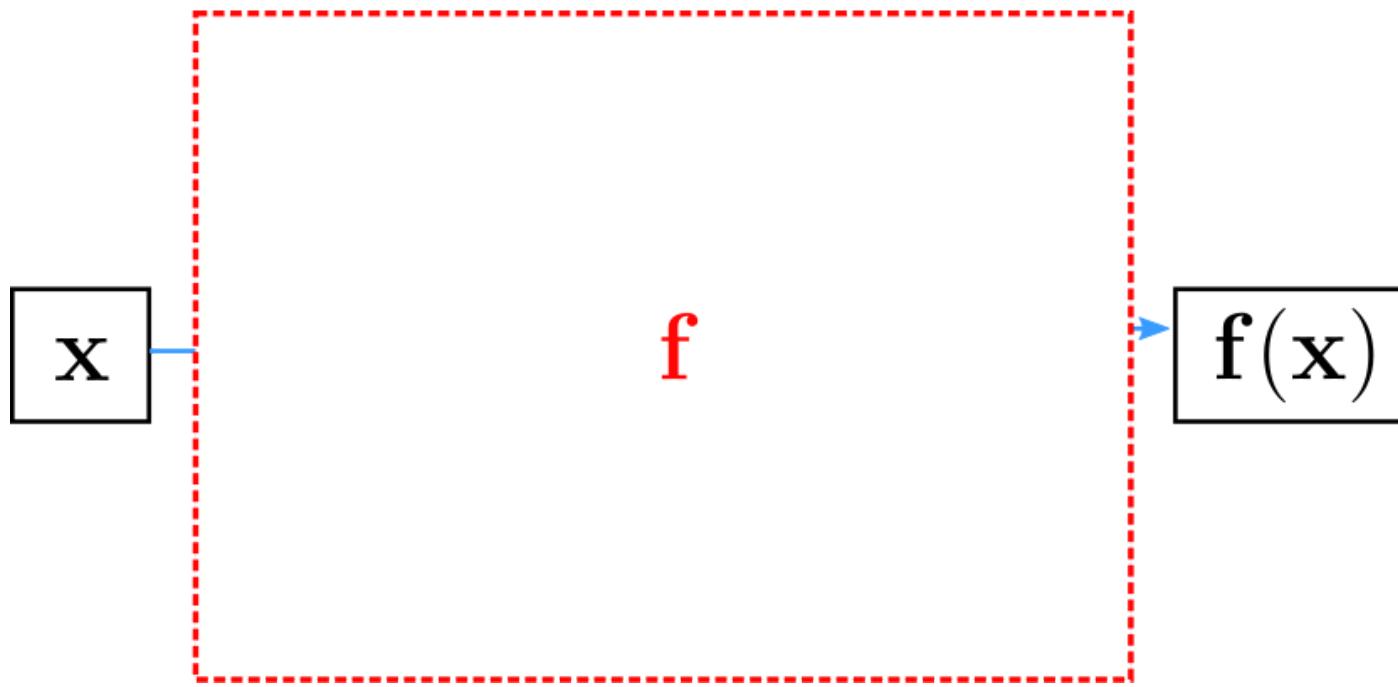
gensim **spaCy**

theano

This lecture is using **Keras**: high level frontend for **TensorFlow** (and MXnet, Theano, CNTK)

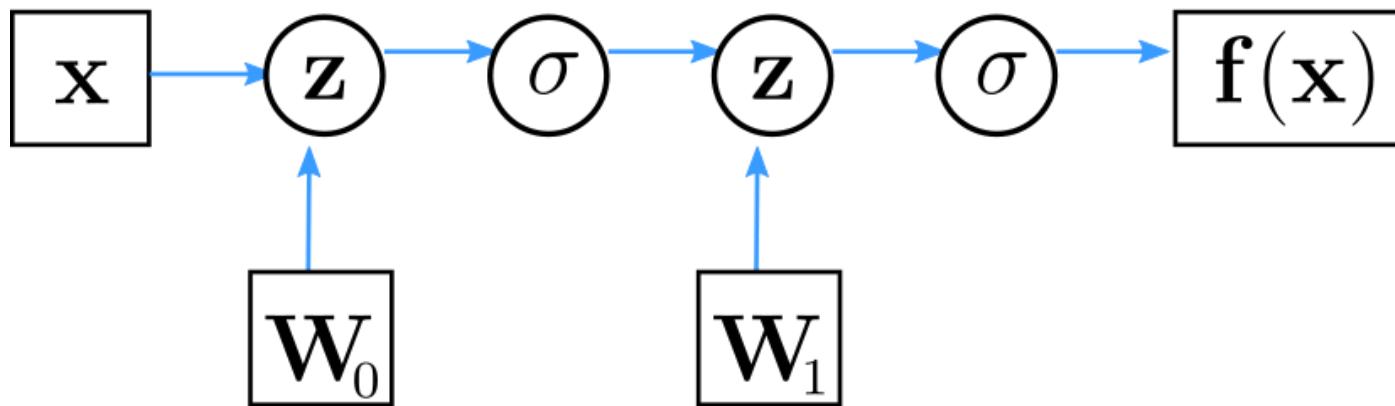
One lab will be dedicated to a short **Pytorch** introduction.

