Course Introduction and Machine (Deep) Learning Refresher

DAVIDE BACCIU – BACCIU@DI.UNIPI.IT

UNIVERSITÀ DI PISA

The Course

- ✓Preliminaries: ML&DL refresher; RL fundamentals
- ✓Fundamentals
  - ✓Markov Decision Processes
  - ✓Planning by Dynamic Programming
  - ✓Model-Free Prediction & Control
- ✓Value Function Methods
- ✓Policy Gradient Methods
- ✓Exploration and Exploitation
- ✓Deep reinforcement learning
- ✓Advanced topics and applications: continual learning, variational methods, RL frameworks, some case studies

13 lectures by me
2 lectures with student seminars

Course Completion

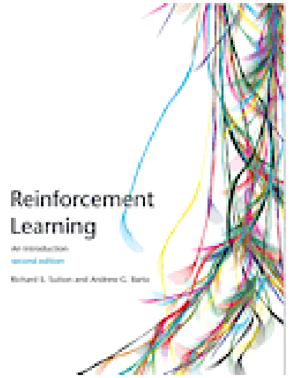
- Allievi
  - Prepare a 15 minutes presentation to be given in one of the 2 student lecture dates
  - 5 minutes Q&A on the content of the presentation
  - Seminar content
    - Read 3 relevant papers on a topic of interest for the course; summarize their content and confront the methods
    - Implement 1 RL method from literature and attempt a validation on a simple application; describe the model, its implementation and the validation results
- Ph.D. Students
  - Read 3 (or more) relevant papers on a topic of interest for the course and summarize their content in a report (6-10 pages single column, NeurIPS format)
  - Sketch/propose a novel RL method/application: report your idea in sufficient detail in a short paper (6-10 pages single column, NeurIPS format)
  - Implement a RL-based application and validate it: prepare a short presentation to report the results (10-15 slides describing the model, the implementation and the results)
  - Contact me and agree on alternative ways (e.g. using RL in your Ph.D. project, …)

Resources

The course webpage (https://elearning.di.unipi.it/course/view.php?id=190)

- ✓Course calendar & news
- ✓Slides, video lectures, additional materials
- ✓Course assignment upload

Reference Book



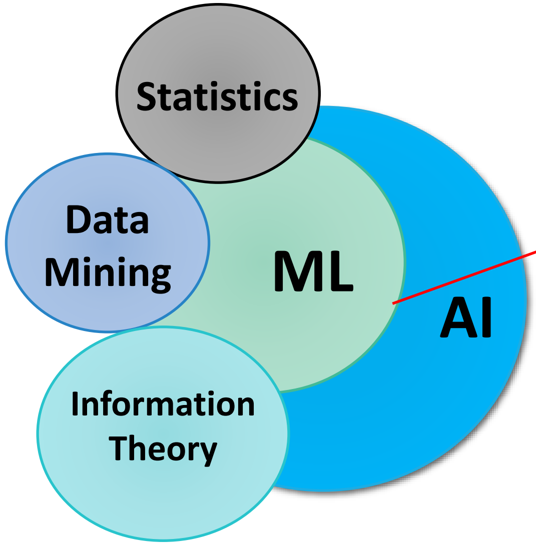
Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, Second Edition, MIT Press (available online)

Lecture Outline

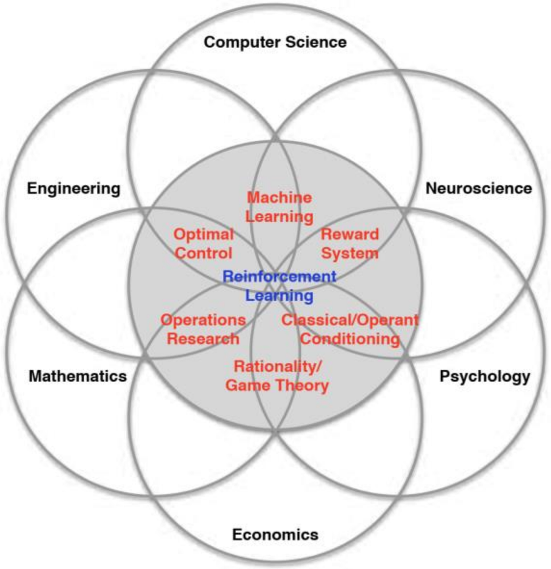
- ✓ A super-condensed refresher of machine learning
  - ✓ Data, principles, model selection, probabilistic ML
- ✓ Fundamentals of neural networks
  - ✓ Neuron model, feedforward networks, backpropagation, fundamental techniques and building blocks
- ✓ Fundamentals of deep learning
  - ✓ Autoencoders
  - ✓ Convolutional Neural Networks
  - ✓ Recurrent neural networks

Fundamentals of Machine Learning

Machine Learning (ML)



Reinforcement learning?



