Imperial College London

Neural Networks

Machine Learning

QSRI summer school – July 2022

Sarah Filippi

Department of Mathematics
Imperial College London

Neural networks

- Neural networks can be used for supervised learning tasks.
- They have demonstrated predictive power on complicated tasks – when trained with a**large number of samples**.
- The representational power of neural networks comes at the price of a non-convex loss function, and a potentially large set of parameters.
- As for other parametric approaches, the parameters are inferred during the training phase by optimising a loss function using gradient-based method.
- The field of Neural network has evolved into what is now known as **deep learning**

Some applications

Outline

From a single neuron classifier to a multilayer perceptron

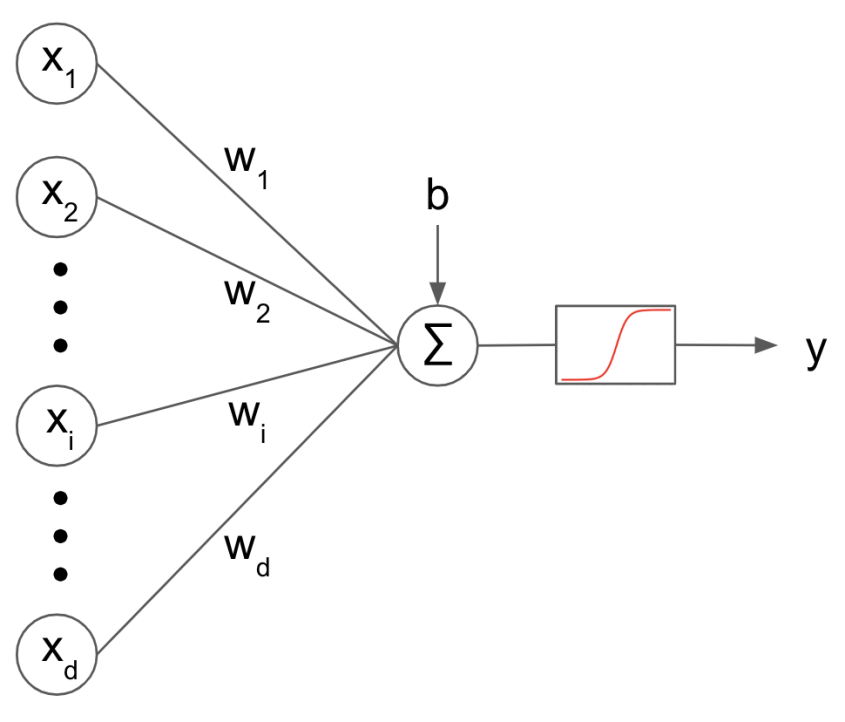
Training a neural network

Other types of networks such as convolutional neural networks

Neural networks

- A common type of neural network is called a **feedforward network** or a **multilayer perceptron** (MLP)
- The MLP is constructed by stacking hidden layers of logistic regression functions, terminating in an output prediction layer

Single Neuron Classifier

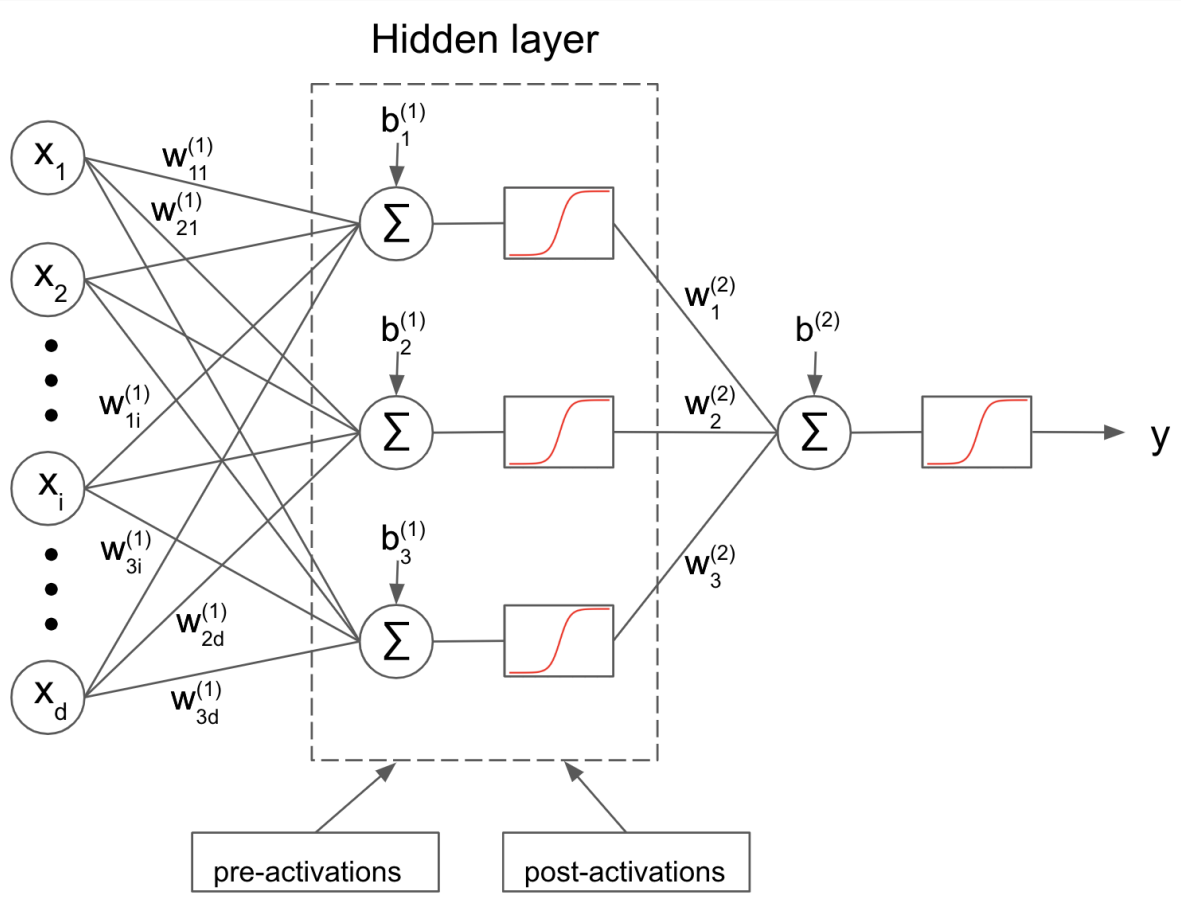


- inputs: $x = (x_1, \ldots x_d)$
- activation: $b + \sum_{j=1}^{d} w_j x_j$ where b is the bias and $w_1 \ldots w_d$ are the weights
- activation/transfer function: $\sigma(\cdot)$

