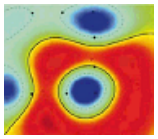
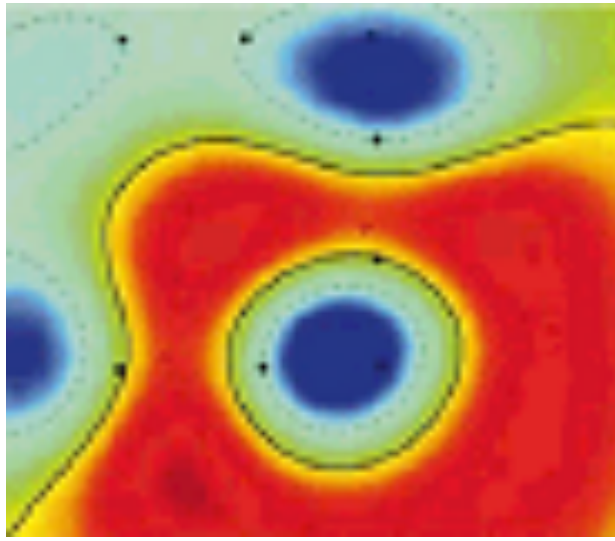


SoSe 2024

Machine Learning 2/2-X

Technische Universität Berlin

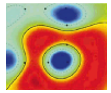


Lecture 8

Structured Input with Neural Networks

Outline

- Neural Networks recapitulation
  - Motivations, Biological vs. Artificial Neural Networks
  - Neural Networks vs. Feature Extraction & Kernels
- Structured Neural Networks
  - Convolutional Neural Networks
  - Non-vector data (text)
  - Recursive Neural Networks
  - Graph Neural Networks



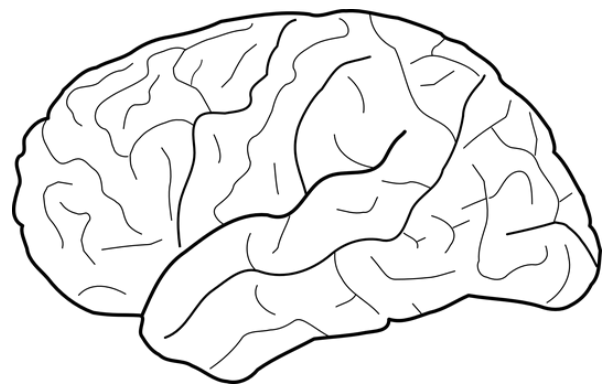
Why Neural Networks?

Humans have shown to be capable of mastering tasks such as visual recognition, motion, speech/language, or games. These tasks have however been challenging for classical AI methods (e.g. linear classifiers, simple kernels, expert systems, etc.).

Question: Can machine learning models take inspiration of some mechanisms in the human's brain in order to achieve similar performance at these tasks?

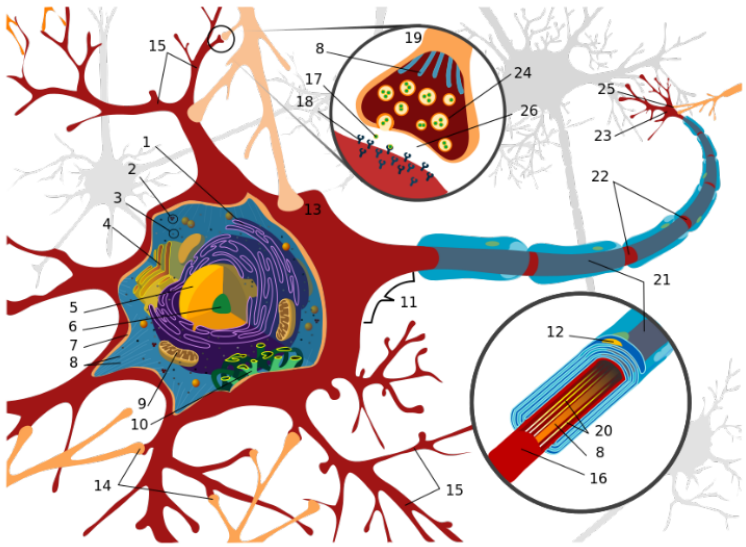
The Human Brain as a Model for Machine Learning

- The human brain is a highly complex (and so far scarcely understood) system.
- Scientific research in the past century has however provided some understanding of what might enable these systems to learn successfully:
  - Complex abstract representations result from the *interconnection* of many simple nonlinear neurons.
  - The property of these neurons to *modify* their response when exposed repeatedly to a certain stimuli enables the brain to learn.

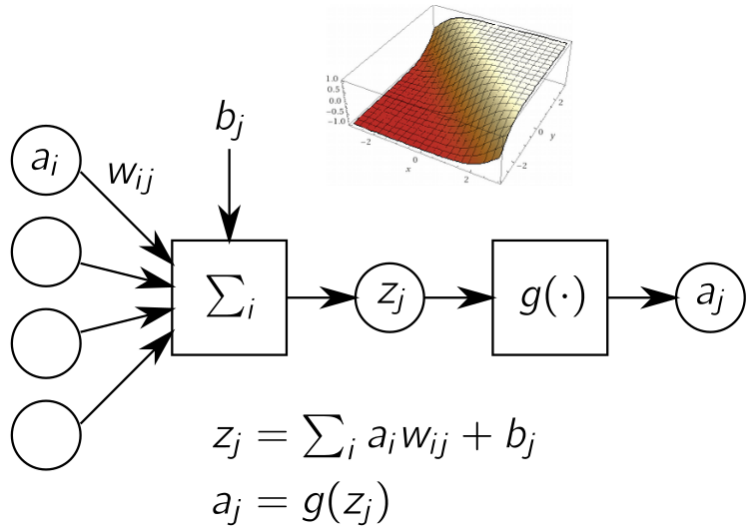


Biological vs. Artificial Neurons

Biological neuron

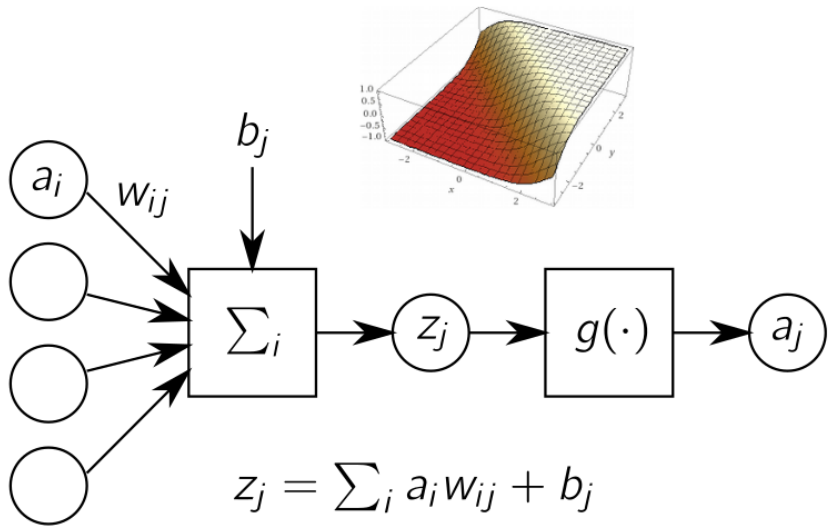


Artificial neuron



A biological neuron and the way it is interconnected to other neurons are highly complex. Artificial neurons only retain the most essential components of the biological neuron for practical purposes: *nonlinearity* and ability to *learn*.

Nonlinearity of the Neuron



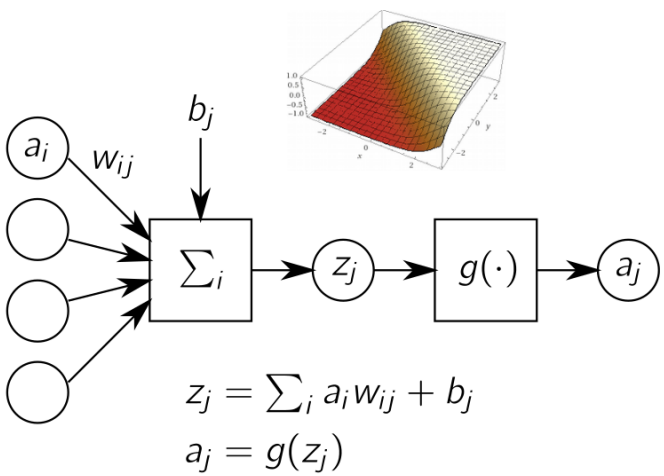
- Nonlinearity is produced by the activation function g. Examples of common activation functions:

$$g(z_j) = I(z_j \geq 0) \quad \text{(threhsold)} \qquad g(z_j) = \frac{\exp(z_j)}{1 + \exp(z_j)} \quad \text{(sigmoid)}$$

$$g(z_j) = \tanh(z_j) \quad \text{(tanh)} \qquad g(z_j) = \max(0, z_j) \quad \text{(ReLU)}$$

- Interconnecting several nonlinear neurons can produce more complex functions, that are potentially useful for solving real-world tasks.

Learning Mechanism of the Neuron

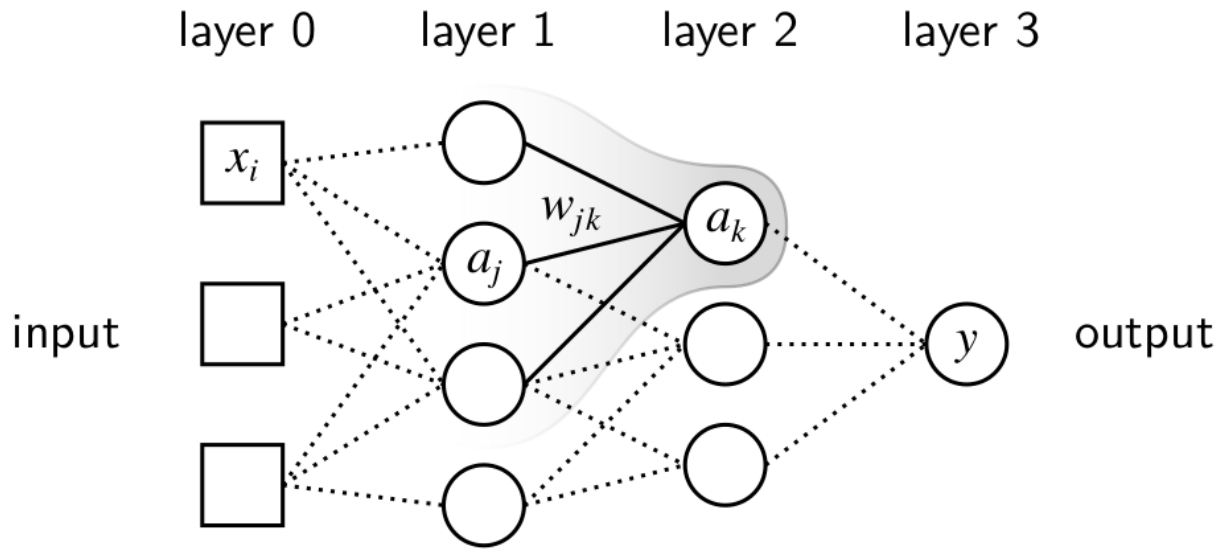


- Consider the quantity $\partial \mathcal{E} / \partial a_j$ which can be interpreted as the way the neuron activation a_j influences locally the prediction error $\mathcal{E}$.
- Using the chain rule for derivatives, one can also compute how the parameters $(w_{ij})_j$ of that neuron influence $\mathcal{E}$:

$$\frac{\partial \mathcal{E}}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}} \cdot \frac{\partial a_j}{\partial z_j} \cdot \frac{\partial \mathcal{E}}{\partial a_j}$$
$$= a_i \cdot g'(z_j) \cdot \frac{\partial \mathcal{E}}{\partial a_j}$$

- Therefore, we know how to update the parameters w_{ij} in a way that the objective $\mathcal{E}$ is decreased. Similar computation can be performed for the neuron bias b_j.