Machine Learning is a field of artificial intelligence dealing with models and methods that allow computer to learn from data

ML – Tasks & Data

Supervised Learning

Learn an unknown function predicting an output in response to an input

- Predicting credit risk given customer profile

(x, y)

Unsupervised Learning

Identification of structures, regularities associations and anomalies in the data

- Signaling anomalous transactions

(x)

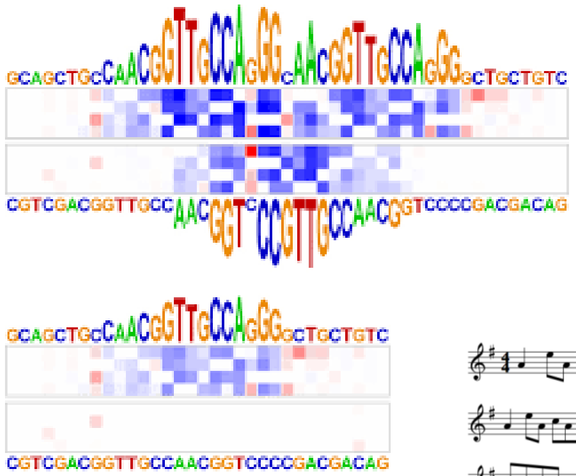
Reinforcement Learning

Learning of a policy or complex behaviour while being allowed to observe only partial responses from the interaction with the environment or the user

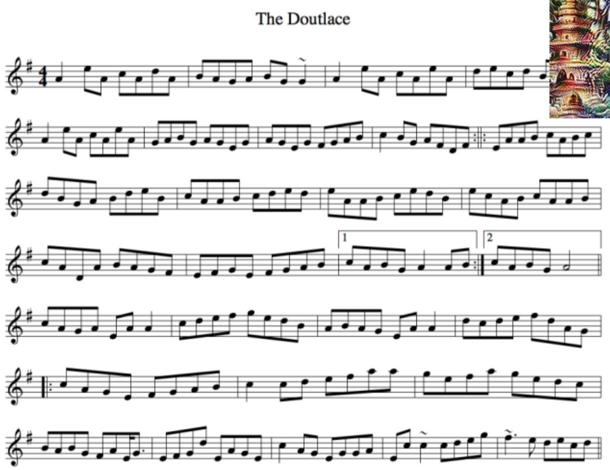
- Autonomous agents

(s, a, r)

Modern ML tasks are often beyond recognition and prediction



Understanding, reasoning and explaining

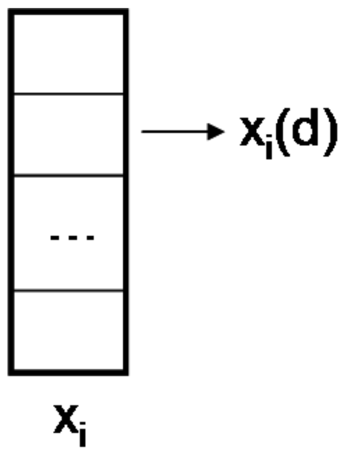


Generation

Creativity

ML – Information Representation

Vectorial data



The *i*-th input sample x_i is a *D*-dimensional numerical vector

- Continuous, categorical or mixed values
- Describes an individual of our world of interest, e.g. patients in a biomedical application

The single dimensions d are called features and numerically represent an attribute of the individual