For logistic regression: $\sigma(a) = \frac{1}{1-e^{-a}}$

Multilayer perceptron for classification

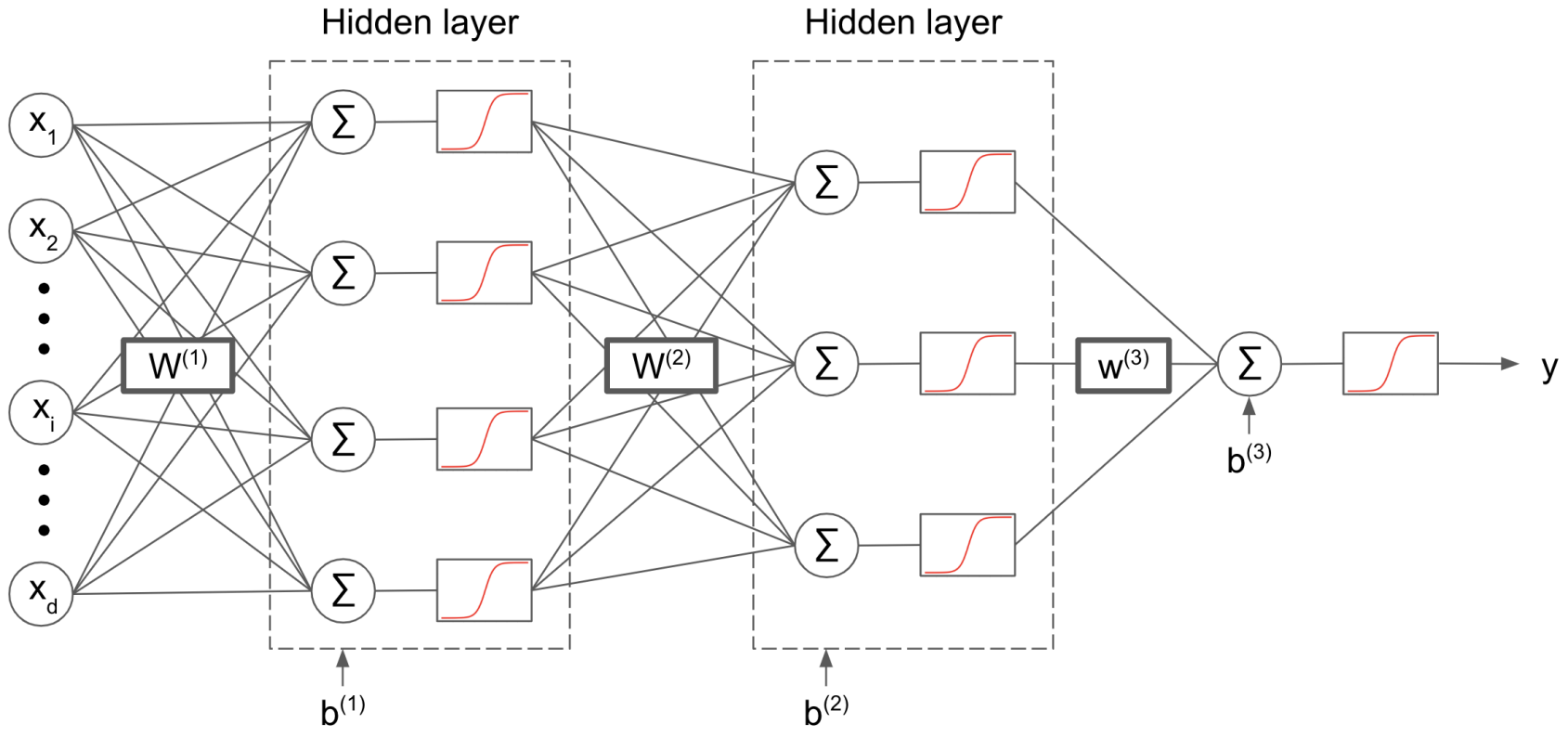
An artificial neural network increases the model flexibility by creating an intermediate layer of neurons, the outputs of which feed into the final output neuron:



How many model parameters if $d = 10$? (menti.com with code 6092 9192)

