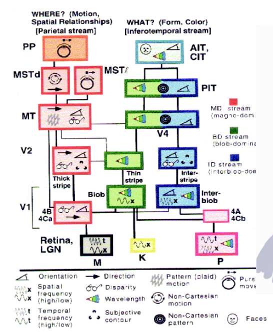
Deep Learning Overview

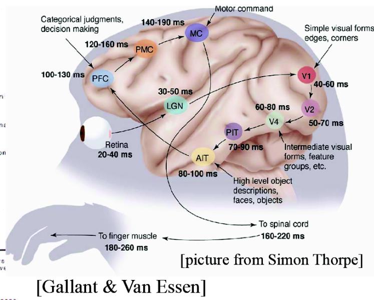
- Fall 2016
- Yoonsuck Choe

What Is Deep Learning?

- Learning higher level abstractions/representations from data.
- Motivation: how the brain represents and processes sensory information in a hierarchical manner.
  - The ventral (recognition) pathway in the visual cortex has multiple stages
  - Retina - LGN - V1 - V2 - V4 - PIT - AIT ....



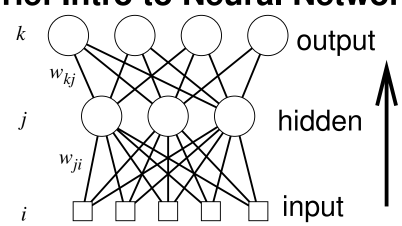
[Gallant & Van Essen]

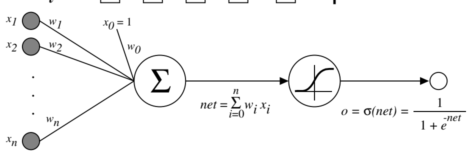


[picture from Simon Thorpe]

From LeCun's Deep Learning Tutorial

Brief Intro to Neural Networks



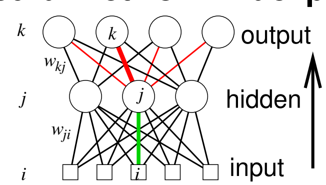


Deep learning is based on neural networks.

- Weighted sum followed by nonlinear activation function.
- Weights adjusted using *gradient descent* (η = learning rate):

$$w_{ij} \leftarrow w_{ij} + \eta \frac{\partial E}{\partial w_{ij}}$$

Intro to Neural Network: Backpropagation



Weight w_{ji} is updated as: $w_{ji} \leftarrow w_{ji} + \eta \delta_j a_i$, where

- a_i : activity at input side of weight w_{ji}.
- Hidden to output weights (thick red weight). T_k is target value.

$$\delta_k = (T_k - a_k)\sigma'(net_k)$$

- Deeper weights (green line in figure above).

$$\delta_j = \left[\sum_k w_{kj}\delta_k\right]\sigma'(net_j)$$

Deep Learning

- Complex models with large number of parameters
  - Hierarchical representations
  - More parameters = more accurate on training data
  - Simple learning rule for training (gradient-based).
- Lots of data
  - Needed to get better generalization performance.
  - High-dimensional input need exponentially many inputs (curse of dimensionality).
- Lots of computing power: GPGPU, etc.
  - Training large networks can be time consuming.

