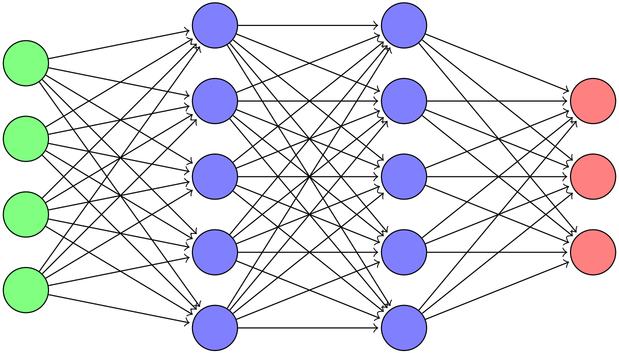
Machine Learning

Artificial neural networks

Maxime Gasse



Course overview

Lecture: 1 x 2h

- perceptron (1958-60s);
- backpropagation (1986-90s);
- deep learning (2006-today).

Practical work: 1 x 4h

- python + pytorch;

Evaluation: final exam + practical work.

Lecture sources:

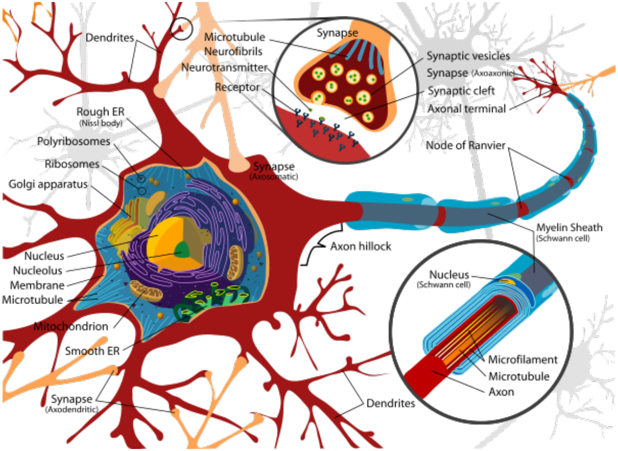
A. Kurenkov (2015). A 'Brief' History of Neural Nets and Deep Learning. http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

I. Goodfellow, Y. Bengio, and A. Courville (2016). Deep Learning.

Perceptron era (1958-60s)

Biological neurons

Extremely complicated cells.

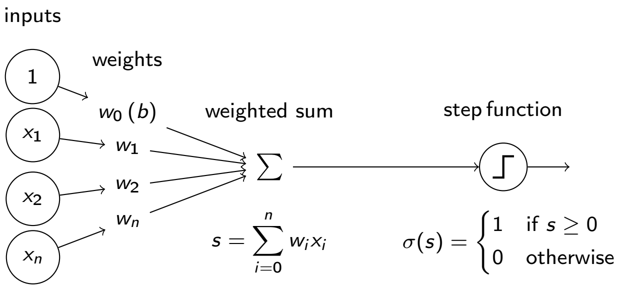


In a nutshell:

- Receives electrical pulses from dendrites (input).
- Accumulates electrical potential.
- Above a certain threshold, fires off along its axon (output).

McCuloch-Pitts artificial neuron

A simplified mathematical model of how neurons operate.


inputs
weights
1
x_1
x_2
x_n
w_0 (b)
w_1
w_2
w_n
weighted sum
step function
$s = \sum_{i=0}^{n} w_i x_i$
$\sigma(s) = \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases}$

Can model logical gates: AND, OR, NOT.

Remember linear SVM ? $h(\mathbf{x}) = sign(\mathbf{w}^\intercal \mathbf{x} + b)$.

W. S. McCulloch and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity.

Rosenblatt's perceptron

$\mathcal{D}$ a training set of input-output pairs $(\mathbf{x}, y)$, with $\mathbf{x} \in \mathbb{R}$ and $y \in \{0, 1\}$.

Start with random weights, and iterate over the training examples

- if target is 1 and output is 0:
  - increase weights w_i where x_i is positive
  - decrease weights w_i where x_i is negative
- if target is 0 and output is 1:
  - decrease weights w_i where x_i is positive
  - increase weights w_i where x_i is negative

Formally: $\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - \sigma(\mathbf{w}^\intercal \mathbf{x}))\mathbf{x}$, with α the learning rate.

Gradient descent ?

