Deep Learning

Robin Devooght
PhD Student @ IRIDIA

Automatic description of images

"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."

"girl in pink dress is jumping in air."

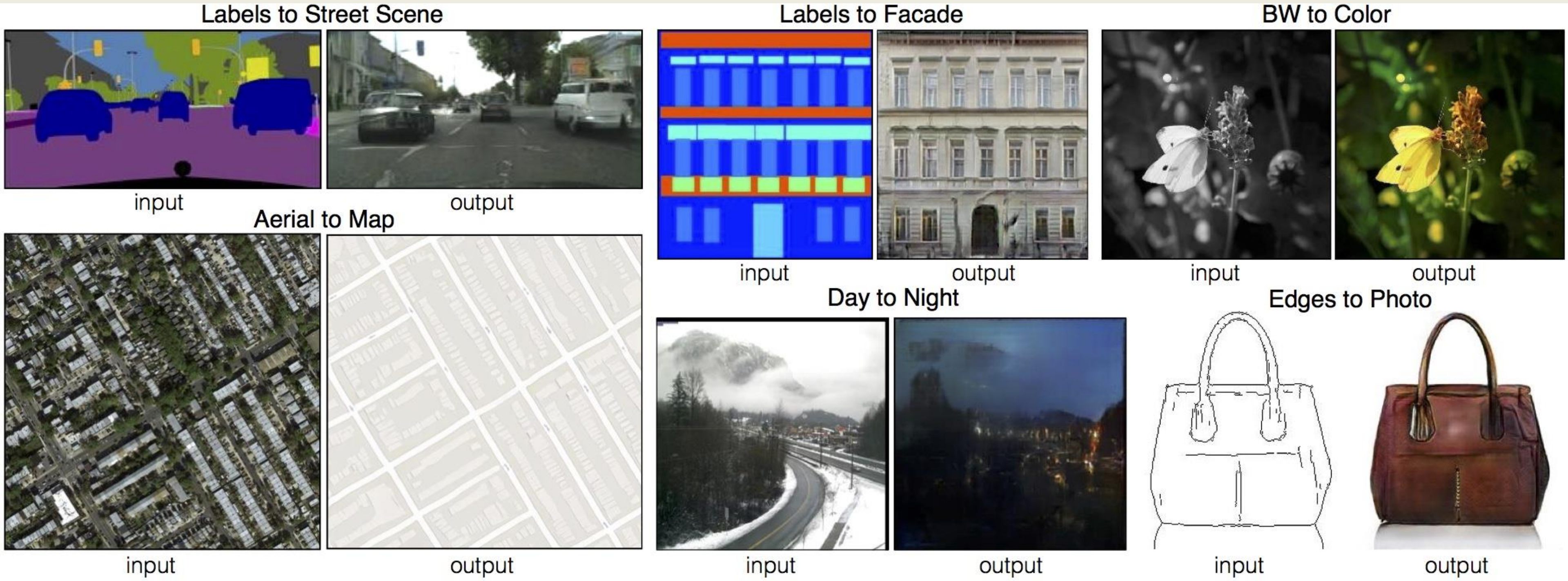
"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

Andrej Karpathy and Li Fei-Fei, http://cs.stanford.edu/people/karpathy/deepimagesent/

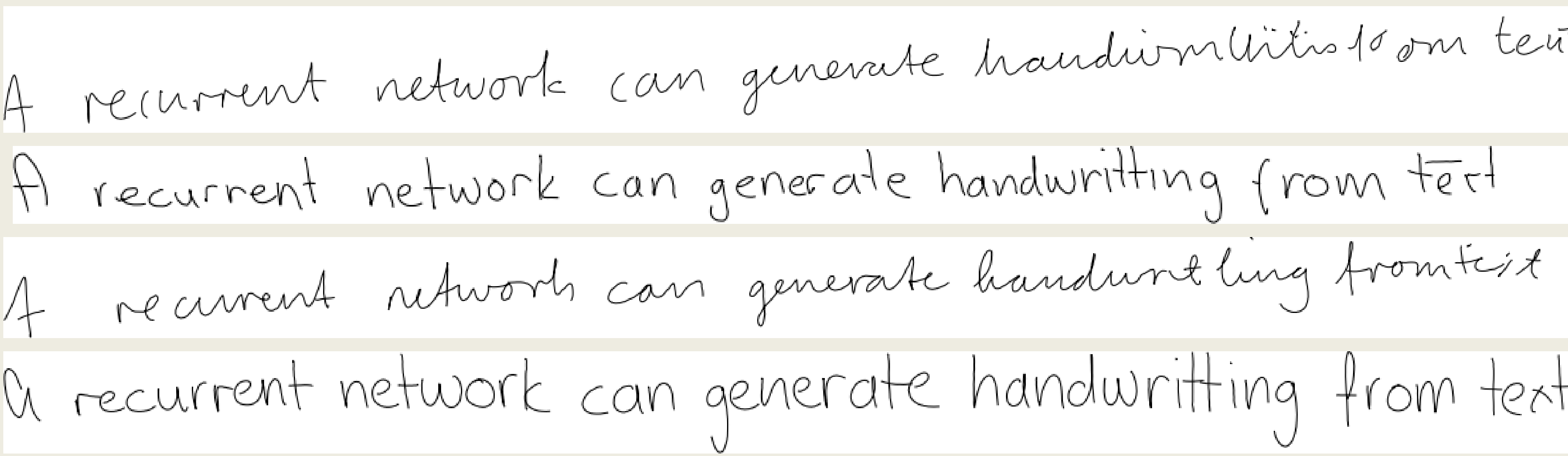
Image to image translation



Philip Isola et al. Image-to-Image Translation with Conditional Adversarial Networks
Demo: https://affinelayer.com/pixsrv/index.html