Deep Learning, in the Context of AI/ML

Deep Learning:
Automating
Feature Discovery

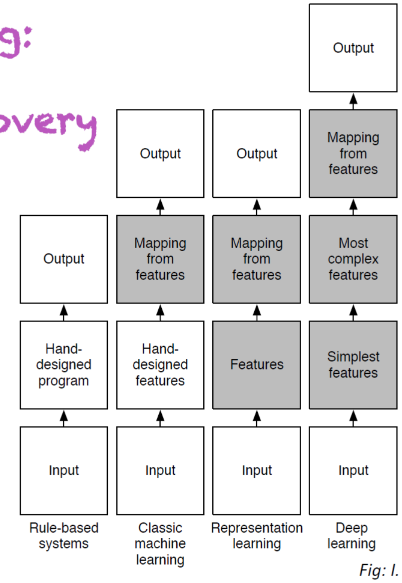


Fig: I. Goodfellow

From LeCun's Deep Learning Tutorial

The Rise of Deep Learning

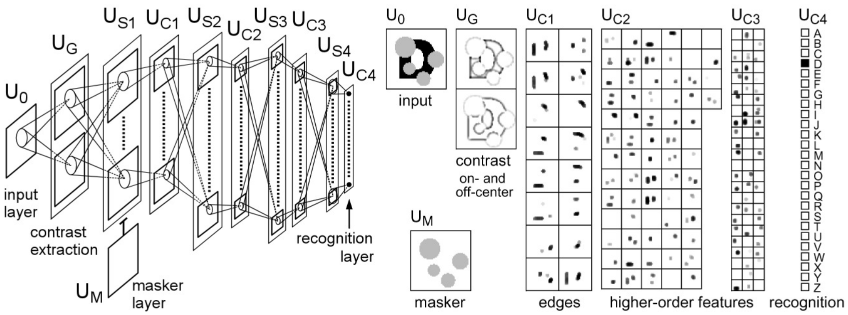
Made popular in recent years

- Geoffrey Hinton et al. (2006).
- Andrew Ng & Jeff Dean (Google Brain team, 2012).
- Schmidhuber et al.'s deep neural networks (won many competitions and in some cases showed super human performance; 2011–). Recurrent neural networks using LSTM (Long Short-Term Memory).
- Google Deep Mind: Atari 2600 games (2015), AlphaGo (2016).
- ICLR, International Conference on Learning Representations: First meeting in 2013.

Long History (in Hind Sight)

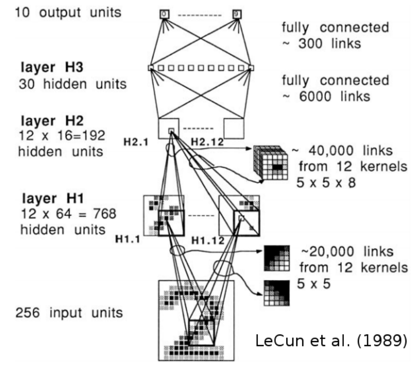
- Fukushima's Neocognitron (1980).
- LeCun et al.'s Convolutional neural networks (1989).
- Schmidhuber's work on stacked recurrent neural networks (1993). Vanishing gradient problem.
- See Schmidhuber's extended review: Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85-117.

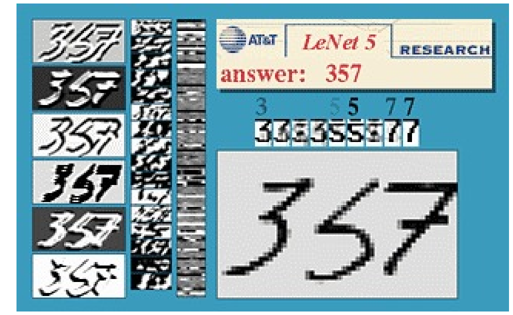
History: Fukushima's Neocognitron



- Appeared in journal *Biological Cybernetics* (1980).
- Multiple layers with local receptive fields.
- S cells (trainable) and C cells (fixed weight).
- Deformation-resistent recognition.

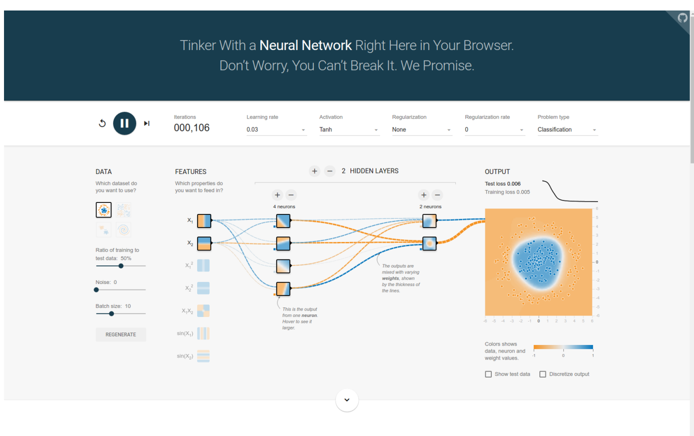
History: LeCun's Colvolutional Neural Nets





- Convolution kernel (weight sharing) + Subsampling
- Fully connected layers near the end.
- Became a main-stream method in deep learning.

Motivating Deep Learning: Tensorflow Demo



- http://playground.tensorflow.org
- Demo to explore why deep nnet is powerful and how it is limited.