F. Rosenblatt (1957). The perceptron, a perceiving and recognizing automaton. Tech. rep. Project Para., Cornell Aeronautical Laboratory

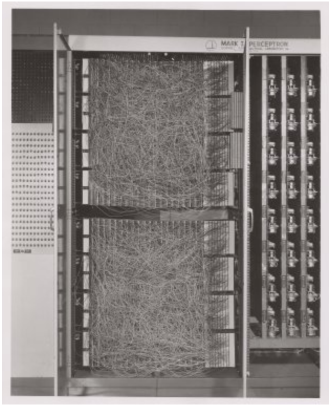
Rosenblatt's perceptron

Custom hardware implementation:
Mark I Perceptron.

- input: 20×20 photocells
- weights: potentiometers
- learning: electric motors

Classifies simple shapes correctly.

Machine learning is born!



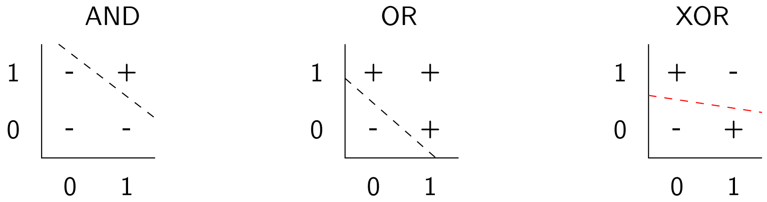
«The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself an be conscious of its existence [...] Dr. Frank Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers. »

New-York Times, 1958

1969: first AI winter

Critical analysis by Minsky and Papert (MIT lab).

Linearity limits, can not learn the simple XOR function.



Non-linearity requires a multilayer perceptron (MLP).

Rosenblatt's algorithm does not work for multiple layers.

$\Longrightarrow$ Massive disillusionment, freeze to funding and publications.

M. Minsky and S. Papert (1969). Perceptrons. An Introduction to Computational Geometry.

Backpropagation era (1986-90s)

Stochastic gradient descent

Gradient descent (GD): $\theta \leftarrow \theta - \alpha \Delta\theta$, with α the learning rate.
$\implies$ reaches a local minima if α small enough (or decaying).

Batch GD: cost over the whole training set

$$\Delta\theta = \frac{\partial \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} L(\mathbf{h}(\mathbf{x}),\mathbf{y})}{\partial\theta} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \frac{\partial L(\mathbf{h}(\mathbf{x}),\mathbf{y})}{\partial\theta}$$

Stochastic GD: cost over a single example $(\mathbf{x},\mathbf{y}) \in \mathcal{D}$

$$\Delta\theta = \frac{\partial L(\mathbf{h}(\mathbf{x}),\mathbf{y})}{\partial\theta}$$

Mini-batch GD: cost over a small subset $\mathcal{D}_{mb} \subset \mathcal{D}$

$$\Delta\theta = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_{mb}} \frac{\partial L(\mathbf{h}(\mathbf{x}),\mathbf{y})}{\partial\theta}$$

Often, many noisy updates converge faster than one unbiased update.
Also, can escape local minima !

Error back-propagation

Idea proposed in the 70s, rediscovered in the 80s by several groups.

$$\begin{tikzcd}
\text{input} & \text{hidden} & \text{output} & \\
x \arrow[r] & z \arrow[r] & h \arrow[r] & L(h, y) \\
 & f(x) & g(z) &
\end{tikzcd}$$

Chain rule of calculus: $\dfrac{\partial h}{\partial x} = \dfrac{\partial h}{\partial z}\dfrac{\partial z}{\partial x}$.

$\Longrightarrow$ all we need is L and σ differentiable.

P. Werbos (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis. Harvard University, Cambridge, MA

D. E. Rumelhart, G. E. Hinton, and R. J. Williams (1986). Learning representations by back-propagating errors.

Error back-propagation

Example (squared error, identity activation function)

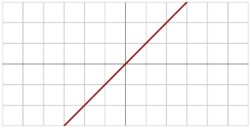
$$\begin{tikzcd}
 & \text{input} & \text{hidden} & \text{output} & \\
\text{Forward} & x \arrow[r] & z \arrow[r] & h & \\
 & & w^{(1)}x + b^{(1)} & w^{(2)}z + b^{(2)} & err = \frac{1}{2}(y-h)^2 \\
\text{Backward} & \frac{\partial z}{\partial x} & \frac{\partial h}{\partial z} \arrow[l] & \frac{\partial err}{\partial h} \arrow[l] & \\
 & w^{(1)} & w^{(2)} & y - h &
\end{tikzcd}$$

$$\Delta w^{(2)} = \frac{\partial err}{\partial h}\frac{\partial h}{\partial w^{(2)}} = (y-h)(w^{(1)}x + b^{(1)}) \qquad \Delta b^{(2)} = \frac{\partial err}{\partial h}\frac{\partial h}{\partial b^{(2)}} = (y-h)$$