Multilayer perceptron for classification

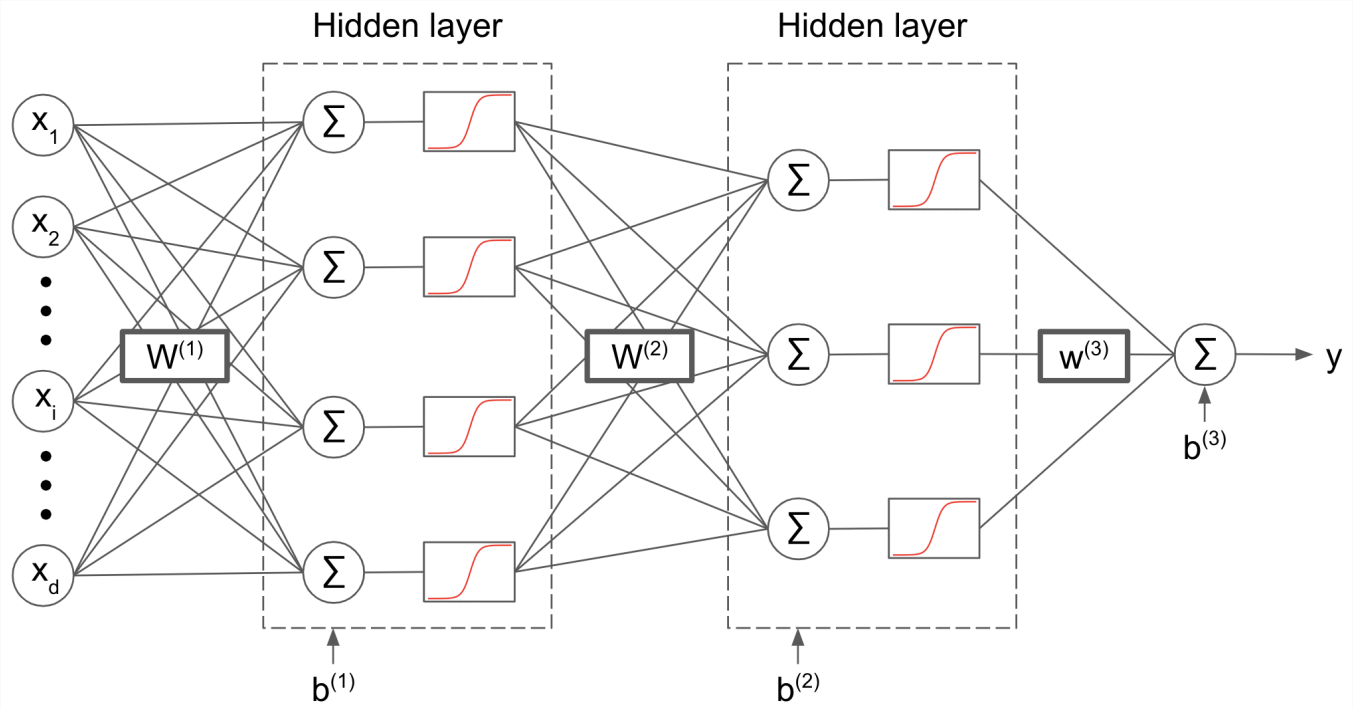
In addition, hidden layers can be stacked up to create deeper networks.



Additional hidden layers increase the flexibility of the function approximation for the distribution $\mathbb{P}(y = 1|x)$.

If $d = 10$, this model has $(10 + 1) \times 4 + (4 + 1) \times 3 + (3 + 1) = 63$ parameters.

Multilayer perceptron for regression



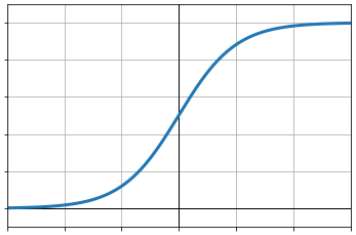
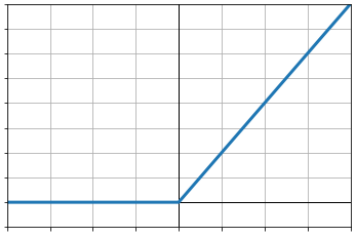
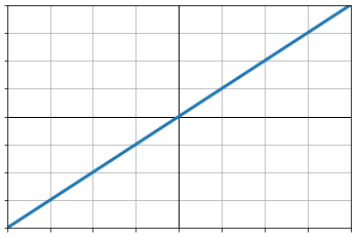
- An MLP can also be used to solve regression problems. The hidden layers are constructed in the same way
- The only difference is in the output layer, which is now a linear combination of the output of the neurons in the last hidden layer

Neural network components

There is a great deal of flexibility when it comes to constructing neural network. One needs to carefully choose:

- the number of hidden layers
- the number of nodes per layer
- the nonlinear activation functions.

Activation functions

Activation function	Plot	Equation
sigmoid		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$
ReLU		$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$
identity		$f(x) = x$

Output layers

Frequently used output activations / layers:

- **Sigmoid output layer** is useful to predict probabilities as the range is in $(0, 1)$.
- **Softplus output layer** is useful to predict e.g. Gaussian variance parameters, as the range is $(0, \infty)$:

$$f(x) = \ln(1 + e^x)$$

- **Softmax output layer** is often used to predict parameters of a categorical distribution:

$$P(\text{Category } j) = \frac{e^{a_j}}{\sum_{k=1}^{M} e^{a_k}},$$

where a_k ($k \in \{1, \ldots, M\}$) are the pre-activations.

Outline

From a single neuron classifier to a multilayer perceptron

Training a neural network

Other types of networks such as convolutional neural networks

Training a Neural Network

- Specify a loss function $\mathcal{L}(\mathcal{D}; \theta)$, e.g. squared error loss (regression), Log Loss (classification).
- Use a method called backpropagation (based on the chain rule) to compute $\nabla_\theta \mathcal{L}$.
- Use gradient descent to update θ:

$$\theta^{t+1} = \theta^t - \epsilon \nabla_\theta \mathcal{L}(\mathcal{D}; \theta^t)$$

Training a Neural Network in Practice

$$\theta^{t+1} = \theta^t - \epsilon \nabla_\theta \mathcal{L}(\mathcal{D}; \theta^t)$$

- Given a huge dataset, computing the gradient $\nabla_\theta \mathcal{L}(\mathcal{D}; \theta^t)$

  $$\nabla_\theta \mathcal{L}(\mathcal{D}; \theta^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(x^{(i)}, y^{(i)}; \theta^t)$$

  is too expensive.