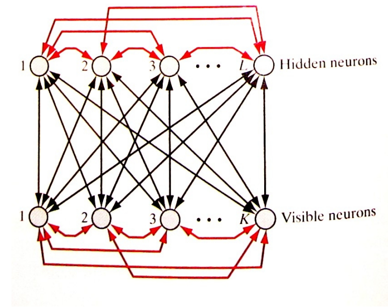
Current Trends

- Deep belief networks (based on Boltzmann machine)
- Convolutional neural networks
- Deep Q-learning Network (extensions to reinforcement learning)
- Deep recurrent neural networks using (LSTM)
- Applications to diverse domains.
  - Vision, speech, video, NLP, etc.
- Lots of open source tools available.

Boltzmann Machine to Deep Belief Nets

- Haykin Chapter 11: Stochastic Methods rooted in statistical mechanics.

Boltzmann Machine



- Stochastic binary machine: +1 or -1.
- Fully connected symmetric connections: $w_{ij} = w_{ji}$.
- Visible vs. hidden neurons, clamped vs. free-running.
- Goal: Learn weights to model prob. dist of visible units.
- Unsupervised. Pattern completion.

Boltzmann Machine: Energy

- Network state: $\mathbf{x}$ from random variable $\mathbf{X}$.
- $w_{ij} = w_{ji}$ and $w_{ii} = 0$.
- Energy (in analogy to thermodynamics):

$$E(\mathbf{x}) = -\frac{1}{2} \sum_i \sum_{j, i \neq j} w_{ji} x_i x_j$$

Boltzmann Machine: Prob. of a State $\mathbf{x}$

- Probability of a state $\mathbf{x}$ given $E(\mathbf{x})$ follows the *Gibbs distribution*:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left(-\frac{E(\mathbf{x})}{T}\right),$$

  - Z: *partition function* (normalization factor – hard to compute)

$$Z = \sum_{\forall \mathbf{x}} \exp(-E(\mathbf{x})/T)$$

  - T: temperature parameter.
  - Low energy states are exponentially more probable.
- State $\mathbf{x}$ changed over time following the probability distribution $P(\mathbf{X} = \mathbf{x})$.

Boltzmann Learning Rule

- Learning based on correlation ρ^+_{ji} (clamped) and ρ^-_{ji} (free-running).

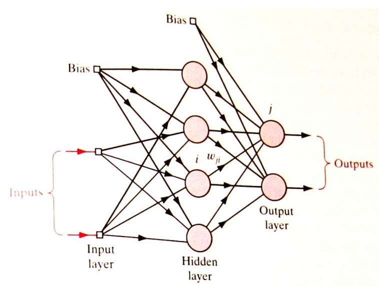
$$\Delta w_{ji} = \eta \frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \eta \left(\rho^+_{ji} - \rho^-_{ji} \right)$$

where $L(\mathbf{w})$ is the log likelihood of the pattern being any of the training patterns, and η is the learning rate. This is *gradient ascent*.

Boltzmann Machine Summary

- Theoretically elegant.
- Very slow in practice (especially the unclamped phase).

Logistic (or Directed) Belief Net



- Similar to Boltzmann Machine, but with directed, acyclic connections.

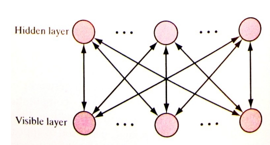
$$P(X_j = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) = P(X_j = x_j | parents(X_j))$$

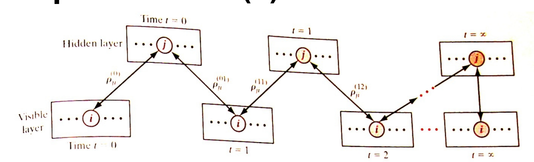
- Same learning rule:

$$\Delta w_{ji} = \eta \frac{\partial L(\mathbf{w})}{\partial w_{ji}}$$

- With dense connetions, calculation of P becomes intractable.

Deep Belief Net (1)





- Overcomes issues with Logistic Belief Net. Hinton et al. (2006)
- Based on Restricted Boltzmann Machine (RBM): visible and hidden layers, with layer-to-layer full connection but no within-layer connections.
- RBM Back-and-forth update: update hidden given visible, then update visible given hidden, etc., then train $\mathbf{w}$ based on

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \rho^{(0)}_{ji} - \rho^{(\infty)}_{ji}$$

Deep Belief Net (2)

Deep Belief Net = Layer-by-layer training using RBM.