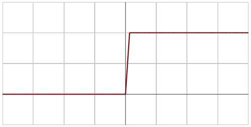
$$\Delta w^{(1)} = \frac{\partial err}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial w^{(1)}} = (y-h)w^{(2)}x \qquad \Delta b^{(1)} = \frac{\partial err}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial b^{(1)}} = (y-h)w^{(2)}$$

Activation functions

Identity: linear model...



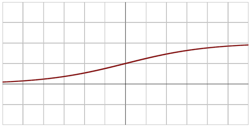
Step function: non-differentiable...



Logistic (sigmoid)

$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$

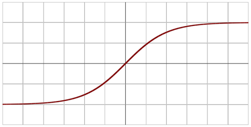
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$



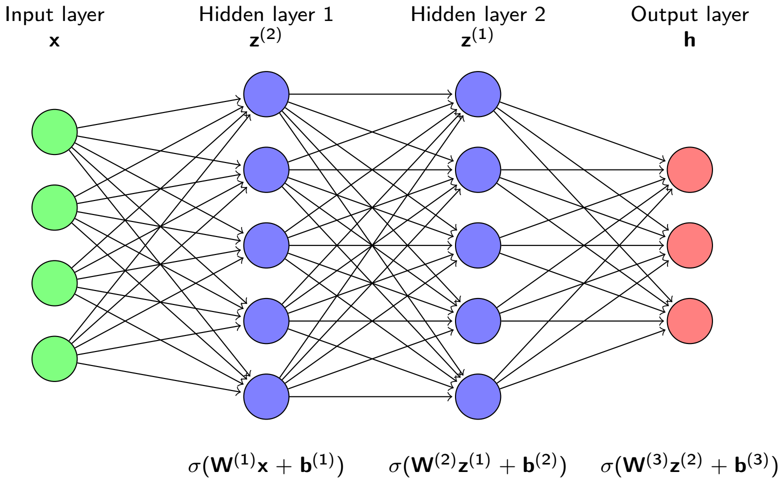
Hyperbolic tangent (tanh)

$$\sigma(x) = \frac{2}{1 + \exp^{-2x}} - 1$$

$$\frac{\partial \sigma(x)}{\partial x} = 1 - \sigma(x)^2$$



Multilayer feed-forward neural network

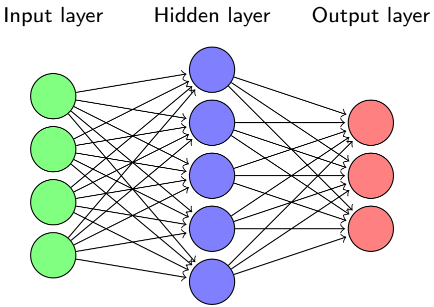

Input layer
$\mathbf{x}$
Hidden layer 1
$\mathbf{z}^{(2)}$
Hidden layer 2
$\mathbf{z}^{(1)}$
Output layer
$\mathbf{h}$
$\sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$
$\sigma(\mathbf{W}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)})$
$\sigma(\mathbf{W}^{(3)}\mathbf{z}^{(2)} + \mathbf{b}^{(3)})$

Demo time: `http://playground.tensorflow.org`

MLPs are universal approximators

A single hidden layer with sufficiently many hidden units can approximate any $\mathcal{X} \mapsto \mathcal{Y}$ mapping to any desired degree of accuracy.

I other words: infinite capacity.

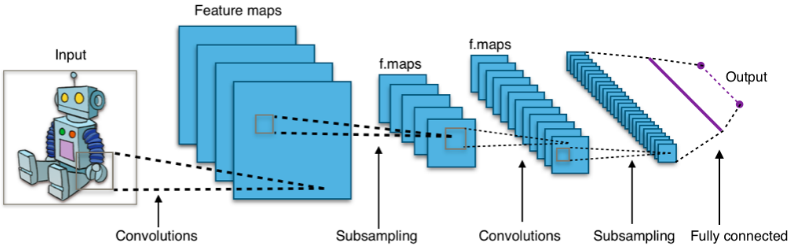


Strong theoretical argument at the time. Still, are NNs useful ?