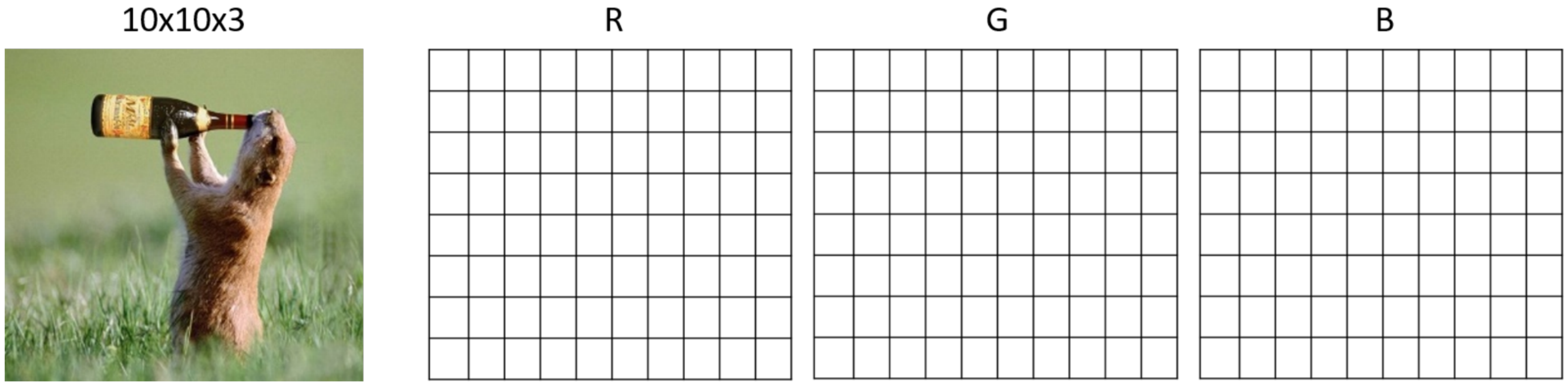
- E.g. if x_i describes a patient, $x_i(d)$ can be his/her age

Also output samples y_i are *D'*-dimensional numerical vectors

ML – Information Representation

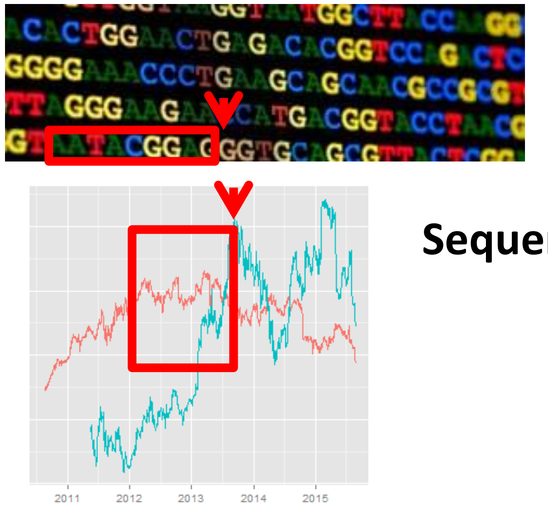
Images

Images are matrices of pixels intensity



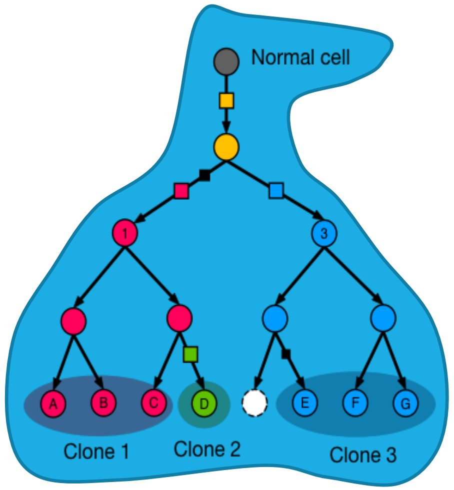
Structured Data

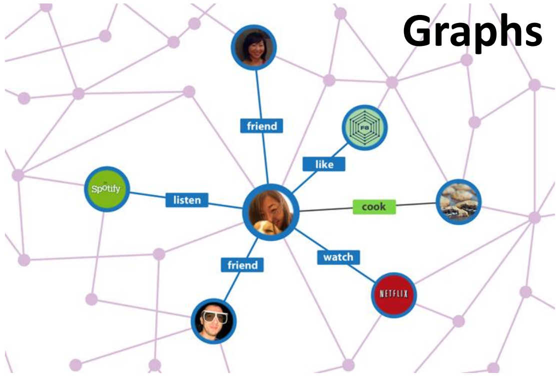
Structured (relational) information comprising atomic elements that needs to be interpreted in the context of the surrounding elements