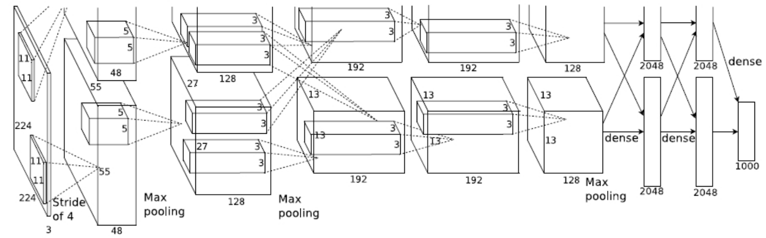
Hybrid architecture: Top layer = undirected, lower layers directed.

1. Train RBM based on input to form hidden representation.
2. Use hidden representation as input to train another RBM.
3. Repeat steps 2-3.

* Similar approach: Stacked denoising autoencoders.

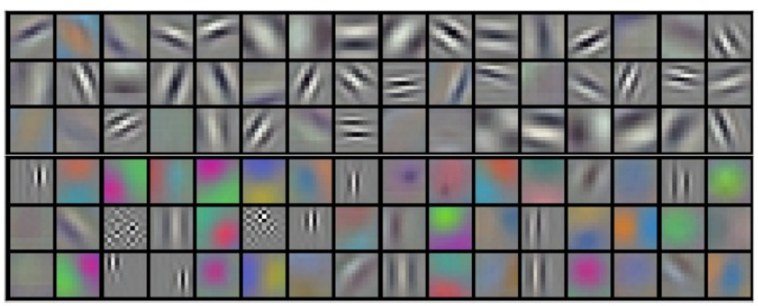
Applications: NIST digit recognition, etc.

Deep Convolutional Neural Networks (1)



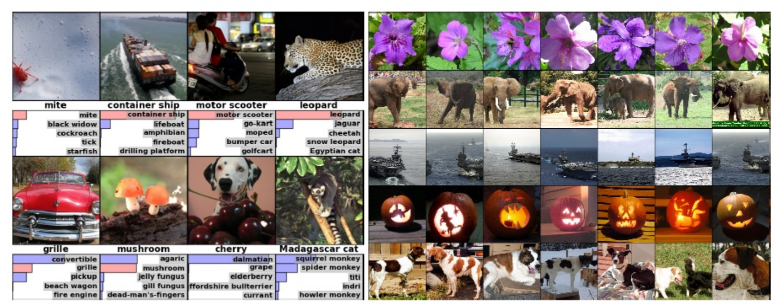
- Krizhevsky et al. (2012)
- Applied to ImageNet competition (1.2 million images, 1,000 classes).
- Network: 60 million parameters and 650,000 neurons.
- Top-1 and top-5 error rates of 37.5% and 17.0%.
- Trained with backprop.

Deep Convolutional Neural Networks (2)



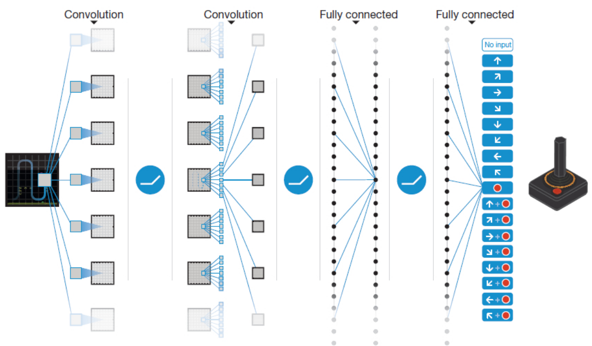
- Learned kernels (first convolutional layer).
- Resembles mammalian RFs: oriented Gabor patterns, color opponency (red-green, blue-yellow).

Deep Convolutional Neural Networks (3)



- Left: Hits and misses and close calls.
- Right: Test (1st column) vs. training images with closest hidden representation to the test data.