- k-NN has infinite capacity;
- decision trees have infinite capacity;
- requires an exponential number of hidden neurons.

K. Hornik, M. B. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators.

Convolutional neural networks



Typical CNN layer: convolution + max pooling

- convolutions → weight sharing, less free parameters
- max pooling → translation invariance, dimensionality reduction

Leverages spatial / temporal structures

- image processing (2D spatial convolutions)
- signal processing (1D temporal convolution)

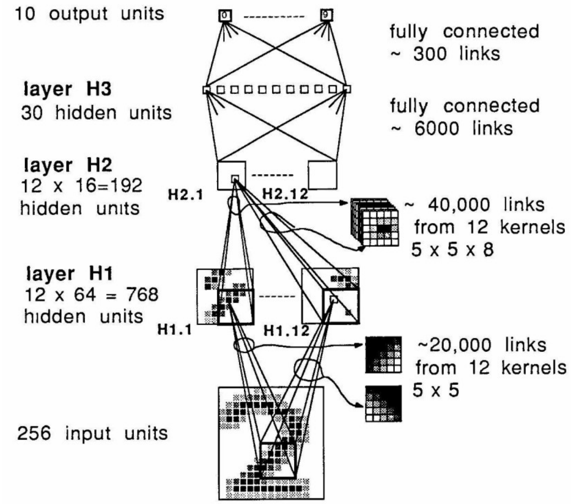
Handwritten digit recognition: LeNet-5

SUN 4/260
16.67 MHz, 128 MB

- *tanh* activations
- L_2 loss
- Stochastic GD

`https://youtu.be/FwFduRA_L6Q`

At some point in the 90s, read about 10-20% of all the checks in the US.


10 output units
0
9
fully connected ~ 300 links
layer H3
30 hidden units
fully connected ~ 6000 links
layer H2
12 x 16=192 hidden units
H2.1
H2.12
~ 40,000 links from 12 kernels 5 x 5 x 8
layer H1
12 x 64 = 768 hidden units
H1.1
H1.12
~20,000 links from 12 kernels 5 x 5
256 input units

Y. LeCun et al. (1989). Backpropagation Applied to Handwritten Zip Code Recognition.

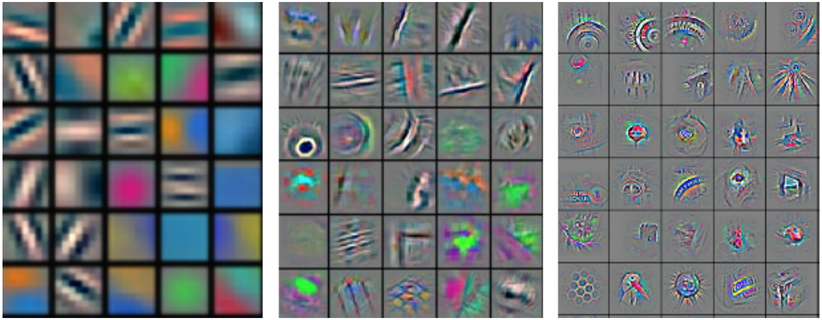
Convolutional neural networks

CNN architecture = strong model prior (regularizer)

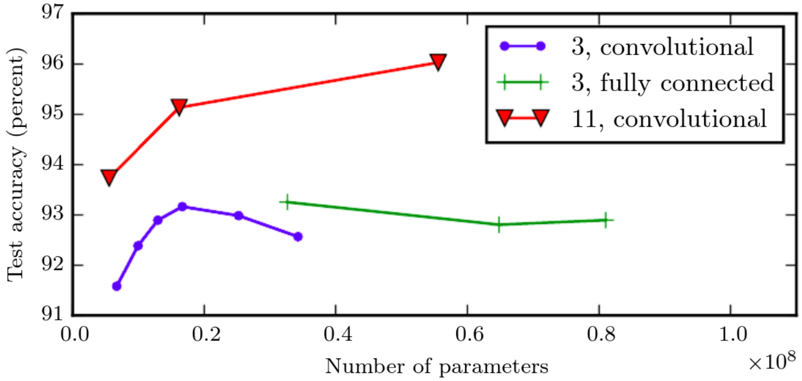
- more stable training
- better generalization

Intuition:

- lower layers = low-level features (edges, blobs)
- higher layers = high-level features



