2 - Deep Learning

Ludwig Krippahl

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

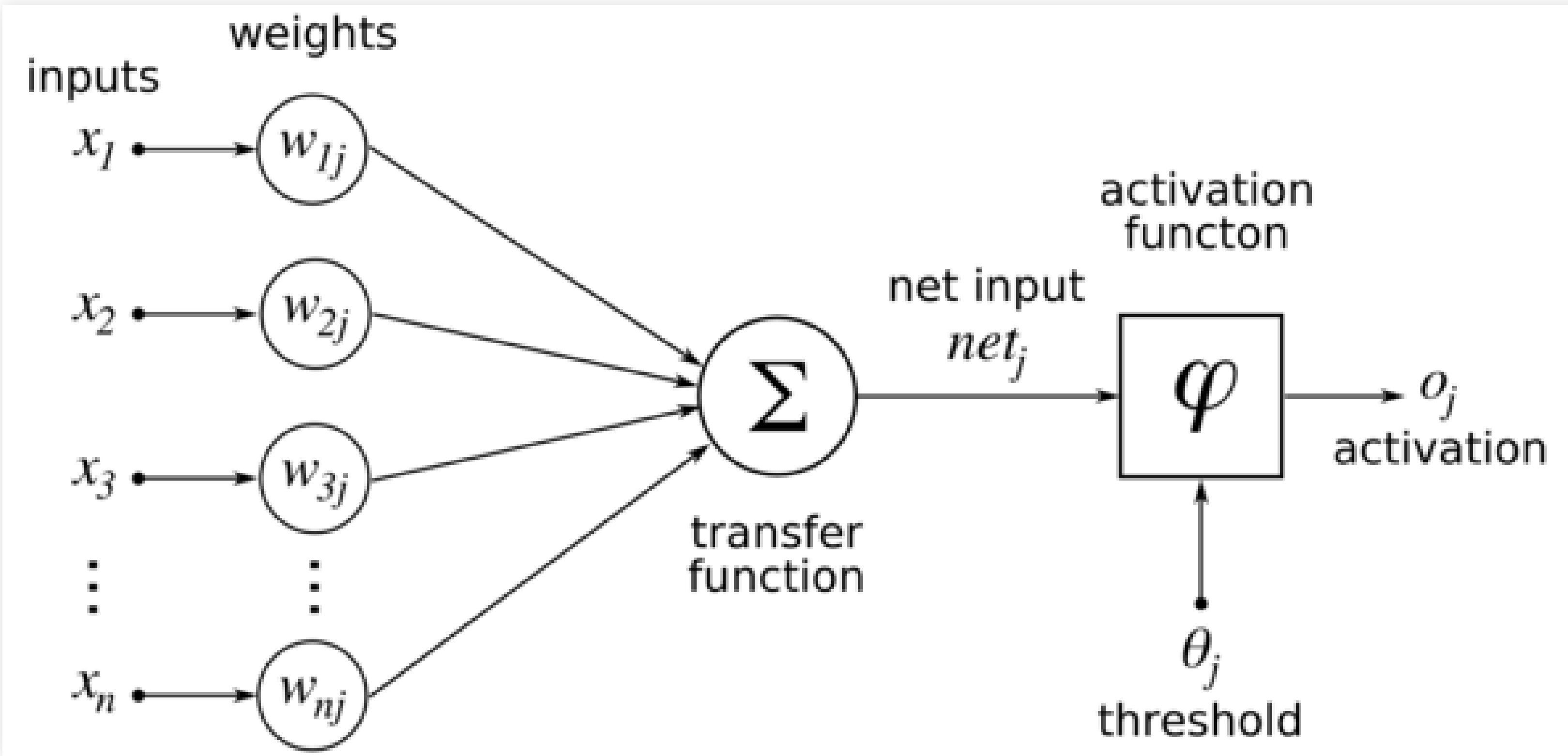
Summary

- Backpropagation
- Stochastic Gradient Descent
- Deep model architectures
- Features and Data
- Why the success now?
- Some examples of deep architectures and applications

Backpropagation

Backpropagation

- Neuron: linear combination of inputs with non-linear activation

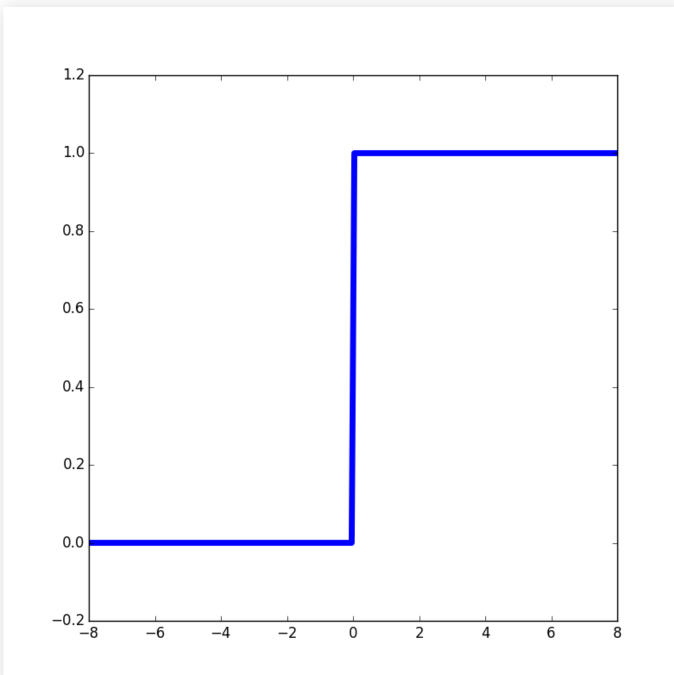


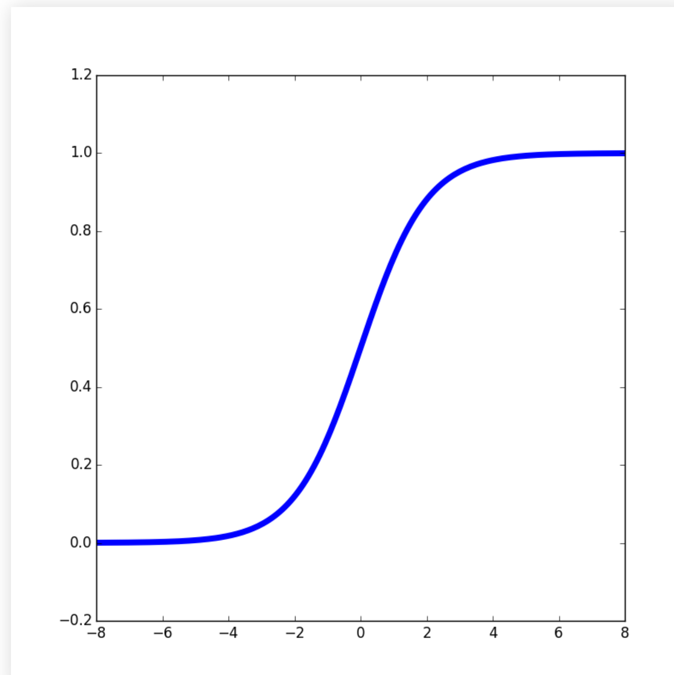
FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Backpropagation

- To propagate error through weight parameters we need derivatives.
  - E.g. sigmoid activation

$$s(y) = \frac{1}{1+e^{-y}} = \frac{1}{1+e^{-\vec{w}^T\vec{x}}}$$





Training a single neuron

- Minimize quadratic error between class and output

$$E = \frac{1}{2}\sum_{j=1}^{N}(t^j - s^j)^2$$

- Like perceptron, present each example and adjust weights.
- Gradient of the error wrt w: $-\frac{\delta E^j}{\delta w_i} = -\frac{\delta E^j}{\delta s^j}\frac{\delta s^j}{\delta net^j}\frac{\delta net^j}{\delta w_i}$

$$E^t = \frac{1}{2}(t^j - s^j)^2 \qquad \frac{\delta E^j}{\delta s^j} = -(t^j - s^j)$$

$$s^j = \frac{1}{1+e^{-net^j}} \qquad \frac{\delta s^j}{\delta net^j} = s^j(1-s^j)$$

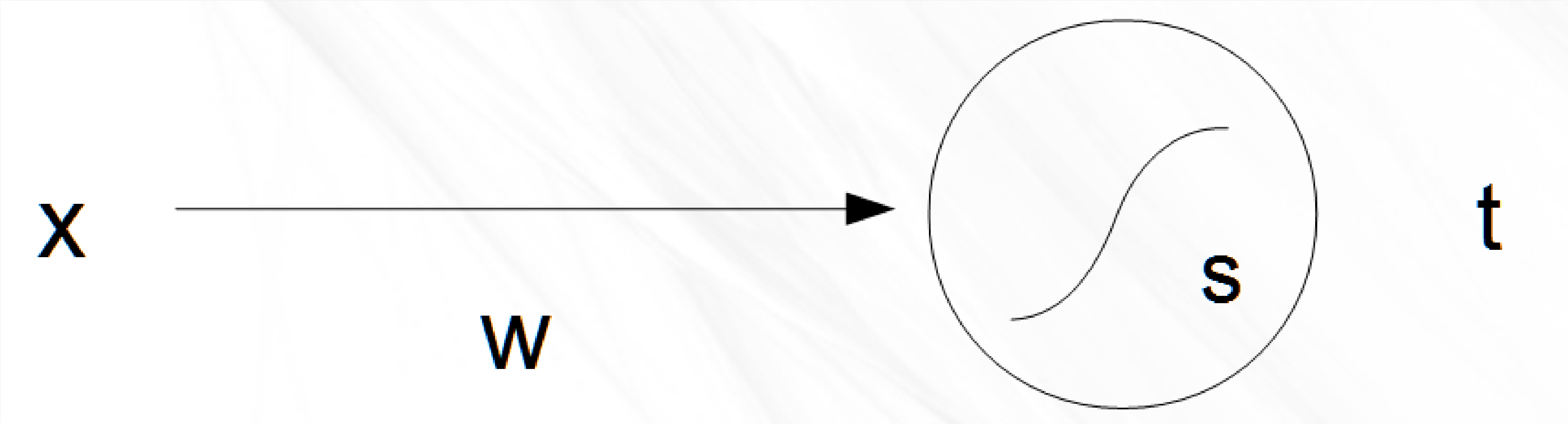
$$net^j = w_0 + \sum_{i=1}^{M} w_i x_i \qquad \frac{\delta net^j}{\delta w_i} = x_i$$

Single Neuron

- Update rule (η generally small, ~0.1):

$$\Delta w_i^j = -\eta \frac{\delta E^j}{\delta w_i} = \eta (t^j - s^j) s^j (1 - s^j) x_i^j$$

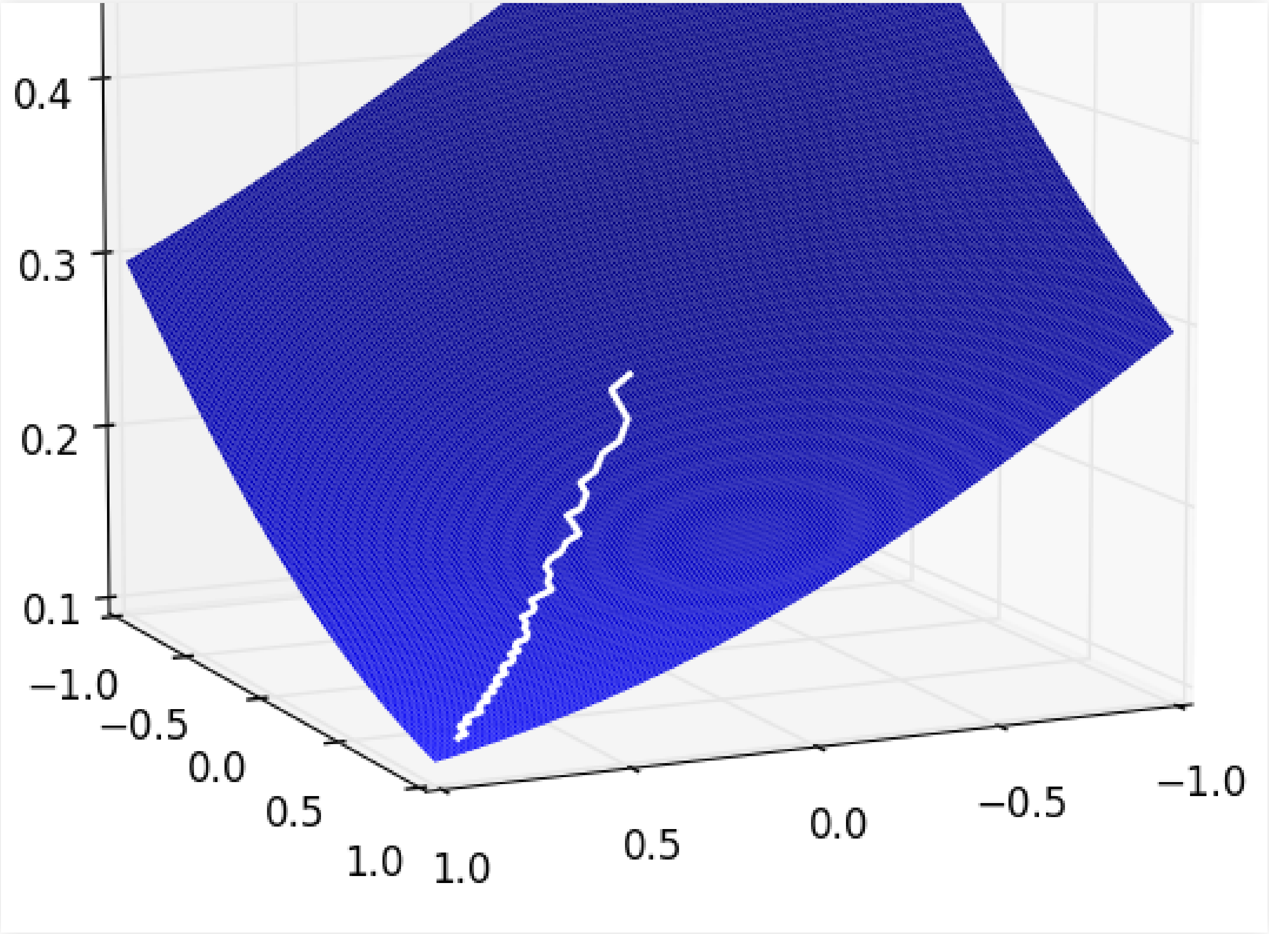
- Intuitive explanation:



Stochastic Gradient Descent

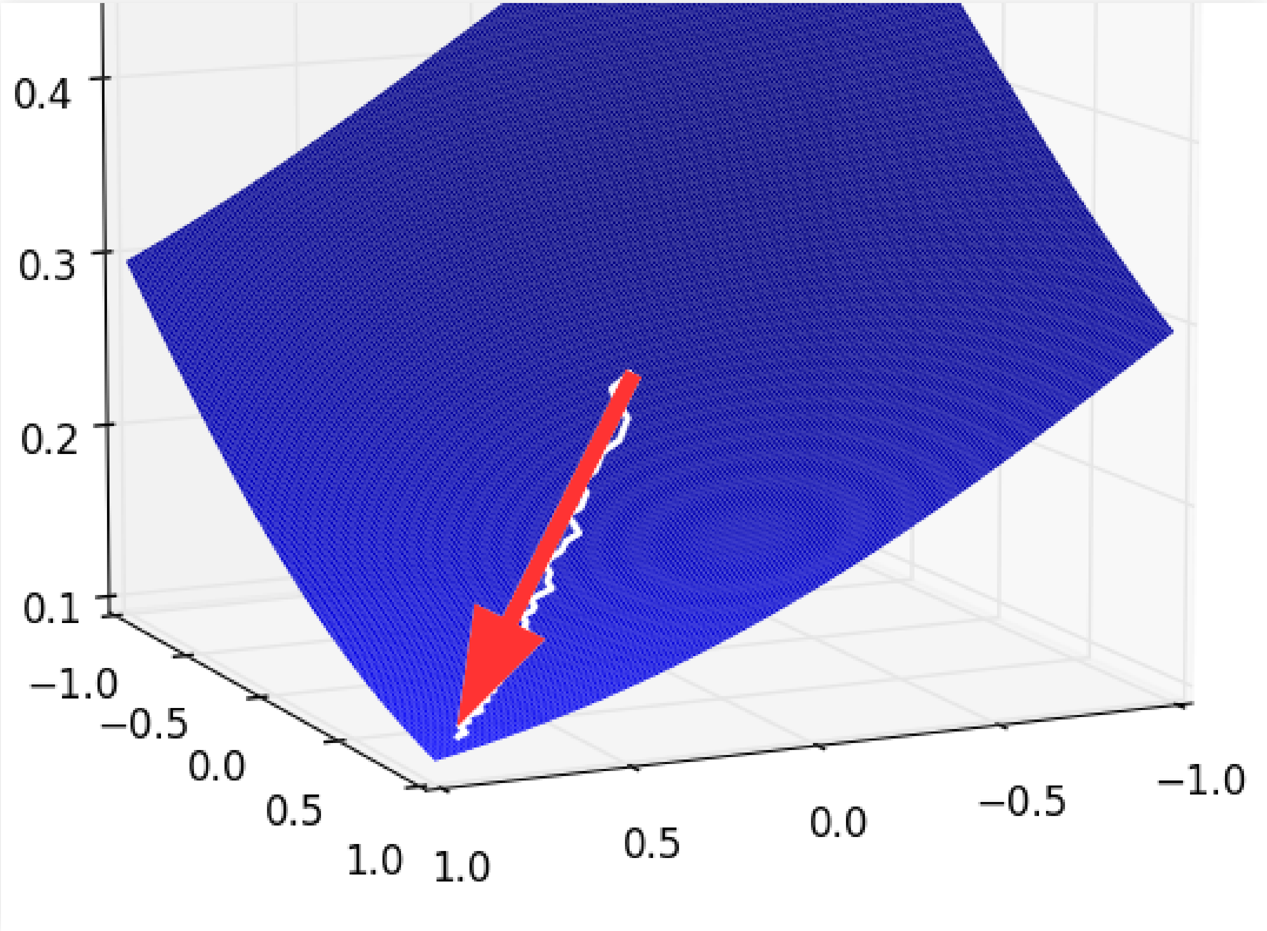
Stochastig Gradient Descent

- Online learning: one step per example, in random order



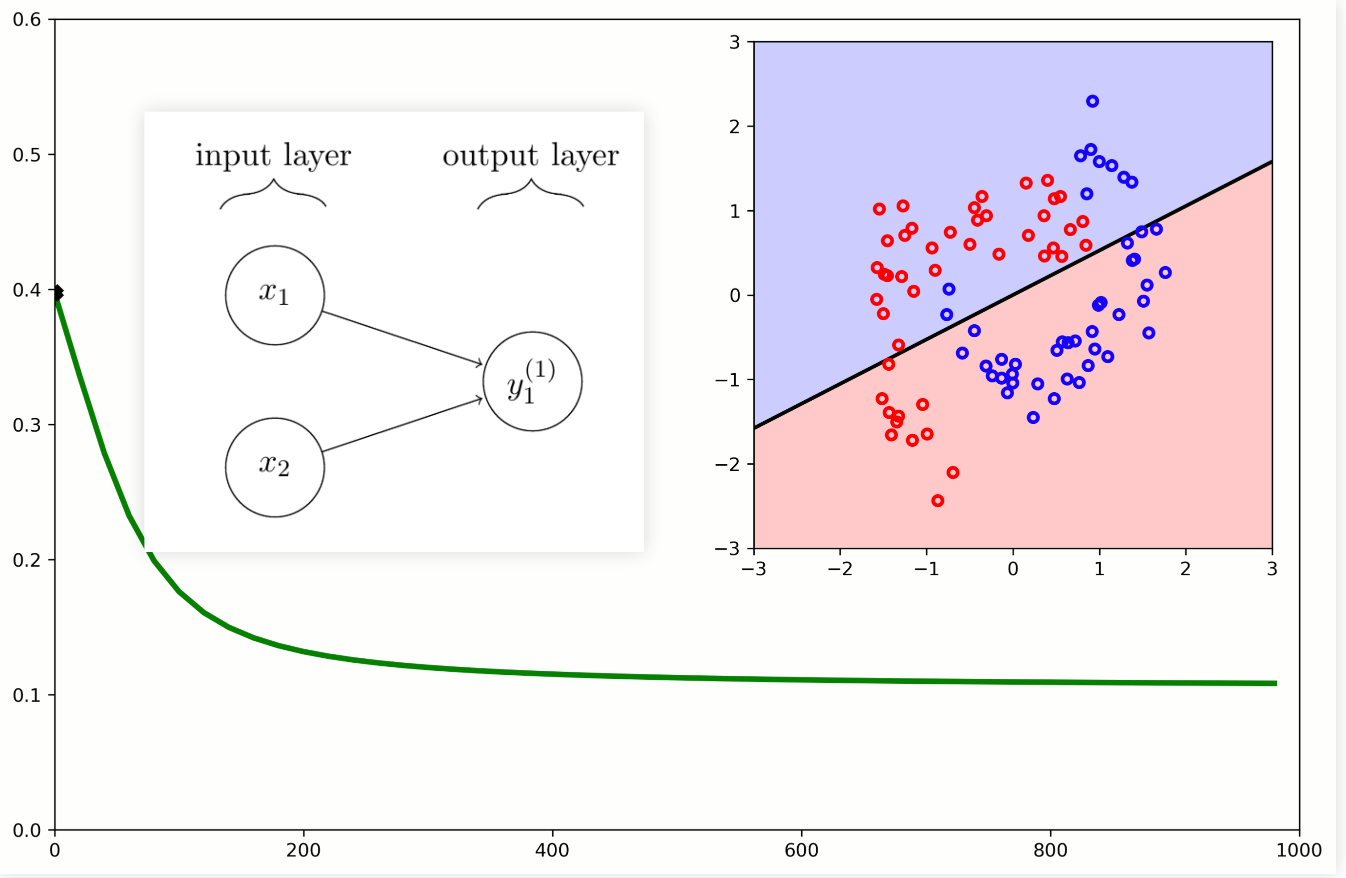
Stochastig Gradient Descent

- Batch training: add Δw_i^j for batch, then update



Single neuron

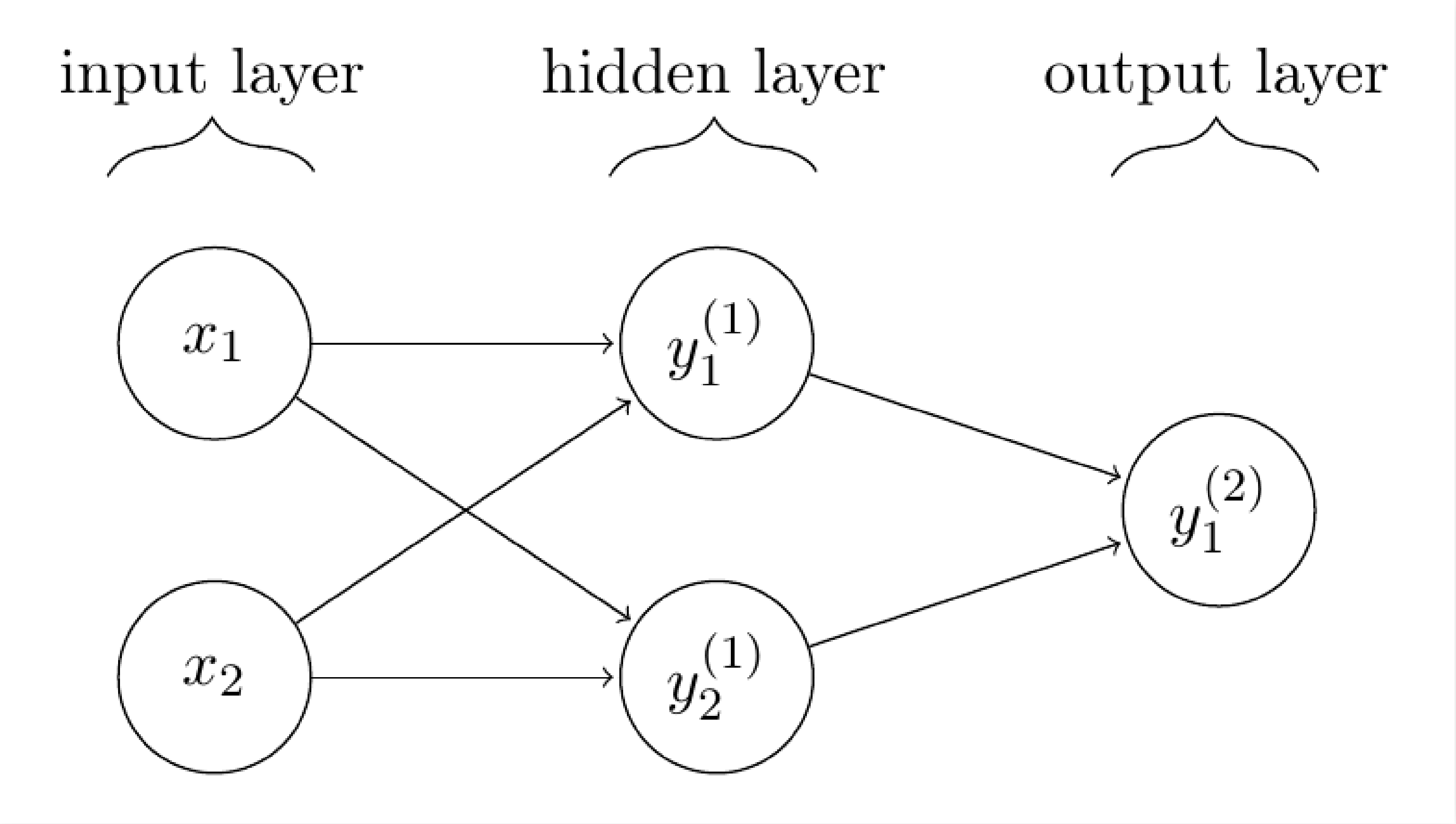
- A single neuron is a linear classifier:



Multilayer Perceptron

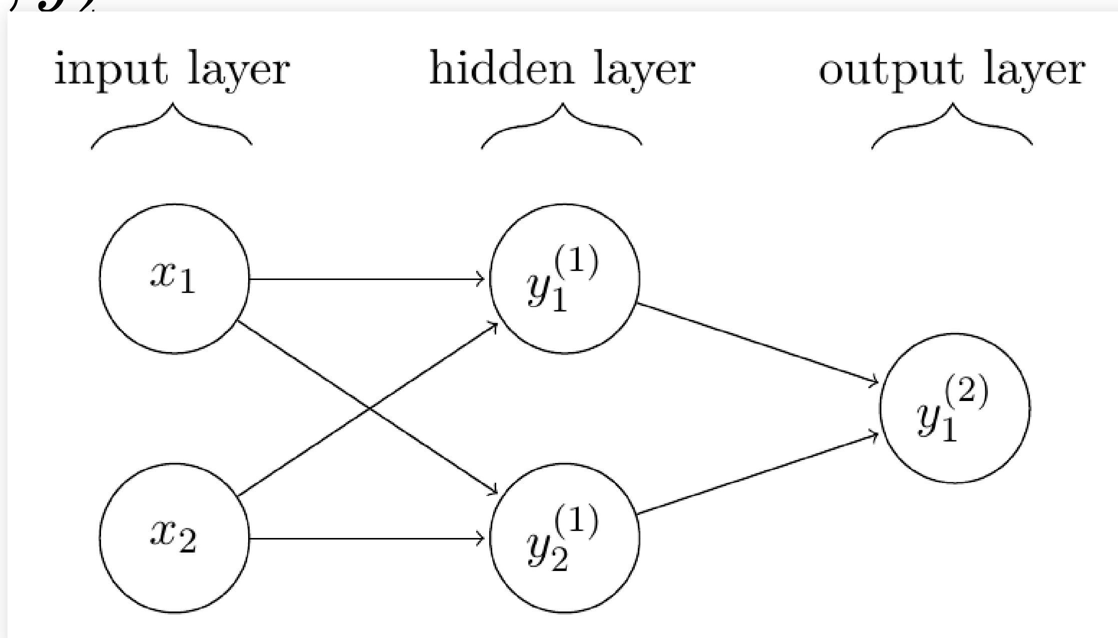
Multilayer Perceptron

- Multilayer Perceptron is a fully connected, feed forward, ANN
  - Layers chain nonlinear transformations



Multilayer Perceptron

- Fully connected
- Feed-forward
- Input layer: x_1, x_2
- Hidden layer(s): $y_1^{(1)}, y_2^{(1)}$
- Output layer: $y_1^{(2)}$



Training a Multilayer Perceptron

- Output neuron n of layer k receives input from m from layer i through weight j
  - Same as single neuron but using output of previous instead of x
- With sigmoid activations:

$$\Delta w^j_{mkn} = -\eta \frac{\delta E^j_{kn}}{\delta s^j_{kn}} \frac{\delta s^j_{kn}}{\delta net^j_{kn}} \frac{\delta net^j_{kn}}{\delta w_{mkn}} = \eta(t^j - s^j_{kn})s^j_{kn}(1 - s^j_{kn})s^j_{im} = \eta\delta_{kn}s^j_{im}$$

- Compute δ for each neuron

$$\delta = (t^j - s^j_{kn})s^j_{kn}(1 - s^j_{kn})$$

Training a Multilayer Perceptron

- For a weight m on hidden layer i, we must propagate the output error backwards from all neurons ahead

• Gradient of error w.r.t. weight of output neuron:

$$\frac{\delta E^j_{kn}}{\delta s^j_{kn}} \frac{\delta s^j_{kn}}{\delta net^j_{kn}} \frac{\delta net^j_{kn}}{\delta w_{mkn}}$$

- Propagate back the errors of all forward neurons (and compute δ):

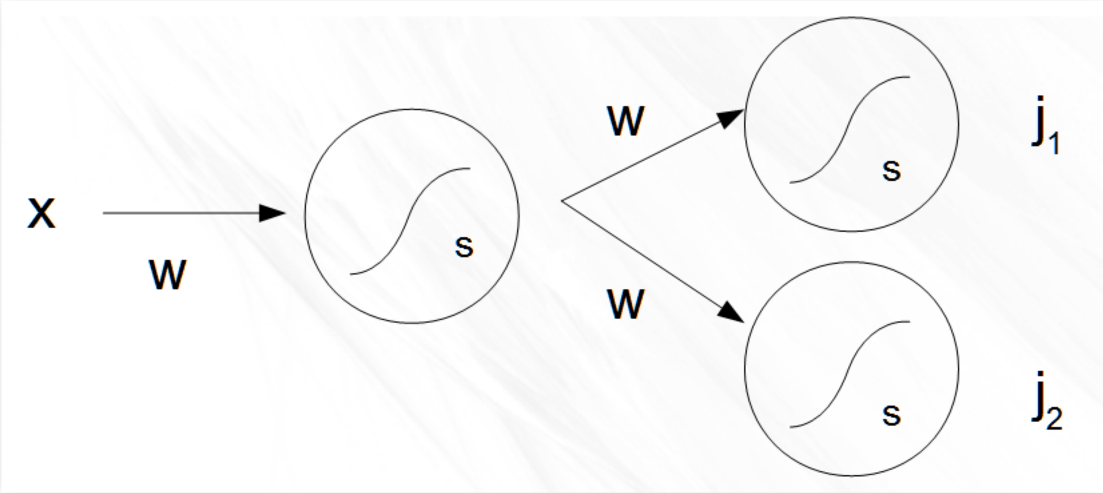
$$\begin{aligned} \Delta w^j_{min} &= -\eta \left(\sum_p \frac{\delta E^j_{kp}}{\delta s^j_{kp}} \frac{\delta s^j_{kp}}{\delta net^j_{kp}} \frac{\delta net^j_{kp}}{\delta s^j_{in}} \right) \frac{\delta s^j_{in}}{\delta net^j_{in}} \frac{\delta net^j_{in}}{\delta w_{min}} \\ &= \eta (\sum_p \delta_{kp} w_{mkp}) s^j_{in} (1 - s^j_{in}) x^j_i = \eta \delta_{in} x^j_i \end{aligned}$$

Training a Multilayer Perceptron

- Intuitive explanation:

$$\Delta w^j_{min} = -\eta \left(\sum_p \frac{\delta E^j_{kp}}{\delta s^j_{kp}} \frac{\delta s^j_{kp}}{\delta net^j_{kp}} \frac{\delta net^j_{kp}}{\delta s^j_{in}} \right) \frac{\delta s^j_{in}}{\delta net^j_{in}} \frac{\delta net^j_{in}}{\delta w_{min}}$$

$$= \eta (\sum_p \delta_{kp} w_{mkp}) s^j_{in} (1 - s^j_{in}) x^j_i = \eta \delta_{in} x^j_i$$



Backpropagation Algorithm

- (MLP, sigmoid activation, quadratic error)
- Propagate the input forward through all layers

$$s(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}^T \vec{x})}}$$

- For output neurons compute

$$\delta_k = s_k(1 - s_k)(t - s_k)$$

- Backpropagate errors to back layers to compute all δ

$$\delta_i = s_i(1 - s_i) \sum_p \delta_p w_{pk}$$

• Note: w_{pk} are weights of "front" neurons connecting to neuron i

- Update weights (for forward layers, x is s of back layer)

$$\Delta w_{ki} = \eta \delta_i x_{ki}$$

Backpropagation Algorithm, general case

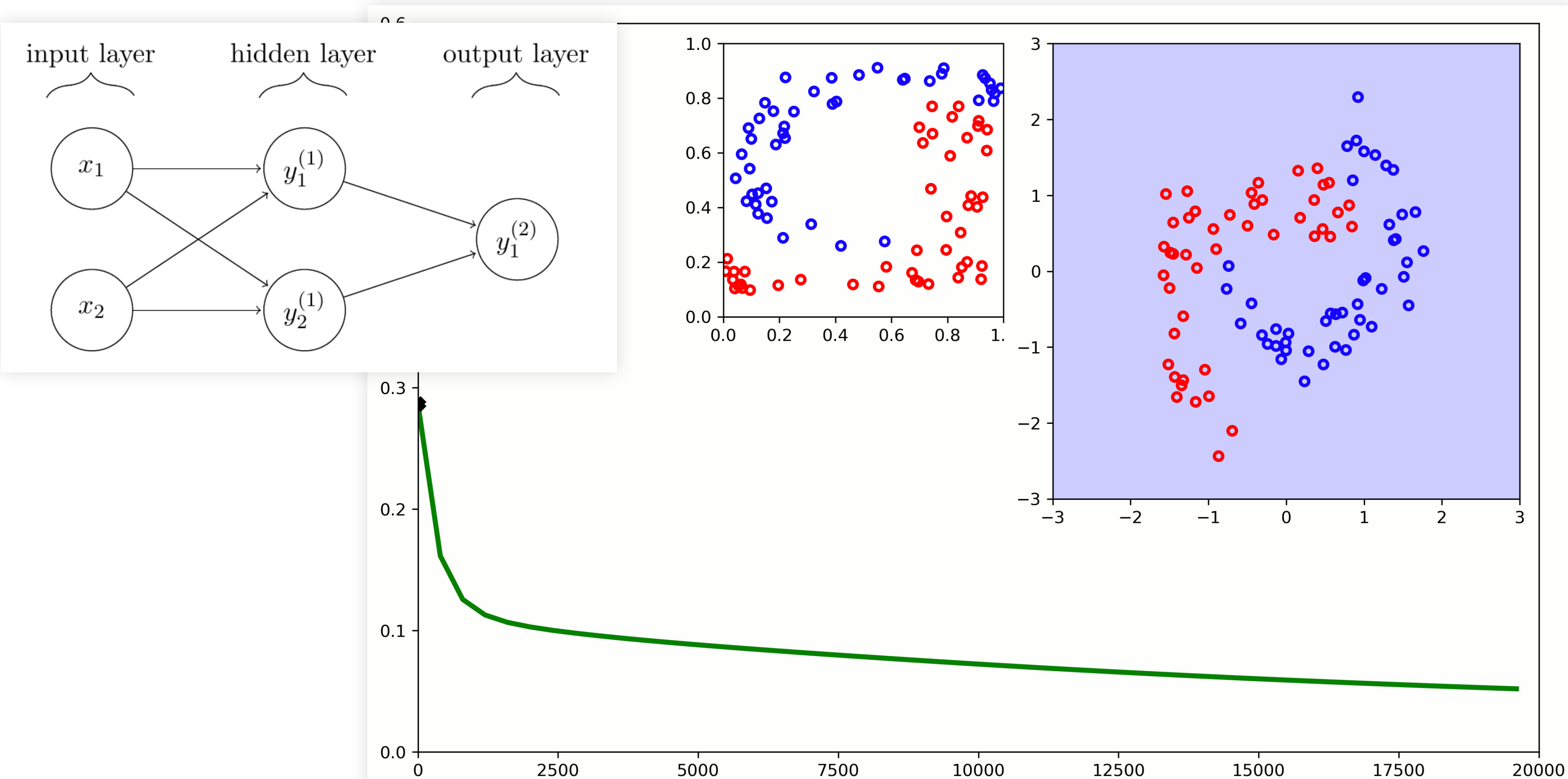
- Propagate the input forward through all layers
  - Compute activations
- For output neurons compute
  - Loss function
  - Derivatives of loss function
- Backpropagate derivatives of loss function to back layers
- Update weights using the computed derivatives

This can be generalized

- Different architectures
- Different activation functions
- Different loss functions, regularization, etc

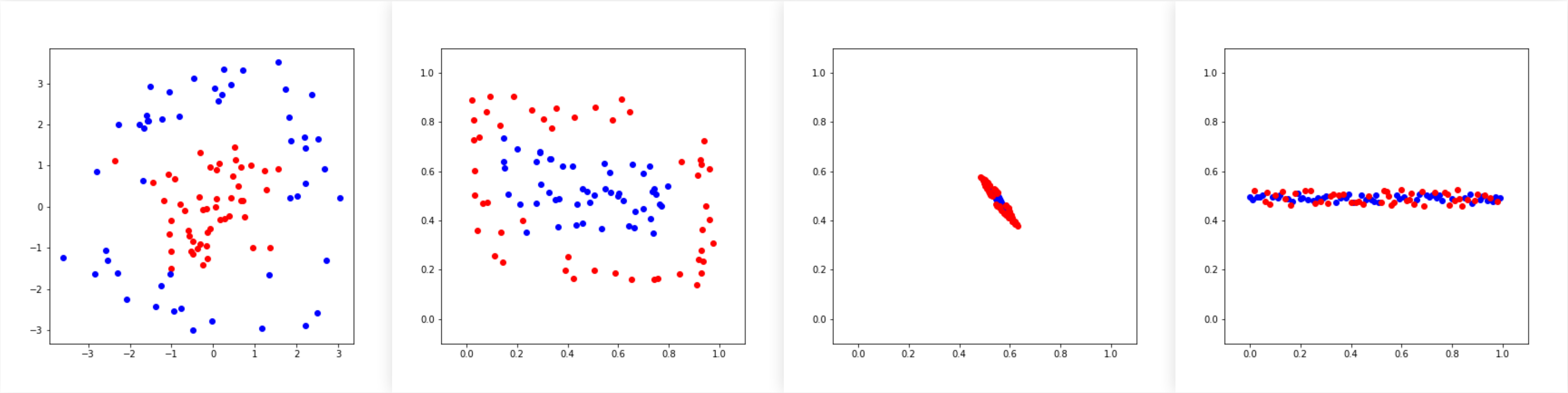
Neural Networks stack nonlinear transformations

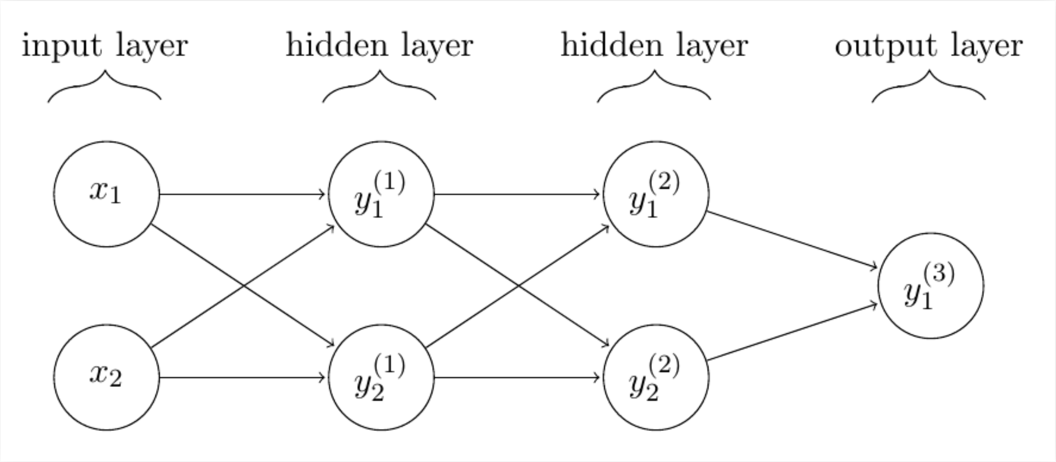
- We can go beyond linear classifiers by stacking layers