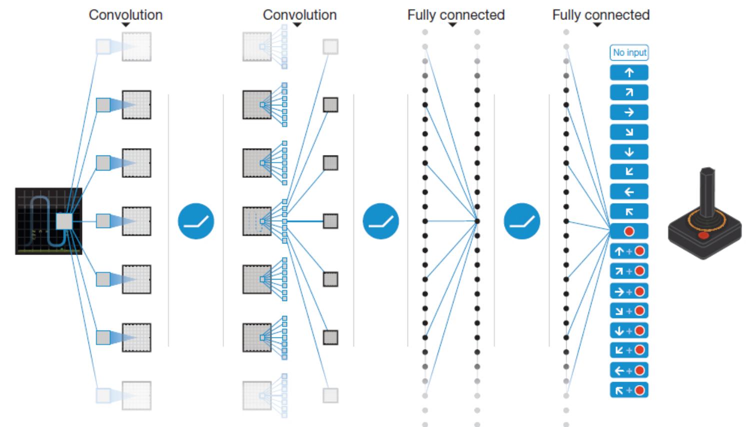
Deep Q-Network (DQN)



Google Deep Mind (Mnih et al. *Nature* 2015).

- Latest application of deep learning to a *reinforcement learning* domain (Q as in Q-learning).
- Applied to *Atari 2600* video game playing.

DQN Overview



- Input: video screen; Output: $Q(s, a)$; Reward: game score.
- $Q(s, a)$: action-value function
  - Value of taking action a when in state s.

DQN Overview

- Input preprocessing
- Experience replay (collect and replay state, action, reward, and resulting state)
- Delayed (periodic) update of Q.
- Moving target $\hat{Q}$ value used to compute error (loss function L, parameterized by weights θ_i).
  - Gradient descent:

$$\frac{\partial L}{\partial \theta_i}$$

DQN Algorithm

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

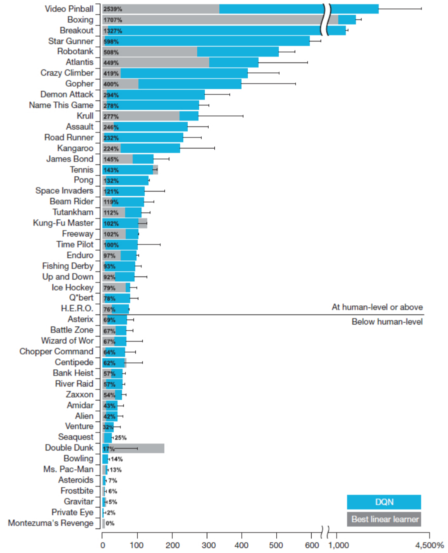
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

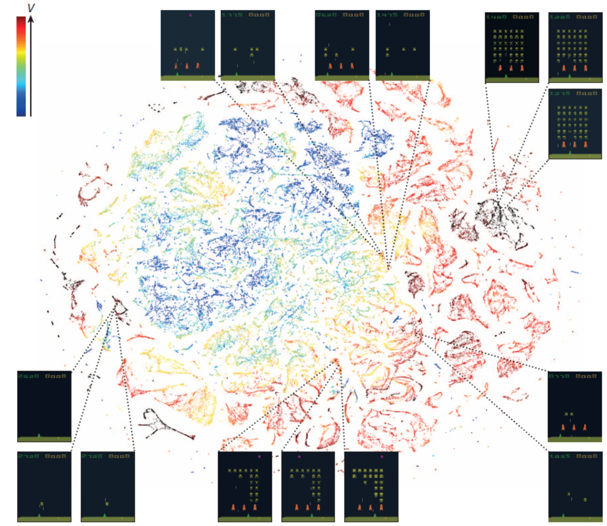
End For

DQN Results



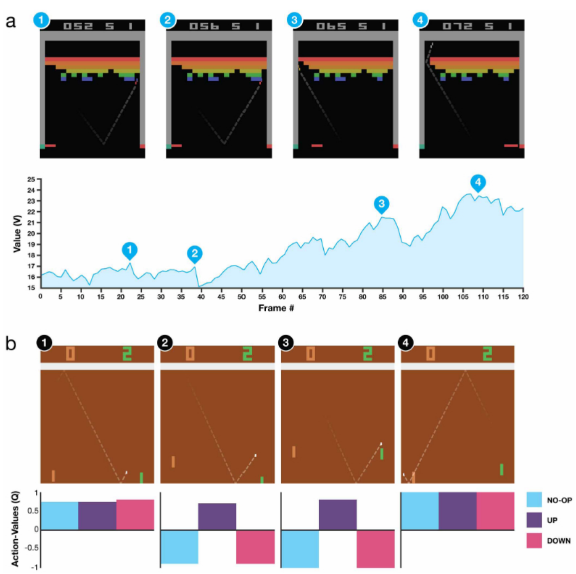
- Superhuman performance on over half of the games.