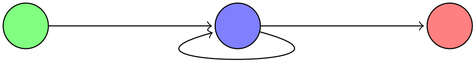
Deep versus shallow networks



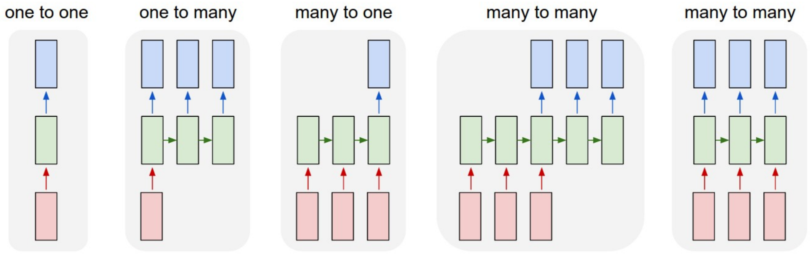
Convolutional architectures are a good prior for visual recognition.

Recurrent neural networks

Time-delayed connections $\Longrightarrow$ memory.



Can process non i.i.d. data (sequences):



Back-propagation through time (BPTT).
$\Longrightarrow$ vanishing / exploding gradients problem...

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

The vanishing gradient problem

Chain rule $\approx$ geometric sequence

- derivatives $\frac{\partial\sigma(x)}{x} < 1 \implies$ vanishing gradient
- derivatives $\frac{\partial\sigma(x)}{x} > 1 \implies$ exploding gradient

Related to:

- activation function σ
- initial weight initialization

Affects MLPs with many layers, and recurrent neural nets.

- hard to train deep networks;
- hard to learn long-term dependencies.

Limitations

Gradient descent limits: no theoretical guarantee

- non-convex optimization problem (hidden layers)
- quality of the local minimum unknown (gradient descent)

Backpropagation limits: vanishing / exploding gradients

- training of deep / recurrent networks is problematic

Limited computer resources at the time

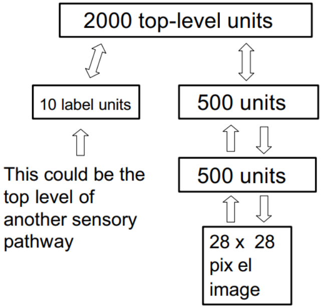
- training is slow

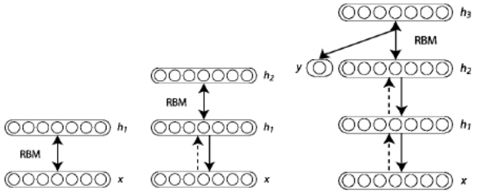
Outperformed by other models in the 2000s (random forest, SVM).

Despite some irrefutable successes, neural networks lose popularity.
$\Longrightarrow$ Second AI winter.

Deep learning era (2006-)

2006 Deep belief network (DBN)





(a) Train RBM for $\mathbf{x}$ (b) Train RBM for $\mathbf{h}^1$ (c) Train RBM for $\mathbf{h}^2$ and y

Unsupervised layer-by-layer pre-training (weight initialization).
$\Longrightarrow$ successful training of deep neural nets possible.

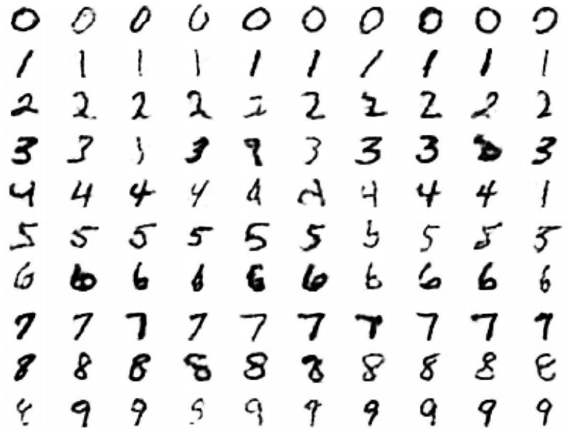
1.25% error on MNIST without convolutions ! (0.95% with CNNs)

Generative model of digit images (MNIST).

G. E. Hinton, S. Osindero, and Y. W. Teh (2006). A Fast Learning Algorithm for Deep Belief Nets.

2006 Resurgence: Deep belief network (DBN)

Generative model: "Looking into the mind of a neural network".



G. E. Hinton, S. Osindero, and Y. W. Teh (2006). A Fast Learning Algorithm for Deep Belief Nets.

Computational power

Back-propagation highly parallel by nature.