Neural Networks: Forward Pass



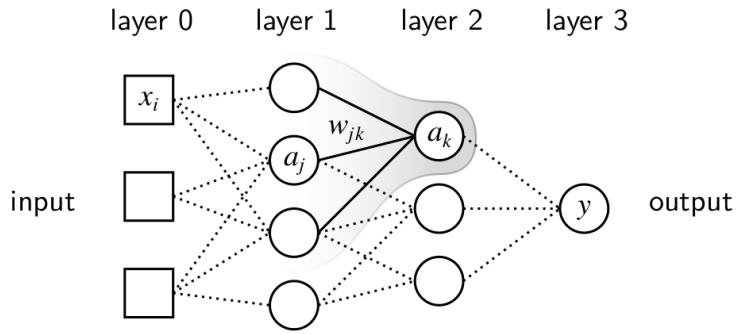
The forward pass mapping the input of the network to the output is given by:

$$z_j = \sum_i x_i w_{ij} + b_j \qquad a_j = g(z_j) \qquad \text{(layer 1)}$$

$$z_k = \sum_j a_j w_{jk} + b_k \qquad a_k = g(z_k) \qquad \text{(layer 2)}$$

$$y = \sum_k a_k v_k + c \qquad \text{(layer 3)}$$

Neural Networks: Backward Pass



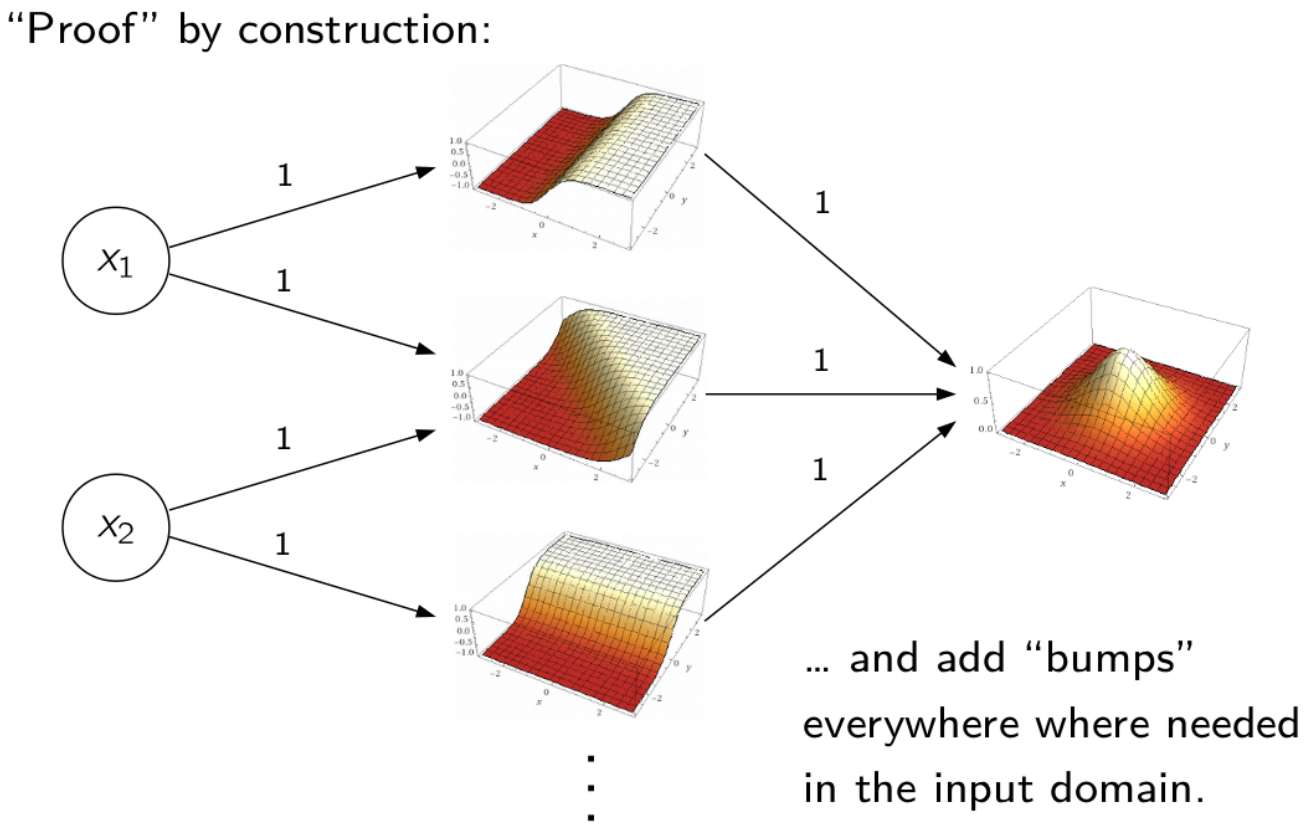
The gradient of the error can be propagated from layer to layer using the chain rule:

$$\underbrace{\frac{\partial \mathcal{E}}{\partial a_j}}_{\delta_j} = \sum_k \underbrace{\frac{\partial a_k}{\partial a_j}}_{w_{jk} g'(z_k)} \cdot \underbrace{\frac{\partial \mathcal{E}}{\partial a_k}}_{\delta_k}$$

and storing intermediate results in variables $\delta_j, \delta_k, \ldots$ so that these quantities do not need to be recomputed. This mechanism allows to compute the gradient of the parameters at all layers, and is known as the error backpropagation algorithm (Rumelhart 1986).

Neural Networks: Universal Representations

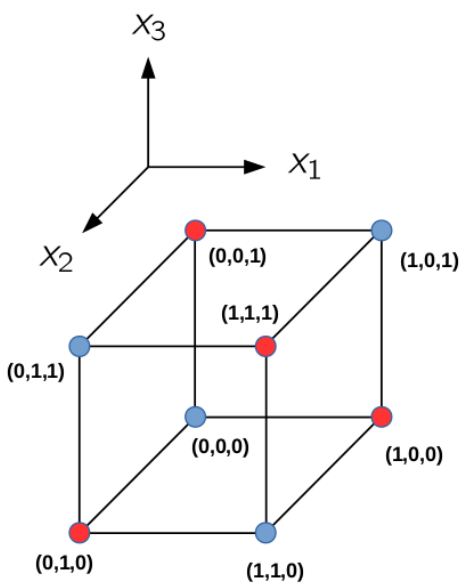
Neural networks with sufficiently many neurons can approximate any function f of its input variables $x_1, x_2, \ldots, x_d$.



Neural Networks: Compactness (1)

Neural networks can express a broad range of 'useful' functions in compact manner (e.g. without having to use exponentially many neurons).

Function to approximate: $f(\boldsymbol{x}) = \text{parity}(x_1, x_2, \ldots, x_d)$



Naive approach:
Use one neuron to handle each corner of the hypercube. (overall 2^d neurons)

Better approach:
Use a deep composition of elementary modules (4 x d neurons).

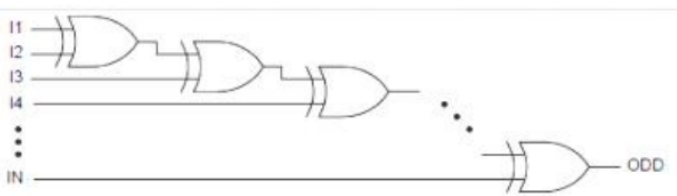
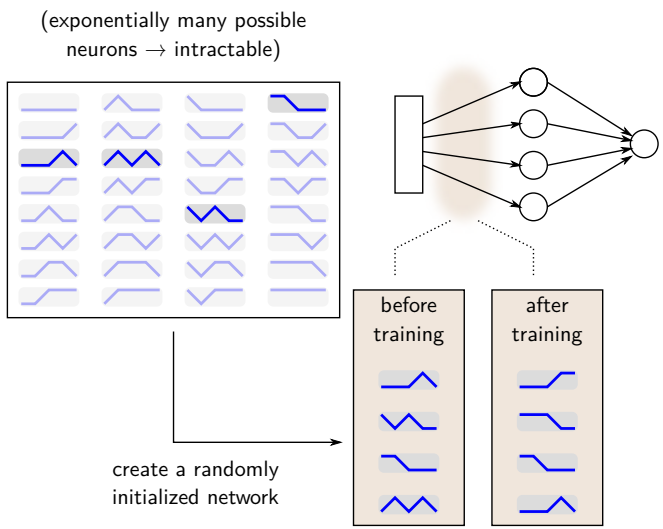


Image source: http://vlsi-design-engineers.blogspot.com/2015/10/exclusive-or-gates-parity-circuits-and.html

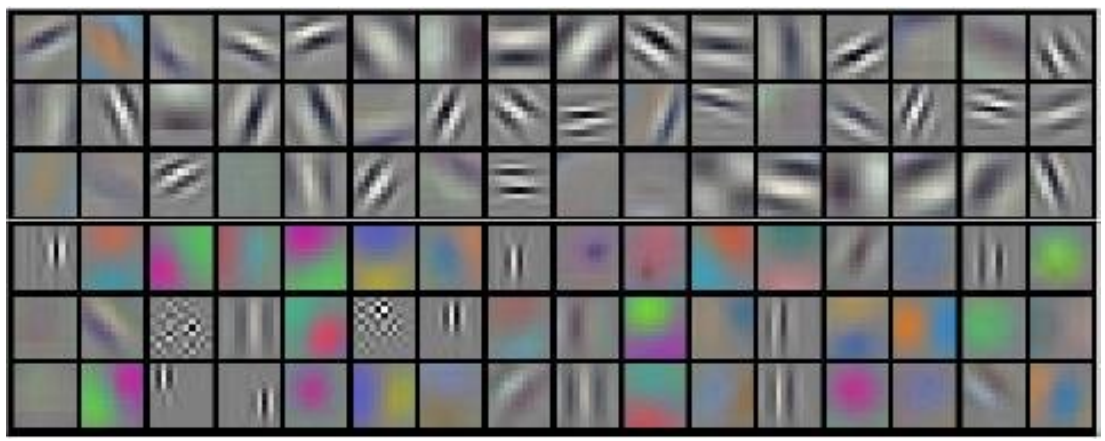
Neural Networks: Compactness (2)



- The neural network starts with a finite and typically small set of randomly initialized neurons (i.e. a subset of all possible neurons).
- The compact problem representation is progressively extracted during training under the simultaneous effect of optimization (minimizing the error) and the finite number of neurons in the model.
- The learned representation is almost as predictive as an exhaustive set of neurons, but much more compact.