Sequences

Trees



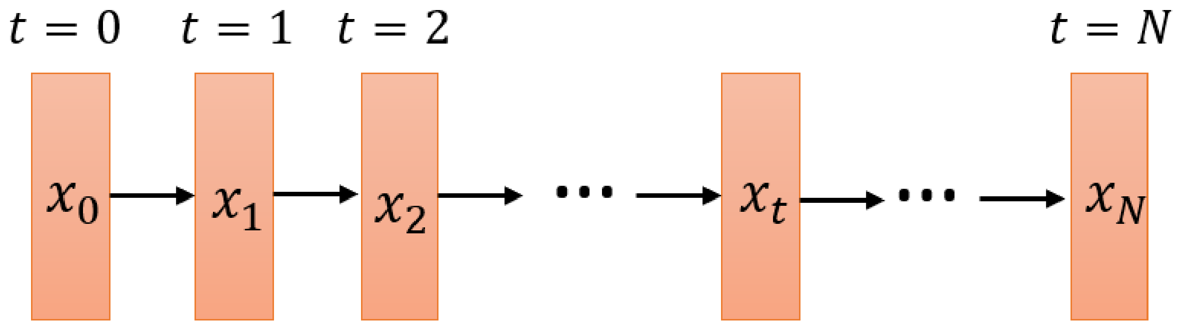


Graphs

Typically varying in size and topology

ML – Information Representation

Sequential data



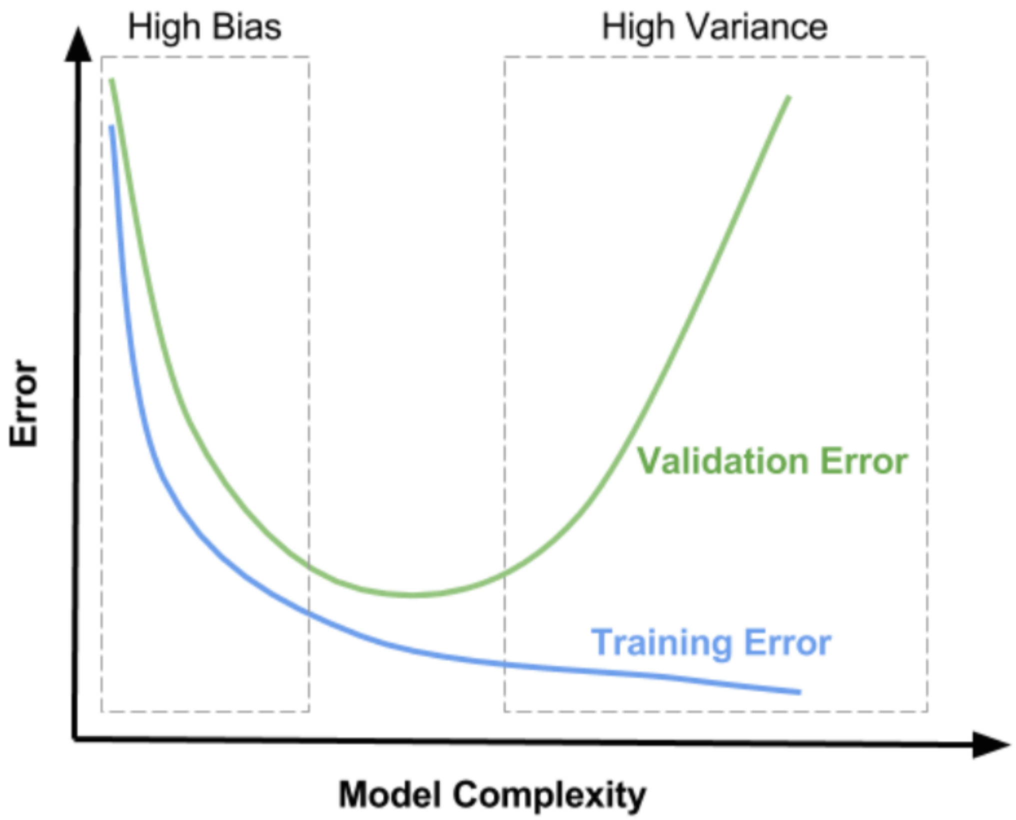
Variable size data characterized by sequentially dependent information

- Examples: financial timeseries, sequences of operations, natural language sentences