Why not using GPUs instead of CPUs for training ?
$\Longrightarrow$ 70x speed-up.

DBN training by Google:

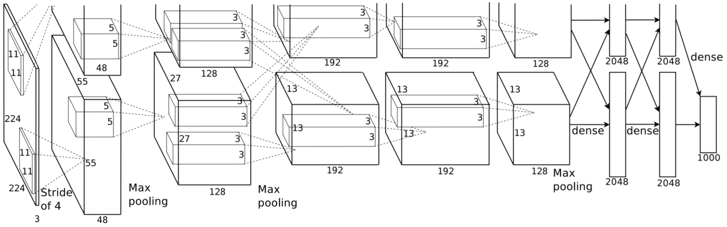
- 16000 parallel CPUs
- $\sim$ 1 billion weights
  (Hinton's 2006 DBN $\sim$ 1 million)

Computational limitations unlocked.

R. Raina, A. Madhavan, and A. Y. Ng (2009). Large-scale deep unsupervised learning using graphics processors.
Q. V. Le et al. (2012). Building high-level features using large scale unsupervised learning.

2012 Tsunami: AlexNet

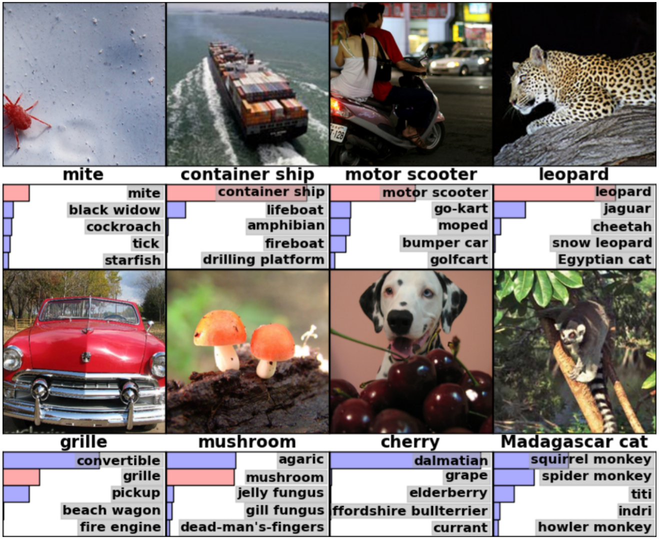
Successful supervised training of a deep convolutional network on GPUs.



- one week training on two gaming GPUs (Nvidia GTX 580, 3GB)
- 5 convolutional layers + 3 fully-connected layers
- relu activation functions (non-saturating neurons)
- dropout (regularization)
- data augmentation (cropping + translations)

A. Krizhevsky, I. Sutskever, and G. E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks.

2012 Tsunami: AlexNet



A. Krizhevsky, I. Sutskever, and G. E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks.

2012 Tsunami: AlexNet

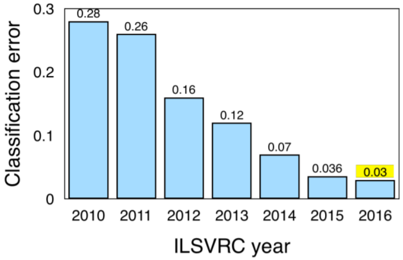
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012

- crowd-sourced image annotation
- 1000 classes
- 1.3 million training examples

The only NN competitor, AlexNet, wins by a large margin

- 15.3% top-5 error rate, compared to 26.2% for second entry

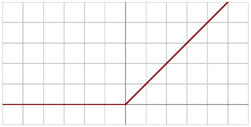
Next years: overflow of CNN competitors, test error continually decreases



Modern activation functions

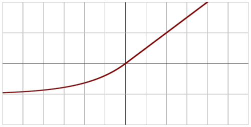
Rectified linear unit (relu)

$$\sigma(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



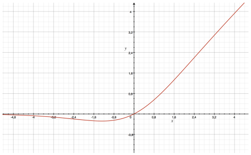
Exponential linear unit (elu)

$$\sigma(x) = \begin{cases} x & \text{if } x \geq 0 \\ \exp(x) - 1 & \text{otherwise} \end{cases}$$



Google's activation (swish)

$$\sigma(x) = \frac{x}{1 + \exp -x}$$



$\implies$ non-saturating neurons.