Neural Networks: Compactness (3)

Example of the set of first-layer filters learned by a neural network trained on image classification (AlexNet):

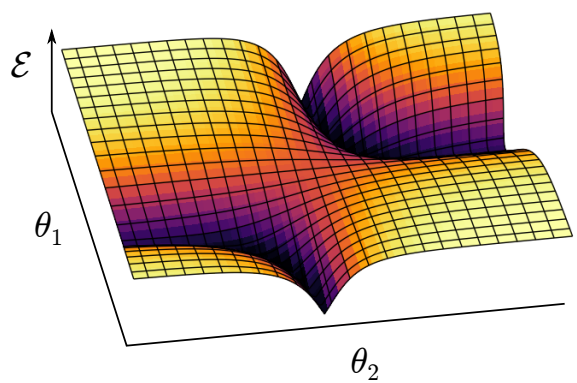


These 96 filters capture most of the important low-level signal for image classification, and are much more compact than the exhaustive set of all possible filters (potentially thousands or millions of possible filters).

Neural Networks: Optimization

Neural networks also have downsides:

- Non-convex objective (e.g. even the simplest two-layer network $\phi(x;\theta) = \theta_1\theta_2 x$ is already non-convex with θ). Many hyperparameters (e.g. initialization, learning rate, etc.) can affect the result of learning.
- Multiple layers can cause pathological curvature, i.e. the gradient vanishes along certain directions of the parameter space. The optimizer may get stuck on large plateaus.



With heuristics on the neural network design (e.g. choice of layers and nonlinearities) and optimization (e.g. momentum, batch-normalization), it is however still possible to train them efficiently.

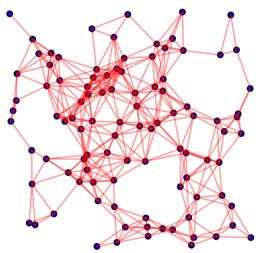
Neural Networks vs. Other Feature Extraction

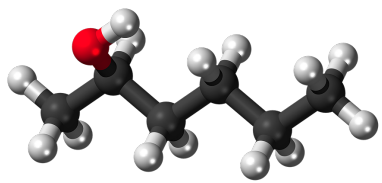
Question: How does the neural network approach compares to other approaches such as (manual) feature extraction, or kernel methods?

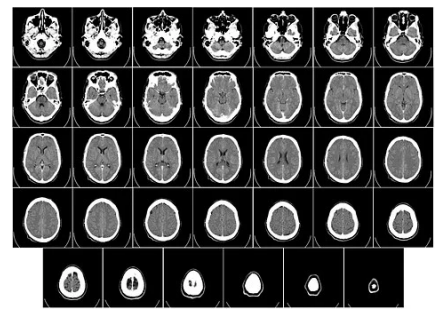
Approach	*Universal*	*Compact*	*Convex*
1. Feature Engineering	✗	✓	✓
2. Kernels/Expansions	✓	✗	✓
3. Neural Networks	✓	✓	✗

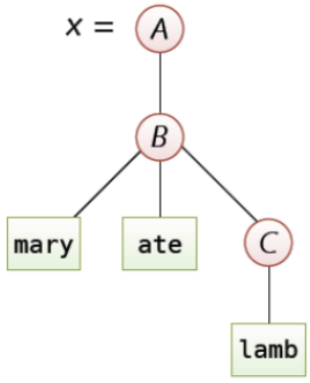
- *Feature engineering:* Discriminative features can be engineered so the task can be modeled with finitely many features (i.e. compact). However, there is no guarantee that with a given set of features, the task can be solved perfectly (i.e. not universal).
- *Kernels:* Any task can in principle be solved perfectly (i.e. universal representation), however that might require kernels with very large associated feature maps (i.e. not compact).

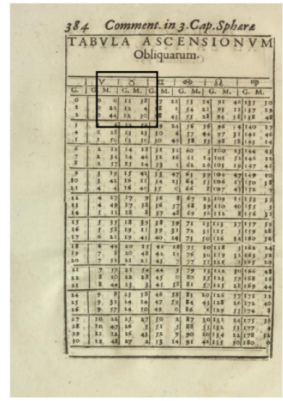
Today's Lecture: NNs for Structured Data





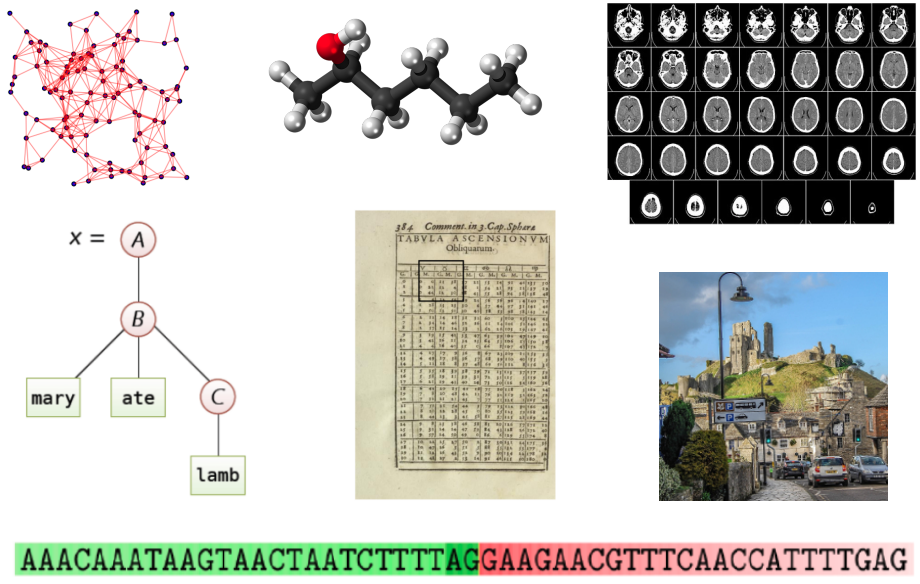






AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG

Why NNs for Structured Data?

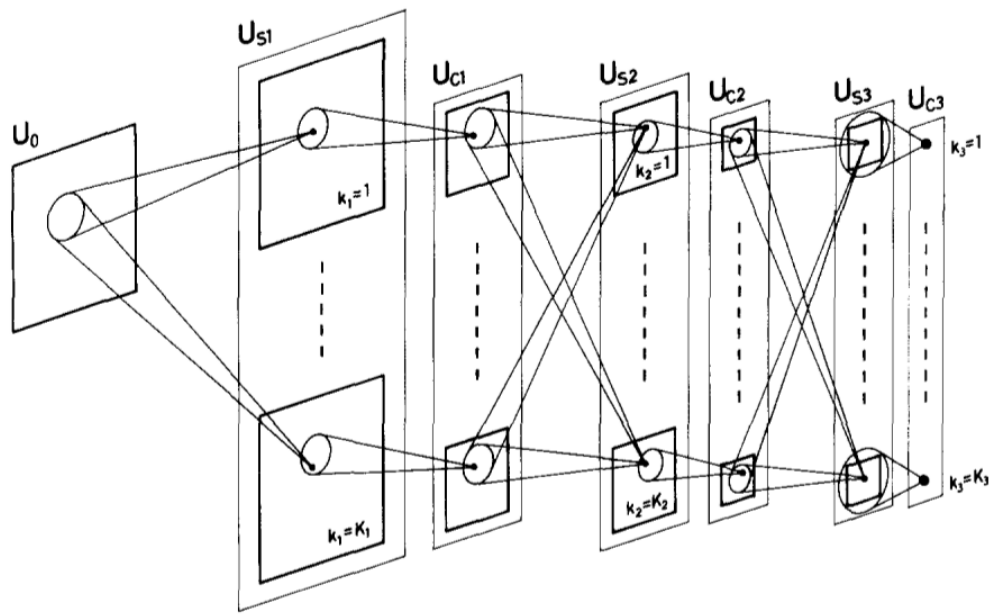


Two Goals:

- *Technical:* Being able to process inputs that are not real-valued and cannot be represented as a fixed-size vector, e.g. text, DNA sequences.
- *Statistical:* Structured data (e.g. images) often come with specific invariances (e.g. translation invariance), that we would like the model to capture in order to generalize well.

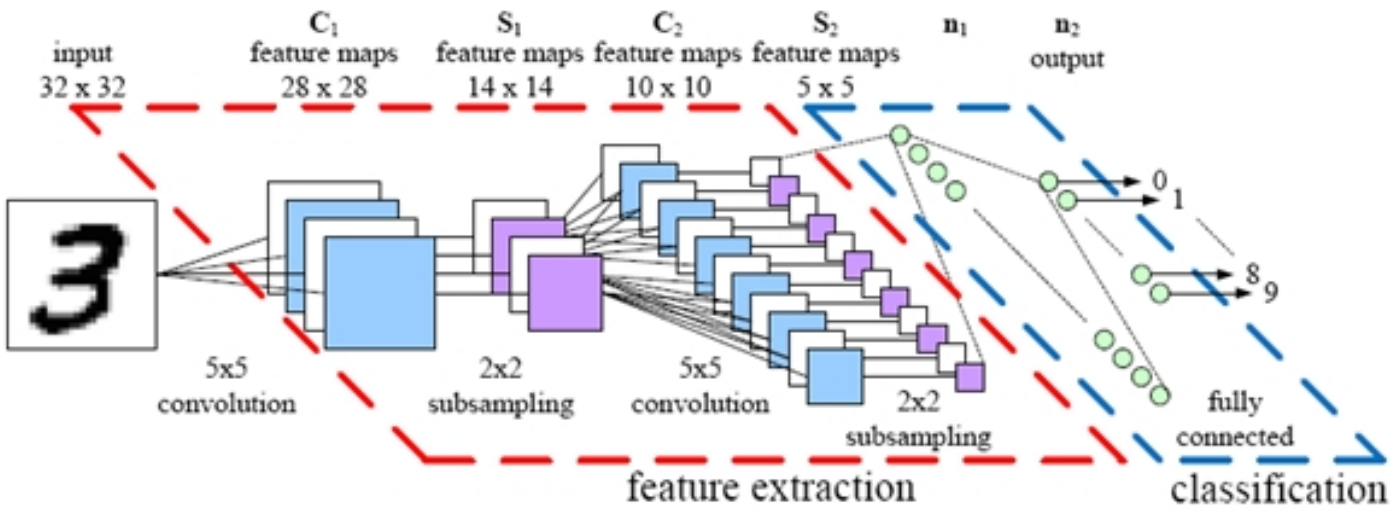
The Neocognitron (1979)

The Neocognitron [2] is an architecture for representing images, and which is designed in a way that the produced representation is invariant to a translation of the input image.