Multilayer Perceptron

Neural Networks stack nonlinear transformations

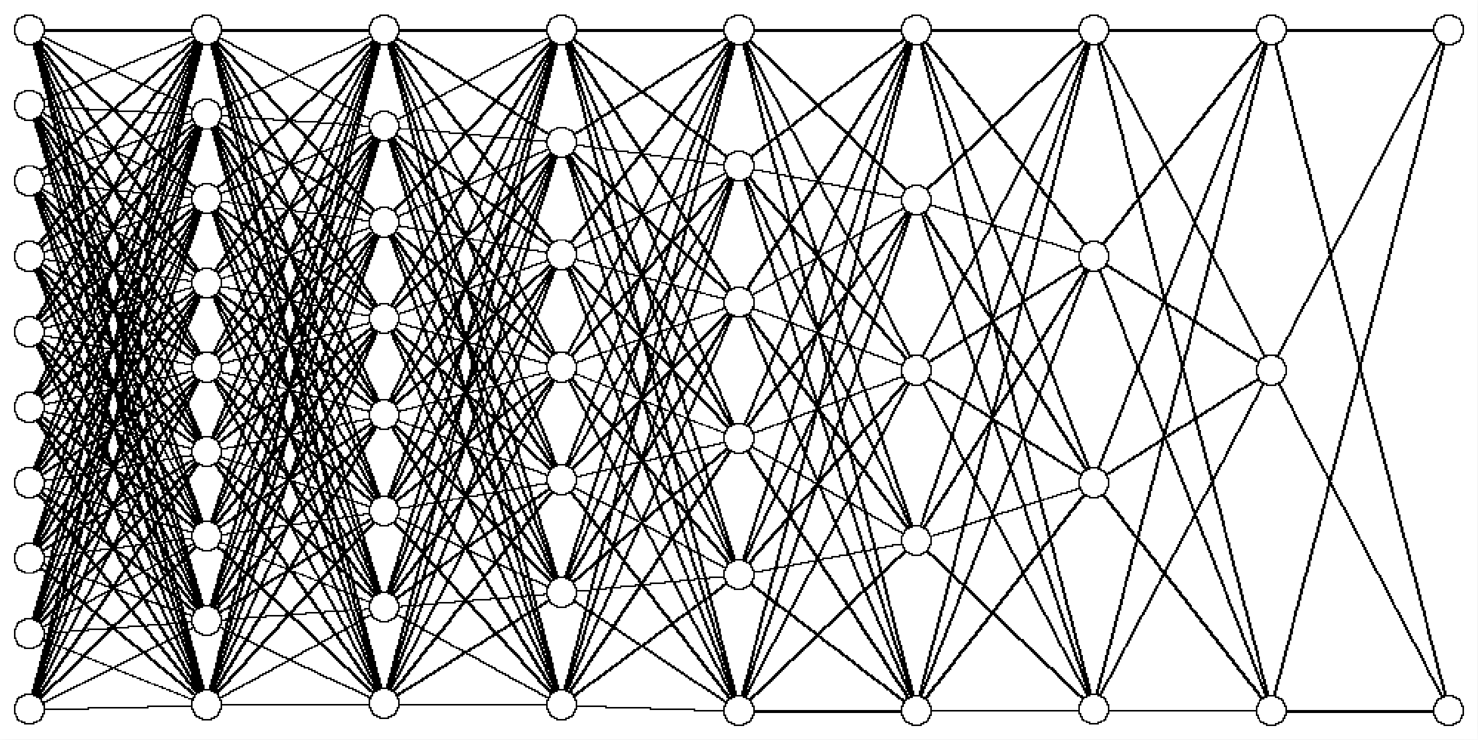




Neural Networks stack nonlinear transformations

- We can build powerful models stacking neurons

- (There are many other details, but this is the core idea)

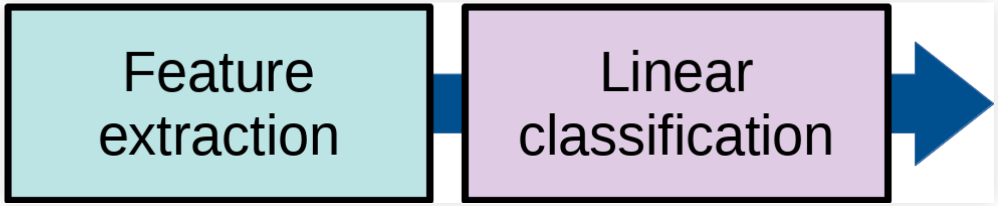


Shallow models

Shallow models

Linear models

- Can only separate linearly separable classifiers
- Require careful (manual) choice of features



Shallow models

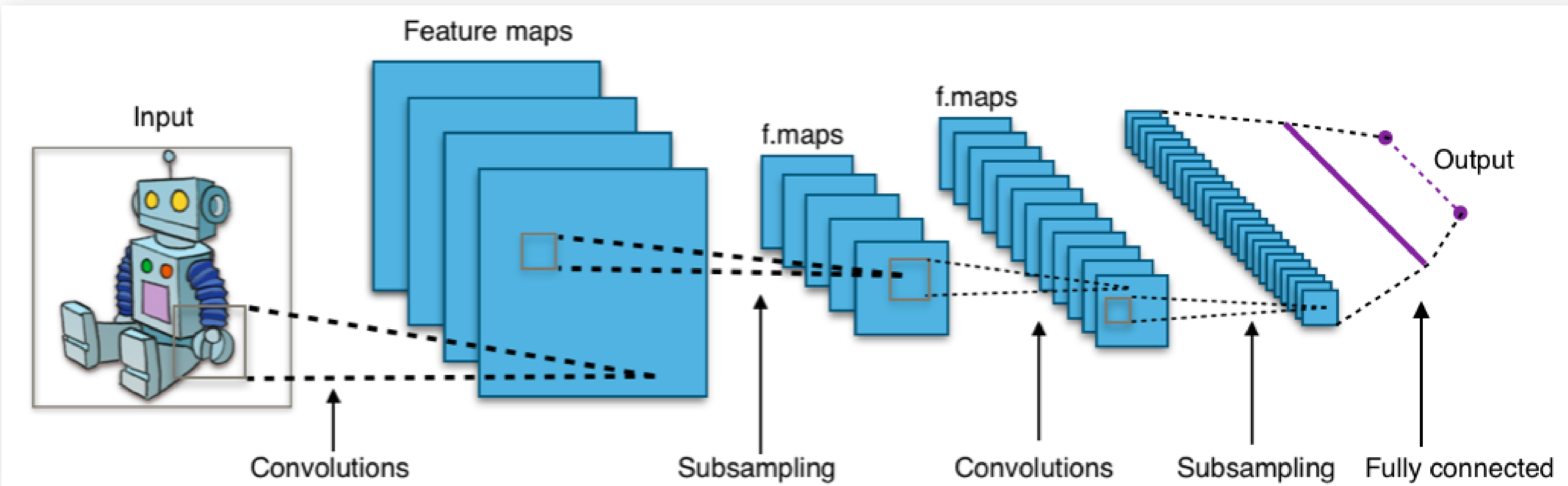
With nonlinear transformation

- In theory, can approximate any function
  - E.g. Gaussian kernels, 1 hidden layer MPL
- But are sensitive to irrelevant information in input, requiring a good choice of features
  - E.g. position and orientation of images, sound pitch, ...

Deep classifiers

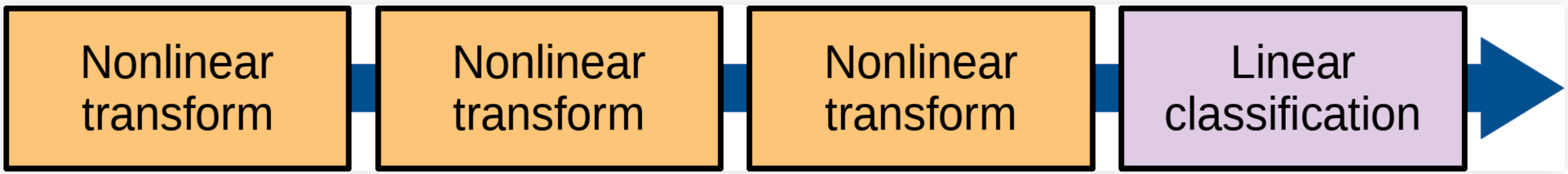
ANN have a nonlinear response because of layers

- Single neurons are linear classifiers, similar to logistic regression
- But deep networks can be extremely nonlinear, combining different representations



ANN have a nonlinear response because of layers

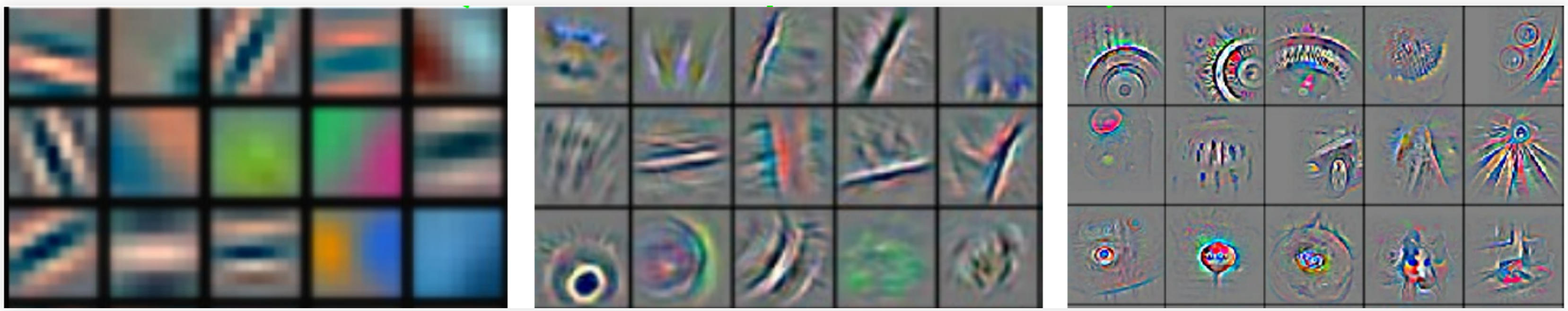
- Single neurons are linear classifiers, similar to logistic regression
- But large networks can be extremely nonlinear, combining different representations



Deep classifiers

ANN have a nonlinear response because of layers

- These layers can find better representations than a single nonlinear transformation and the representations are learned



Zeitler, 2014, Visualizing and Understanding Convolutional Networks

- More efficient representations
  - Instead of one transformation to arbitrarily large space
- Effectively automates feature extraction

Features and Data

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Features and Data

Structured Data: conforms to a (tabular) data model

- Well defined semantics, within some context
  - e.g. business model, hospital, banknotes
- All examples have the same set of attributes
  - With specified relations and allowed values
- Each value "means" the same thing on all examples
  - e.g. Blood pressure, glucose levels, temperature, ...

Features and Data

Structured Data: conforms to a (tabular) data model

- Easy to use in machine learning models:
  - Known attributes
  - Fixed-size inputs
  - Fixed match between attributes of different examples
- Requires human preparation and maintenance
  - Data does not naturally structure itself
- Examples:
  - Client data, seismic events table, gene activities, ...

Semi-structured Data: not a table, but some structure

- Schema contained in the data (self-describing)
  - XML, JSON, NoSQL, Email metadata
- Technical text (with standard terms)
- Also requires some intervention to organize the data model
  - But much semi-structured data is machine generated
- Not trivial to use in machine learning
  - Input size varies
  - Attributes are not the same across examples
- Examples:
  - Access logs, keywords, metadata

Unstructured Data: no predefined structure

- This is most data:
  - Video, email bodies, phone conversations, images, free text documents
- Natural state of most data when generated
  - Before human curation and feature extraction
- Often found within structured data
  - E.g. "comments" field in database, or call-center phone recordings
- Difficult to use with classical machine learning

Features and Data

Classical approach

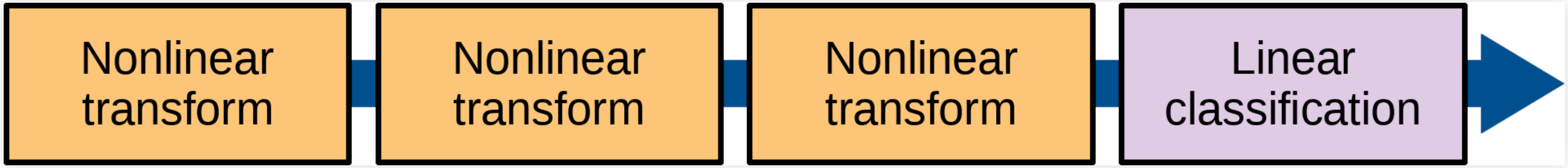
- Explicit conversion to structured format, with specific feature extraction methods
  - Example: Text mining
  - Categorization, clustering, concept extraction, sentiment analysis, ...
  - Example: Image segmentation
  - Edge detection, thresholding, histograms, ...
- Labour-intensive, takes years to perfect feature extraction methods
- Many particular tricks and fine tuning required

Features and Data

Artificial Intelligence

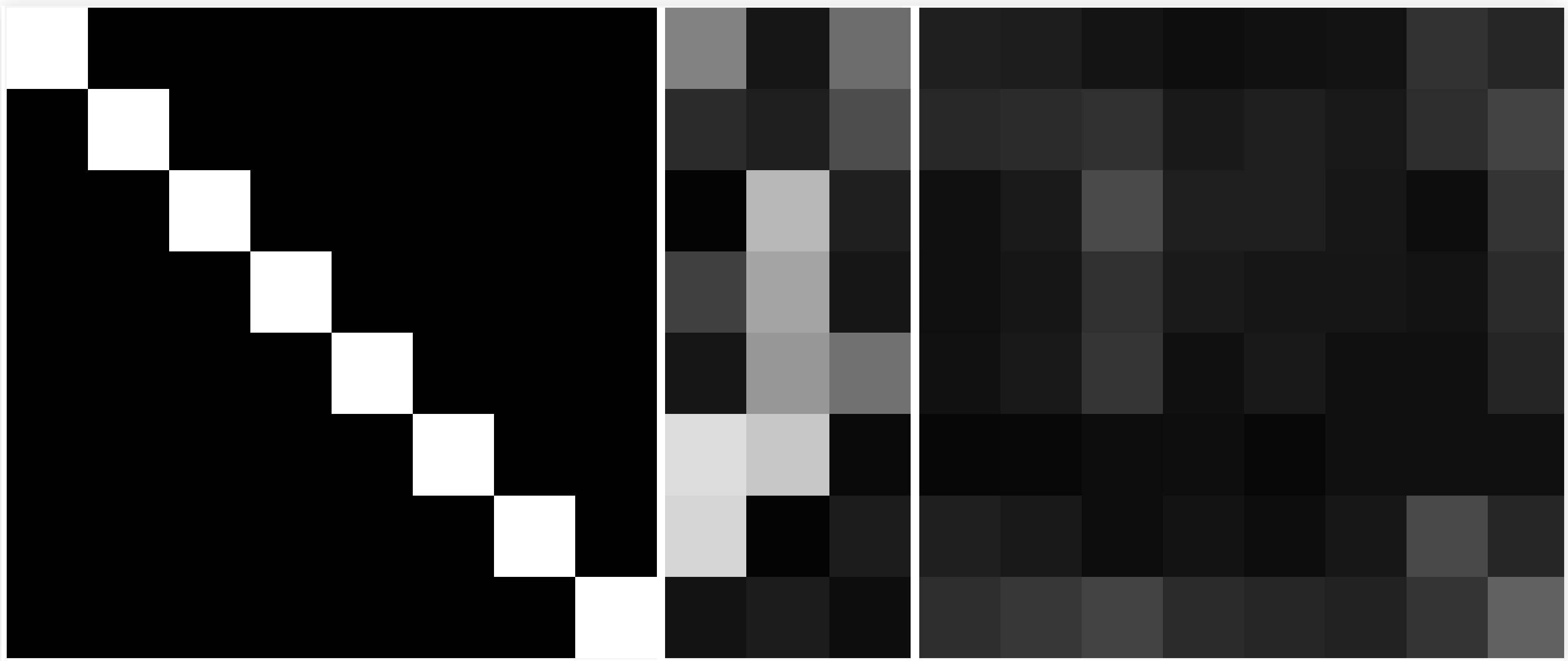
- Goal in AI: have the computer solve the problems, not us
- Custom-made feature extraction is not good for adapting quickly
  - Need to respond to changing conditions
  - Adversarial systems (credit fraud, fake social media accounts)