Google deep learning research :)

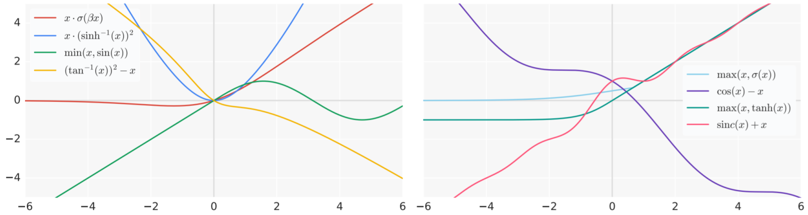
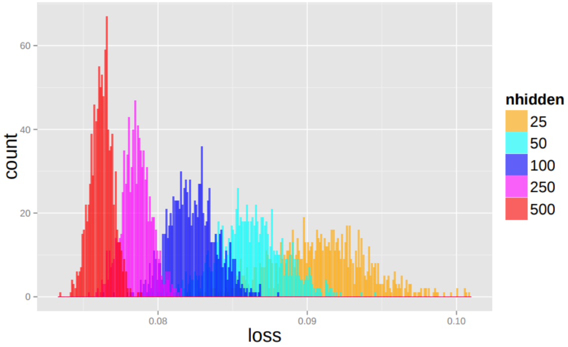


Figure 3: The top novel activation functions found by the searches. Separated into two diagrams for visual clarity. Best viewed in color.

P. Ramachandran, B. Zoph, and Q. Le (2017). Swish: a Self-Gated Activation Function.

Deep learning analogy to spin glasses

«For large-size decoupled networks the lowest critical values [. . .] are located in a well-defined band lower-bounded by the global minimum. The number of local minima outside that band diminishes exponentially with the size of the network. »



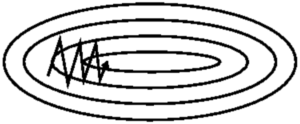
$\Longrightarrow$ local minimum is not a problem with large networks.

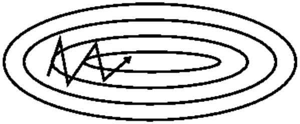
A. Choromanska et al. (2015). The Loss Surfaces of Multilayer Networks.

Modern gradient descent algorithms

Momentum:

$$\Delta\theta \leftarrow \beta\Delta\theta + (1-\beta)\frac{\partial L}{\partial\theta} \quad \text{with } \beta \in]0,1[,$$
$$\theta \leftarrow \theta - \alpha\Delta\theta.$$


No momentum


Momentum

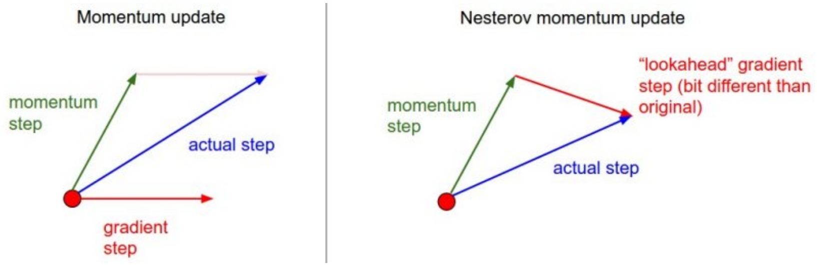
$\Longrightarrow$ faster convergence, especially with SGD.

Modern gradient descent algorithms

Nesterov momentum:

$$\Delta\theta \leftarrow \beta\Delta\theta + (1-\beta)\frac{\partial L}{\partial\theta - \alpha\beta\Delta\theta} \quad \text{with } \beta \in]0,1[,$$

$$\theta \leftarrow \theta - \alpha\Delta\theta.$$



$\Longrightarrow$ often a bit faster than vanilla momentum.

Modern gradient descent algorithms

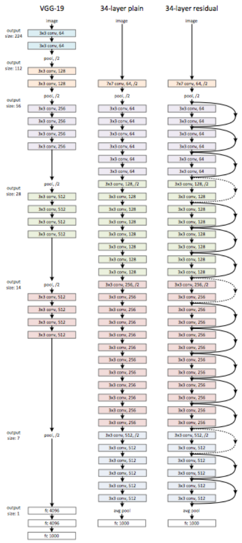
Adaptive moment estimation (adam): first and second order statistics

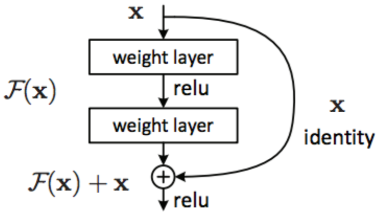
$$
\begin{aligned}
m &\leftarrow \beta_1 m + (1-\beta_1)\frac{\partial L}{\partial \theta} && \text{(mean)},\\
v &\leftarrow \beta_2 v + (1-\beta_2)\left(\frac{\partial L}{\partial \theta}\right)^2 && \text{(uncentered variance)},\\
\hat{m} &= \frac{m}{1-\beta_1} && \text{(bias correction)},\\
\hat{v} &= \frac{v}{1-\beta_2} && \text{(bias correction)},\\
\theta &\leftarrow \theta - \alpha \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon}.
\end{aligned}
$$

$\implies$ really fast, used consistently for deep learning.