The Neocognitron consists of an alternation of 'simple cells' (convolutions) and 'complex cells' (pooling), which progressively build the desired representation. It was a precursor of modern convolutional neural network architectures.

Convolutional Neural Nets (1989, 1998, 2012)



- The Convolutional Neural Network (CNN) [4] is also composed of convolution and pooling layers, and comes with a way of backpropagating the error through the convolution/pooling layers.
- CNNs are nowadays state-of-the-art (by a wide margin) on many image recognition tasks (e.g. ILSVRC 2012).
- CNN training is resource-hungry. Often, people use pretrained models.

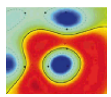
Standard vs. Convolutional Neural Networks

Standard neural network:

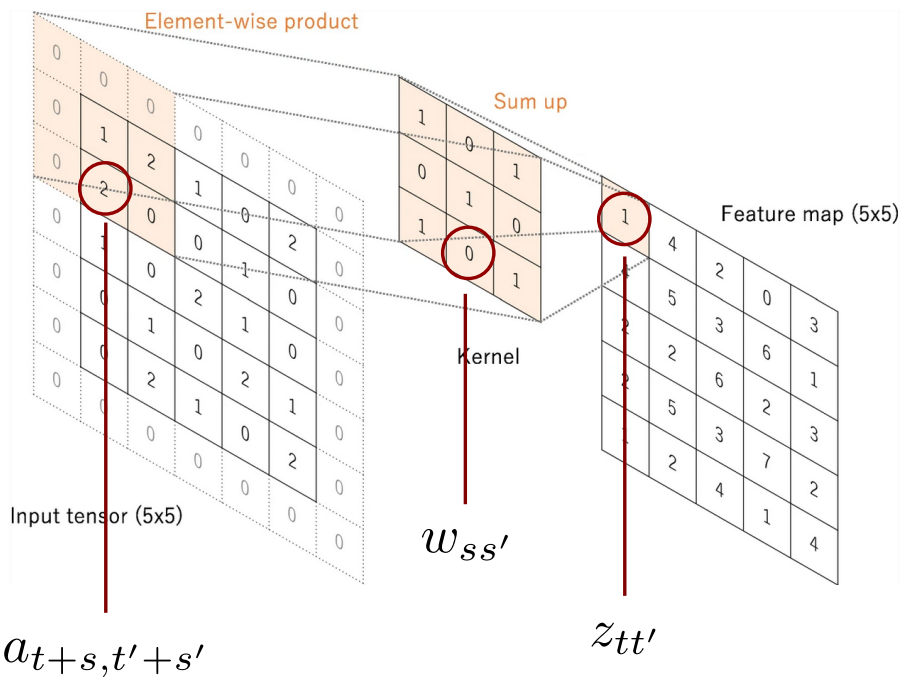
- Neurons and weights are scalars, i.e. $a_i, w_{ij} \in \mathbb{R}$.
- Mapping between two layers is a sum of products: $z_j = \sum_i w_{ij} a_i + b_j$

Convolutional neural network:

- Neurons and weights are 2D arrays, i.e. $\mathbf{a}^{(i)} \in \mathbb{R}^{W \times H}$, $\mathbf{w}^{(ij)} \in \mathbb{R}^{W' \times H'}$.
- Mapping between two layers is a sum of 2D cross-correlations: $\mathbf{z}^{(j)} = \sum_i \mathbf{w}^{(ij)} \star \mathbf{a}^{(i)} + \mathbf{b}^{(j)}$
- Convolution layers are interleaved with pooling layers in order to progressively reduce the spatial resolution and simultaneously increase the number of neurons.



Understanding the CNN (1)



adapted from Yamashita et al. 2018
Convolutional neural networks: an overview [...]

Step 1: The Cross-Correlation

- A *cross-correlation* slides a kernel (or filter) through the whole input tensor and records responses at each location.

$$z_{tt'} = [w \star a]_{tt'}$$
$$= \sum_{s=0}^{2} \sum_{s'=0}^{2} w_{ss'} \cdot a_{t+s,t'+s'}$$

- Zero-padding can be applied to the input signal in order to produce an output signal of same size as the input.

Backpropagation in Cross-Correlation

Forward pass (one-dimensional case, infinite filter size):

$$z_t = [w \star a]_t = \sum_{s=-\infty}^{\infty} w_s \cdot a_{t+s}$$

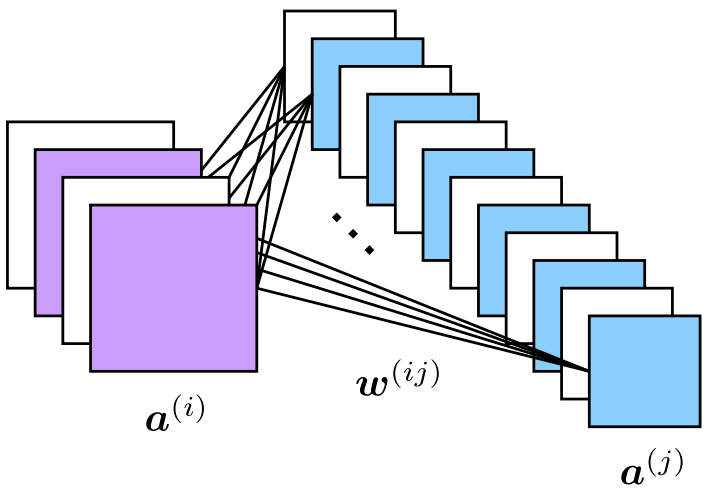
Backward pass:

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial a_u} &= \sum_{t=-\infty}^{\infty} \frac{\partial \mathcal{E}}{\partial z_t} \cdot \frac{\partial z_t}{\partial a_u} \\
&= \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} \frac{\partial \mathcal{E}}{\partial z_t} \cdot w_s \cdot 1_{\{t+s=u\}} \\
&= \sum_{t=-\infty}^{\infty} \frac{\partial \mathcal{E}}{\partial z_t} \cdot w_{u-t} \\
&= \left[\frac{\partial \mathcal{E}}{\partial z} * w\right]_u
\end{aligned}$$

Parameter gradient:

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial w_u} &= \sum_{t=-\infty}^{\infty} \frac{\partial \mathcal{E}}{\partial z_t} \cdot \frac{\partial z_t}{\partial w_u} \\
&= \sum_{t=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} \frac{\partial \mathcal{E}}{\partial z_t} \cdot a_{t+s} \cdot 1_{\{s=u\}} \\
&= \sum_{t=-\infty}^{\infty} \frac{\partial \mathcal{E}}{\partial z_t} \cdot a_{u+t} \\
&= \left[\frac{\partial \mathcal{E}}{\partial z} \star a\right]_u
\end{aligned}$$

Understanding the CNN (2)



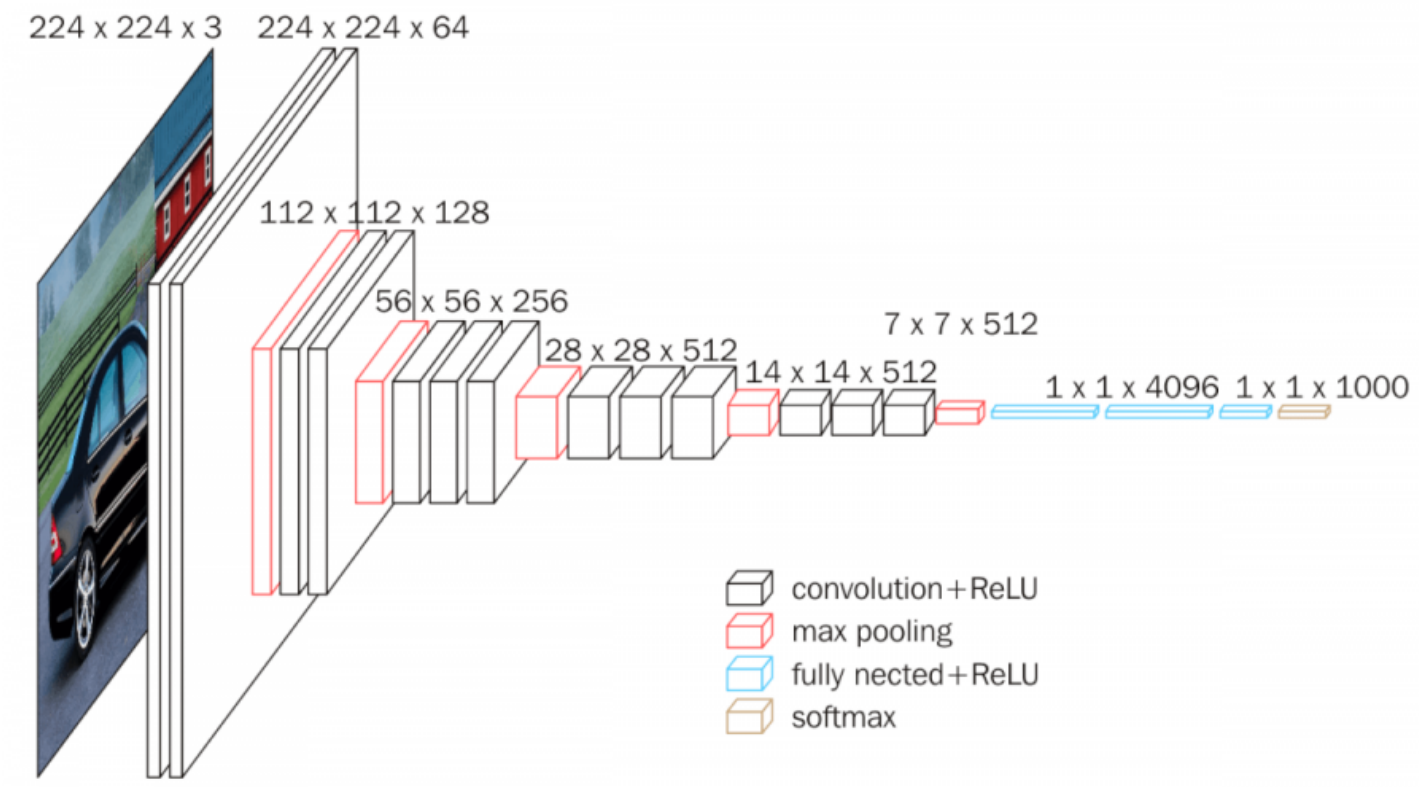
Step 2: The Convolution Layer

- A *convolution layer* has its output given by a sum of cross-correlations associated to each incoming neurons:

$$\mathbf{a}^{(j)} = g\big(\sum_i \mathbf{w}^{(ij)} \star \mathbf{a}^{(i)}\big)$$

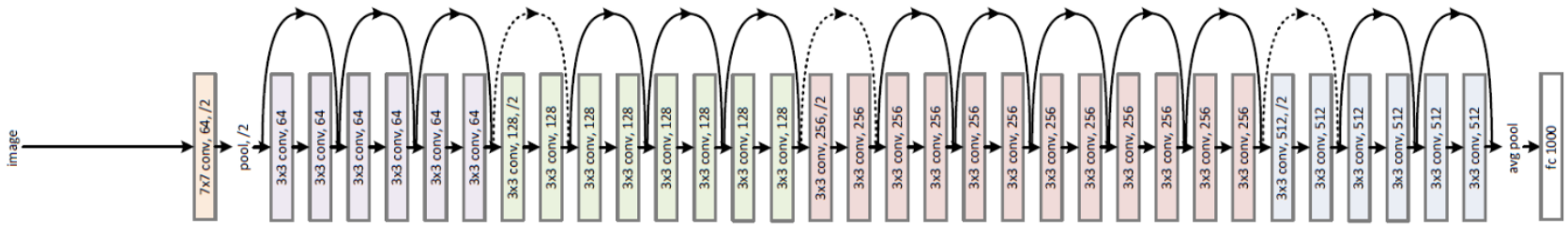
- The number of trainable parameters in a convolution layer is given by (# input neurons $\times$ # output neurons $\times$ filter size).

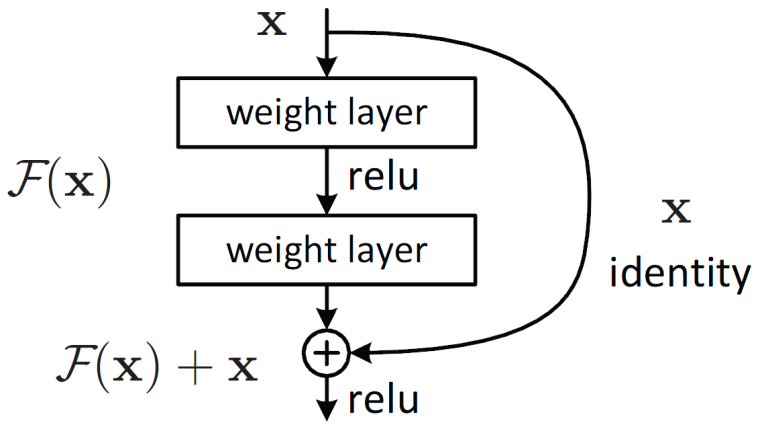
The VGG-16 Architecture



- Multiple blocks of two or three consecutive convolutions (each with kernel size 3×3). Blocks are interleaved by max-pooling layers.
- The spatial resolution is progressively reduced, which allows to use more neurons in the top layers.

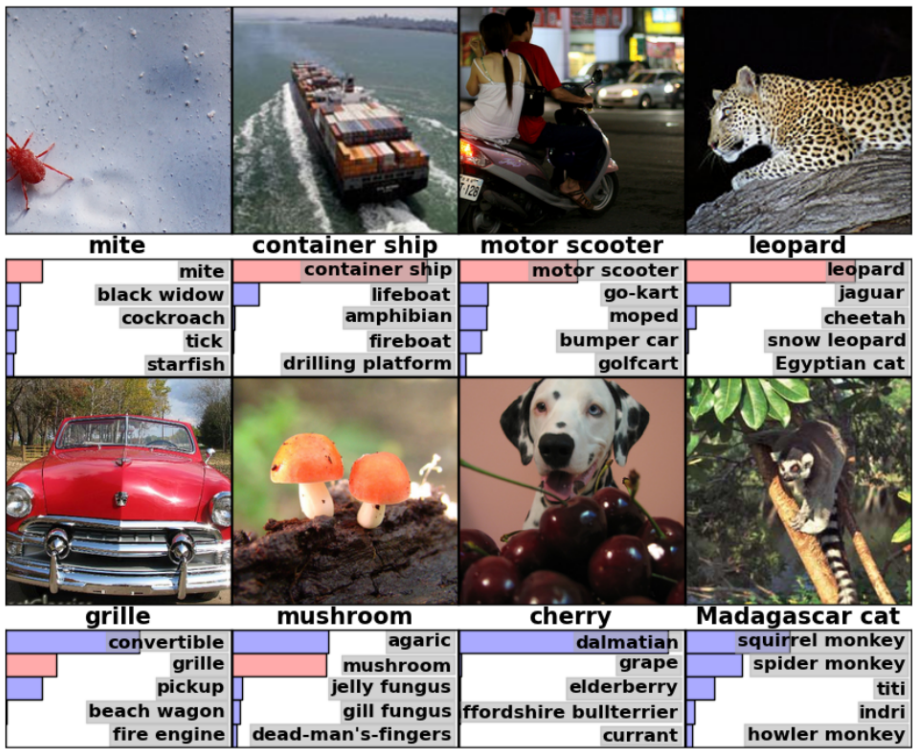
The ResNet Architecture





- Deeper than VGG-16 (ResNets have between 34 and a hundred of layers depending on the variants).
- Includes skip-connections (identity) every two or three layers in order to make prediction/training more stable.

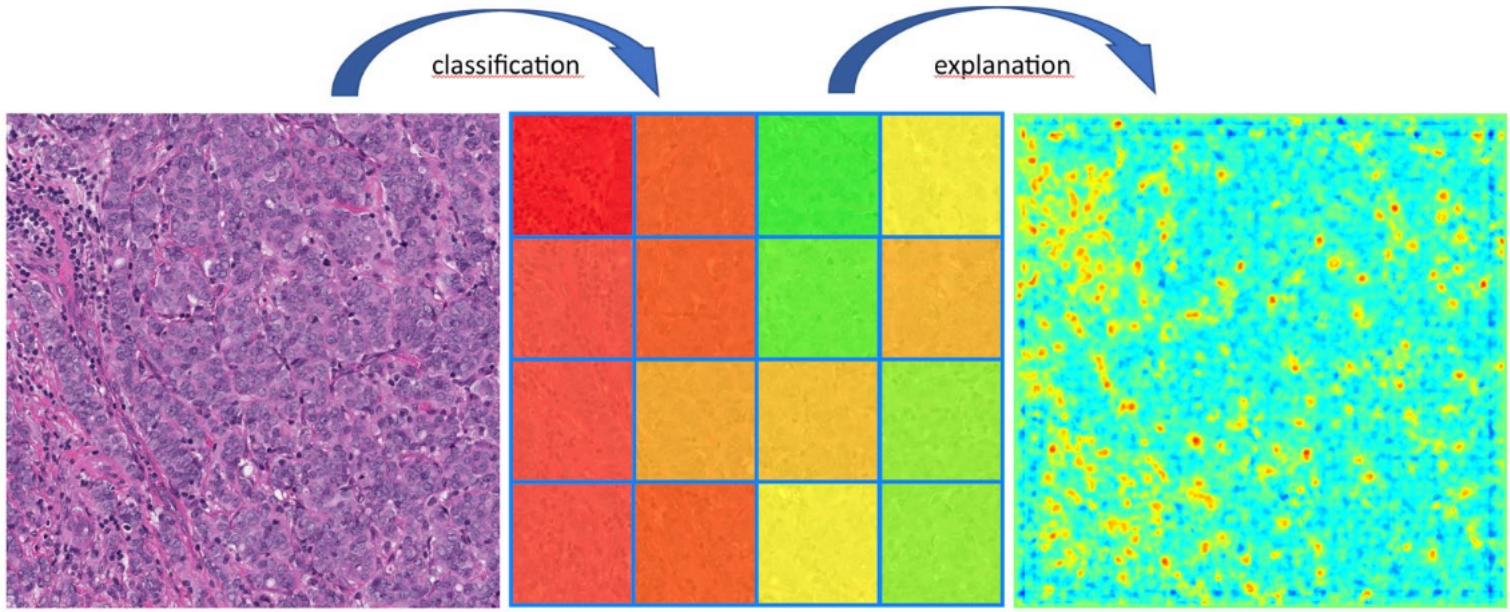
Application of CNNs: ImageNet Classification



Source: Krizhevsky et al. (2012) ImageNet Classification with Deep Convolutional Neural Networks

- Ground-truth prediction very often in the top-5 predicted classes (among 1000 predicted classes!).
- Errors are often similar to errors a human could do (e.g. similar class, or co-located pattern of another class on the same image).

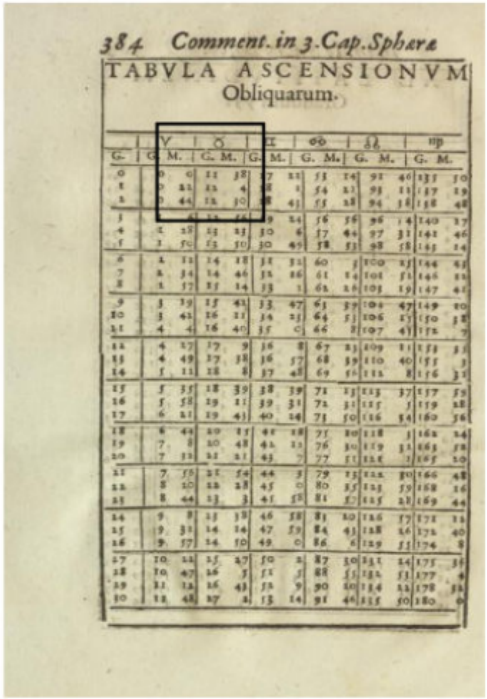
Application of CNNs: Histopathological Images



Klauschen et al. (2018) Scoring of tumor-infiltrating lymphocytes: from visual estimation to machine learning

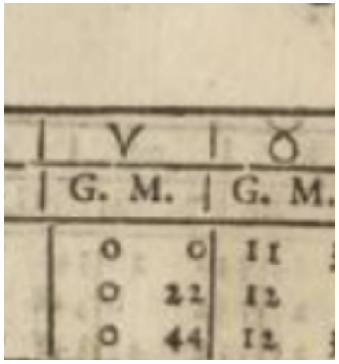
- Histopathological images can be very large ($\sim$ gigapixel).
  Convolutional neural networks are trained and applied on small patches.
- Patch-level predictions can be sharpened spatially by using pixel-wise explanation techniques such as LRP (cf. Lecture 12).

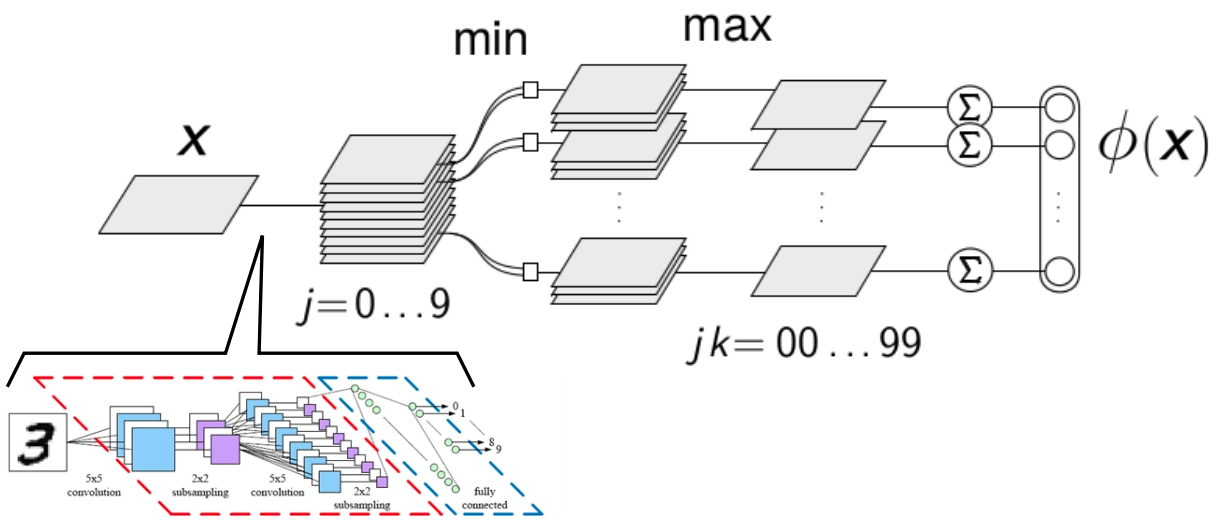
Specialized Architectures for Table Images



- Predicting similarity between scanned books content (e.g. from the early-modern era), is a recent direction for making historical inferences.
- The type of data we would like to predict is very specialized, and standard convnet architectures for image recognition do not work well at predicting this data.
- We need ad-hoc neural network architectures that take into consideration what types of features are predictive of table similarity, and also that make an efficient usage of data and labels.

The ‘Bigram Network’ [1]





- Compound architecture composed of a digit recognizer followed by soft-logic operations to extract histogram of digit bigrams.
- Only the digit recognizer is trained, the rest is hard-coded. This avoids the necessity of having full-table labels.

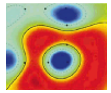
Structured Neural Networks for Text

Image classification

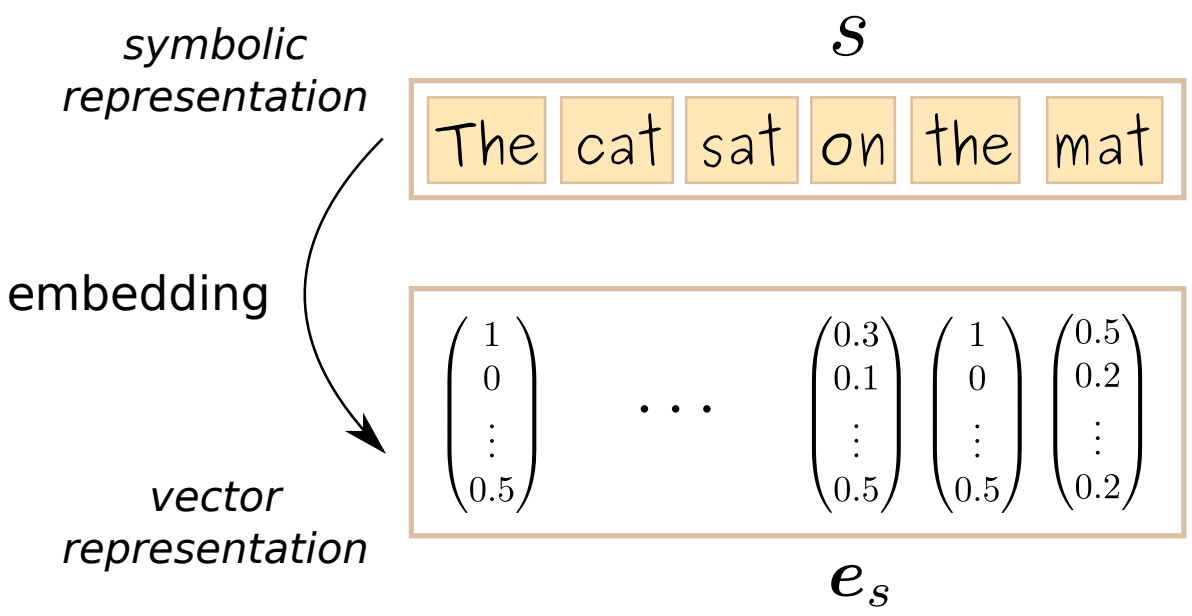
- Images are arrays of pixels.
- Pixels p of an image are given by their RGB components (i.e. $x_p \in [0,1]^3$). Therefore, they can be readily given as input to the neural network.

Text classification

- Texts are collections of words.
- Words w are abstract symbols that do not have a numerical representation. $\Rightarrow$ They need to be embedded into one such representation so that they can be given as input to a neural network.



Vector Embedding of a Sentence



Types of Vector Embedding

1. **One-hot encoding:** Represent the word with indicator functions:

$$e[w] = \left(1_{\{w=\text{'a'}\}}, 1_{\{w=\text{'able'}\}}, 1_{\{w=\text{'about'}\}}, \dots\right) \in \mathbb{R}^{\#\text{words}}$$

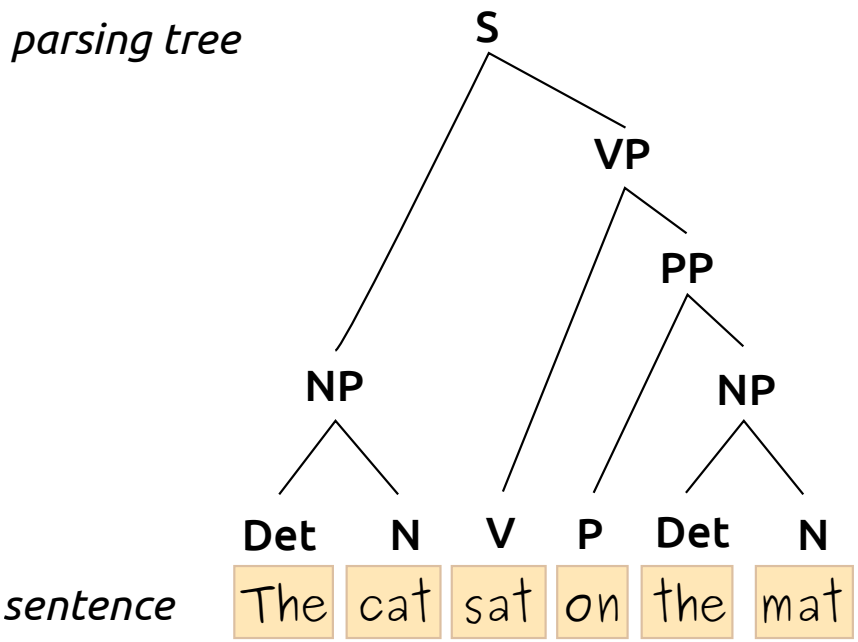
2. **kPCA embedding:** Structured kernel $k(w, w')$ measures similarity between words (e.g. co-occurrence or lexical similarity). Embedding is obtained by diagonalizing the Gram matrix: $K = U\Lambda U^\top$, and defining:

$$e[w] = U_{w,:d} \odot (\lambda_1^{0.5}, \dots, \lambda_d^{0.5}) \in \mathbb{R}^d$$

3. **Learned embedding (e.g. Word2Vec):** Start with a randomly initialized embedding $e[w] \in \mathbb{R}^d$ and learn the embedding on the supervised task directly or on some related unsupervised task:

$$e[w] \leftarrow e[w] - \gamma \cdot \partial\mathcal{E}/\partial e[w].$$

Recursive Neural Networks [5]

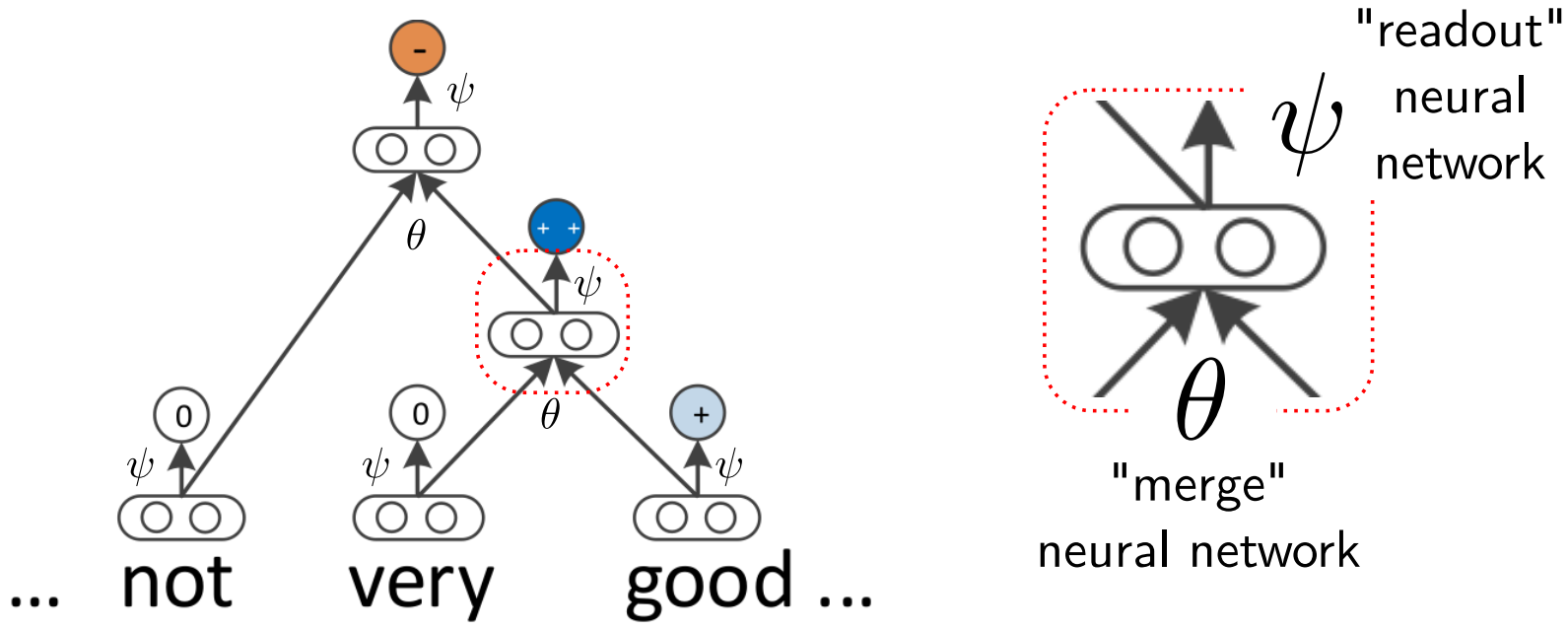


Ideas:

- Make use of the parsing tree of a sentence to produce a better text classification model.
- Build the representation by recursively applying a neural network from the leaves of the tree up to the root.

RecNNs for Sentiment Analysis

The recursive neural network (RecNN) consists of two subnetworks: "merge" and "readout". Both are used at multiple locations in the parse tree.

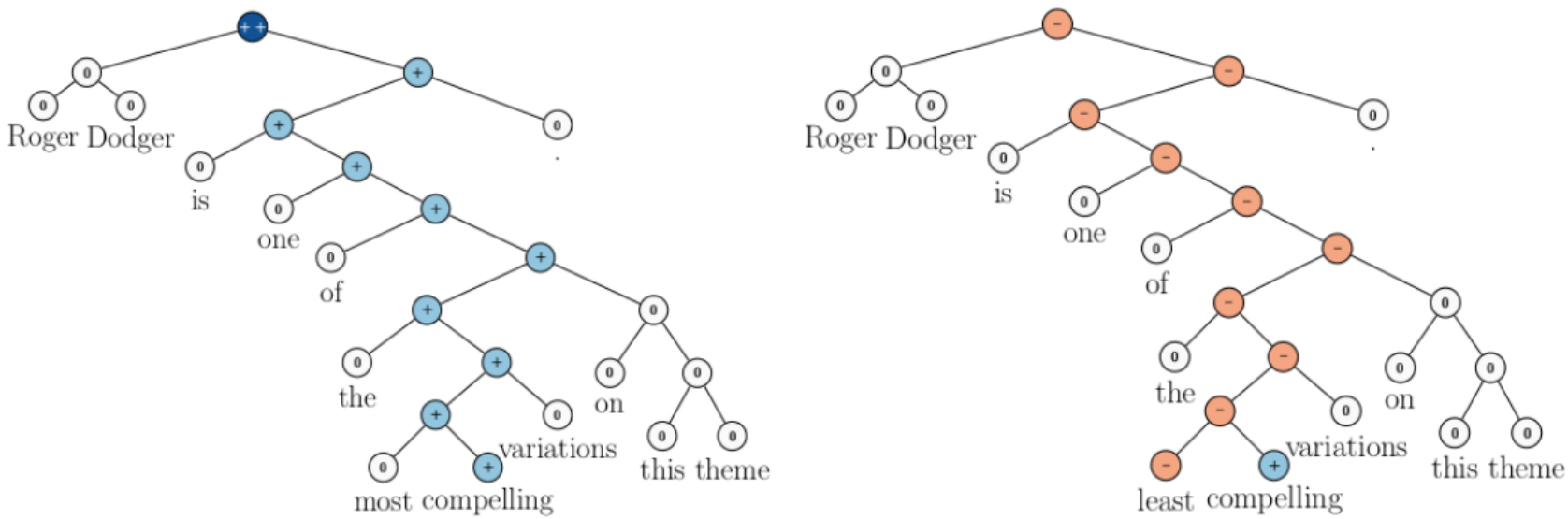


Based on Socher et al. (2013) Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

It can be applied to any sentence that comes with a binary parse tree.

RecNNs for Sentiment Analysis

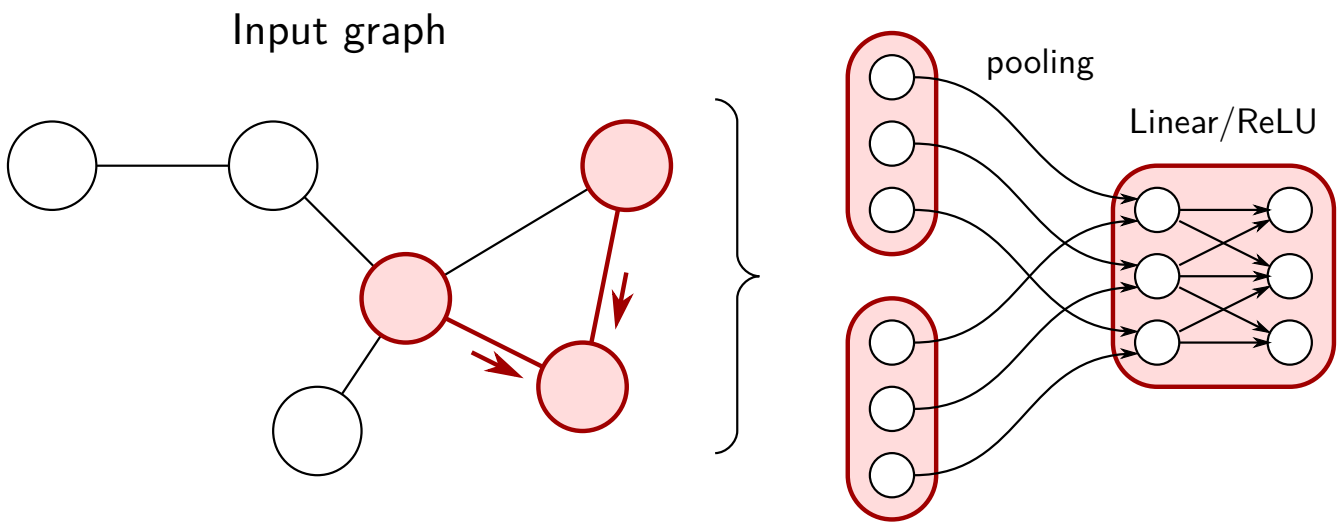
Applying the readout function on each node allows to visualize how the sentiment builds up in the recursive neural network.



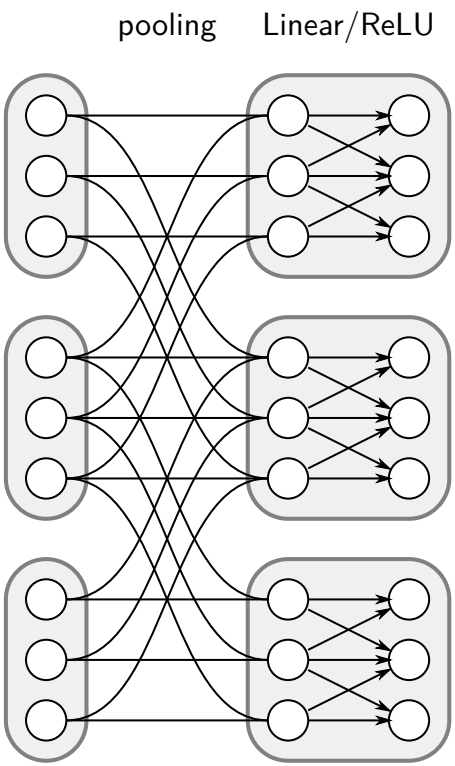
Source: Socher et al. (2013) Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Graph Neural Networks

- Graph neural networks (GNNs) are networks that are specialized for classifying graphs. Graphs are general data representation, that include pixel lattices, trees, sequences as special cases.
- GNNs receive a graph as input and implement at each layer a propagation step along its edges (via a pooling function).



Graph Neural Networks



- Consider a graph of N nodes. We denote the representation of the graph at layer l as $\mathbf{H}_l = (\mathbf{h}_1^{(l)}, \ldots, \mathbf{h}_N^{(l)})$ a matrix of dimensions $N \times d$.
- Define $\mathbf{\Lambda}$ of size $N \times N$ a connectivity matrix, e.g. a normalized version of the graph adjacency matrix.
- Define $\mathbf{W}_l$ a matrix of size $d \times d'$, which we use to map nodes representations from one layer to another.