- Solution: **stochastic gradient descent** (SGD). Use a small random subset of the data (minibatch $\mathcal{B}$) to approximate the gradient:

  $$\nabla_\theta \mathcal{L}(\mathcal{D}; \theta^t) \approx \nabla_\theta \mathcal{L}(\mathcal{B}; \theta^t)$$

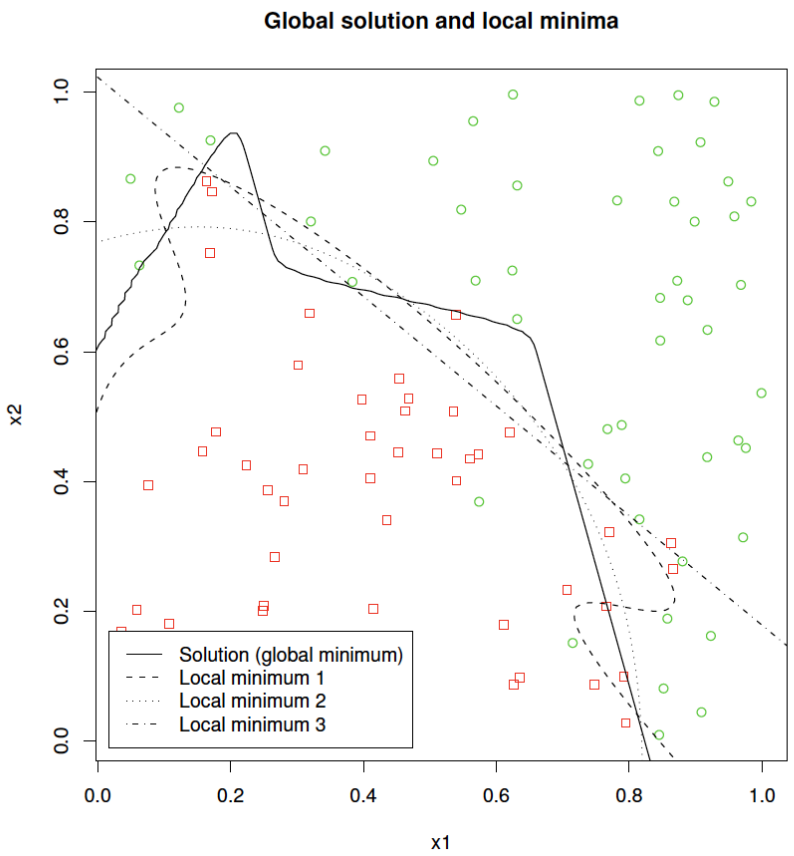
- Consider splitting $\mathcal{D}$ into (roughly) equally-sized minibatches $\mathcal{B}_1, \ldots, \mathcal{B}_k$. Size of each minibatch is the **batch size**. One pass through all of the minibatches is called an "epoch". An epoch takes k iterations.

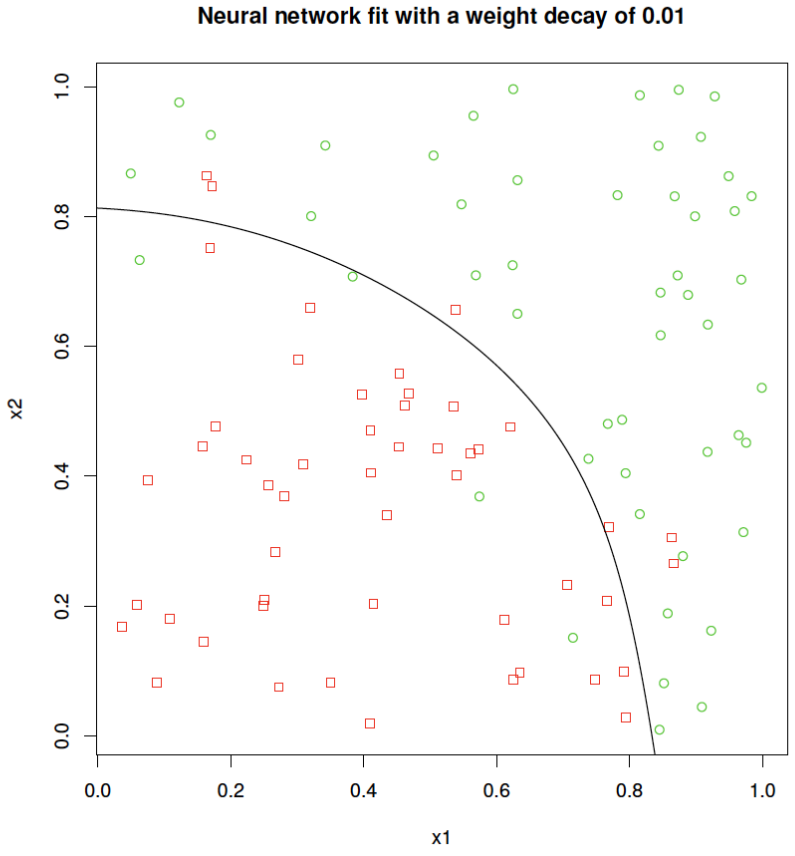
Regularisation

Regularisation is important to avoid overfitting. There are several approaches to regularisation in deep learning including

- Weight decay (L2)
- Patience/early stopping
- Dropout
- Weight sharing

Example


Global solution and local minima
x2
x1
0.0
0.2
0.4
0.6
0.8
1.0
Solution (global minimum)
Local minimum 1
Local minimum 2
Local minimum 3


Neural network fit with a weight decay of 0.01
x2
x1
0.0
0.2
0.4
0.6
0.8
1.0

Neural network in R

- There exists many packages.
- None are ideal.
- In the tutorial today: `deepnet` package as we only consider simple networks in this course.
- For more complex networks you can use the R Interface to Keras, which is a wrapper around the Python module TensorFlow.

Outline

From a single neuron classifier to a multilayer perceptron

Training a neural network

Other types of networks such as convolutional neural networks

Deep learning

- The 'deep' in deep learning refers to the number of hidden layers in the network
- In general, greater depth in networks allow for a richer class of approximating functions
- Intuition is that deep networks allow the model to build hierarchies of concepts
- Concepts are built on top of each other in layers
- This reduces the need for hand-engineered features (basis functions)
- Deep learning can be seen as part of *representation learning*, which aims to discover the best representations or features of the data

Convolutional neural networks

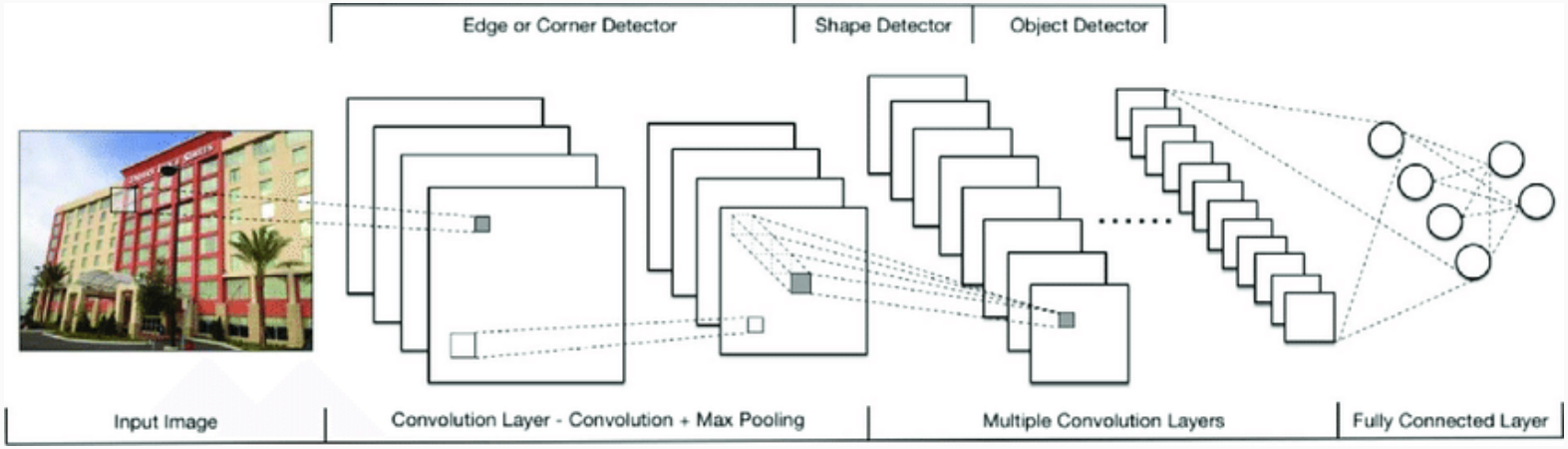
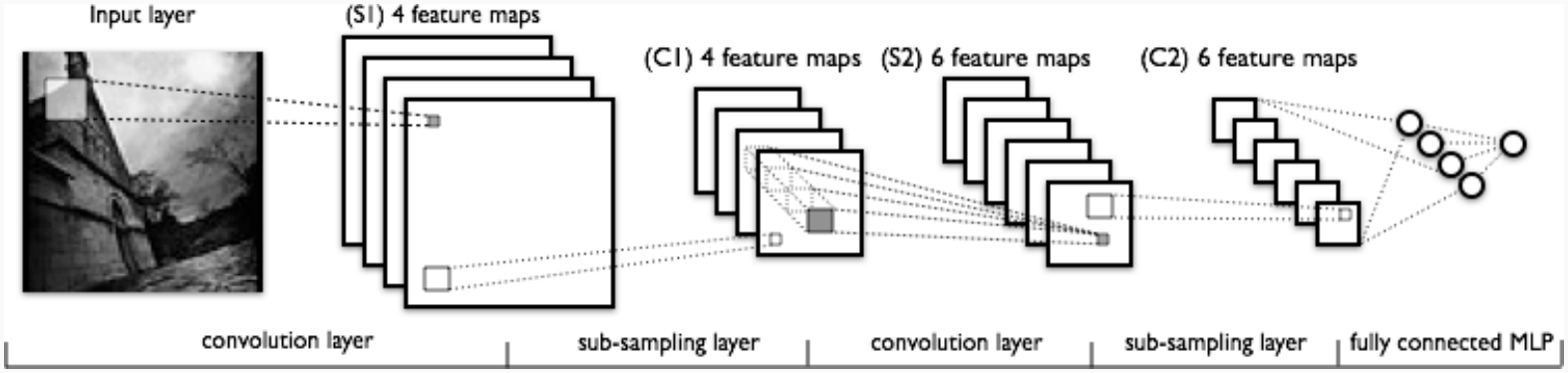


Image from the article "Effects of user-provided photos on hotel review helpfulness: An analytical approach with deep leaning" by Ma et al (2018)

- **Convolutional neural network** (CNN or ConvNet) is a class of neural network with a special structure
- Breakthrough improvements in image processing
- Also used in NLP, audio waveform analysis and generation, and Reinforcement Learning models for games

Convolutional Neural Networks



- Input is a black-and-white 2D image, $X \in \mathbb{R}^{p \times q}$
- *Convolution layer*: detects simple object parts or features

$$A^m = \sigma(X * W^m) \qquad A^m_{jk} = s\left(b^m + \sum_{fg} X_{j-f,k-g} W^m_{fg}\right)$$

  Weights W^m now correspond to a *filter* to be learned - typically much smaller than the input thus encouraging sparse connectivity.
- *Pooling and Sub-sampling*: replace the output with a summary statistic of the nearby outputs, e.g. max-pooling (allows invariance to small translations in the input).

Visualizing and understanding ConvNet

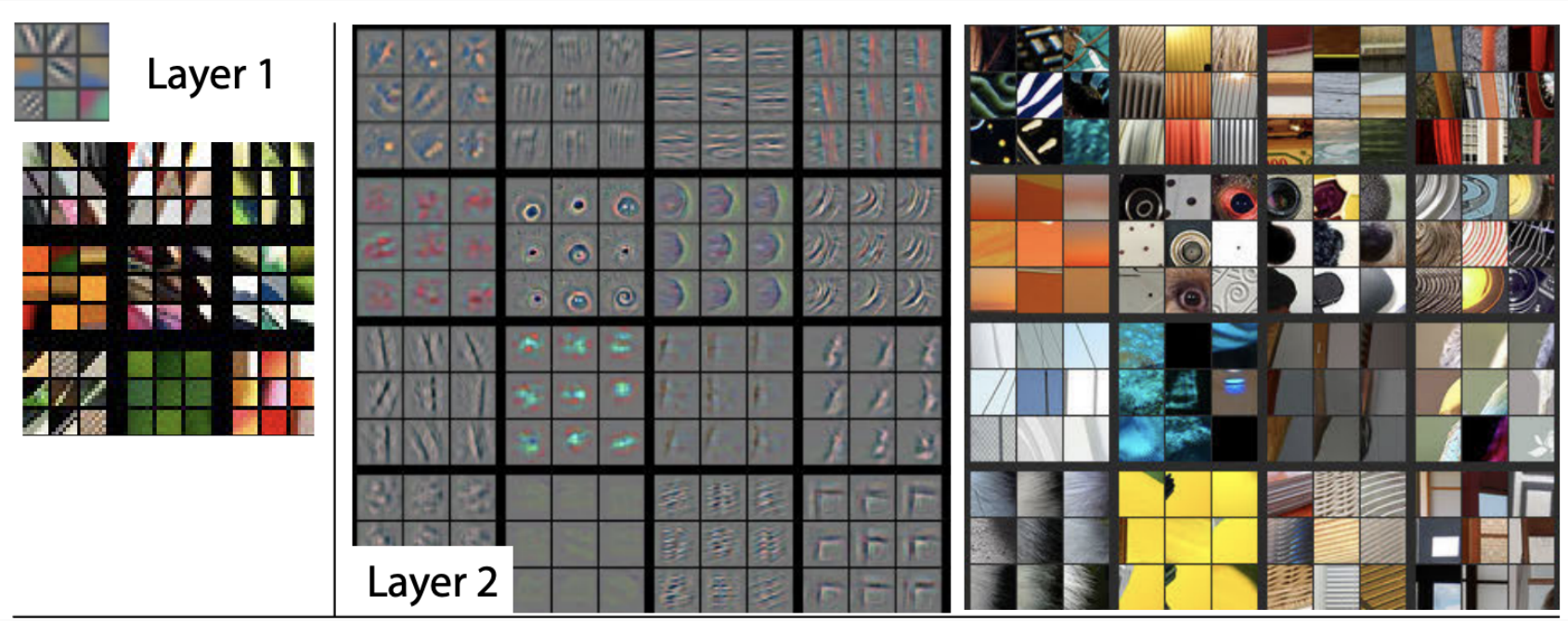


Image from "Visualizing and Understanding Convolutional Networks" by Zeiler and Fergus (2013)

CNN architectures

- Certain CNN architectures have contributed to significant progress in image recognition
- ImageNet is a standard dataset to compare networks against each other
- ImageNet has been running an annual competition since 2010: the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**
- 1.2 million images with 1000 classes for recognition

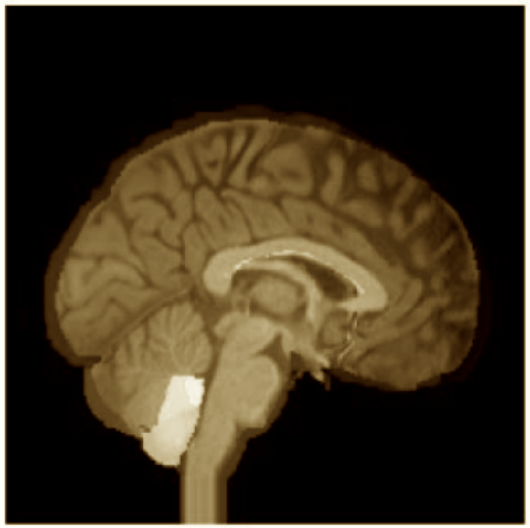
CNN architectures and ILSVRC

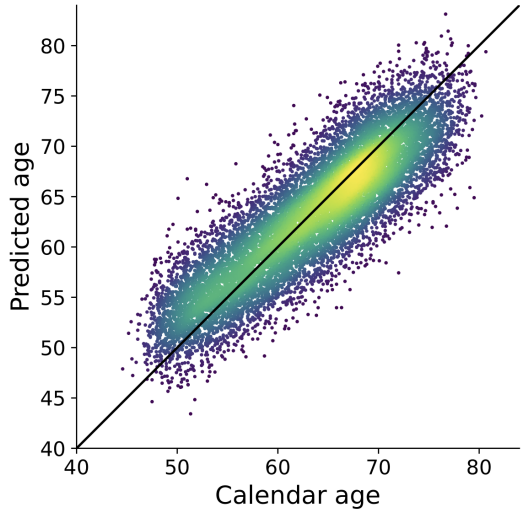
Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

Research example: MRI-predicted brain ageing

Can we predict age from brain MRI?

- Data from UK Biobank – population-based cohort study recruited from UK general population between 2006 and 2010; 21,382 participants, participants with brain MRI data
- CNN trained on 3000 MRI scans and validated on around 4000 scans.
- Age predicted by the deep neural network plotted against calendar age for all participants in the test set.

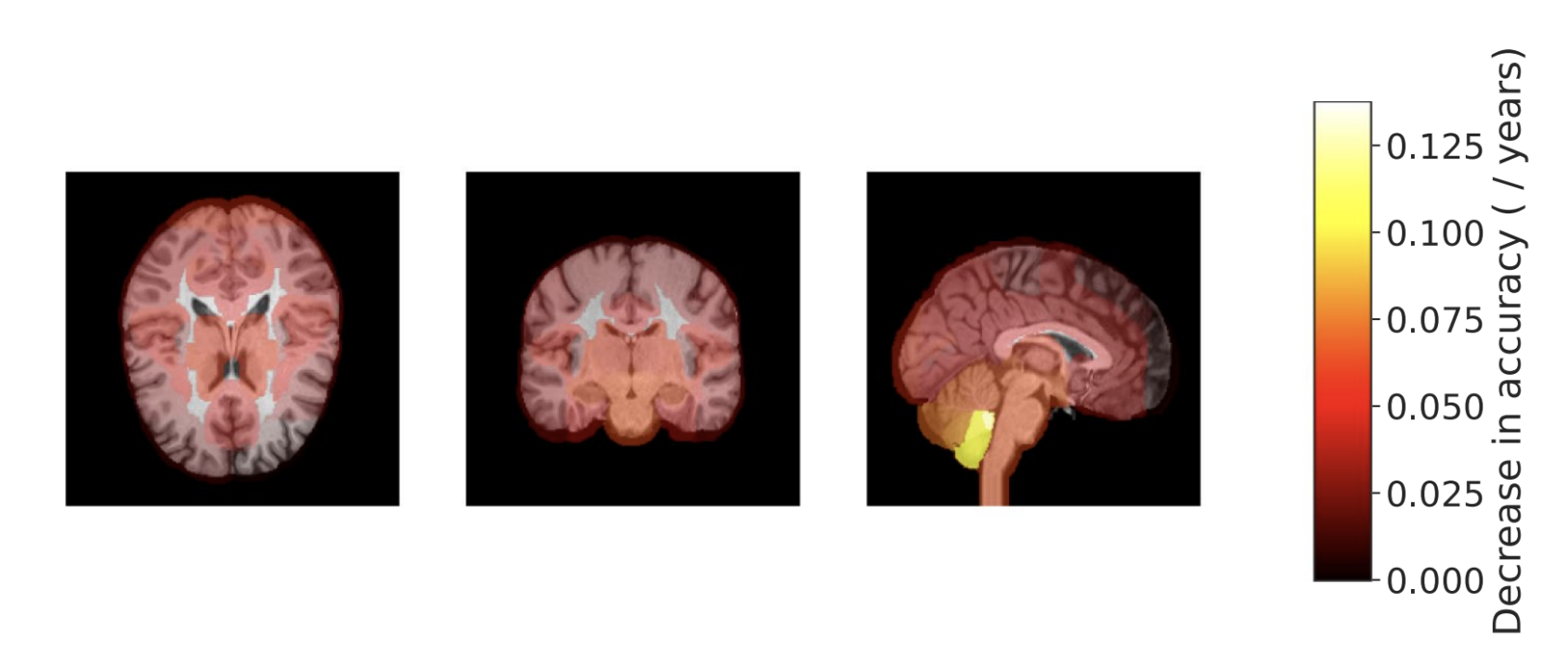




Kolbeinsson et al., *Scientific reports* (2020).

Example: MRI-predicted brain ageing

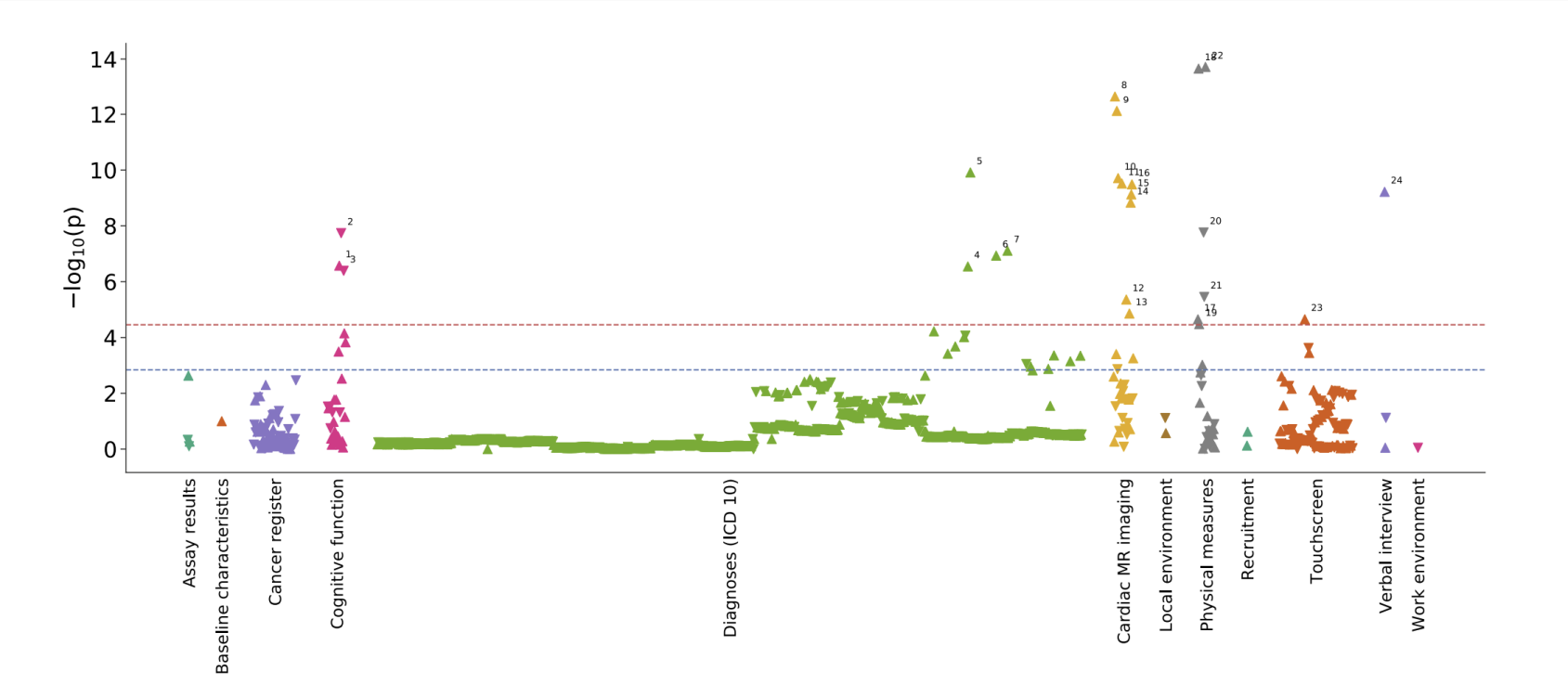
Contributions of different brain regions to brain age predictions



Kolbeinsson et al., *Scientific reports* (2020).