Deep learning solves these problems:



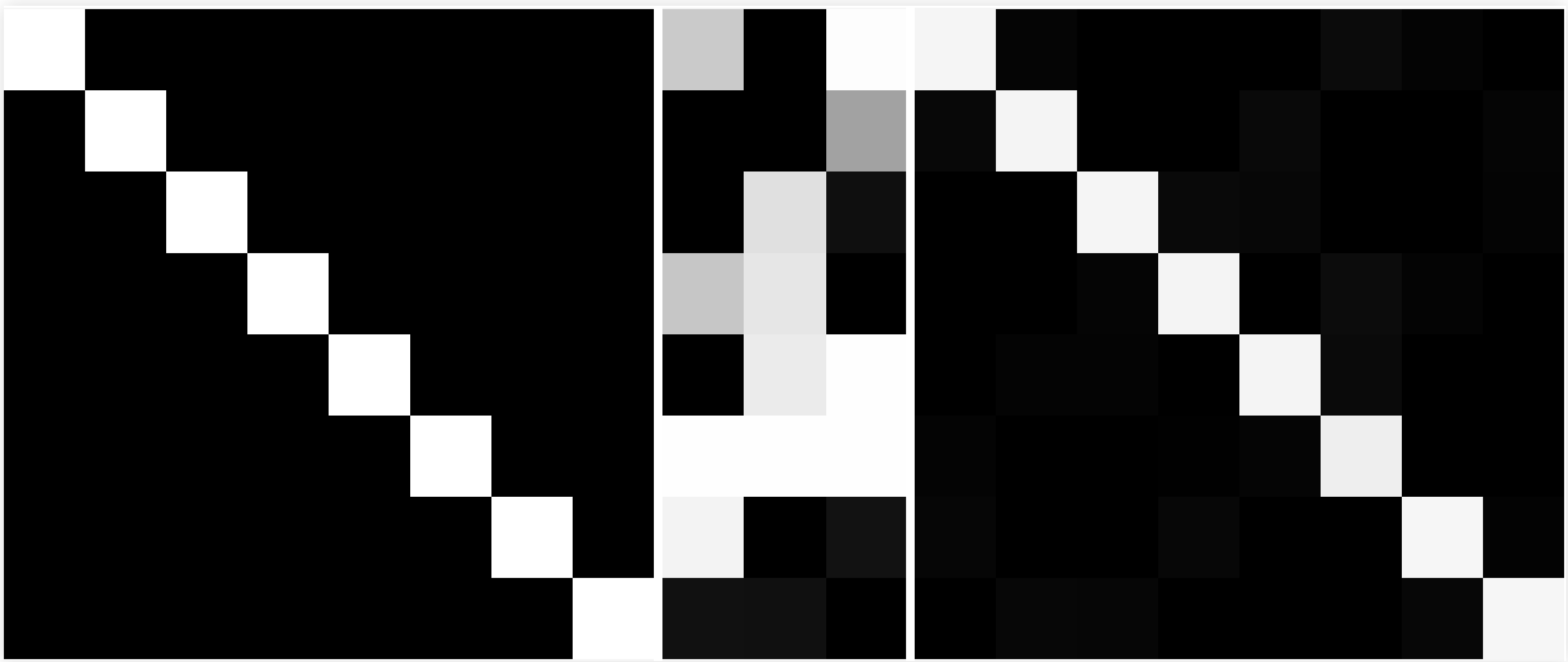
Deep Learning

- Neural networks can learn useful representations
  - With or without labeled examples
- Mitchell's autoencoder, hidden layer of 3 neurons



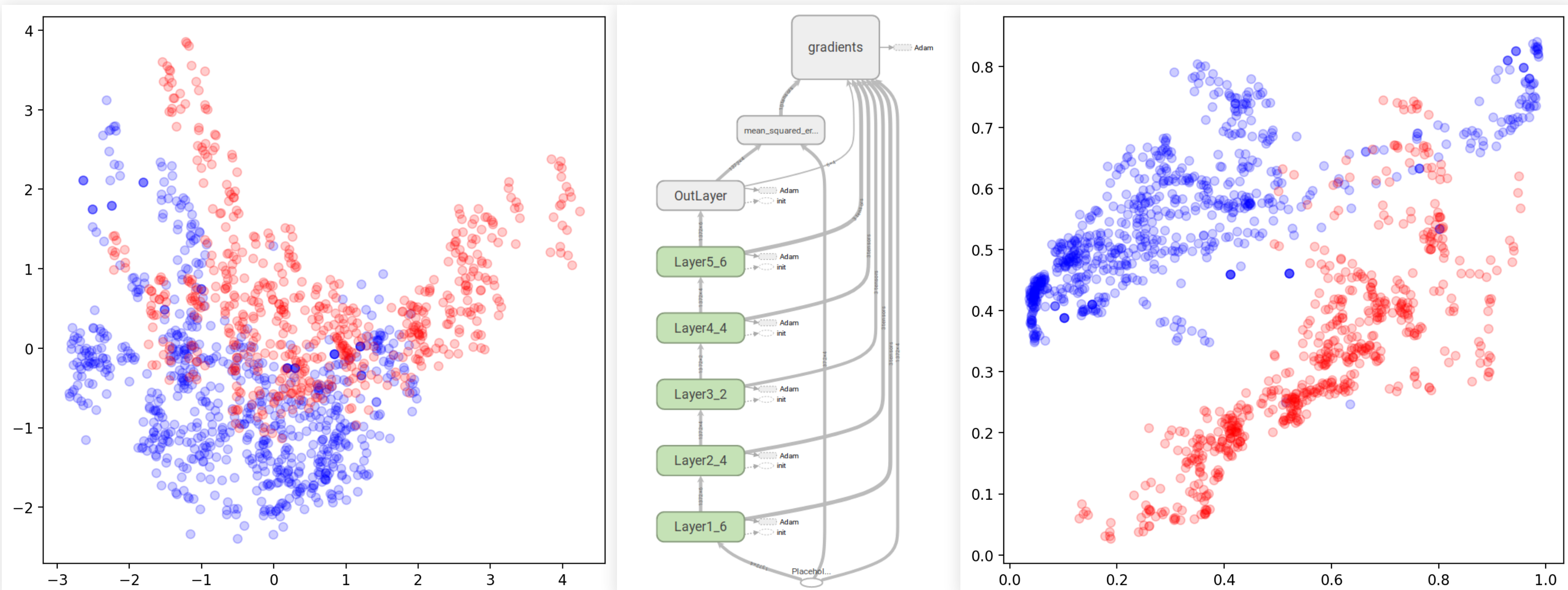
Deep Learning

- Neural networks can learn useful reprsentations
  - With or without labeled examples
- Mitchell's autoencoder, hidden layer of 3 neurons



Deep Learning

- Neural networks can be used to recode data even without labels
- PCA vs autoencoder (4),6,4,2,4,6,4 UCI bannknote dataset



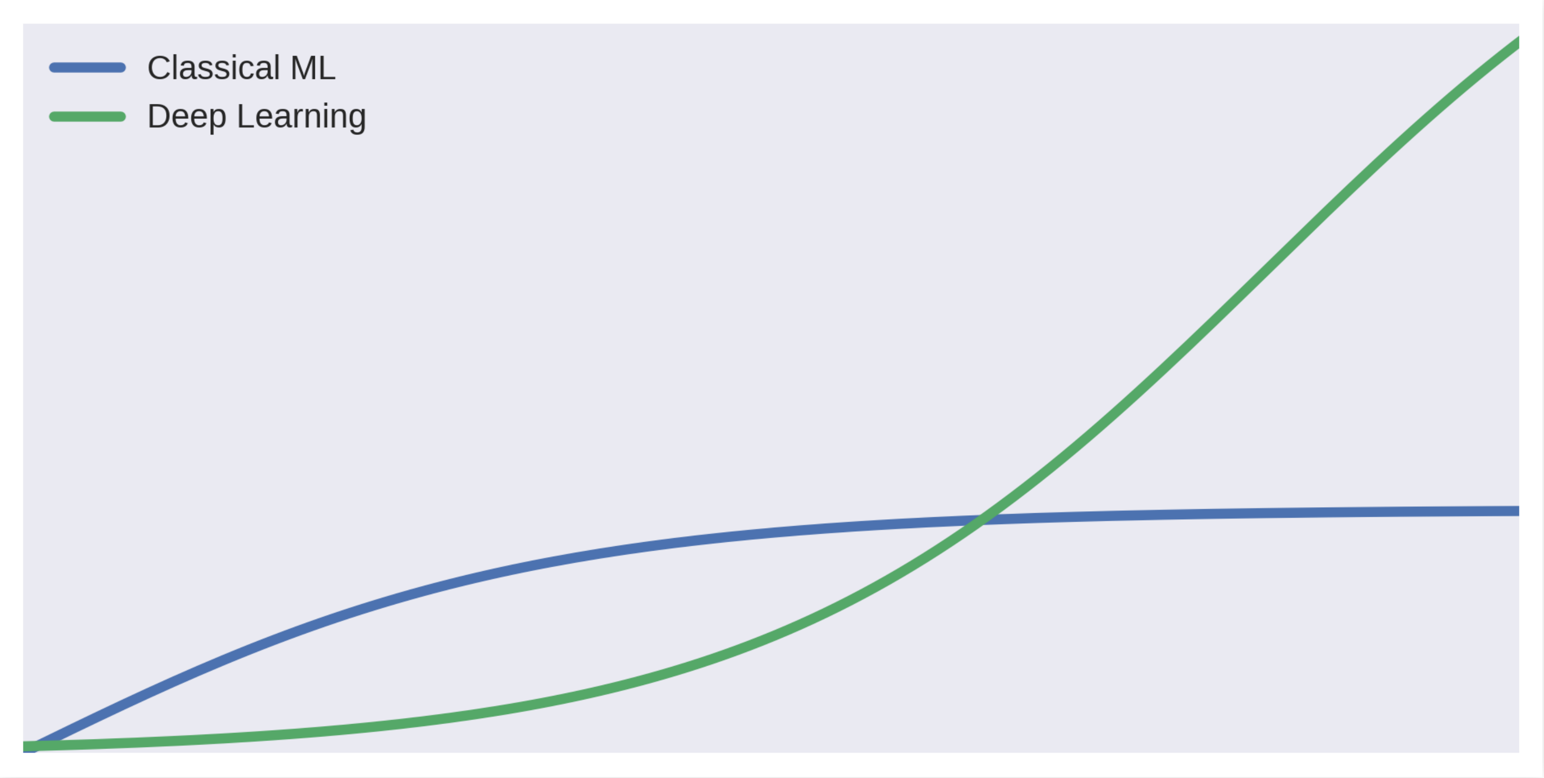
Deep Learning

- Deep models learn to extract the best features
  - Even from unstructured data (e.g. convolution networks for images)
- Deep models can learn to encode data in useful ways
  - Even without labels (autoencoders)
- This is done automatically by the model, using the data

The Triumph of Deep Learning

Deep models are very powerful

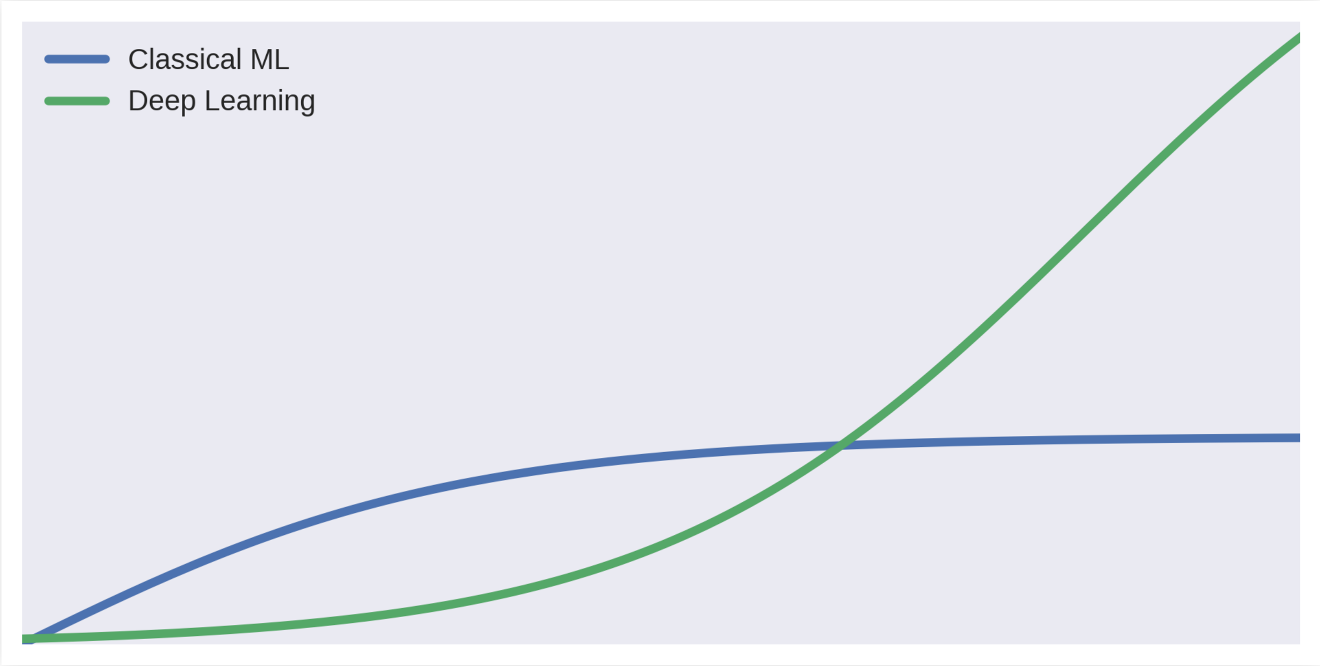
- Not good with less data (overfitting), but excellent for big data



Why now?