Computation Graph



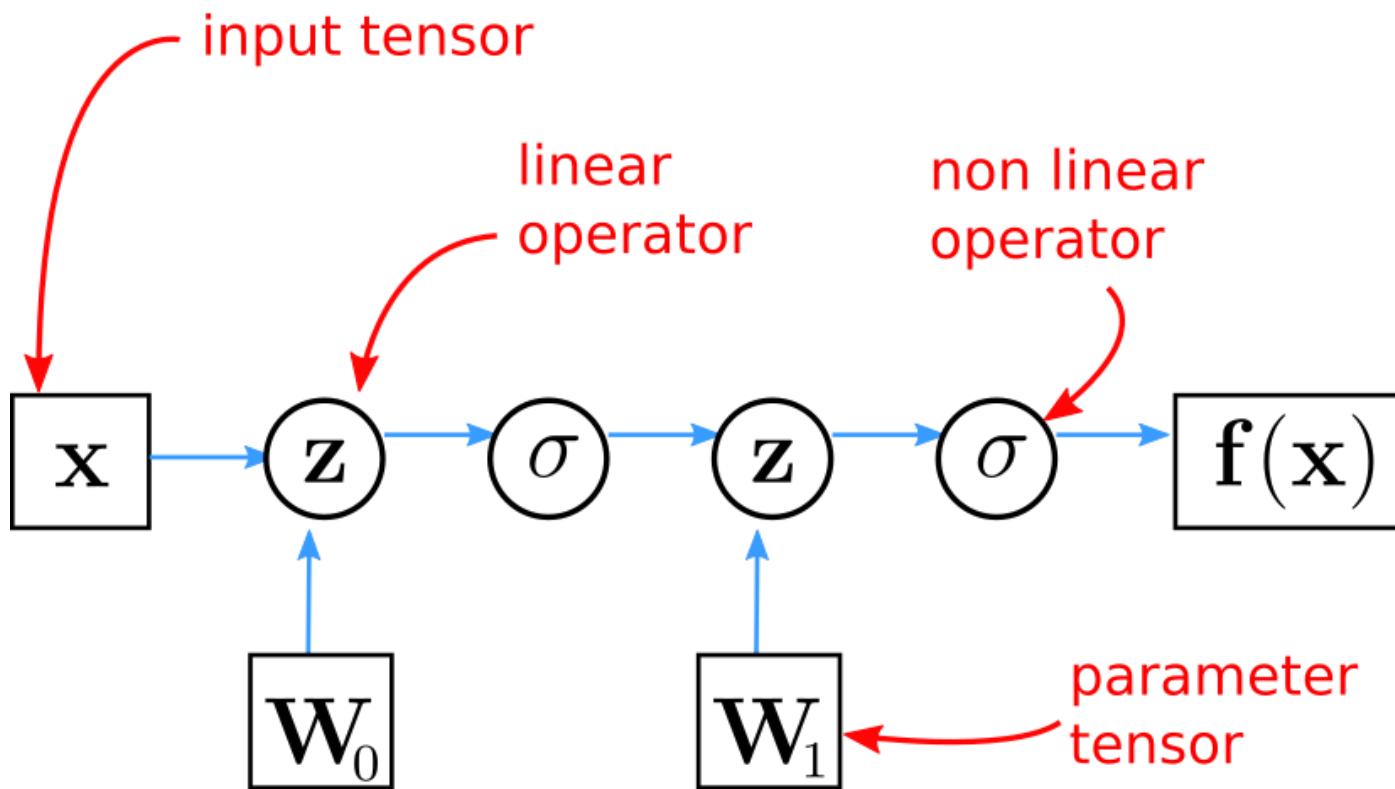
Neural network = parametrized, non-linear function