D. P. Kingma and J. L. Ba (2015). Adam: a Method for Stochastic Optimization.
A. Veit, M. J. Wilber, and S. J. Belongie (2016). Residual Networks Behave Like Ensembles of Relatively Shallow Networks.

Residual networks



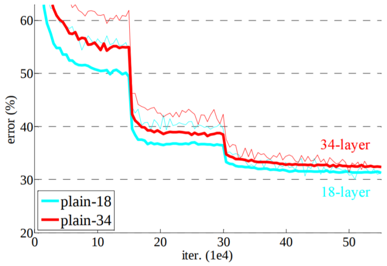


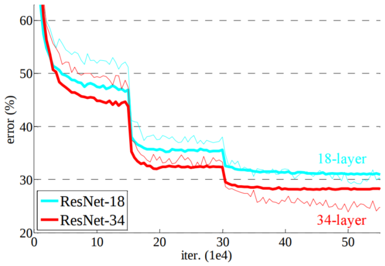
A residual block

$\Longrightarrow$ state-of-the-art on many tasks (AlphaGo Zero)

K. He et al. (2015). Deep Residual Learning for Image Recognition.

Residual networks





Scales up to thousands of layers (but only exploits paths of $<$ 30 layers).

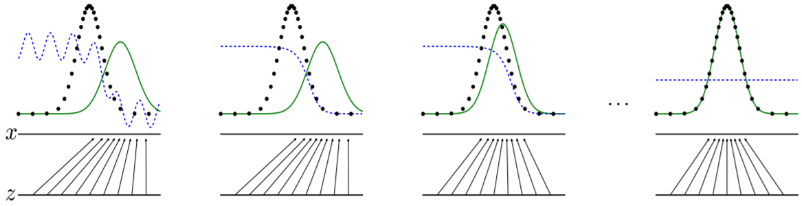
K. He et al. (2015). Deep Residual Learning for Image Recognition.

Generative Adversarial Networks (GANs)

«Adversarial training is the coolest thing since sliced bread. »
Yann LeCun

Observations $\mathbf{x} \sim p(\mathbf{x})$, samples $\mathbf{z} \in [0,1]^M$ (latent space).

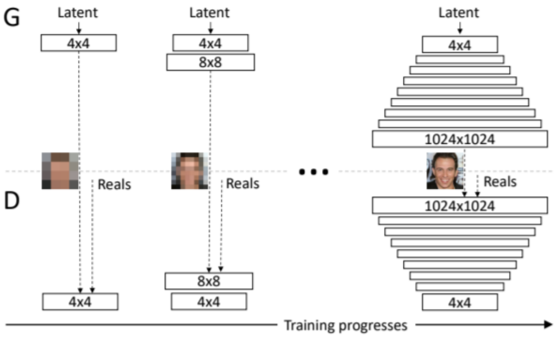
Generator network $G : \mathcal{Z} \to \mathcal{X}$.
Discriminator network $D : \mathcal{X} \to [0,1]$.
$G^\star = \arg\max_G \mathbb{E}_{\mathbf{z}}[D(G(\mathbf{z}))]$
$D^\star = \arg\max_D \mathbb{E}_{\mathbf{x},\mathbf{z}}[D(\mathbf{x}) - D(G(\mathbf{z}))]$

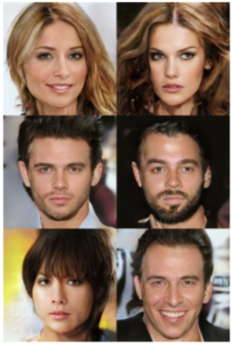


Optimal point: Nash equilibrium.

I. J. Goodfellow et al. (2014). Generative Adversarial Nets.

Realistic face image generation





`https://youtu.be/XOxxPcy5Gr4`

T. Karras et al. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation.