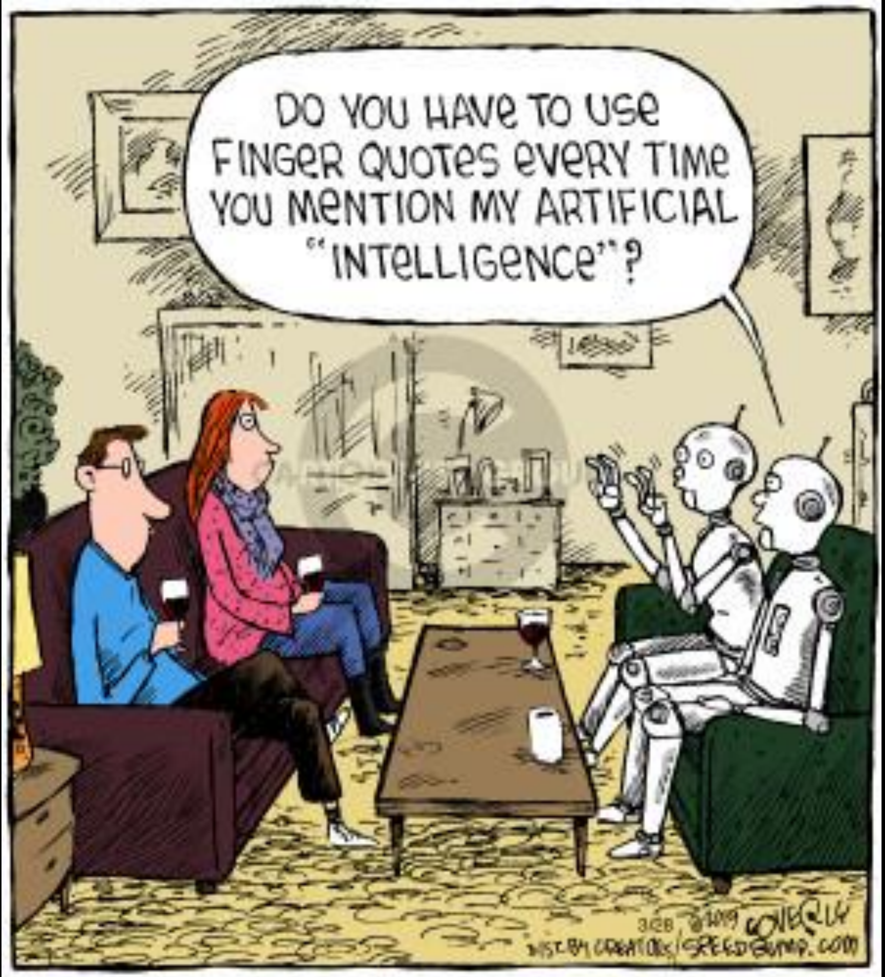


INFO 251: Applied Machine Learning

Deep Learning

Course Outline

- Causal Inference and Research Design
  - Experimental methods
  - Non-experiment methods
- Machine Learning
  - Design of Machine Learning Experiments
  - Linear Models and Gradient Descent
  - Non-linear models
  - **Neural models**
  - Unsupervised Learning
  - Practicalities, Fairness, Bias
- Special topics

Key Concepts (last lecture)

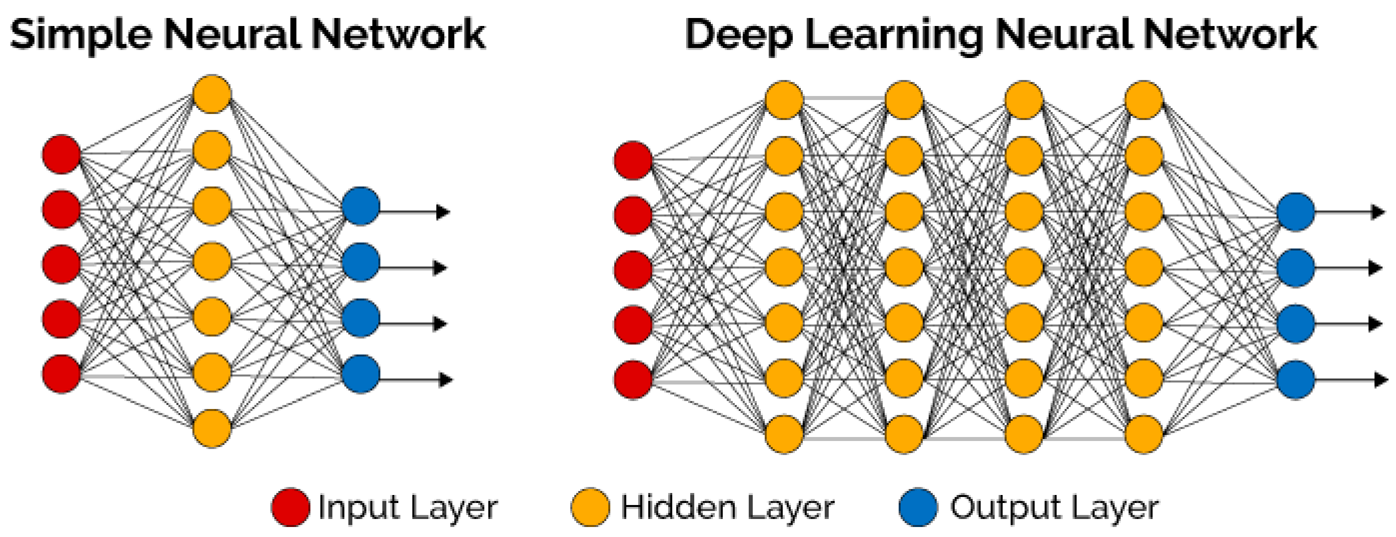
- Multilayer networks
- Activation and non-convexity
- Why GD doesn’t work on MLP’s
- Backpropagation

Today's outline

- "Deep" Learning
- Autoencoders
- Convolutions
- Pooling
- Convolutional Neural Networks
- Recurrent Neural Networks and LSTM's

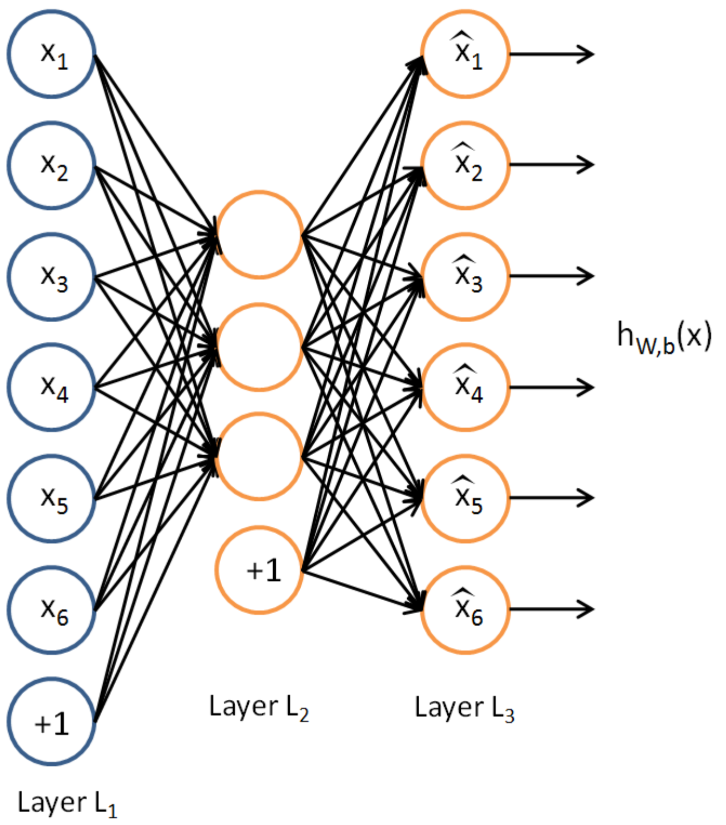
Deep Learning

- What is “deep” learning?



Example NN building block: Autoencoders

- Auto-encoders map inputs to inputs (i.e., it approximates the identity function)
  - $\hat{y}_i(x_i) = x_i$
- Why bother?
  - Data compression
  - Data abstraction
- Example
  - 96 x 96 pixel image: 9216 features
  - This NN compresses this to *h* features, where *h* is the number of hidden nodes in L_2



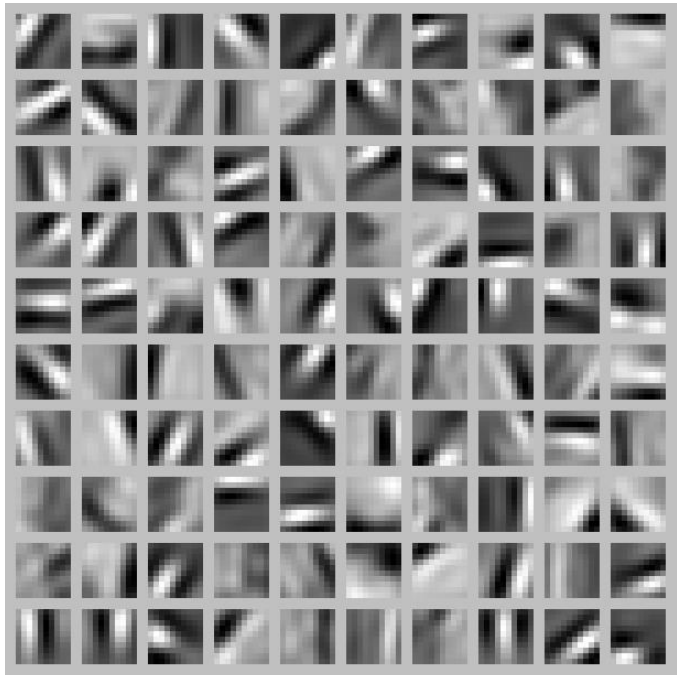
Sparse Autoencoders

- "Sparsity"
  - Even if number of hidden units h is large, can impose sparsity constraints on those hidden units
  - Achieve sparsity by
    - Constraining avg. activation of each hidden neuron to be small (e.g., <0.05) – see Ng's lecture notes on bCourses for details of this approach
    - Choosing the *k* highest activations and setting the rest to zero (see Makhzani et al., 2013). Error backpropagated only through these active nodes

Sparse Autoencoders

- Take-away
  - Often, autoencoders are not competitive with hand-crafted features
  - But features themselves are often useful

- Example: Image features
  - Start with 10x10 pixel images
  - 100 hidden units
  - Find images that maximally
  activate each of those hidden units:



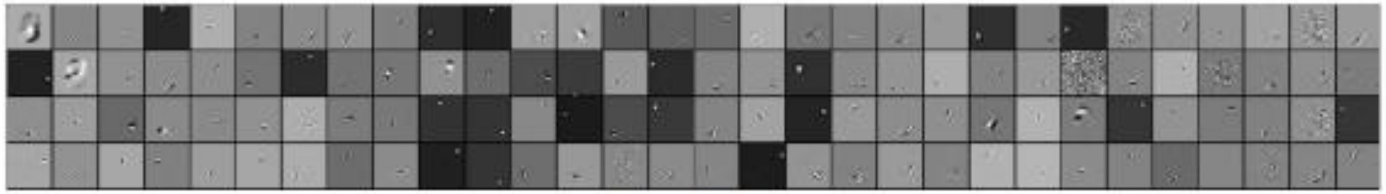
Sparse Autoencoders

- *k*-Sparse Autoencoders Example
  - MNIST data (28x28 images)
  - Hidden layer retains only the *k* largest elements

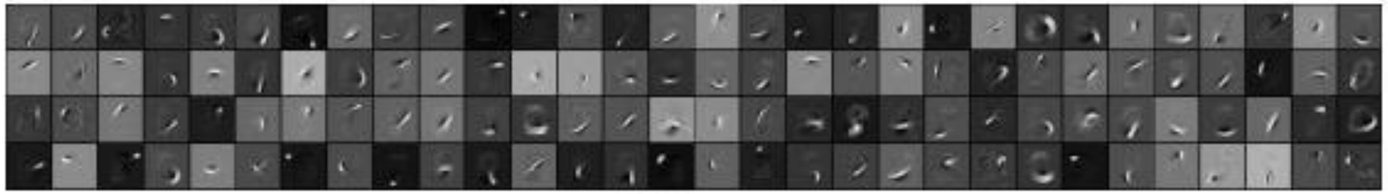
k-Sparse Autoencoders

Alireza Makhzani MAKHZANI@PSI.UTORONTO.CA
Brendan Frey FREY@PSI.UTORONTO.CA
University of Toronto, 10 King's College Rd. Toronto, Ontario M5S 3G4, Canada

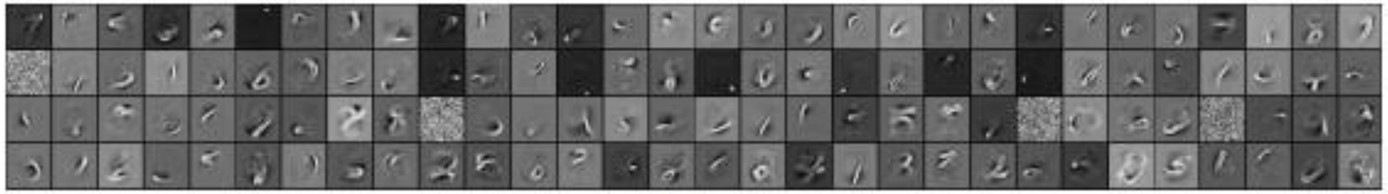
Sparse Autoencoders



(a) $k = 70$



(b) $k = 40$



(c) $k = 25$

(d) $k = 10$

Figure 1. Filters of the k-sparse autoencoder for different sparsity levels k, learnt from MNIST with 1000 hidden units.