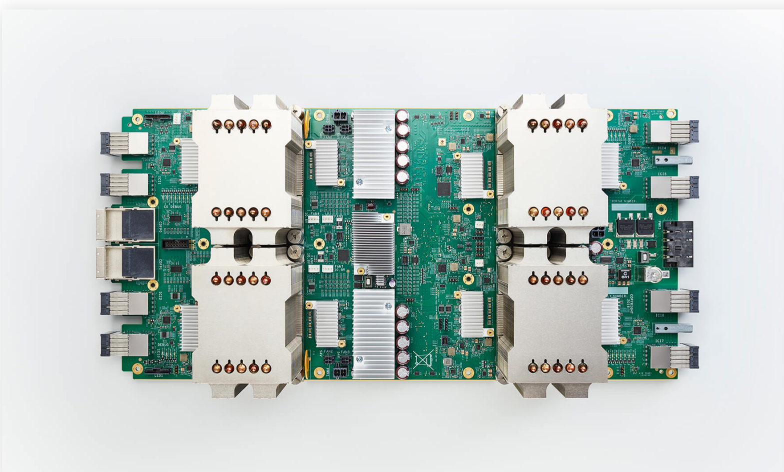
Example: computer vision

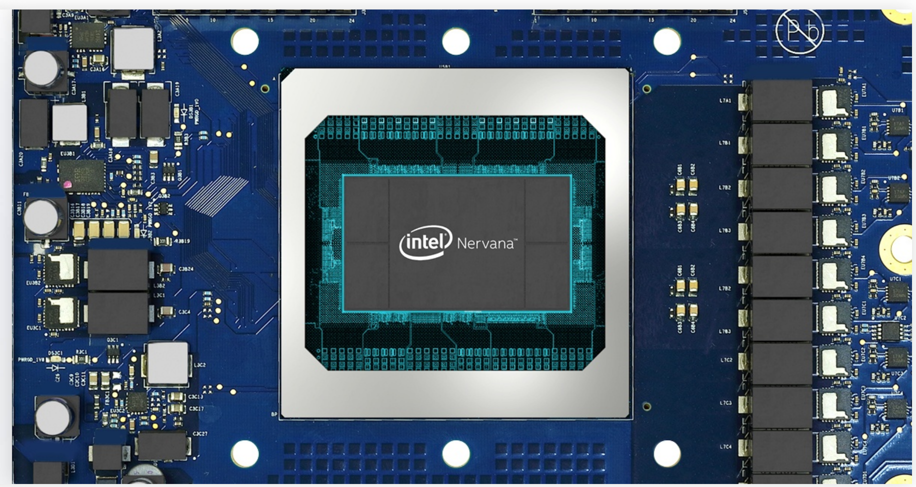
- 2006: Caltech 101 dataset, ~50 images per category (40-800)
  - Classical CV models, 26% error
  - Also, slow computers



Better methods and hardware

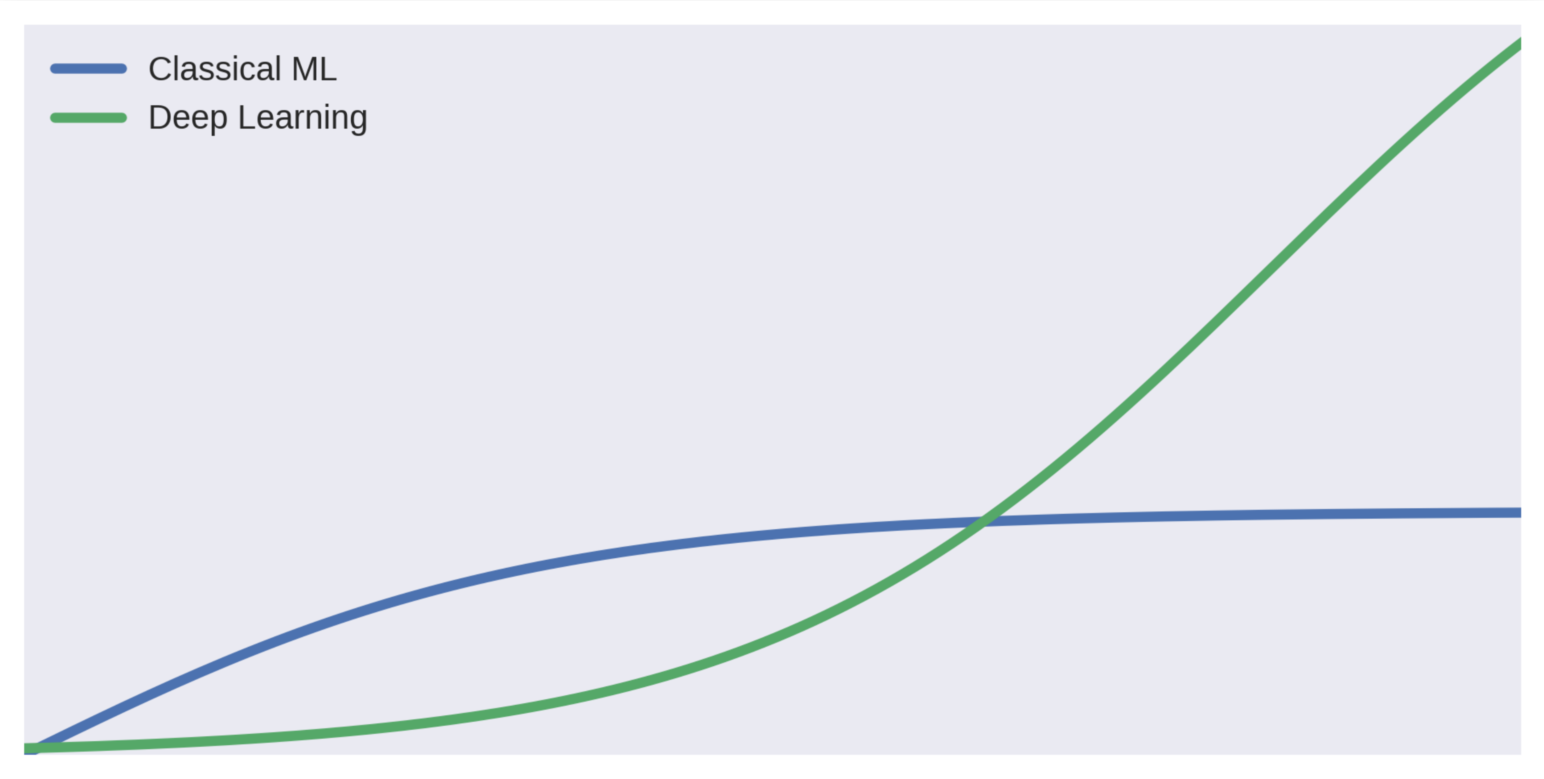
- Improvements in algorithms (examples)
  - 2007: Hinton, pre-training of deep feedforward ANN
  - 2011: Bengio, rectified linear unit (ReLU)
- New hardware: GPGPU
  - Google Tensor Processing Unit
  - Intel Nervana Neural Network Processor





Larger data sets

- 2012: ImageNet dataset (1.2 M images, 1000 categories), GPGPU
  - Large convolution networks became dominant



Why now?

Example: computer vision

- 2012: ImageNet dataset (1.2 M images, 1000 categories)
- More computing power (GPGPU)

Deep CNN became dominant:

- AlexNet [Krizhevski, Sutskever, Hinton 2012], 15% top-5 error
- OverFeat [Sermanet et al. 2013], 13.8% error
- VGG Net [Simonyan, Zisserman 2014], 7.3% error
- GoogLeNet [Szegedy et al. 2014] 6.6% error
- ResNet [He et al. 2015] 5.7% error

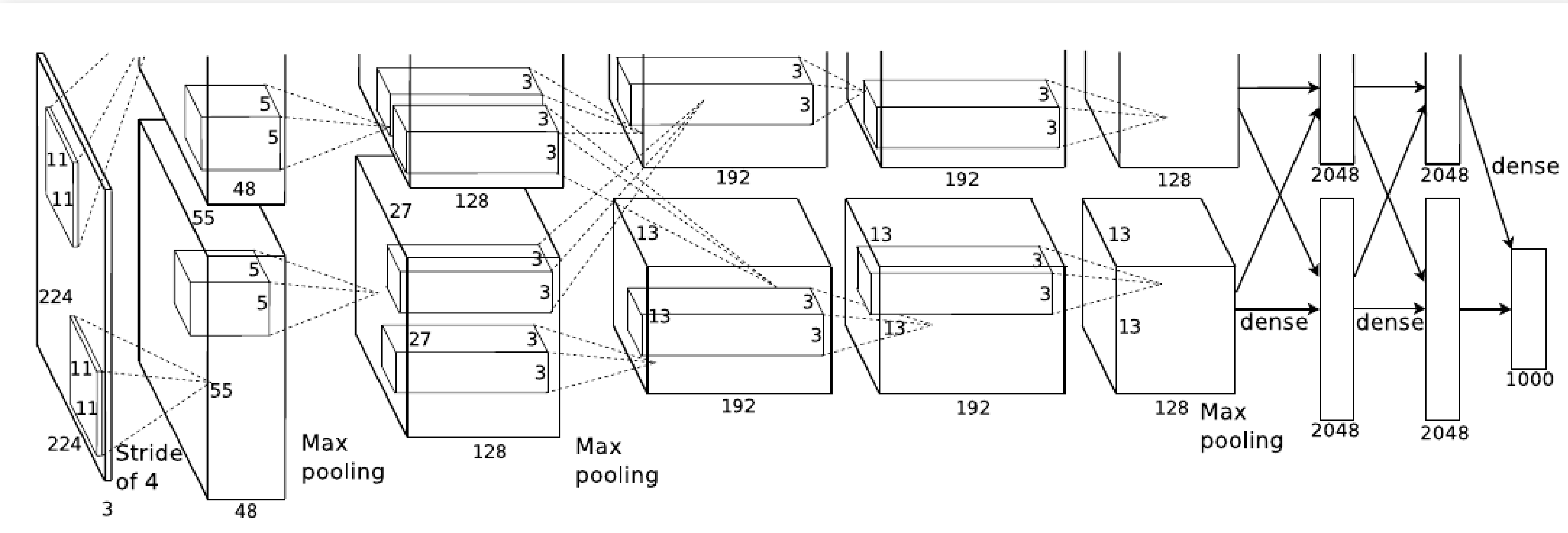
(Yann Le Cun, CERN, 2016)

Examples

ImageNet Classification with Deep CNN

- Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012

- Used 2 GPU (NVIDIA GeForce GTX 580, 3GB), 2 parallel streams
- 15.4% top-5 error (second best, classical CV, 26.2%)

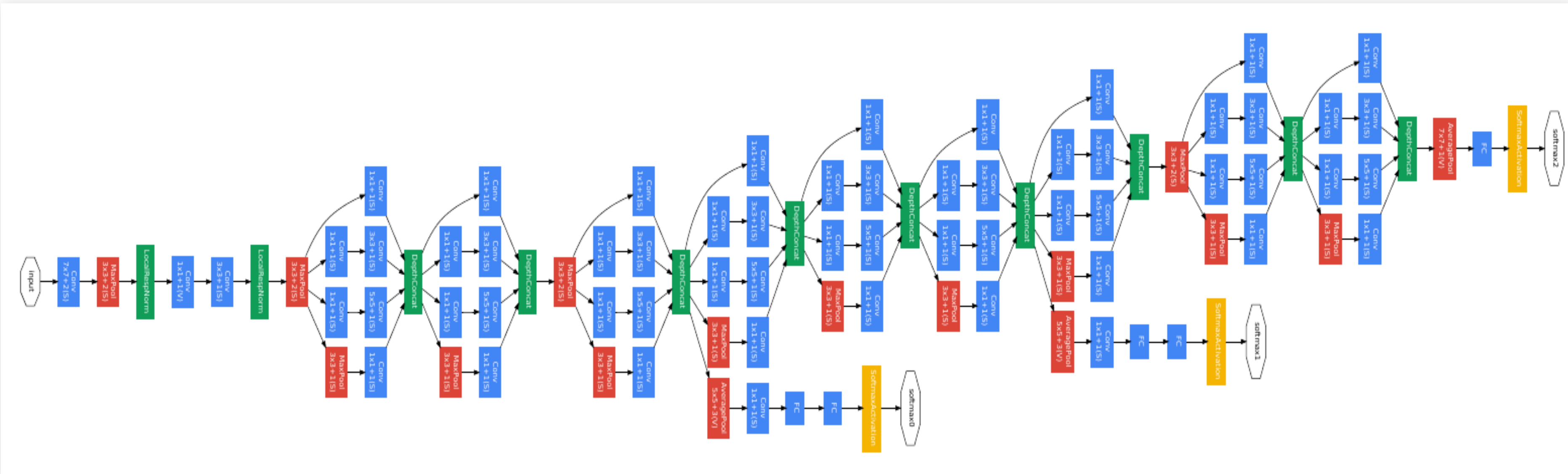


Inputs: 150,528 253,440 186,624 64,896 64,896 43,264 4096 4096 1000

GoogLeNet

Going Deeper with Convolutions (GoogLeNet)

- Szegedy et. al. 2015, 6.67% top-5 error in ImageNet



22 layers with parameters (+5 pooling) in 100 "inception" blocks

Images and Deep Learning

Examples for image problems

- VGG-19: 75% top 1, 92% top 5 accuracies on ImageNet
- ResNet: 85% top 1, 97.7% top 5 accuracies on ImageNet
- ENet: real time semantic segmentation
- ShuffleNet: Imnage classification on mobile devices

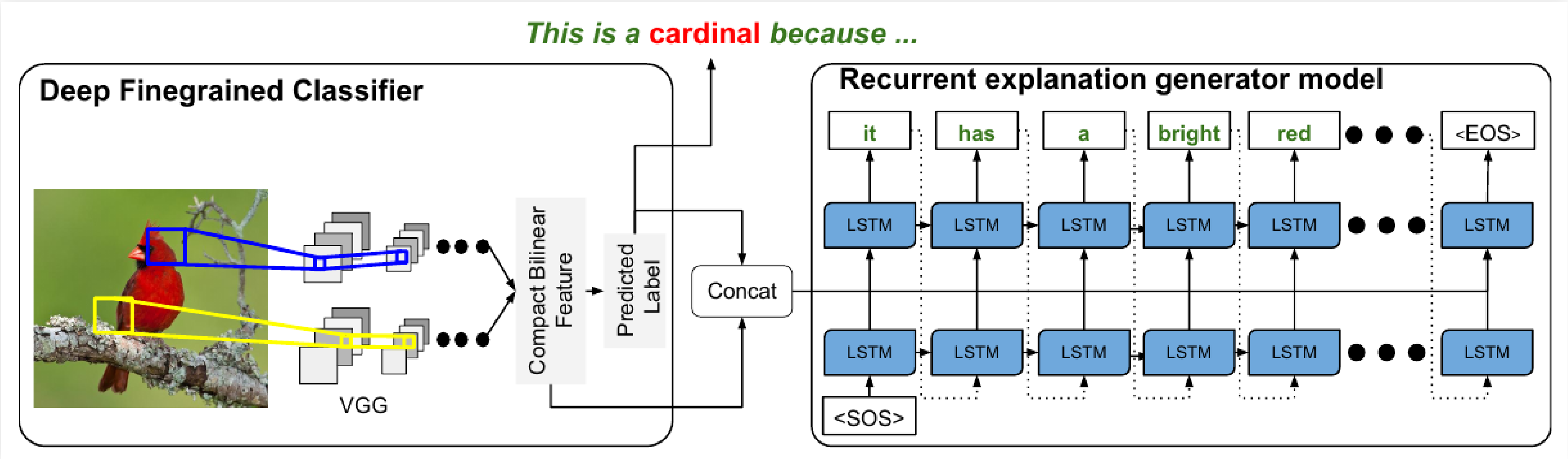
Generating Visual Explanations

- Hendricks et. al., 2016, generate visual explanations for images:

"be class discriminative and accurately describe a specific image instance"

- Data: Caltech-UCSD Birds-200-2011
  - 12k images, 200 categories, attributes and class labels
  - 5 sentences for each image (Reed et. al. Amazon Turk)
- Baselines and metrics:
  - Description (from images), definition (from labels) and humans
- Explanation, minimize:
  - Relevance loss: probability of correct words
  - Discriminative loss: reinforcement $R_D(\tilde{w}) = p(C|\tilde{w})$

Visual Explanations

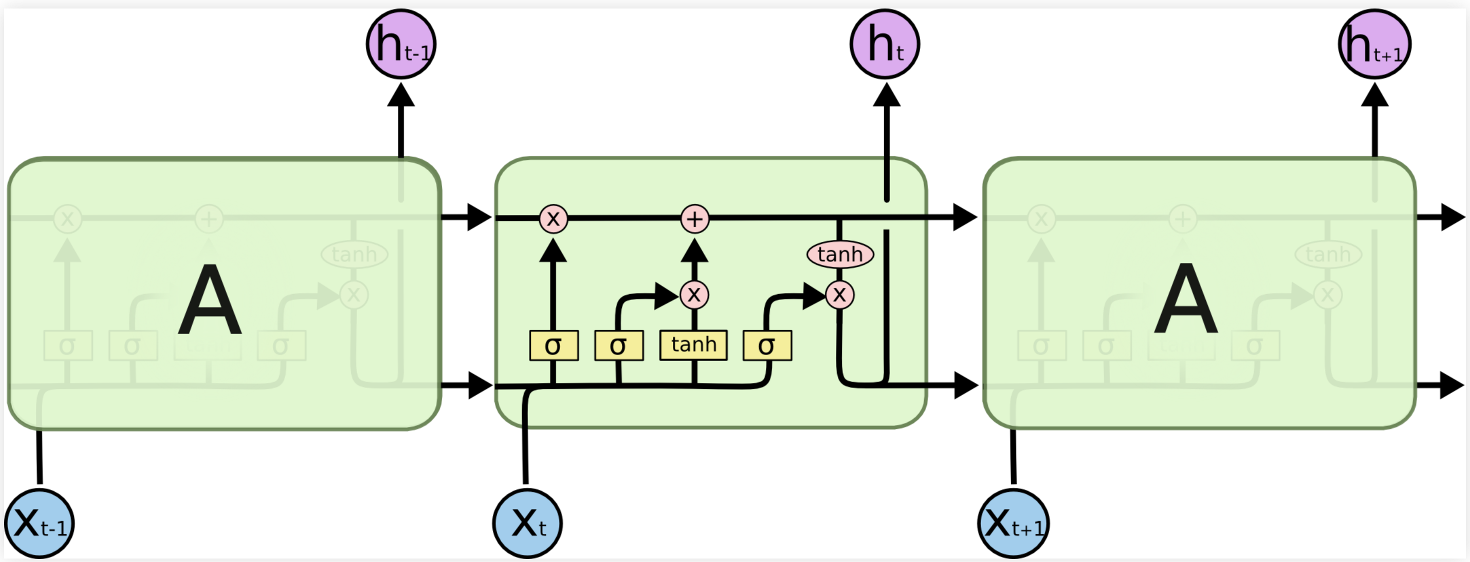


- Compact bilinear model for image classification
- 2 LSTM: Long Short Term Memory networks for sentence generation
  - One receives a w_{t-1} (or start) as input, outputs l_t
  - The other receives l_t and image feature and produces $p(w_t)$
  - Then w_t is generated by sampling $p(w_t)$

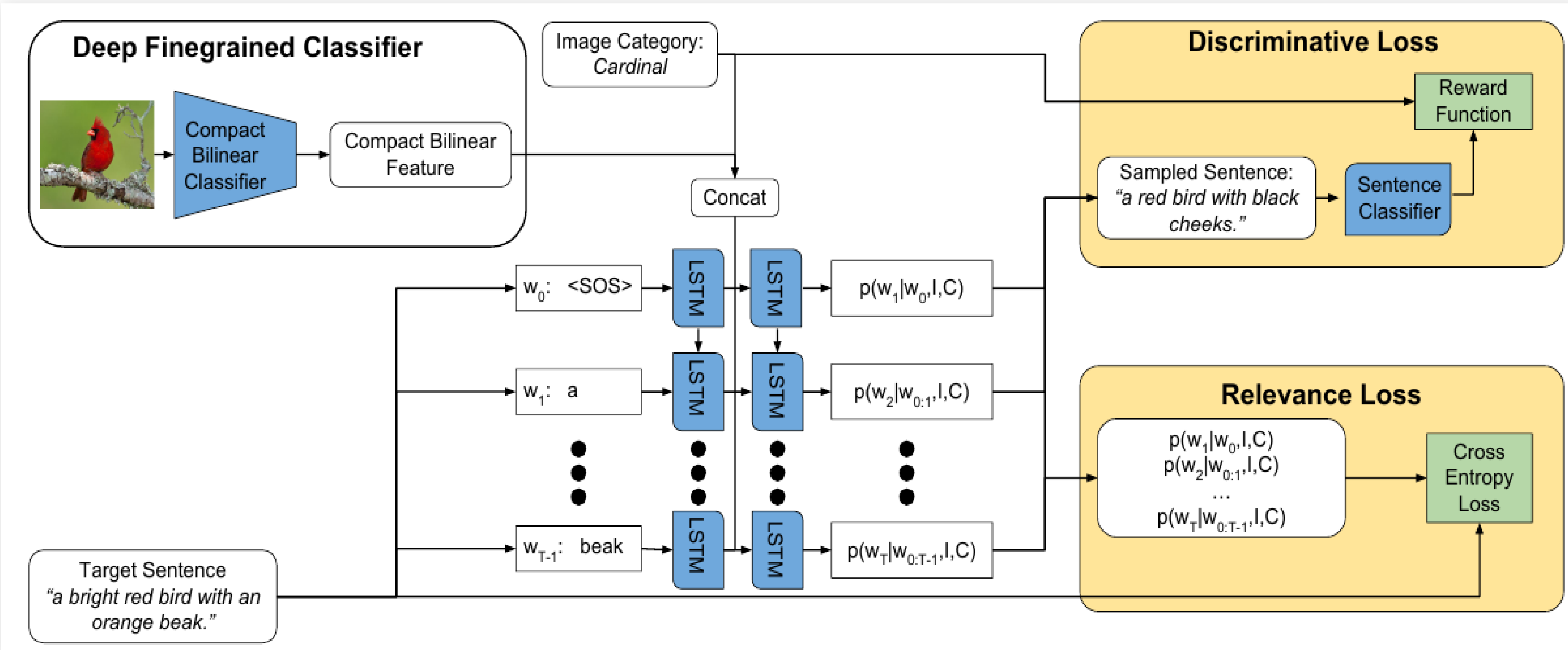
Visual Explanations

LSTM: Long Short Term Memory

- Multiplication to block (0 or 1) and forget
- Sum to force new values (remember, or not, $-1 \leq tanh \leq 1$)
- Decide what to output based on memory and current input



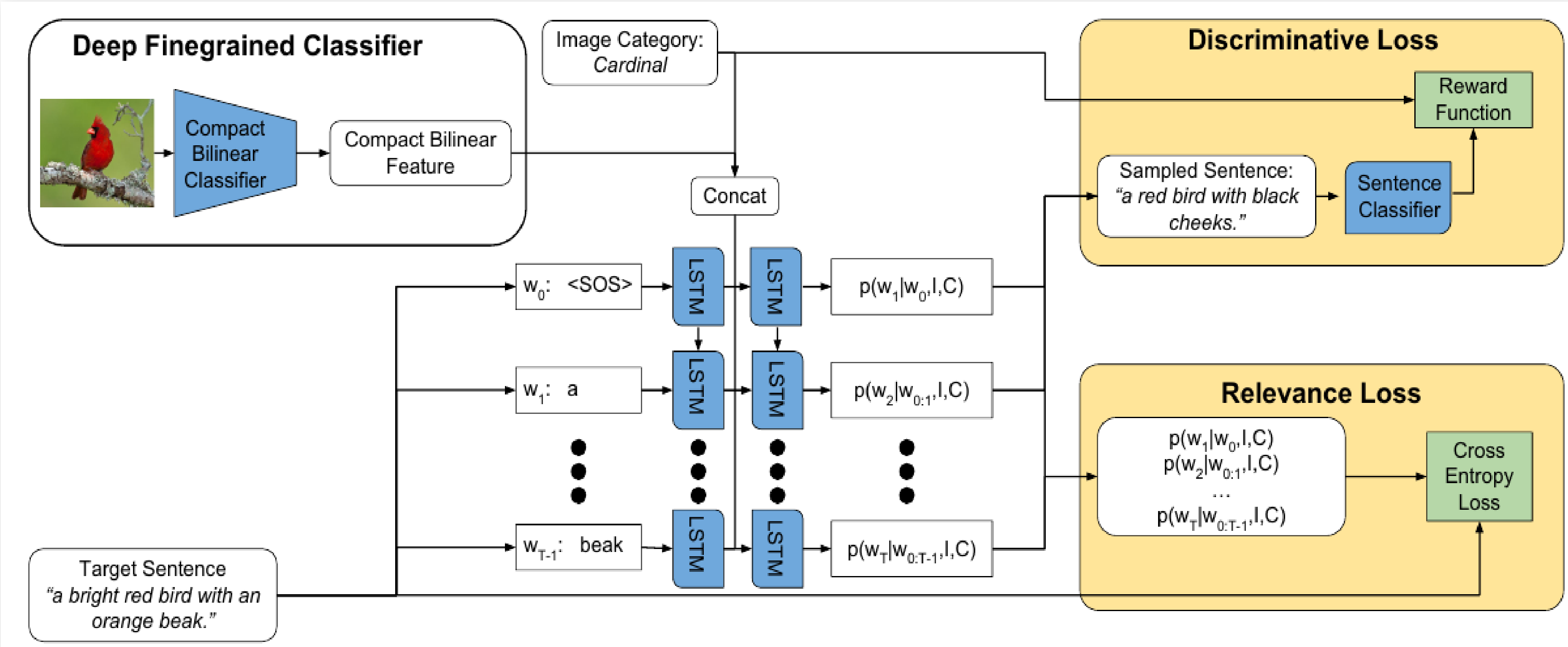
Visual Explanations



- Relevance loss:
  - p of word given previous, image and class label

$$L_R = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} \log p(w_{t+1} | w_{0:t}, I, C)$$