Each element of the sequence is a vector

In ML can be used both as input and output information

Fundamental concepts



ML model– Computational model $M_{\alpha}\ (D, \theta)$ that can be applied to data D and whose behavior is regulated by adaptive parameters θ and by hyperparameters α (externally set)

Training – Process through which model M parameters θ are modified to adapt to training data D_{Tr} by optimizing a cost function $E(\theta | D_{Tr})$

Generalization – Sought property of a model M that, trained on D_{Tr}, generalizes well its output on new/fresh data D_{Tst} (test)

Overfitting – Problem inducing poor generalization in a trained model, which behaves excellently on training data while being very poor on test

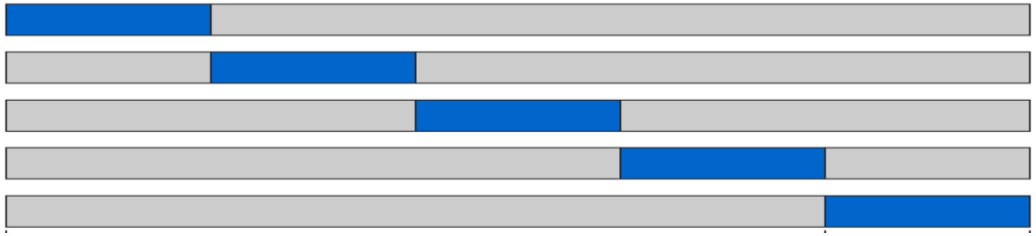
Model Selection

Set of techniques from robust statistics to measure generalization, avoid overfitting and reduce the effect of model bias

1. Separate training phase, from the choice of model configuration (including hyperparameters), from model generalization assessment

Data | Training | Validation | Testing

2. Iterate the process changing data to obtain robust performance estimates



k-fold validation

Probabilistic ML Refresher

On the blackboard

Posteriors, Marginals and Likelihood

A (general) probabilistic learning model comprises

- ✓ Observable random variables X (data)
- ✓ Hidden random variables Z (latent)
- ✓ Model parameters θ

Also model parameters θ can have their prior $\mathrm{P}(\theta)$ (Bayesian Learning)

$$\underbrace{P(Z|X,\theta)}_{\text{Posterior}} = \frac{\overbrace{P(X|Z,\theta)}^{\text{Likelihood}}\overbrace{P(Z|\theta)}^{\text{Prior}}}{\underbrace{P(X|\theta)}_{\text{Marginal}}} = \frac{P(X|Z,\theta)P(Z|\theta)}{\int P(X|Z,\theta)P(Z|\theta)dz}$$

Maximum Likelihood Learning

Find model parameters by maximizing model likelihood

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log P(X|\theta) = \underset{\theta}{\operatorname{argmax}} \log \int P(X|Z,\theta)P(Z|\theta)dz$$

Expectation-Maximization Algorithm

(E) Given the current model parameters θ^k compute

$$Q\left(\theta|\theta^k\right) = \mathbb{E}_{Z|X,\theta^k}\left[\log P\left(X, Z|\theta^k\right)\right]$$

(M) Given the current posterior expectation update parameters

$$\theta^k = \arg\max_{\theta} Q\left(\theta|\theta^k\right)$$

Evidence Lower Bound (ELBO)

Posterior is not always easily computable or available in closed-form so we minimize a lower bound with respect to a variational distribution $Q(Z|\lambda)$ with parameters λ

$$\log P(X|\theta) \geq \underbrace{\mathbb{E}_{Q(Z|\lambda)}[\log P(X,Z|\theta)]}_{\text{Expectation of complete likelihood}} - \underbrace{\mathbb{E}_{Q(Z|\lambda)}[\log Q(Z|\lambda)]}_{\text{Entropy}} = \underbrace{\mathcal{L}(X,\theta,\lambda)}_{\text{ELBO}}$$

- ✓ Optimize model (θ) and variational (λ) parameters
- ✓ Equality holds when $Q(Z|\lambda) = P(Z|\theta)$ (bound is tight)

Sampling Approximations

Alternatively (to the variational approximation) the incomputable posterior can be estimated by sampling

$$\lim_{L\to\infty} \frac{1}{L} \sum_{l=1}^{L} \mathbb{I}[x^l = i] = p(x = i)$$

- ✓Ancestral sampling
- ✓Gibbs sampling
- ✓Markov Chain Monte Carlo Methods
- ✓Importance sampling (particle filtering)