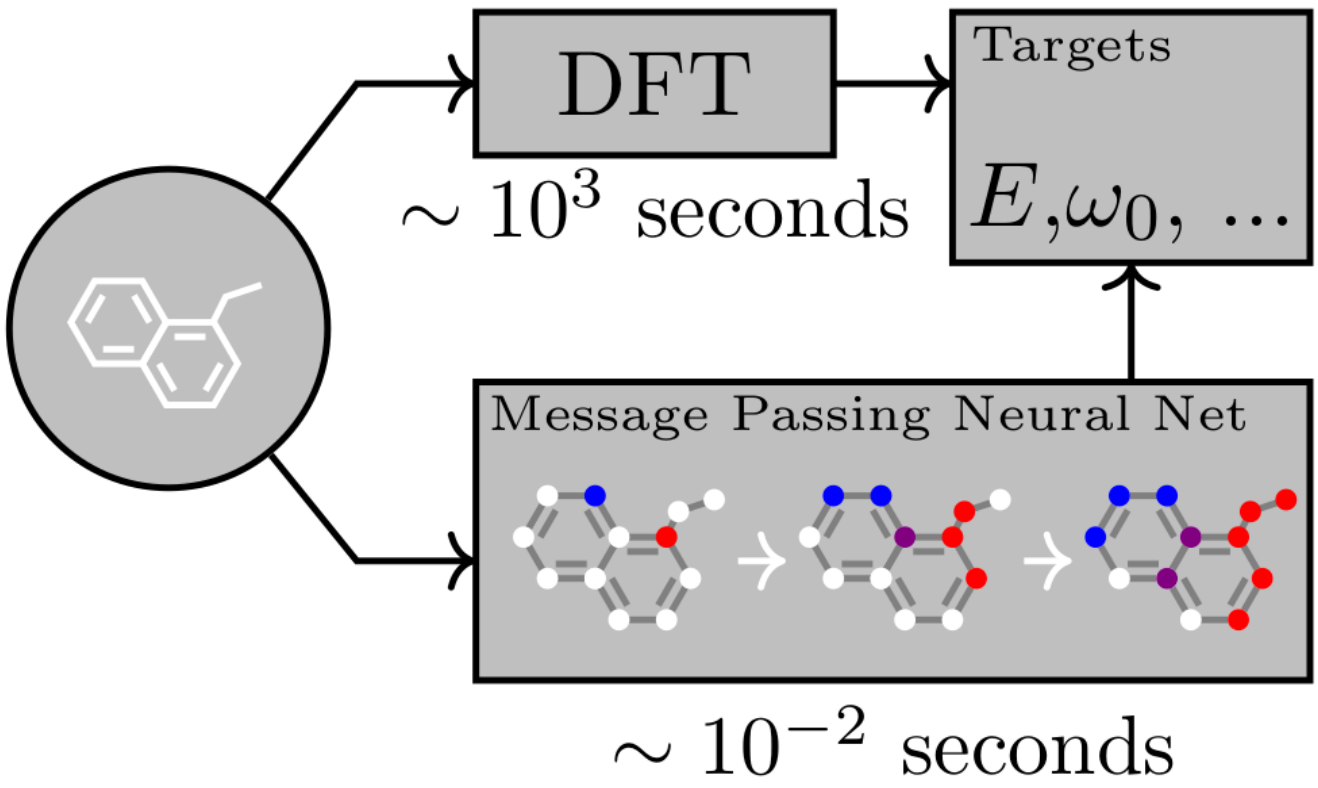
  The graph convolutional network [3] defining the following layer-wise mapping:

$$\mathbf{A}_l = \mathbf{\Lambda}\mathbf{H}_l \qquad \text{(pooling)}$$
$$\mathbf{H}_{l+1} = \rho(\mathbf{A}_l\mathbf{W}_l) \qquad \text{(linear/ReLU)}$$

  where ρ is a ReLU function applied element-wise.

Applications of GNNs

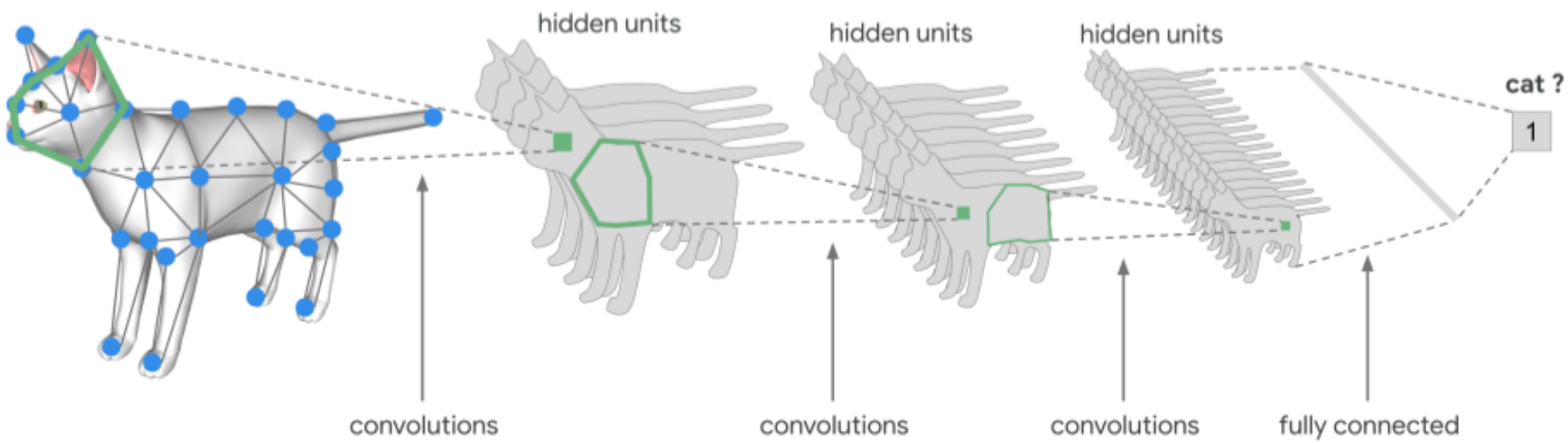
Predicting atomization energy to bypass expensive DFT simulations:



Source: Gilmer et al. (2017) Neural Message Passing for Quantum Chemistry

Applications of GNNs

Recognizing objects from meshes:



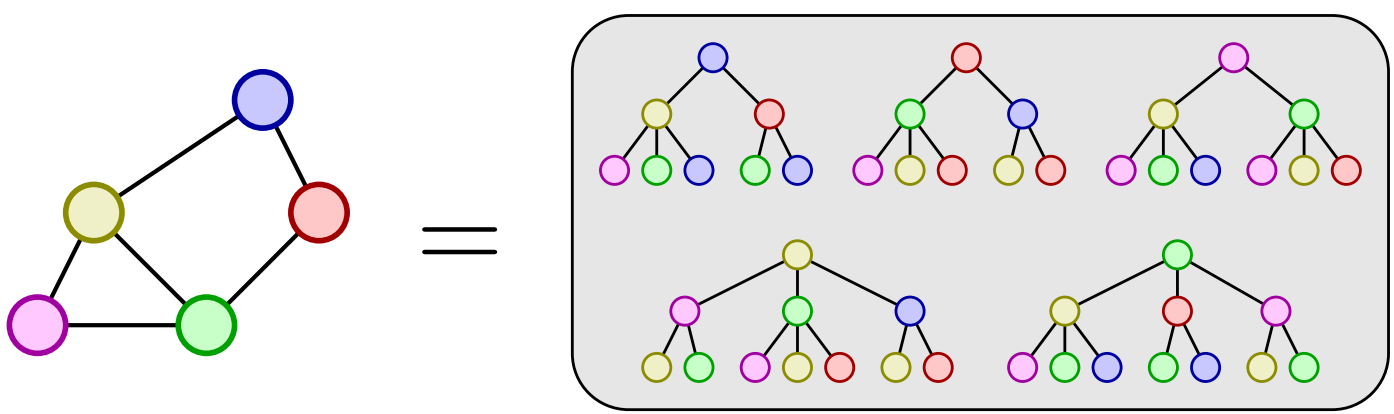
Source: https://colab.research.google.com

Relation between GNNs and Recursive Nets

Graph neural networks models can be expanded as recursive neural networks applying to trees. Each tree represents a search of depth L into the graph starting from a given node, and where L is the number of layers of the graph neural network.

graph neural network

collection of recursive networks



Relation between GNNs and Convolution Layers

Observation:

- The convolution layer can be seen as a special case of a graph neural network where the graph is a two-dimensional lattice.

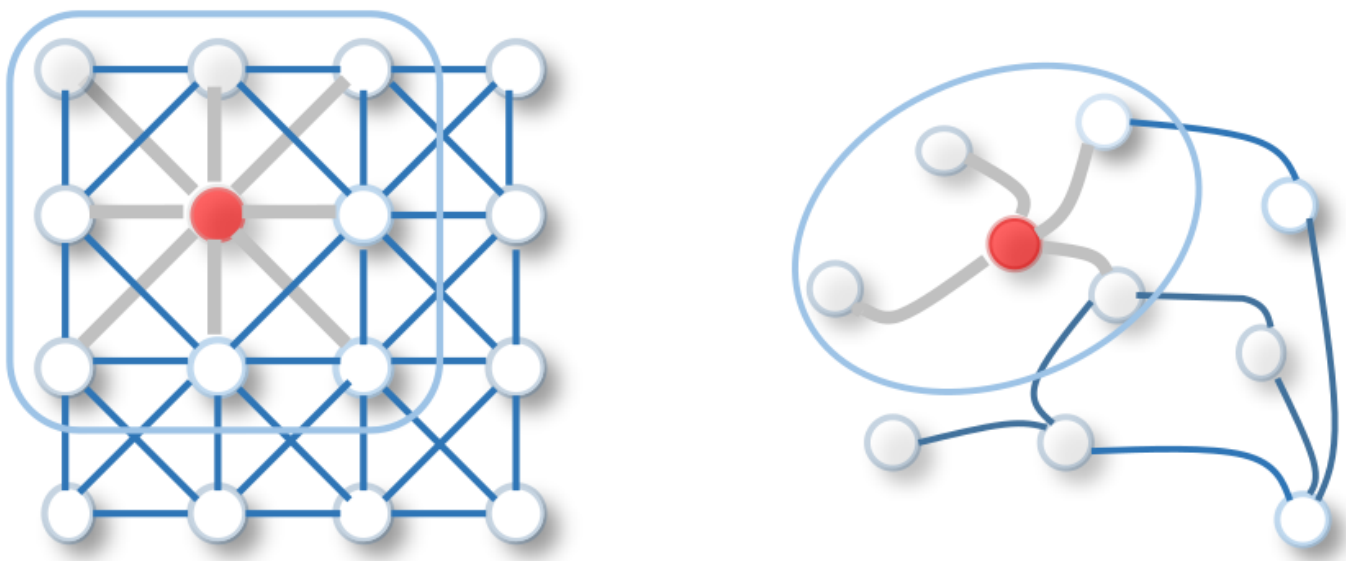
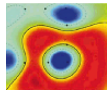


Image source: Wu et al. 2019, A Comprehensive Survey on Graph Neural Networks

Summary

- Neural networks can learn compact representations for any relevant task. This capability comes at the cost of a more difficult optimization problem.
- Structured networks are a generalization of standard neural networks that allows to handle various real-world data (e.g. images, text, graphs).
- Convolutional neural networks efficiently extract image representations by progressively expanding the semantic content while compressing the pixel-wise information.
- Graph neural networks generalize neural networks to any graph-structured input (including trees and pixel lattices).



References

O. Eberle, J. Büttner, F. Kräutli, K. Müller, M. Valleriani, and G. Montavon.

Building and interpreting deep similarity models.

CoRR, abs/2003.05431, 2020.

K. Fukushima.

Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.

Biological Cybernetics, 36(4):193–202, 1980.

T. N. Kipf and M. Welling.

Semi-supervised classification with graph convolutional networks.

In *ICLR (Poster)*. OpenReview.net, 2017.

Y. LeCun.

Generalization and network design strategies.

In *Connectionism in Perspective*. Elsevier, 1989.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts.

Recursive deep models for semantic compositionality over a sentiment treebank.

In *EMNLP*, pages 1631–1642. ACL, 2013.