Today's outline

- "Deep Learning"
- Autoencoders
- **Convolutions**
- Pooling
- Convolutional Neural Networks
- Recurrent Neural Networks and LSTM's

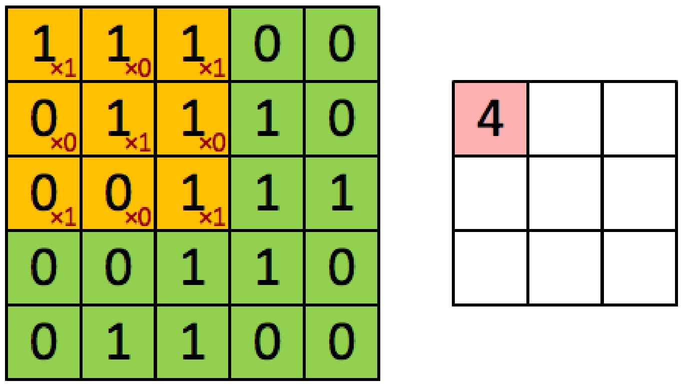
Convolutions and Features

- Fully connected networks have lots of parameters
  - 96x96 pixels => 9,216 input units
  - Assume 100 output features => 921,600 parameters
  - Also, no notion of local structure when fully connected – far away pixels treated same as neighboring pixels
- Locally connected networks
  - Each hidden unit connects to a subset of input units
  - For images, these are often contiguous
  - This is sort of how the visual cortex works
  - For other data, depends on how “contiguous” is defined

Convolutions and Features

http://ufldl.stanford.edu/tutorial/

- "Convolutions" apply filter/kernel to input
  - Filter: common set of "shared" weights applied to all subregions
  - E.g., instead of 96 x 96 weights/neuron, 3 x 3 weights/neuron
    - Number of parameters: 9,216 => 9
  - Much more efficient than fully connected network!



Image

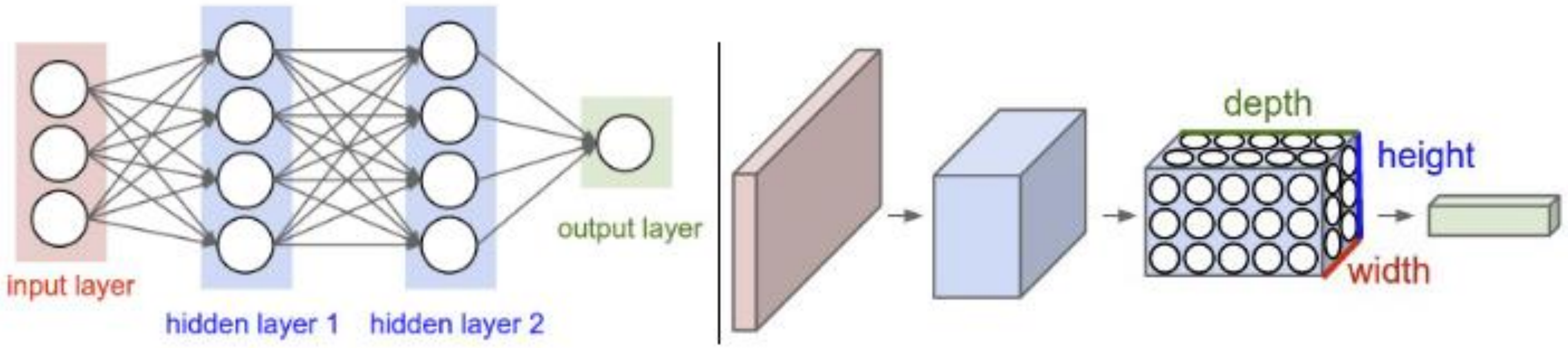
Convolved Feature

Animation shows a single convolutional filter:

$$F = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Convolutions and Features

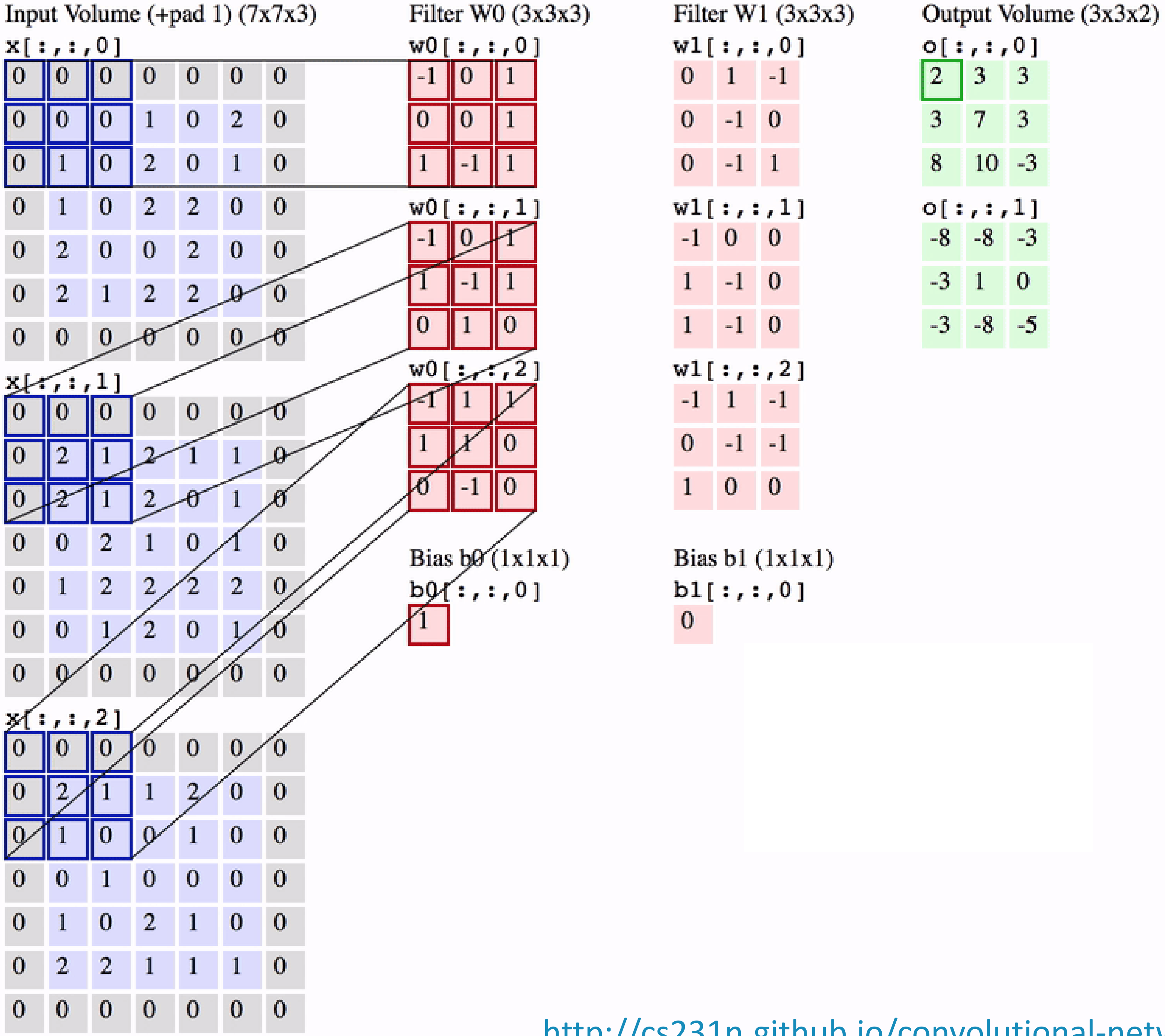
- Each layer of a ConvNet transforms an input 3D volume into an output 3D volume
  - "3D" because of depth, e.g., RGB has depth 3
  - Hidden layer might have depth = number of filters