Fundamentals of Neural Networks

Neural Networks and Inductive Bias

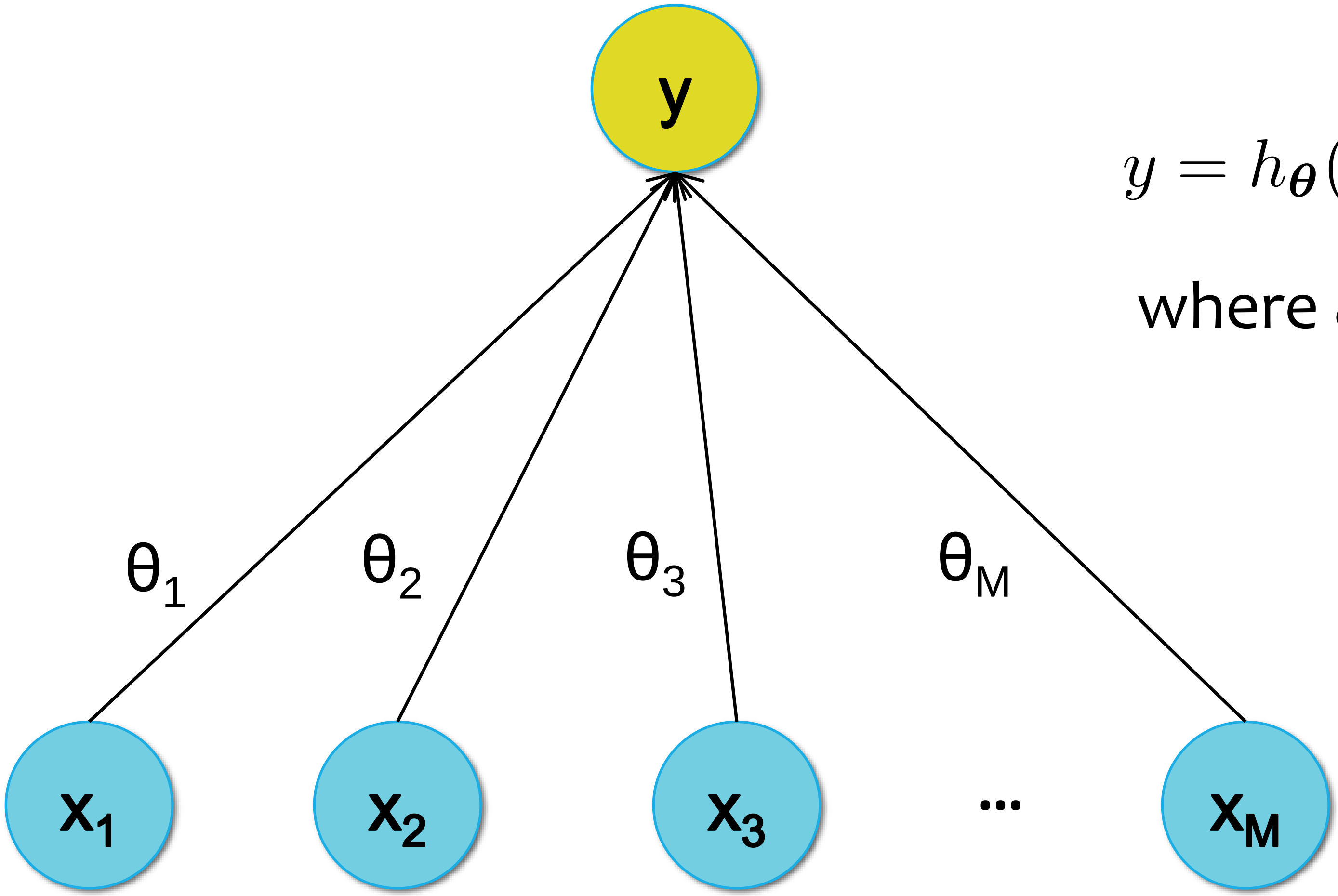
Neural network architectural design influences deeply

- The type of tasks it can solve
- The type of data it can handle
- The quality of generalization of its results

Architectural choices

- Topology and weight sharing
- Activation functions
- Regularization strategies
- Loss functions

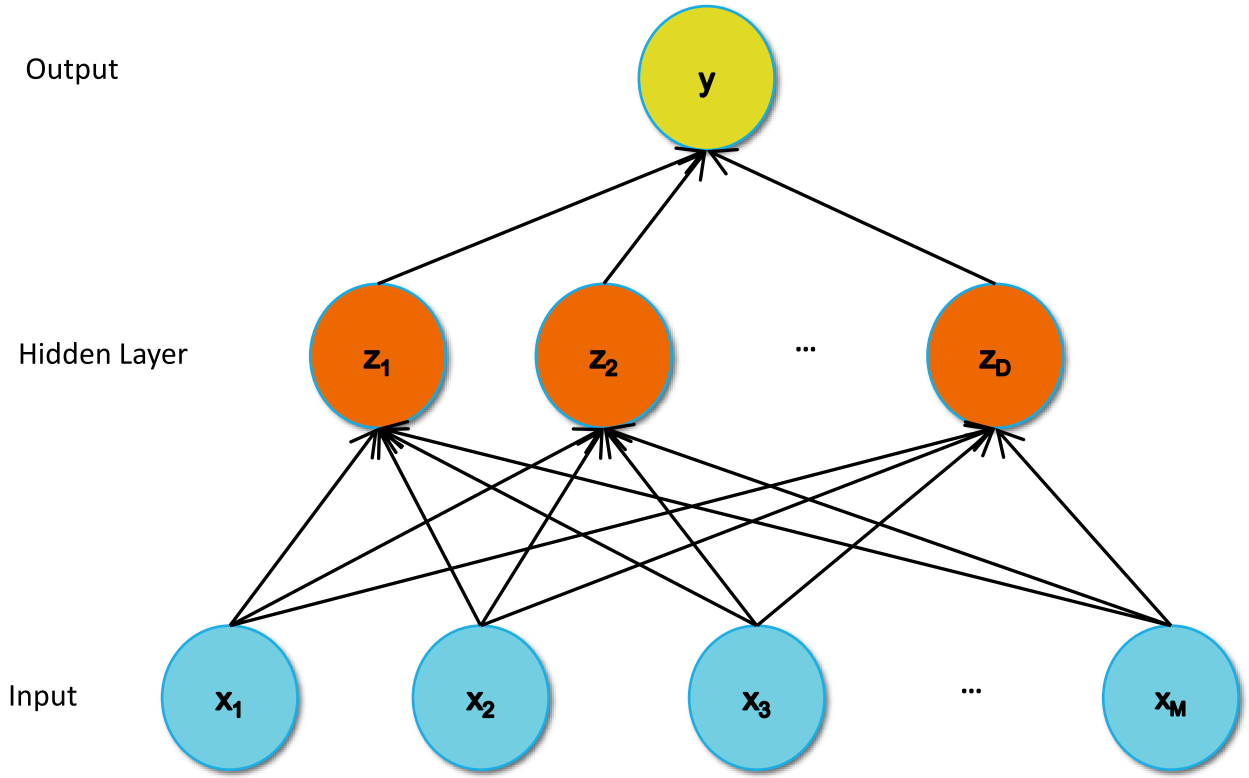
Logistic Neuron



$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

where $\sigma(a) = \dfrac{1}{1 + \exp(-a)}$

Multilayer Perceptron (Single Output)