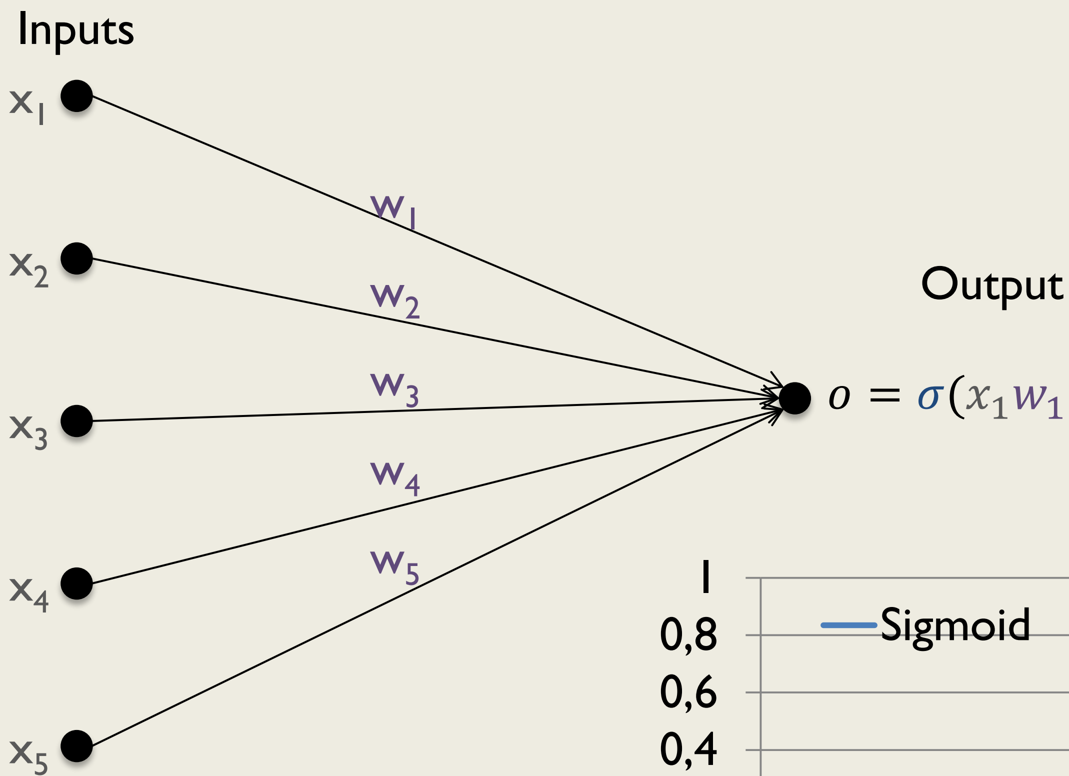
Handwritting generation

A recurrent network can generate handwritting from text

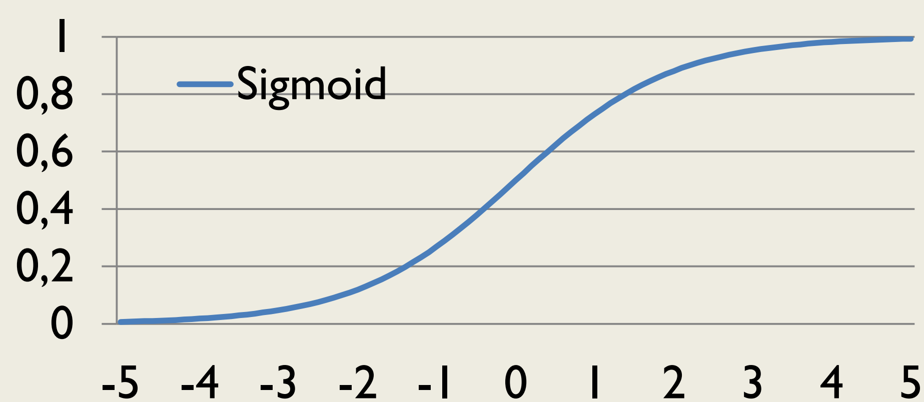


Alex Graves, https://www.cs.toronto.edu/~graves/handwriting.cgi

Deep Learning = Neural Networks

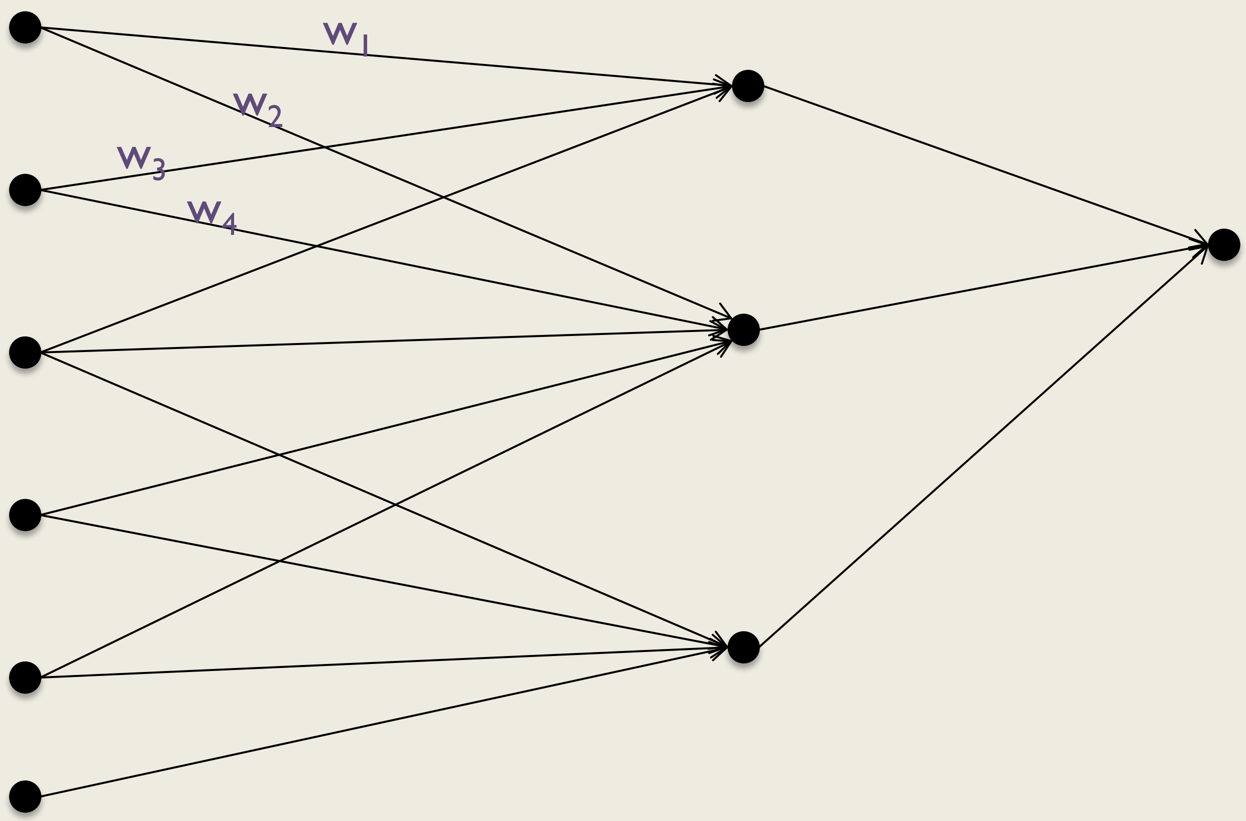


$$o = \sigma(x_1 w_1 + x_2 w_2 + \cdots + x_5 w_5)$$



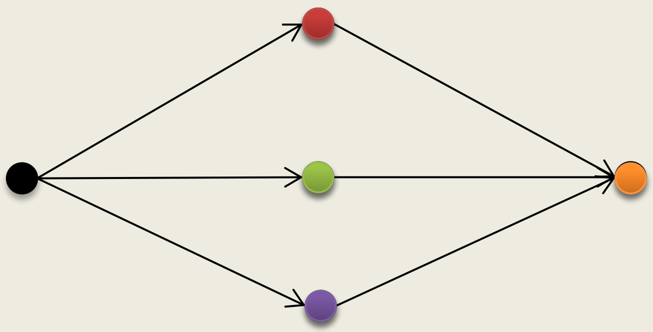
Deep Learning = Neural Networks

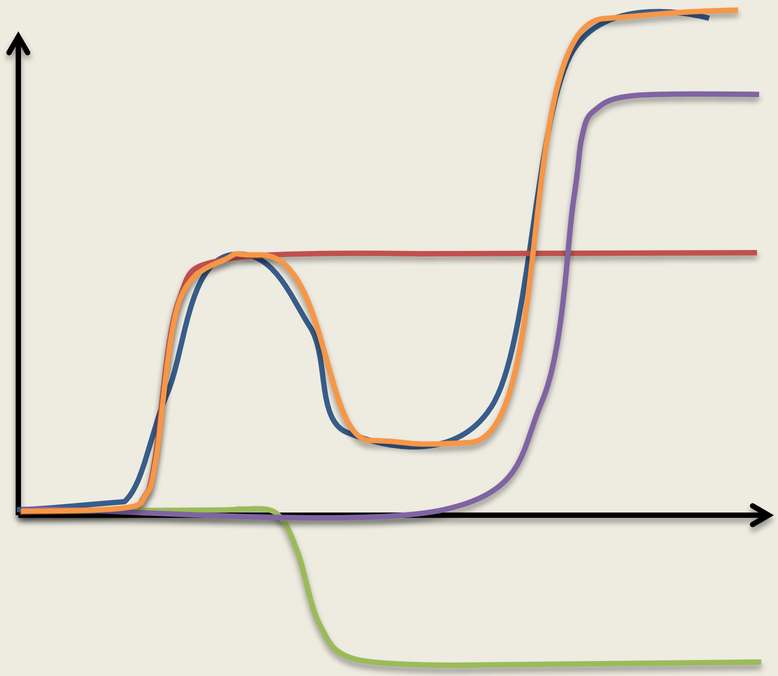
Chaining neurons makes a neural network



Goal: learning the weights

Neural Networks can approximate any function





Original proofs:
Cybenko., G. (1989) "[Approximations by superpositions of sigmoidal functions]"
Kurt Hornik (1991) "[Approximation Capabilities of Multilayer Feedforward Networks]"

Good explanation:
Michael Nielsen, http://neuralnetworksanddeeplearning.com/chap4.html

What’s new in Deep Learning ?

- Larger datasets
- More complex networks
- Faster Hardware
- Accumulation of knowledge

Good models need large datasets

IMAGENET

14M images
30k categories

http://www.image-net.org/

Google books N-grams

2,2TB of text

https://aws.amazon.com/fr/datasets/google-books-ngrams/

MILLION SONG DATASET

1M songs
Audio features,
Labels, lyrics, etc.

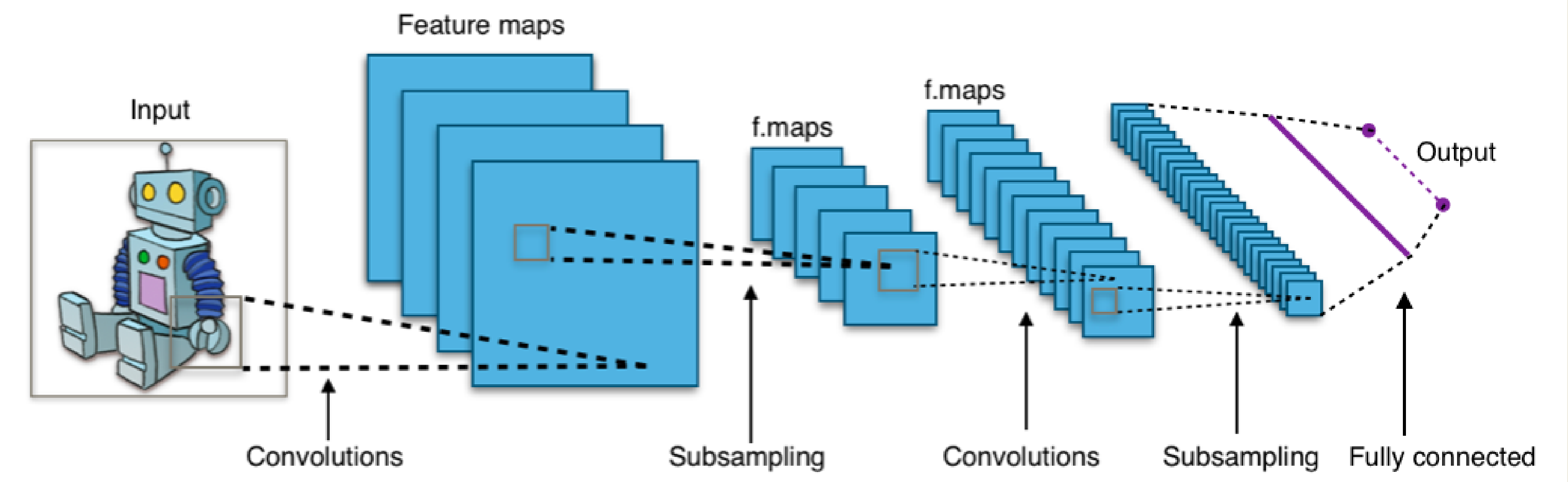
https://labrosa.ee.columbia.edu/millionsong/

Find more datasets at https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research

Beyond the Multi-Layer Perceptron

You can learn any function with one hidden layer,
but it's not the best way to do it

Convolution layer for images:



Beyond the Multi-Layer Perceptron

Gated memory for sequences:

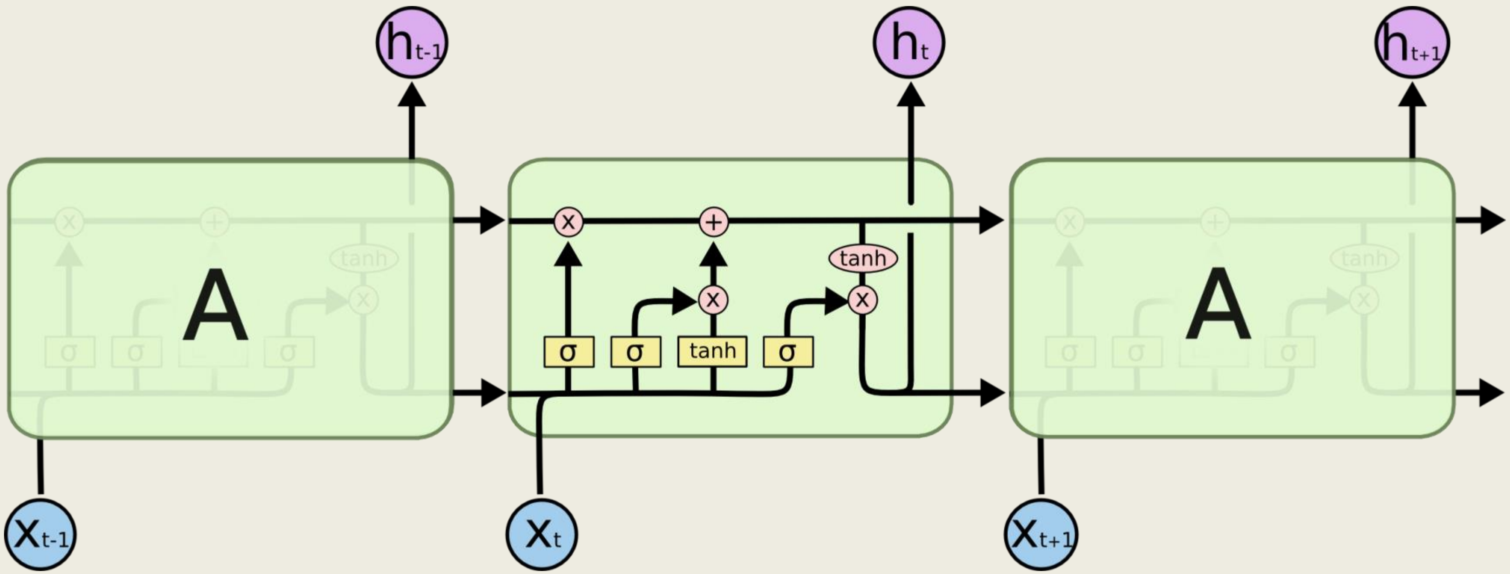


Image by Chris Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
(great explanation of modern recurrent neural nets)

The Neural Network Zoo: http://www.asimovinstitute.org/neural-network-zoo/

Training is faster on new hardware

Large datasets + complex models

⬇

Speed is important

GPUs can compute the output of each neuron in parallel

5 weird tricks to improve training

- How to initialize the model
- How to choose a nonlinearity
- How to avoid over-fitting
- How to pre-process the data
- …

Easy to use !

All tips included

TensorFlow

Compute gradients for you

Works on GPU

Theano is a good alternative to TensorFlow

Neural Networks are trained with Stochastic Gradient Descent

1) Define an objective function:

$$\sum_{(X_i, y_i) \in \{Samples\}} (y_i - o(W, X_i))^2$$

2) Compute partial derivative w.r.t. one training sample:

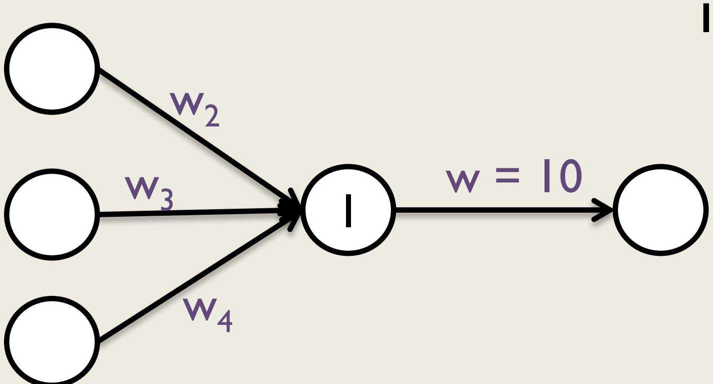
$$\frac{\partial (y_i - o(W, X_i))^2}{\partial W}$$

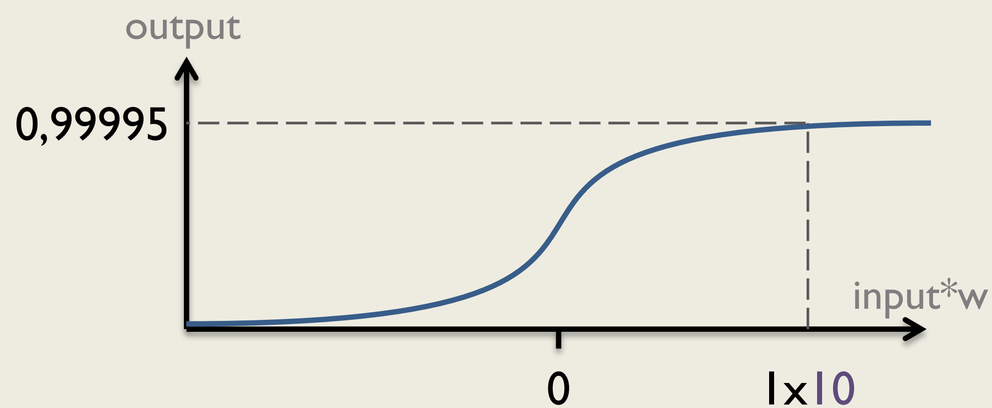
3) Slightly change the parameters in the direction of the gradient:

$$W \leftarrow W - \lambda \frac{\partial (y_i - o(W, X_i))^2}{\partial W}$$

Saturating nonlinearities: it's a trap !

Goal: if x > 0, y=0
if < 1 y=1



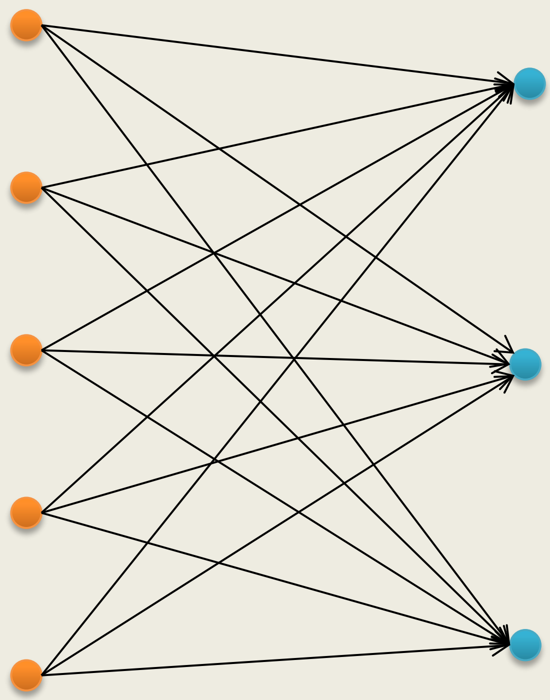


The error is large, but the gradient small
→The network doesn't learn

Deep networks make it worse

Trick 1: Better initialization

(Avoid weights too large or too small)



$$w \sim N\left(0, \sqrt{\frac{2}{n+m}}\right)$$

Xavier Glorot and Yoshua Bengio

Understanding the difficulty of training deep feedforward neural networks. (2010)

Trick 2: Batch Normalization

(Avoid inputs too large or too small)

Goal:
Ensure that the output of each neuron has a reasonable variance

Solution:
Treat inputs by small batches (16 – 100).
After each layer, compute variance over the batch, and normalize

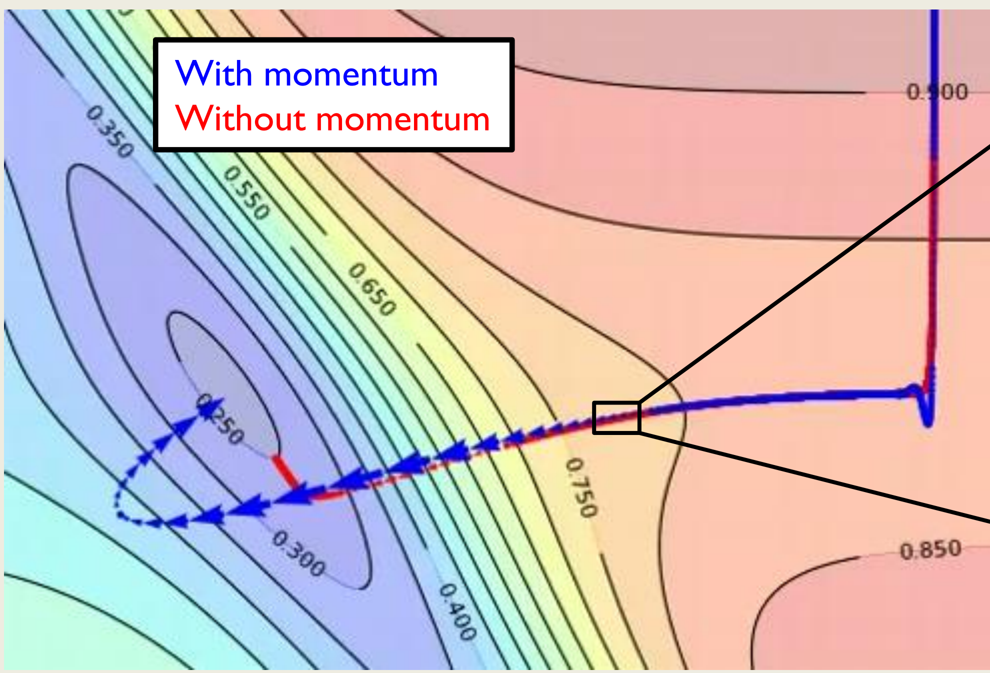
Ioffe, Sergey and Szegedy, Christian
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. (2015)

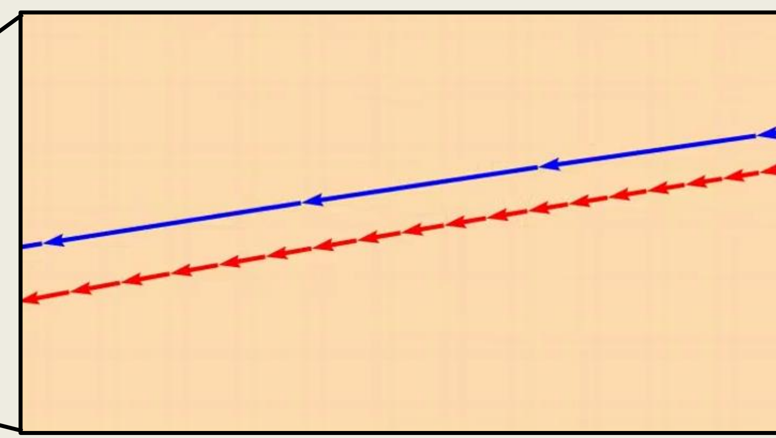
Trick 3: Momentum

(Get out flat areas)

Keep a memory of past updates, and
tend to keep moving in the same direction
→ Accelerate on mostly flat areas

$$W \leftarrow W - \lambda(\text{Gradient} + \alpha \text{ Last update})$$





Momentum seem to take a longer path,
But is in fact 10x faster

Image from a video by Ryan Harris (https://www.youtube.com/watch?v=7HZk7kGk5bU)

Trick 4: Adaptive Gradients

(Solve one problem at a time)

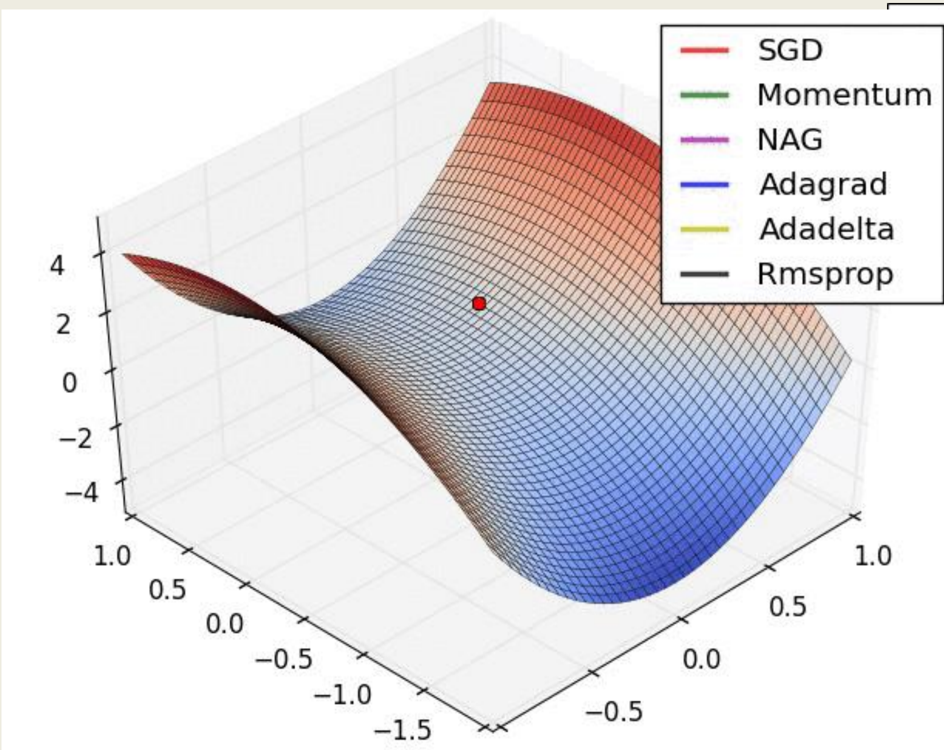
Reduce the learning rate of weights
that have accumulated large gradients

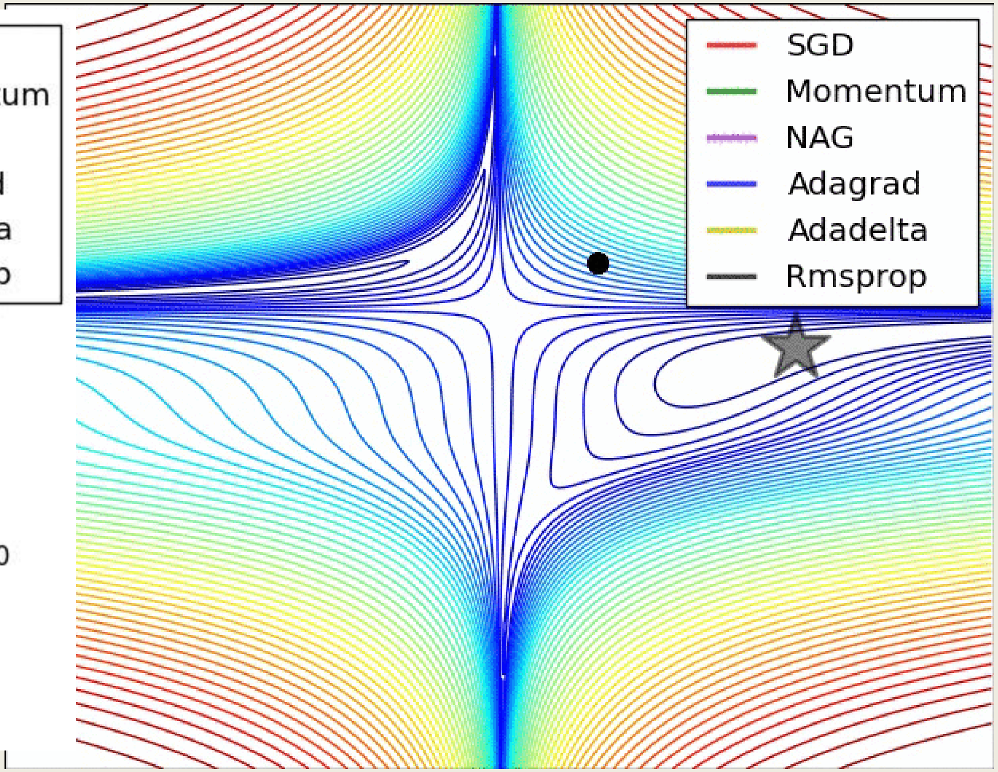
➡ Don't let parameters oscillate indefinitely

➡ Reduce the learning rate of each weight independently

Adam combines momentum and adaptive gradients
Kingma, Diederik, and Jimmy Ba
Adam: A Method for Stochastic Optimization. (2014)

Visualization of gradient descent





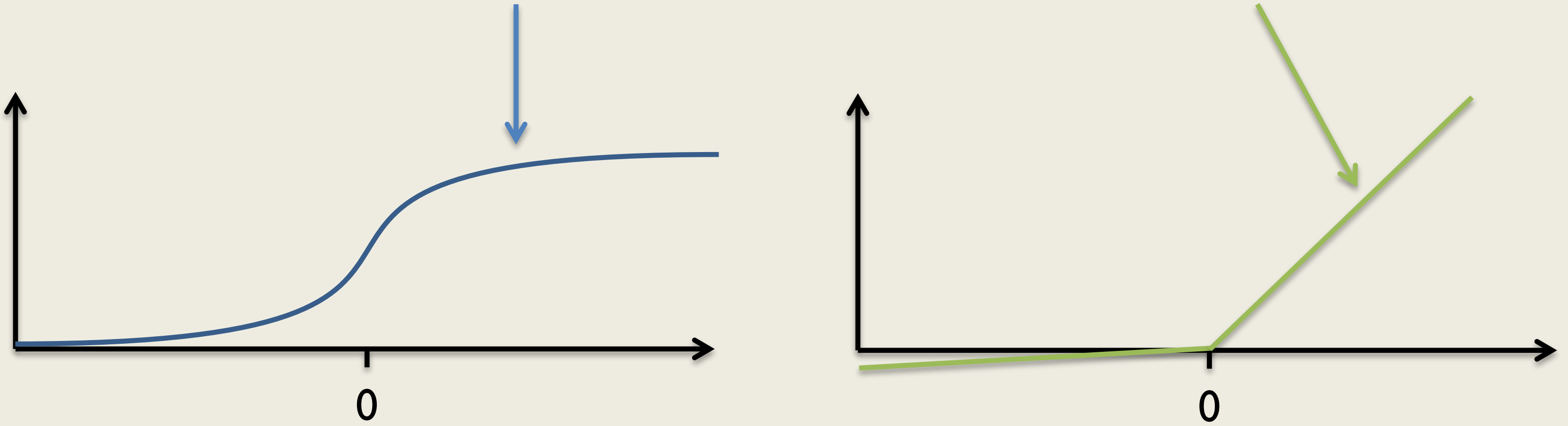
Visualisations from http://imgur.com/a/Hqolp

Good comparison of gradient descent methods:
http://sebastianruder.com/optimizing-gradient-descent

Trick 5: ReLu

(Simpler, Faster, Better, Stronger)

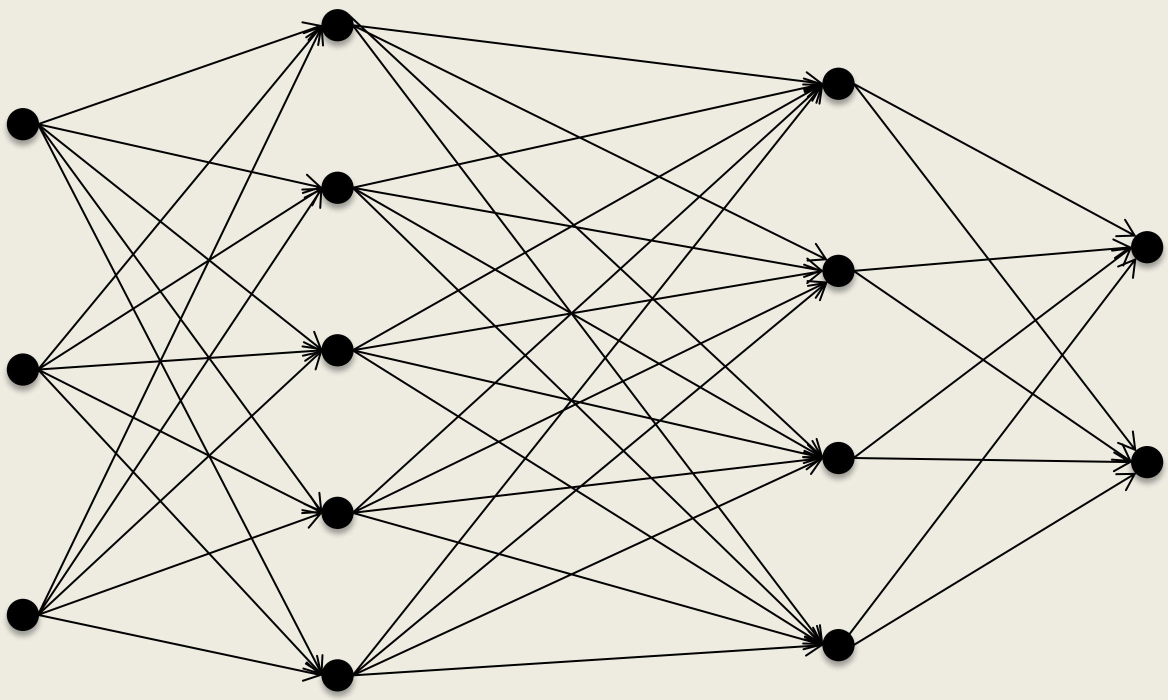
Replace the sigmoid by the « Rectified Linear Unit » (ReLu)



- ➡ Larger domain with non-zero gradient
- ➡ Much faster to compute

One more trick: Dropout

For each training sample, 50% of the hidden neurons are randomly turned off



- Avoids complex co-adaptation
- Works with noisy data
- Similar to training an ensemble model

Simple in theory, hard in practice

- Use knowledge accumulated over the years
- Use a framework
- Don’t fear local minima, fear saturated nonlinearities

What's next ?

- Adversarial Learning
- Deep Q-Learning
- Memory Networks