DQN Hidden Layer Representation (t-SNE map)



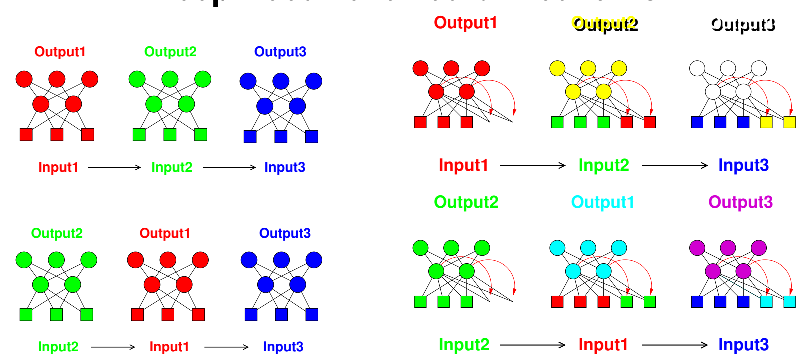
- Similar perception, similar reward clustered.

DQN Operation



- Value vs. game state; Game state vs. action value.

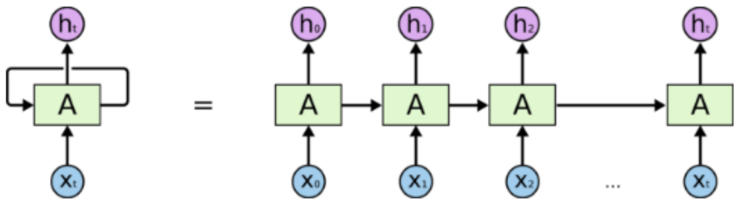
Deep Recurrent Neural Networks



Feedforward **Recurrent**

- Feedforward: No memory of past input.
- Recurrent:
  - Good: Past input affects present output.
  - Bad: Cannot remember far into the past.

RNN Training: Backprop in Time



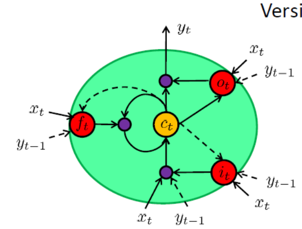
An unrolled recurrent neural network.

- Can unfold recurrent loop: Make it into a feedforward net.
- Use the same backprop algorithm for training.
- Again, cannot remember too far into the past.

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

Long Short-Term Memory

Version 1



i_t, f_t, o_t - input, forget and output gates from 0 to 1

c_t - memory

x_t - input, y_t - output

$$i_t = \sigma(w_{ix}x_t + w_{ic}c_{t-1} + w_{iy}y_{t-1} + b_i)$$

$$f_t = \sigma(w_{fx}x_t + w_{fc}c_{t-1} + w_{fy}y_{t-1} + b_f)$$

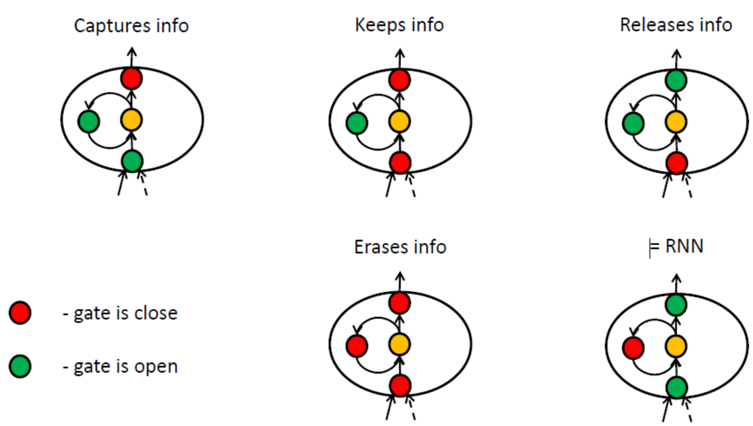
$$o_t = \sigma(w_{ox}x_t + w_{oc}c_t + w_{oy}y_{t-1} + b_o)$$

$$c_t = f_t c_{t-1} + i_t \cdot tanh(w_{cx}x_t + w_{cy}y_{t-1}) \qquad y_t = o_t \cdot tanh(c_t)$$

- LSTM to the rescue (Hochreiter and Schmidhuber, 2017).
- Built-in recurrent memory that can be written (Input gate), reset (Forget gate), and outputted (Output gate).

From http://www.machinelearning.ru/wiki/images/6/6c/RNN_and_LSTM_16102015.pdf

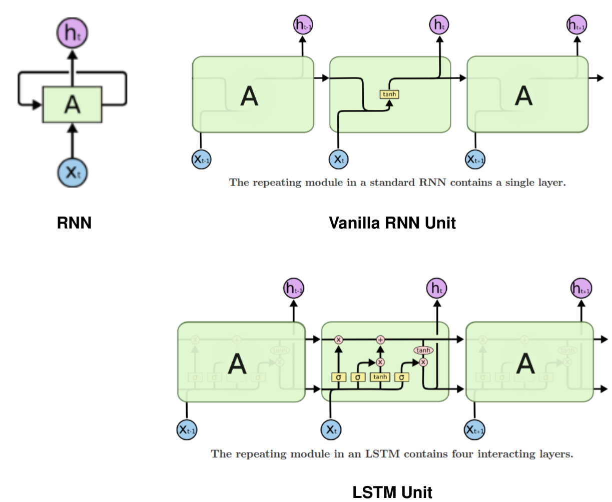
Long Short-Term Memory



- Long-term retention possible with LSTM.

From http://www.machinelearning.ru/wiki/images/6/6c/RNN_and_LSTM_16102015.pdf

Long Short-Term Memory in Action

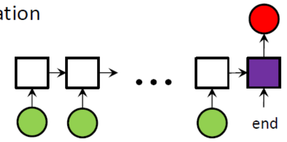


- Unfold in time and use backprop as usual.

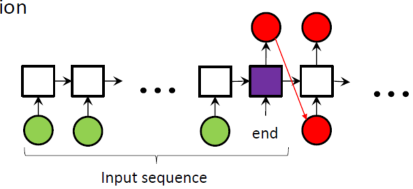
Fig from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Applications

- Sequence classification



- Sequence translation



- Applications: Sequence classification, Sequence translation.

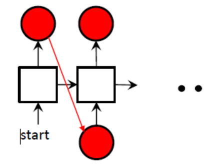
From http://machinelearning.ru

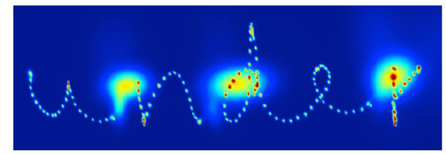
LSTM Applications

handwriting -> handwriting

Next pen position (we predict parameters):
x1,x2 - mixture of bivariate Gaussians
x3 - Bernoulli distribution

Current pen position:
x1,x2 – pen offset
x3 – is it end of the stroke





- Applications: Sequence prediction

From http://machinelearning.ru

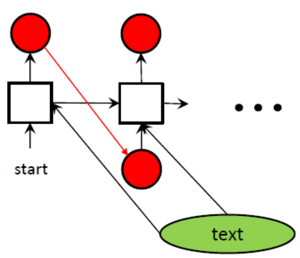
LSTM Applications

text -> handwriting

Next pen position

Current pen position

Which letter we write now



- Applications: Sequence classification, Sequence prediction, Sequence translation.

From http://machinelearning.ru

Deep Learning Applications: Vision

- Give the name of the dominant object in the image
- Top-5 error rates: if correct class is not in top 5, count as error
  - Red:ConvNet, blue: no ConvNet

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

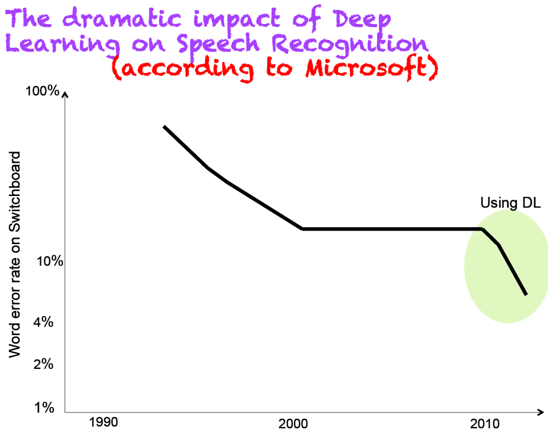
2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

- ConvNet sweeping image recognition challenges.