Computation Graph



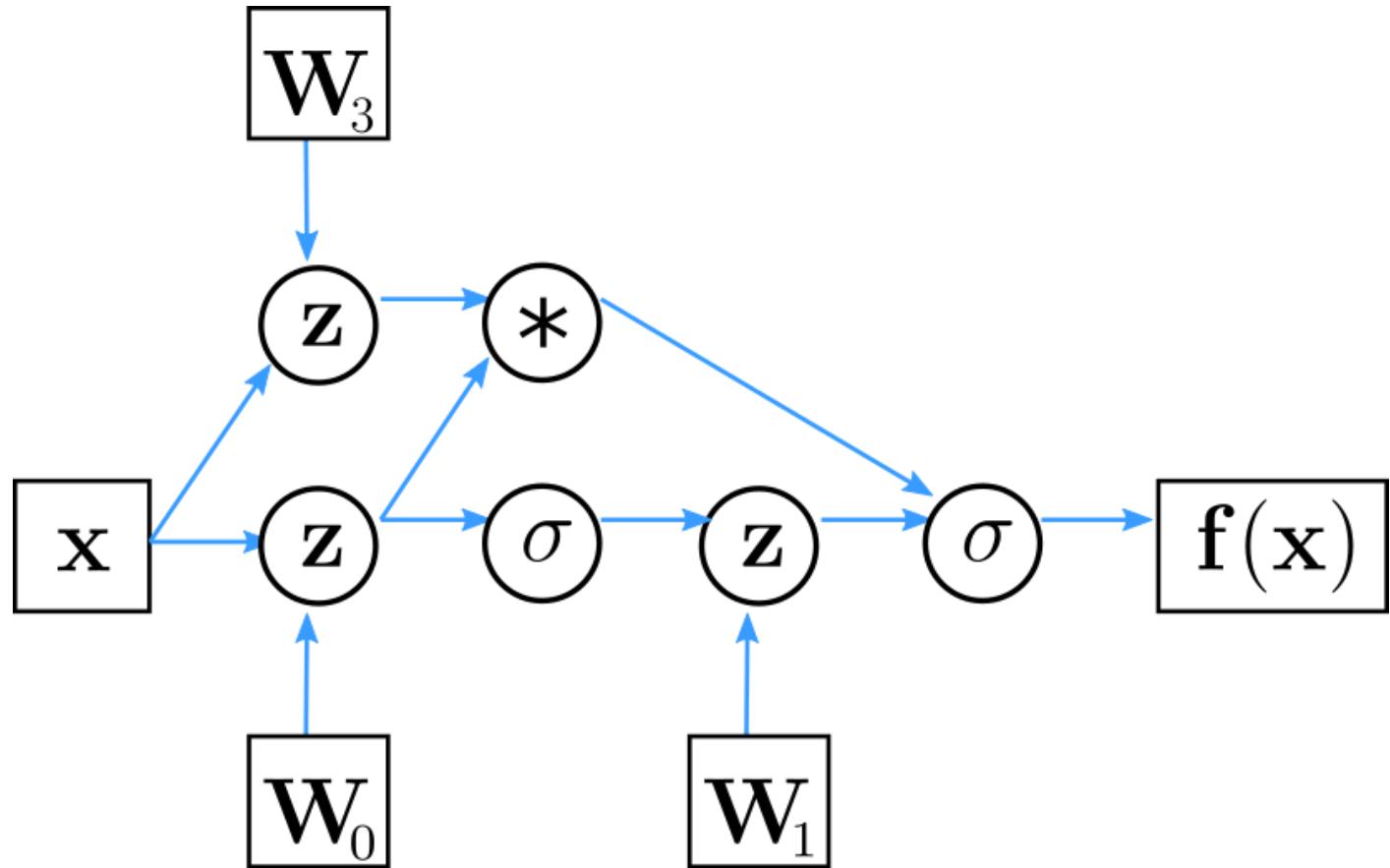
Computation graph: Directed graph of functions, depending on parameters (neuron weights)