Example: MRI-predicted brain ageing

Is the difference between model predicted and chronological ages meaningful?



Kolbeinsson et al., *Scientific reports* (2020).

Types of neural networks

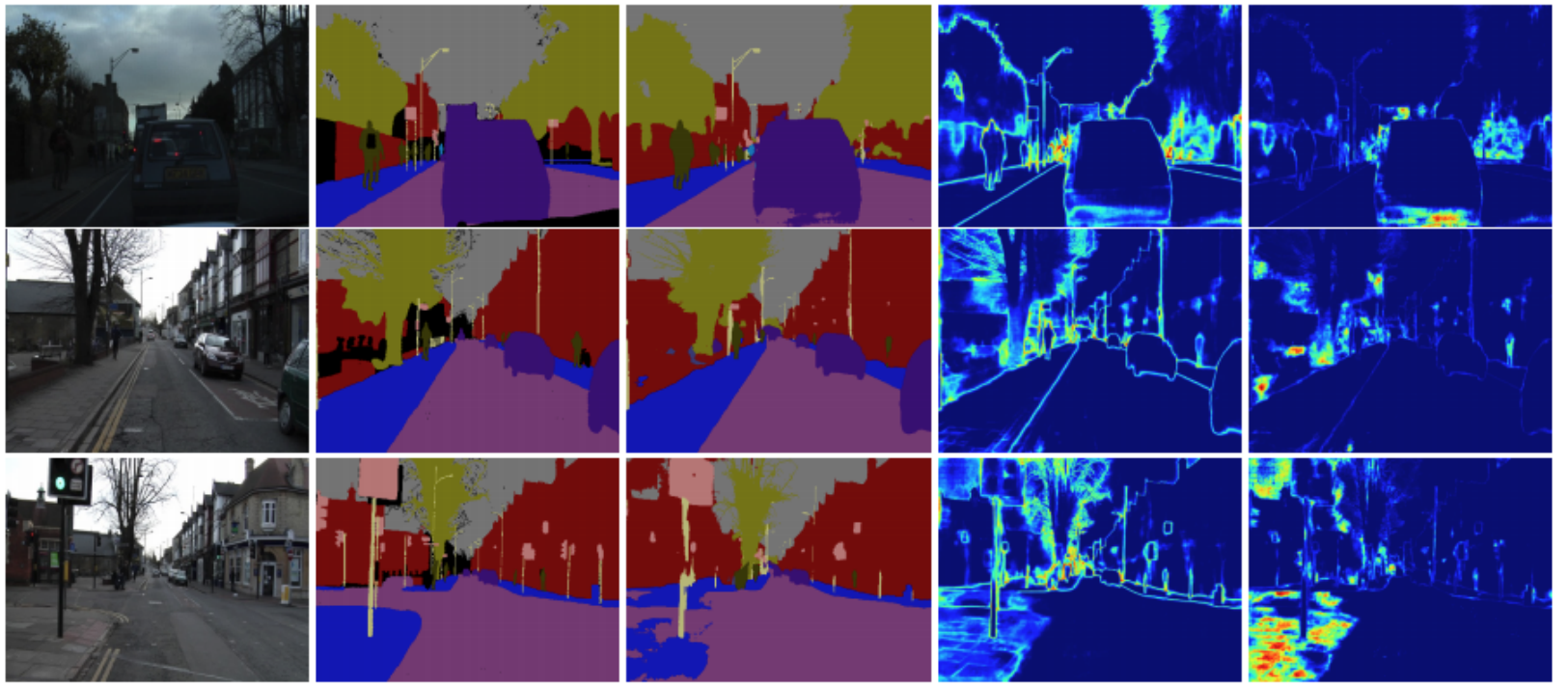
We have mentioned so far the multilayer perceptron and the convolutional neural network but there exists many other types of neural networks, each with different types of architectures.

For example, recurrent neural network have been developed for sequences/longitudinal data.

Neural networks have also been proposed for unsupervised learning tasks. For example, auto-encoder are used for representation learning.

Bayesian approaches

There exists also Bayesian Deep learning approaches.



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

Image from "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" by Kendall and Gal (2017)

Summary and remarks

- Nonlinear hidden units introduce modelling flexibility, hierarchical representations.
- Neural networks with sufficiently many hidden units can model arbitrarily complex functions.
- Training a neural network typically require a very large training dataset.
- Optimization problem is not convex, and objective function can have many local optima, plateaus and ridges.
- On large scale problems, often use stochastic gradient descent, along with a whole host of techniques for optimization, regularization, and initialization.
- Fine tuning techniques can speed up the training and overcome small dataset size.
- Even though some methods start to focus on interpretation, neural networks are typical blackboxes.
- Note also the lack of theory.

More ressources

Explosion of interest in the field recently and many new developments not covered here. See for example http://deeplearning.net/.

Tutorials and courses including:

- http://cs231n.github.io/convolutional-networks/
- https://www.coursera.org/learn/machine-learning
- http://videolectures.net/deeplearning2015_salakhutdinov_deep_learning/
- https://www.youtube.com/watch?v=F1ka6a13S9I