From LeCun's Deep Learning Tutorial

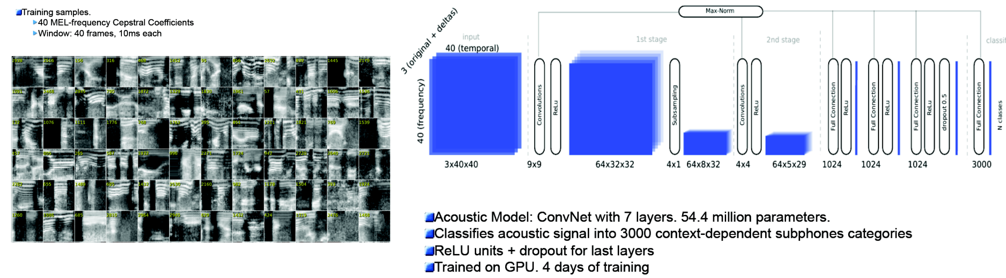
Deep Learning Applications: Speech



- Deep learning led to major improvement in speech recognition.

From LeCun's Deep Learning Tutorial

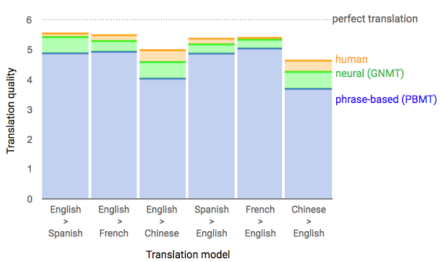
Deep Learning Applications: Speech

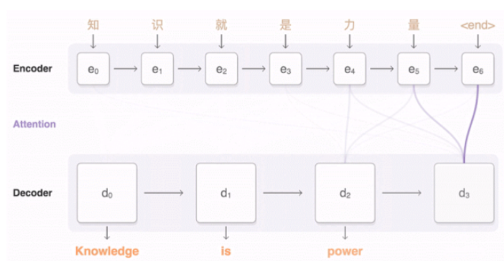


- ConvNet applied to speech recognition.
- Use spectrogram and treat it like a 2D image.

From LeCun's Deep Learning Tutorial

Deep Learing Applications: NLP

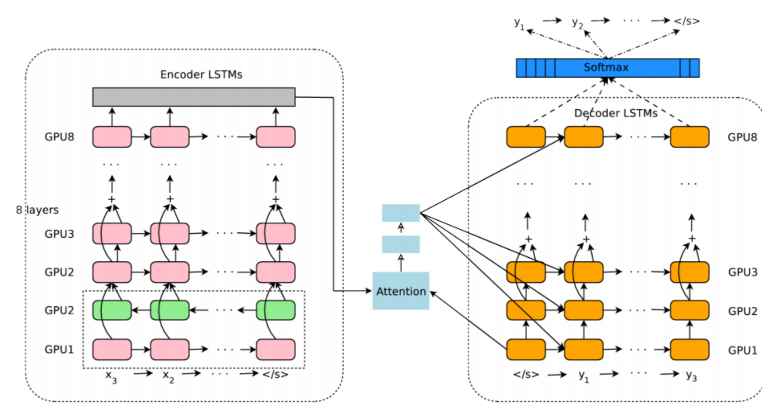




- Based on encoding/decoding and attention.

From https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

Deep Learing Applications: NLP



- Google's LSTM-based machine translation.

Wu et al. *arXiv:1609.08144* (2016).

Limitations

- Discriminative vs. generative learning.
  - Discriminative: $P(class|data)$. Can easily be fooled with adversarial input.
  - Generative: $P(class, data) = P(class|data)P(data)$. Explicitly models the data.
- Deep neural nets mostly use discriminative learning, so can be fooled by adversarial input. Generative adversarial learning can overcome this (Goodfellow et al. *arXiv:1406.2661* (2014)).

Deep Learning Tools

- Kaffe: UC Berkeley's deep learning tool box
- TensorFlow (Google)
- Deep learning modules for Torch (Facebook)
- Microsoft CNTK (Computational Network Tool Kit)
- Other: Apache Mahout (MapReduce-based ML)

Summary

- Deep belief network: Based on Boltzmann machine. Elegant theory, good performance.
- Deep convolutional networks: High computational demand, over the board great performance.
- Deep Q-Network: unique apporach to reinforcement learning. End-to-end machine learning. Super-human performance.
- Deep recurrent neural networks: sequence learning. LSTM a powerful mechanism.
- Diverse applications. Top performance.
- Flood of deep learning tools available.