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Example

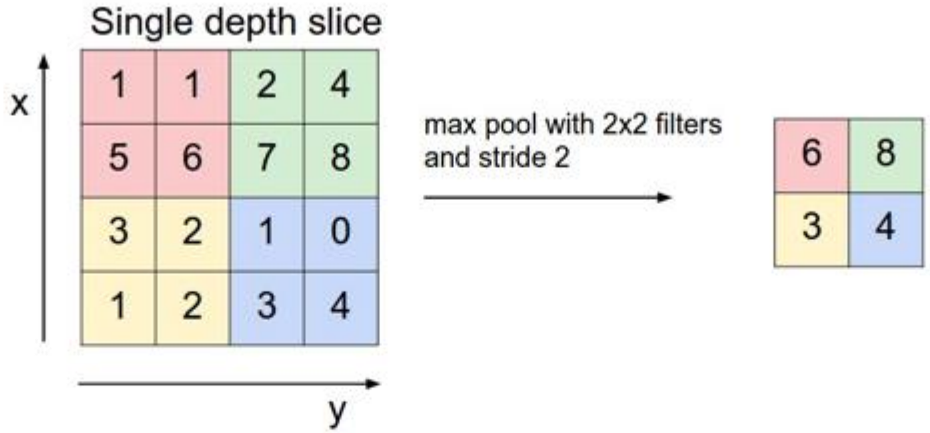
- Input is 5 x 5 x 3
- Output is 3 x 3 x 2
- 2 filters (w0 and w1)
- Stride = 2
- Input padding = 1

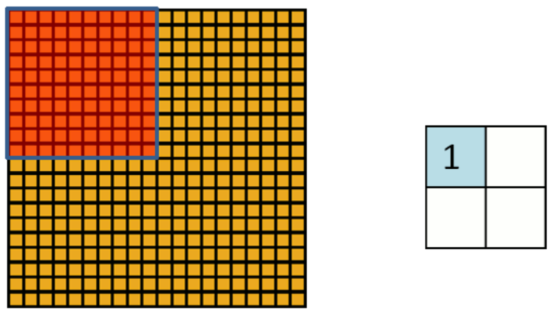


http://cs231n.github.io/convolutional-networks/

"Pooling"

- Even after convolutions, can have lots of features
  - (but a much smaller set of shared weights!)
- "Pooling" aggregates regions of a convolved layer
  - Typically: mean or max feature activation in a region
  - Pooling progressively shrinks dimensionality of network
  - Often used to reduce overfitting





Convolved feature

Pooled feature

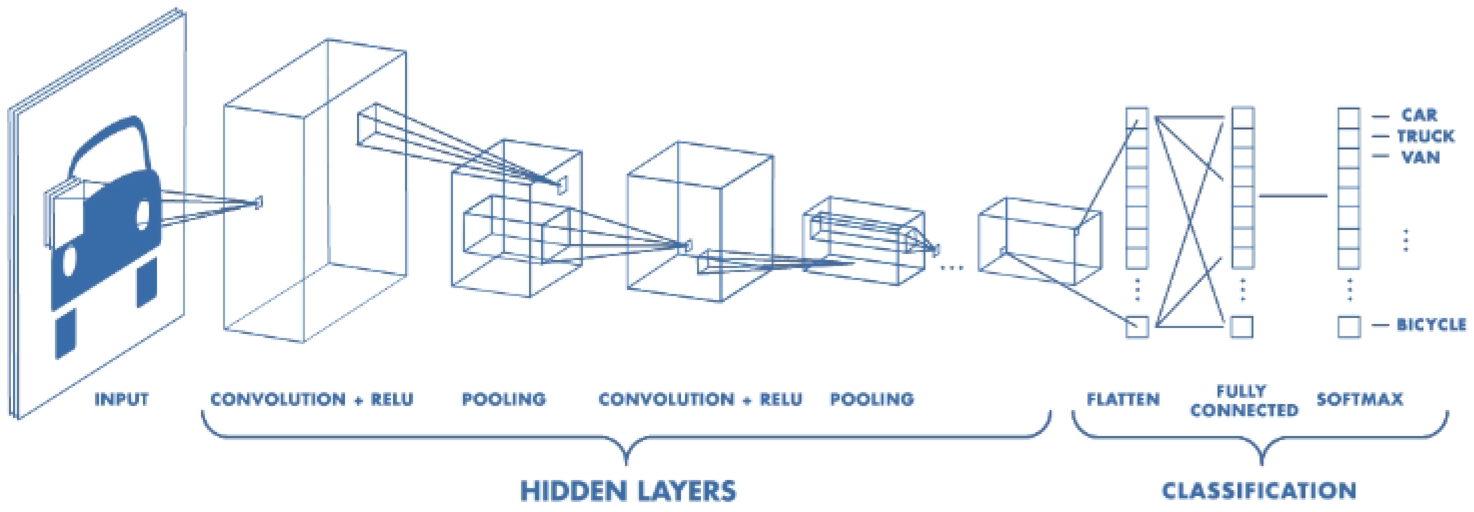
Today's outline

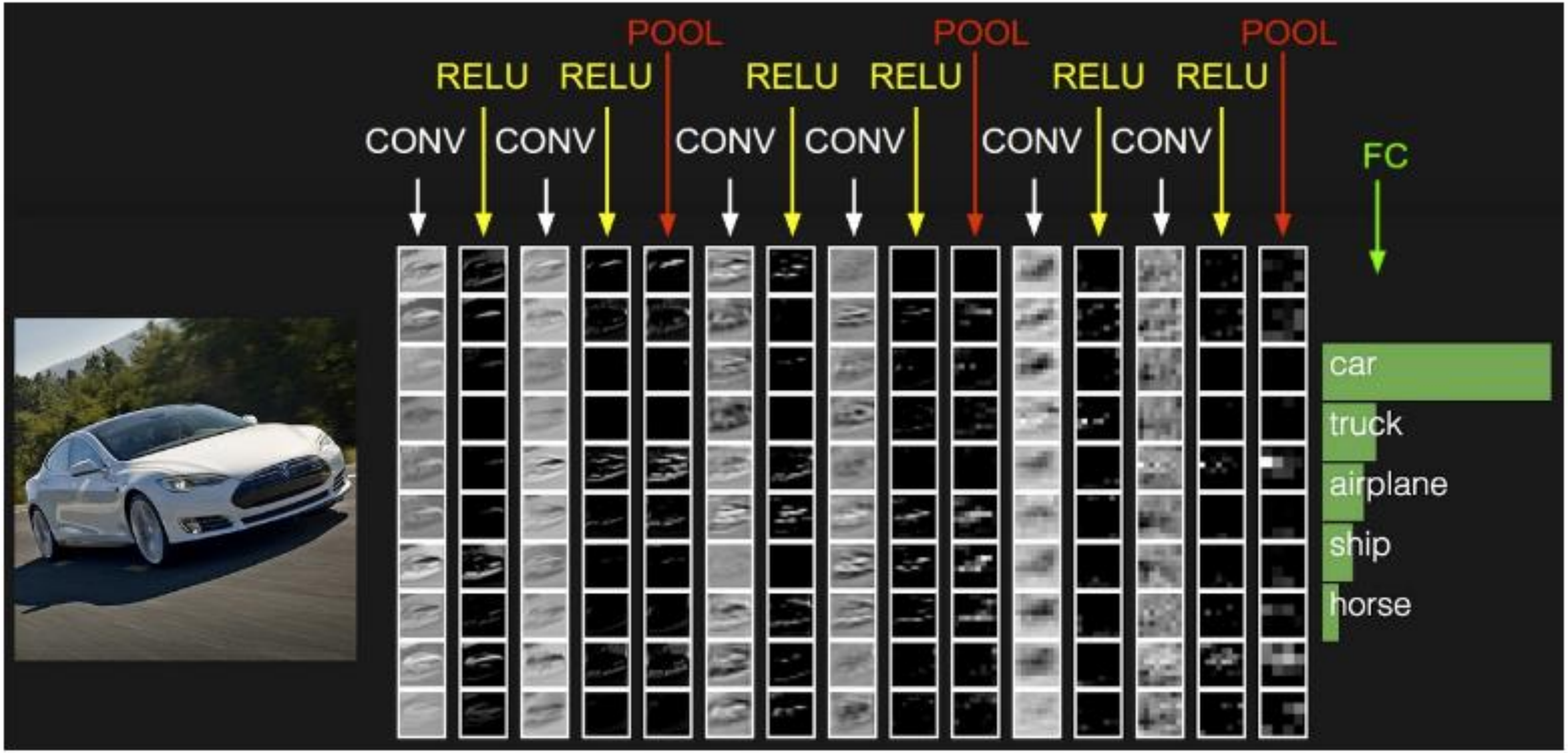
- "Deep Learning"
- Autoencoders
- Convolutions
- Pooling
- **Convolutional Neural Networks**
- Recurrent Neural Networks and LSTM's

Convolutional Neural Networks

- ConvNets typically have different layers
  - Input
    - e.g, for a 32-pixel image with RGB channels: 32 x 32 x 3
  - Convolutions
    - e.g., after applying 12 filters: 8 x 8 x 12
  - ReLU (Rectified Linear Units): activation function
    - e.g., max(0,*x*)
    - leaves volume unchanged
  - Pooling
    - e.g., 3 x 3 x 12
  - Fully connected layers
    - e.g., final 'softmax' classification layer: 1 x 1 x 10
    - (here, output vector might indicate class probabilities)

Convolutional Neural Networks



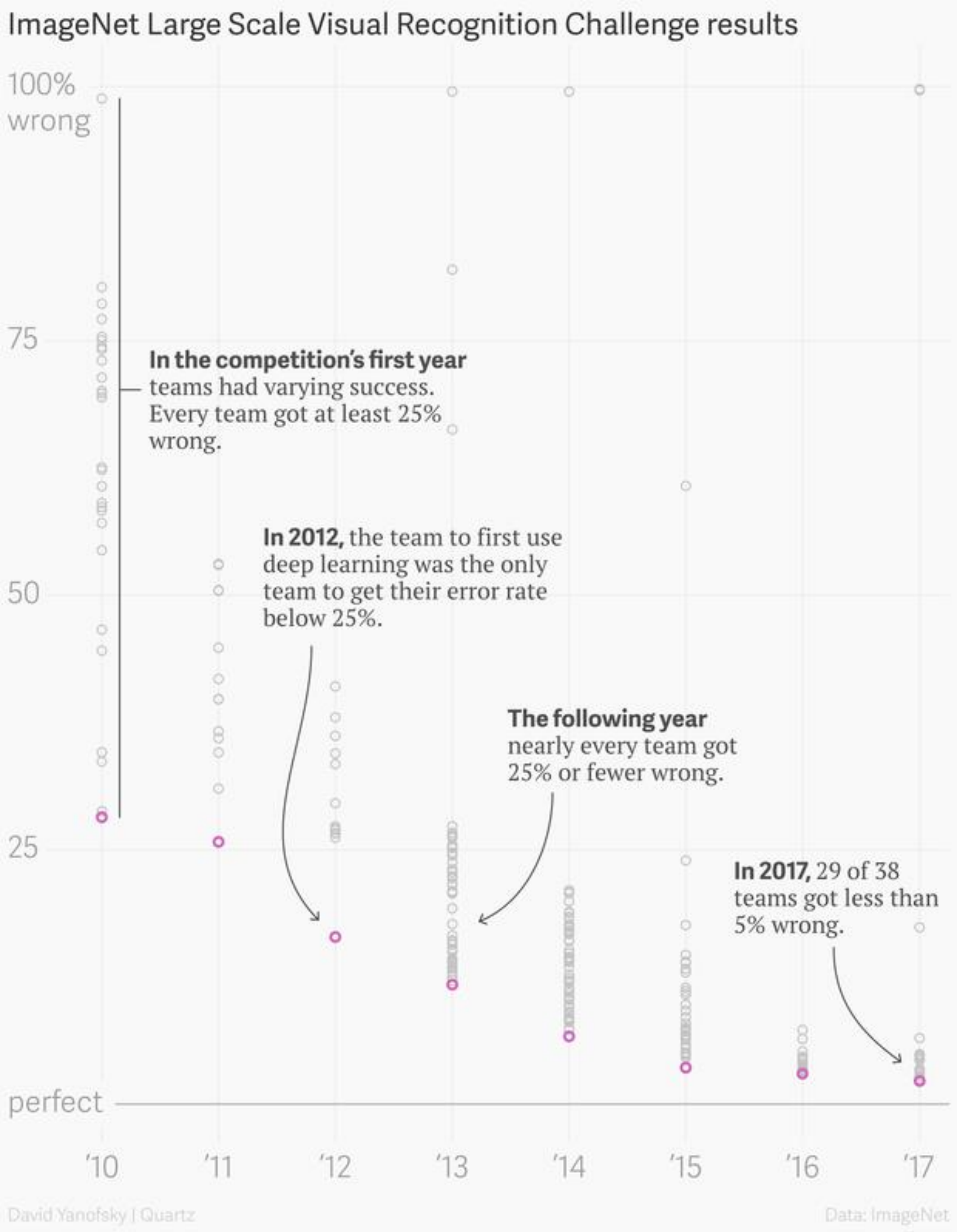


The activations of an example ConvNet architecture. The initial volume stores the raw image pixels (left) and the last volume stores the class scores (right). Each volume of activations along the processing path is shown as a column. Since it's difficult to visualize 3D volumes, we lay out each volume's slices in rows. The last layer volume holds the scores for each class, but here we only visualize the sorted top 5 scores, and print the labels of each one. The full web-based demo is shown in the header of

Convolutional Neural Networks

- Distinguishing features of CNN's
  - 3D volumes of neurons: different types of locally or fully connected layers are stacked to form CNN
  - Local connectivity: small regions of one layer connected to next layer. Early layers learn local features; later layers learn larger areas
  - Shared weights: dramatically reduces number of free parameters

Taking Stock



Today's outline

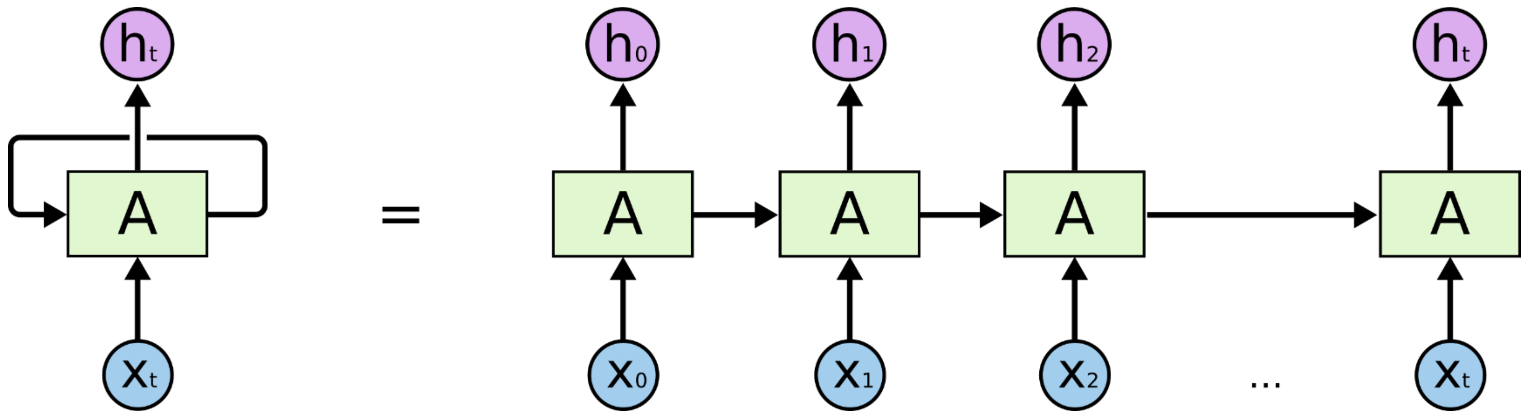
- "Deep Learning"
- Autoencoders
- Convolutions
- Pooling
- Convolutional Neural Networks
- **Recurrent Neural Networks and LSTM's**

Recurrent Neural Networks

- CNNs vs. RNNs
  - CNNs: accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output (e.g. probabilities of different classes)
  - Each new input is a new input; ordering of inputs is largely irrelevant (and often randomized during training)
  - "No persistence" and "no memory"

Recurrent Neural Networks

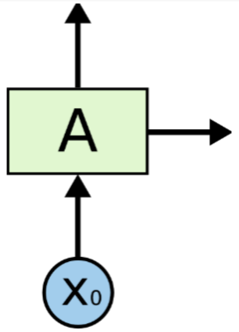
- Recurrent Neural Networks (RNNs)
  - A class of neural networks "with loops", i.e., where previous outputs can be used as inputs

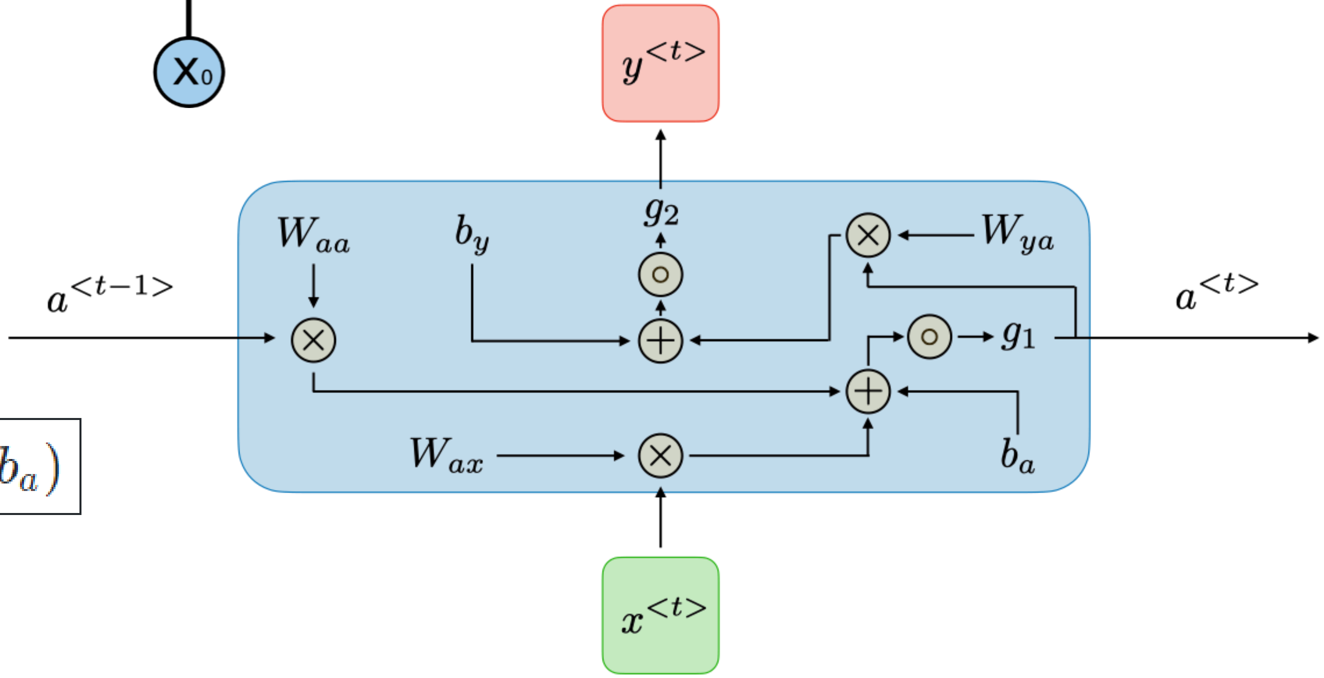


  - Each node accepts input vector, emits output vector
    - Output influenced by input x, as well as prior state

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

- Computation: Unpacking the 



  - For each timestep *t:*

$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)}$$

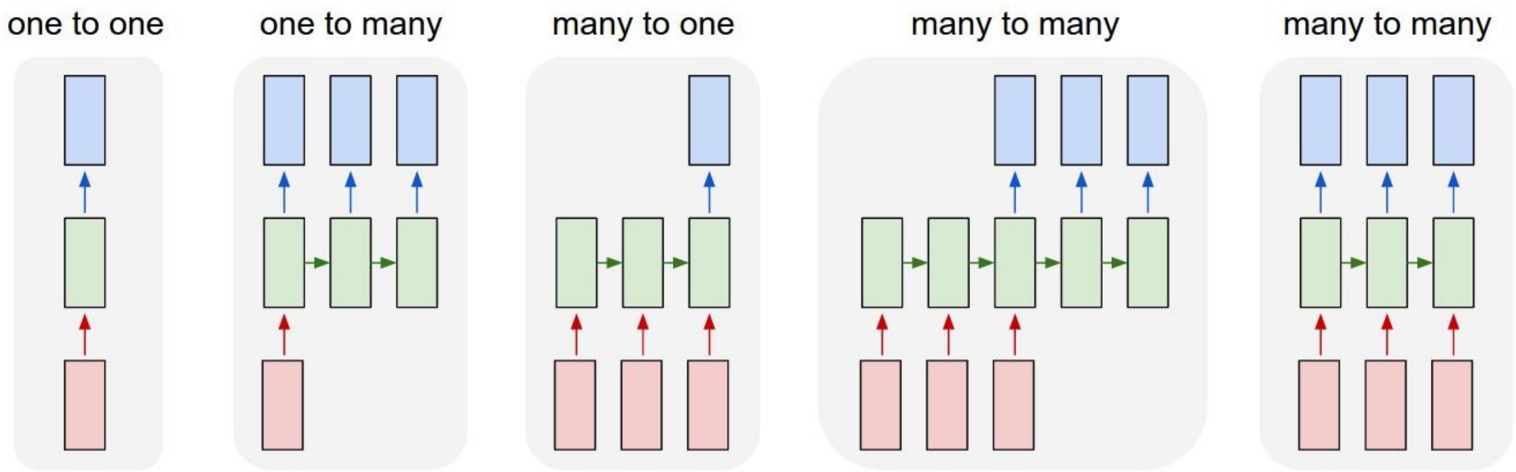
$$\boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally

g_1, g_2 activation functions.

Example RNN architectures

- Recurrent Neural Networks (RNNs)
  - Current classification depends on past information
  - Makes it possible to operate over *sequences* of vectors: Sequences in input, output, or both



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Networks

- In theory, RNNs can simulate arbitrary programs
  - Similar to universal approx. theorem

On The Computational Power Of Neural Nets

Hava T. Siegelmann
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
siegelma@paul.rutgers.edu

Eduardo D. Sontag
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903
sontag@hilbert.rutgers.edu

  - In practice, they can't

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 5, NO. 2, MARCH 1994

Learning Long-Term Dependencies with Gradient Descent is Difficult

Yoshua Bengio, Patrice Simard, and Paolo Frasconi, *Student Member, IEEE*

- In theory, RNNs can handle long-term dependencies…
  - In practice, they work better with shorter dependencies
  - *"Standard RNNs fail to learn in the presence of time lags greater than 5 – 10 discrete time steps between relevant input events and target signals"* (Gers, 2000)

Long Short Term Memory networks

- Some pros and cons of RNN's

Advantages	Drawbacks
• Possibility of processing input of any length • Model size not increasing with size of input • Computation takes into account historical information • Weights are shared across time	• Computation being slow • Difficulty of accessing information from a long time ago • Cannot consider any future input for the current state

https://cs230.stanford.edu/

LONG SHORT-TERM MEMORY

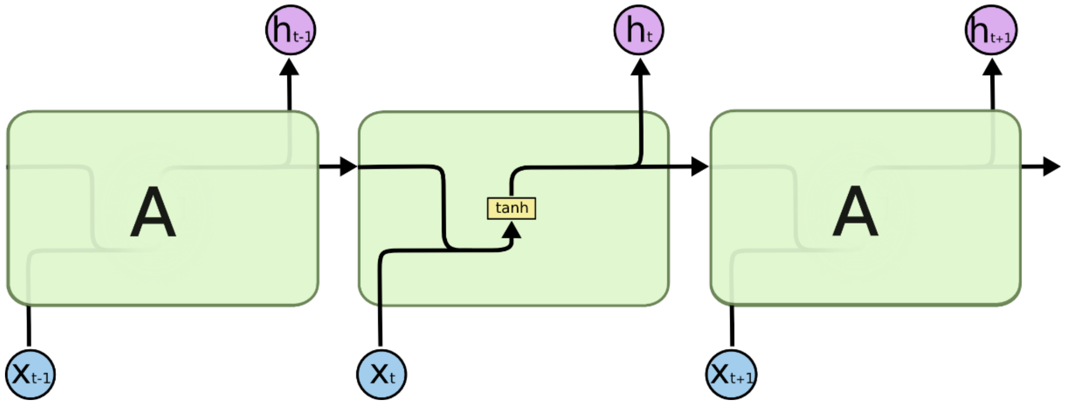
NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter   Jürgen Schmidhuber

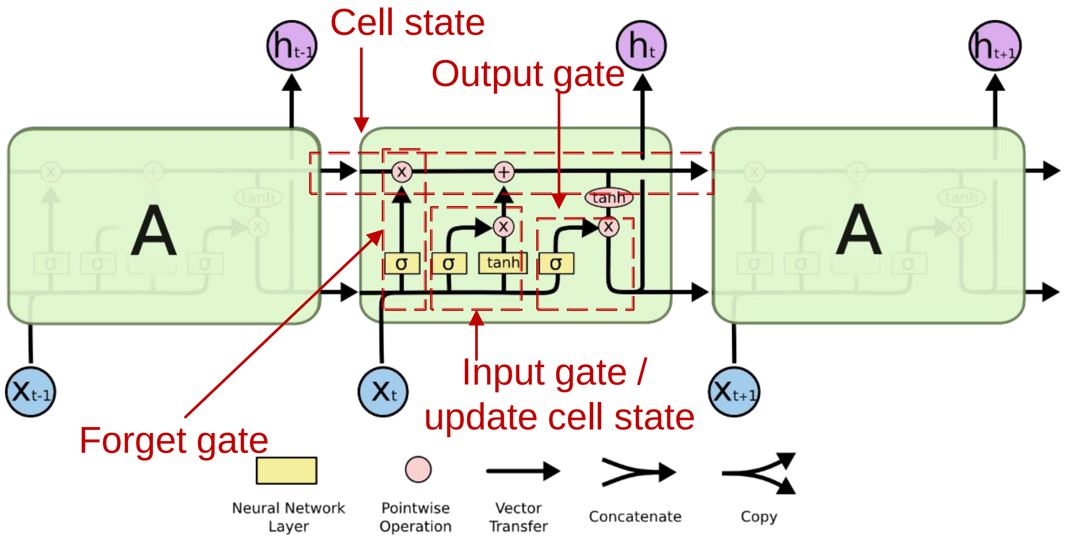
- Enter the LSTM
  - *"An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each one contains one or more recurrently connected memory cells and three multiplicative units – the input, output and forget gates – that provide continuous analogues of write, read and reset operations for the cells."*

LSTM architecture

- Standard RNN

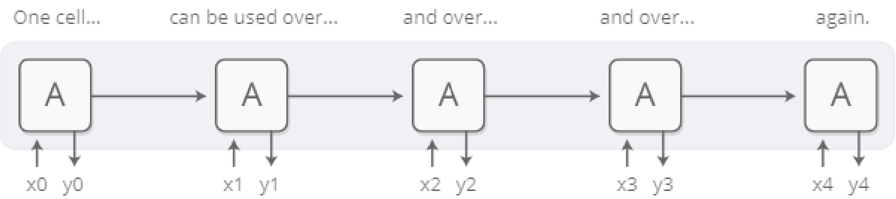


- Example LSTM



RNN/LSTM summary

- Strengths
  - Can be used to represent a **sequence** with high-level understanding, to annotate sequences, and to generate new sequences from scratch



  - Very successful applications:
    - In 2017, Facebook performed ~4.5 billion automatic translations every day using long short-term memory networks (source)
    - Google uses LSTM for Android's speech recognition,[13][14] Google Translate[16][17]
    - Apple uses LSTM for "Quicktype" on the iPhone [18][19] and for Siri.[20]
    - Amazon uses LSTM for Amazon Alexa.[21]

Beyond LSTM's

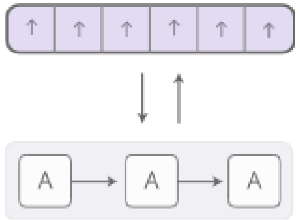
- "LSTM's are so 2015"

The fall of RNN / LSTM

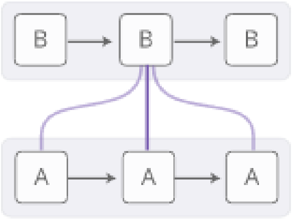
Eugenio Culurciello Follow
Apr 13, 2018 · 8 min read

We fell for Recurrent neural networks (RNN), Long-short term memory (LSTM), and all their variants. **Now it is time to drop them!**

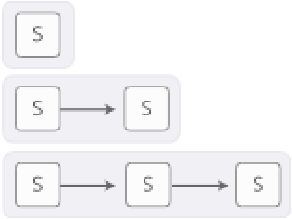
- What else?



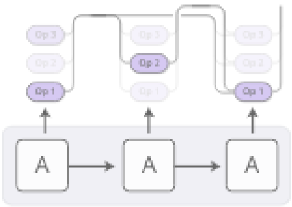
Neural Turing Machines have external memory that they can read and write to.



Attentional Interfaces allow RNNs to focus on parts of their input.



Adaptive Computation Time allows for varying amounts of computation per step.



Neural Programmers can call functions, building programs as they run.

https://distill.pub/2016/augmented-rnns/

Further Reading

- Tutorials on bCourses
  - ConvNets for Visual Recognition: http://cs231n.github.io/
  - TensorFlow tutorial on CNNs: https://www.tensorflow.org/tutorials/images/deep_cnn
  - The Unreasonable Effectiveness of Recurrent Neural Networks http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- At Berkeley
  - STAT 157: Introduction to Deep Learning
  - CS 285: Deep Reinforcement Learning
  - CS 294-131: Special Topics in Deep Learning
  - CS294-158: Deep Unsupervised Learning