Visual Explanations



- Discriminative loss:
  - Gradient (MC) weighted by reward

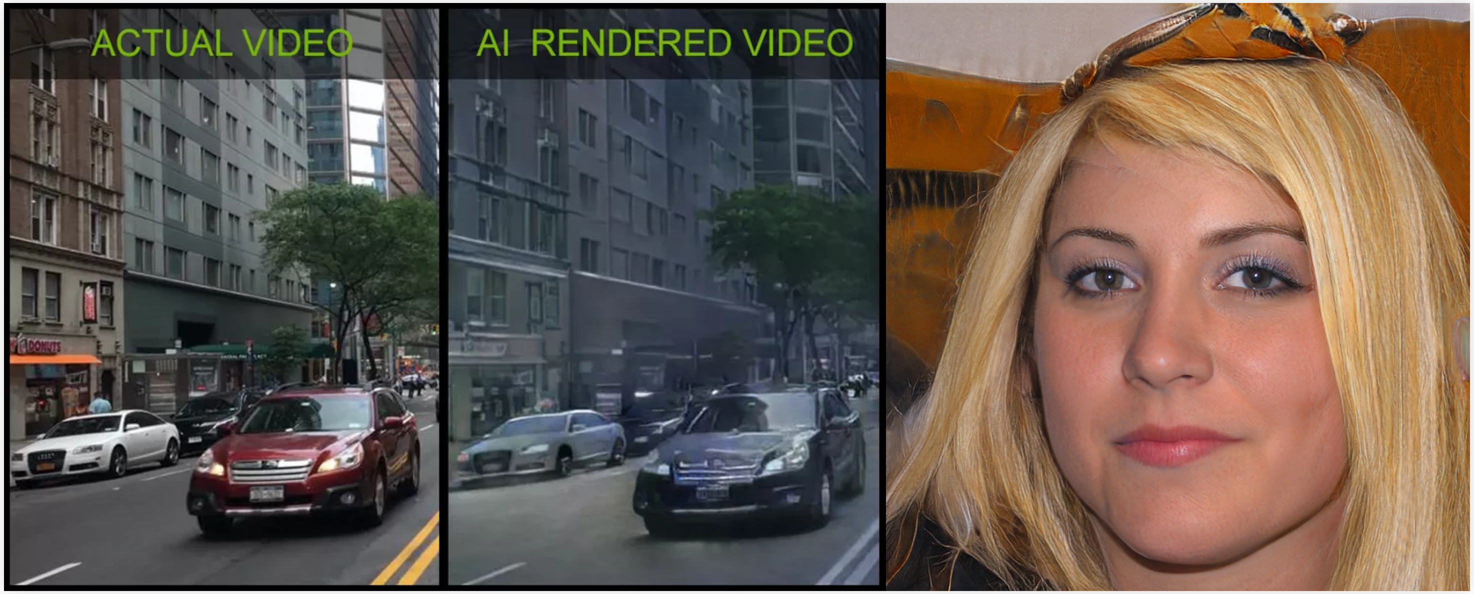
$$R_D(\tilde{w}) = p(C|\tilde{w})$$

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

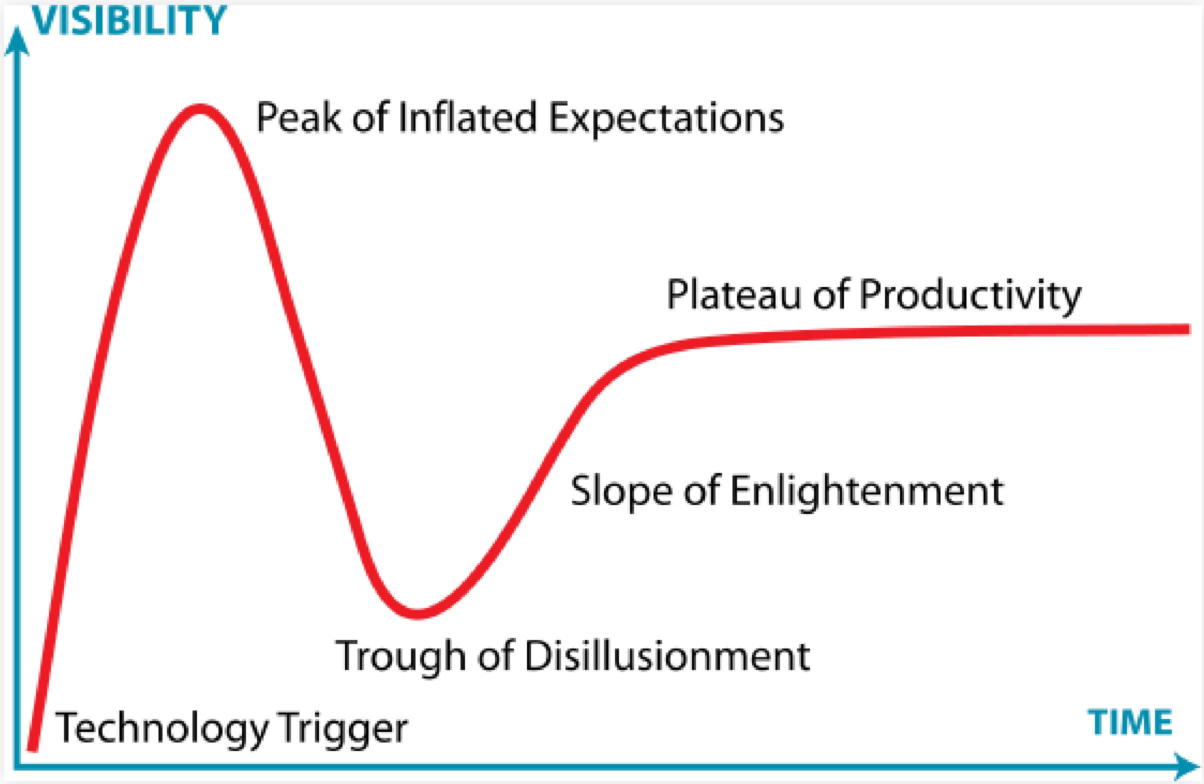
Results, example

Other applications of Deep Learning

- Speech recognition, translation, image classification and generation, video generation



Are we over the "Hype Cycle"?



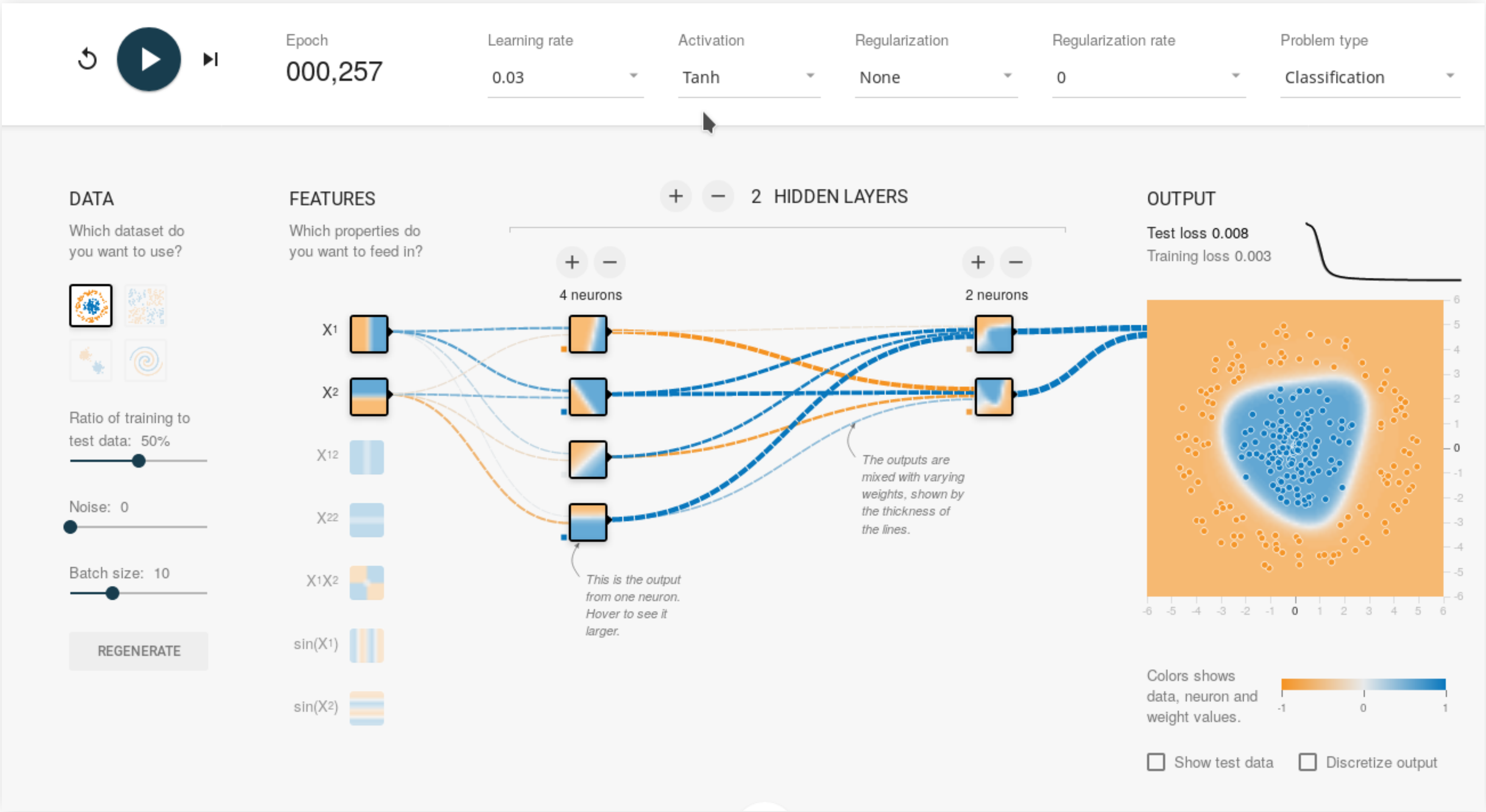
Jeremykemp, CC BY-SA 3.0

Tensorflow Playground

Tensorflow Playground

ANN demo, fun to play with

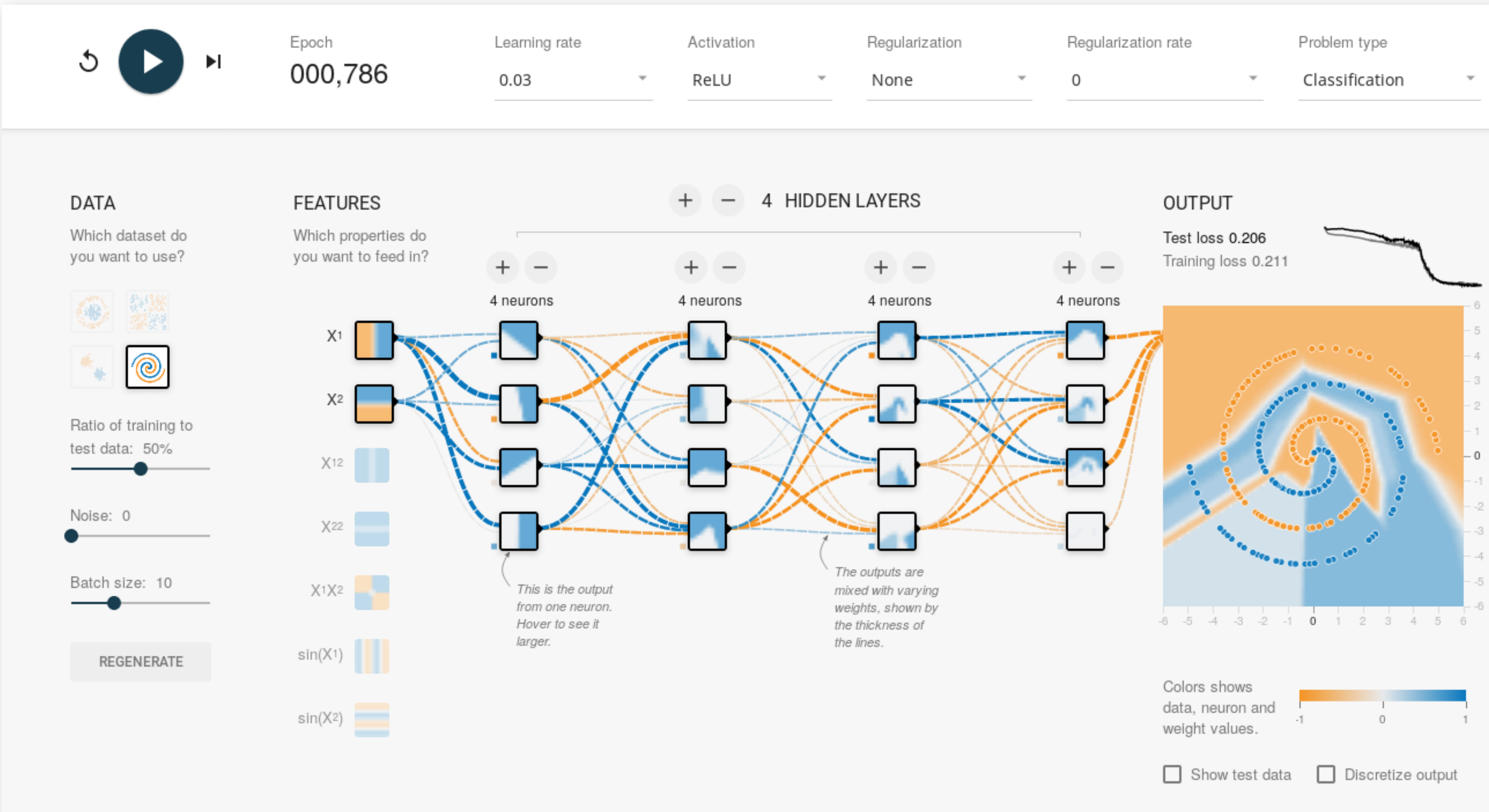
- Try out different nets and data sets: http://playground.tensorflow.org



Tensorflow Playground

ANN demo, fun to play with

- Try out different nets and data sets: http://playground.tensorflow.org



FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Homework

Setup for next classes

- Install (or update) Anaconda
  - https://www.anaconda.com/distribution/
- Install tensorflow 2 (choose gpu if you have NVIDIA):

```
conda install tensorflow-gpu
[or]
conda install tensorflow
```

- Setup Google Colab:
  - Login to Google (GMail, ...) and go to https://colab.research.google.com
  - Create a Jupyter notebook and execute this:

```
from google.colab import drive
drive.mount('/content/drive')
%tensorflow_version 2.x
%cd "drive/My Drive/[your working folder]"
```

Summary

Summary

- Artificial Neural Networks and backpropagation
- Deep learning: automated feature extraction
- Why now?
  - Alorithms, hardware and data
- Examples

Further reading:

- Goodfellow, chapter 5 and beginning of 6 (sections 6.1 and 6.2)
- Skansi, Introduction to Deep Learning, Sections 4.1 through 4.6