Computation Graph



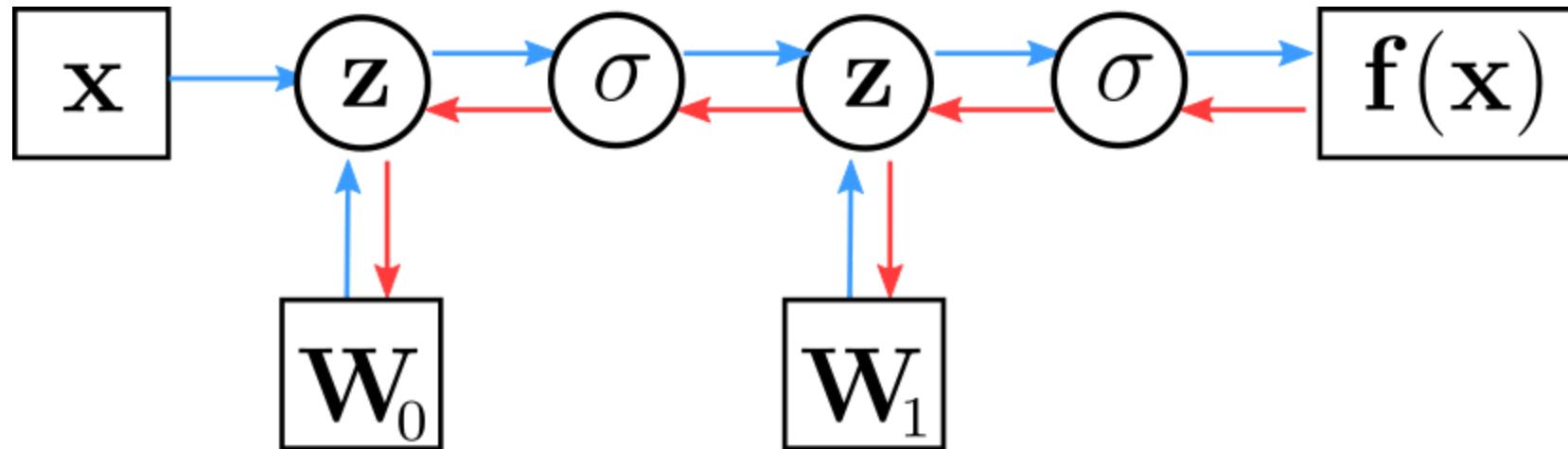
Combination of linear (parametrized) and non-linear functions

Computation Graph



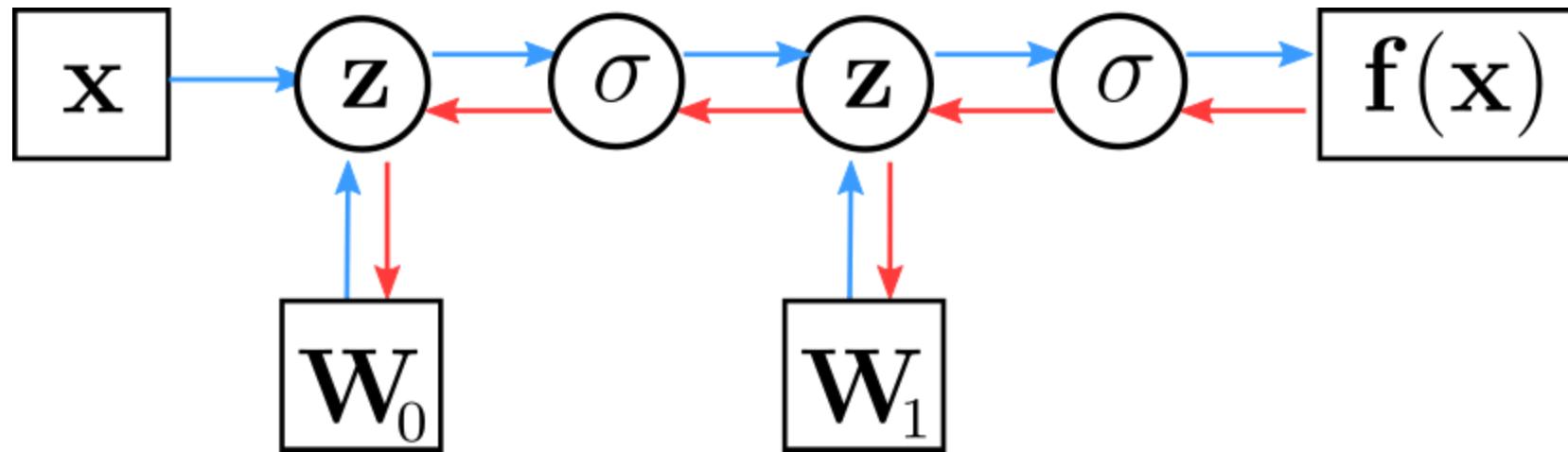
Not only sequential application of functions

Computation Graph



- Automatic computation of gradients: all modules are **differentiable!**
- Theano (now Aesara), **Tensorflow 1**, etc. build a static computation graph via static declarations.
- **Tensorflow 2**, **PyTorch**, **JAX**, etc. rely on dynamic differentiable modules: "define-by-run".
- Vector computation on **CPU** and accelerators (**GPU** and **TPU**).

Computation Graph



Simple keras implementation

```
model = Sequential()
model.add(Dense(H, input_dim=N))      # defines w0
model.add(Activation("tanh"))
model.add(Dense(K))                  # defines w1
model.add(Activation("softmax"))
```

Next: Lab 1!