(E) **Output (sigmoid)**
$y = \frac{1}{1+\exp(-b)}$

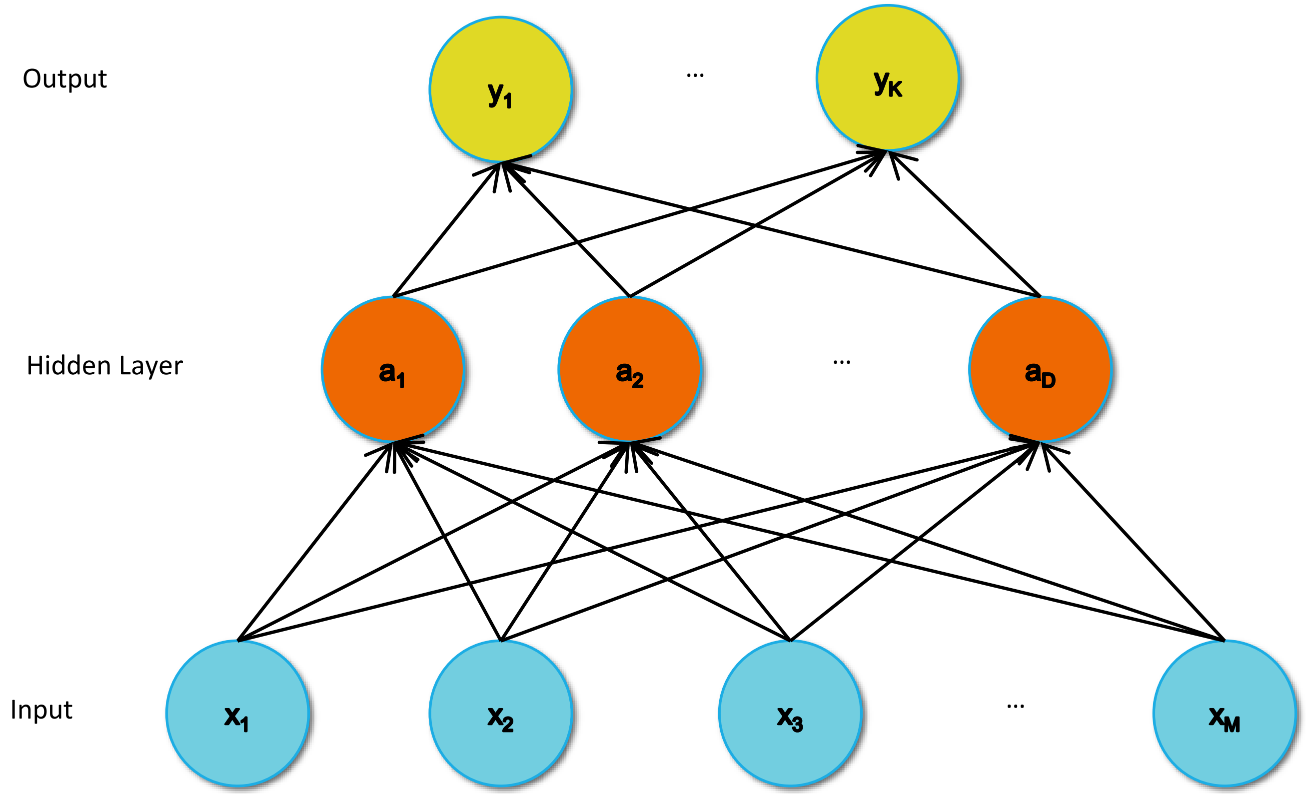
(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) **Hidden (sigmoid)**
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

(A) **Input**
Given $x_i, \ \forall i$

Multilayer Perceptron (Multi-class output)



(E) **Output (softmax)**
$$y_k = \frac{\exp(b_k)}{\sum_{l=1}^{K}\exp(b_l)}$$

(D) **Output (linear)**
$$b_k = \sum_{j=0}^{D} \beta_{kj} z_j \ \forall k$$

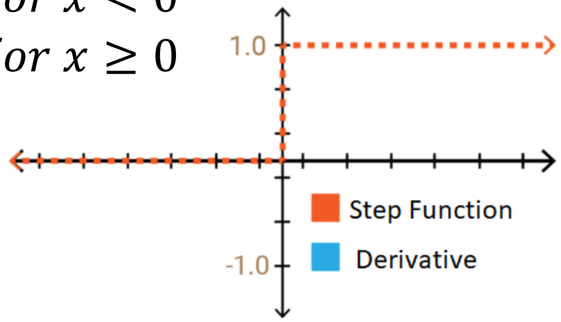
(C) **Hidden (nonlinear)**
$$z_j = \sigma(a_j), \ \forall j$$

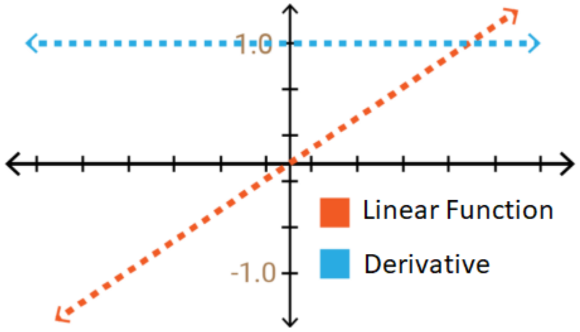
(B) **Hidden (linear)**
$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$$

(A) **Input**
Given $x_i, \ \forall i$

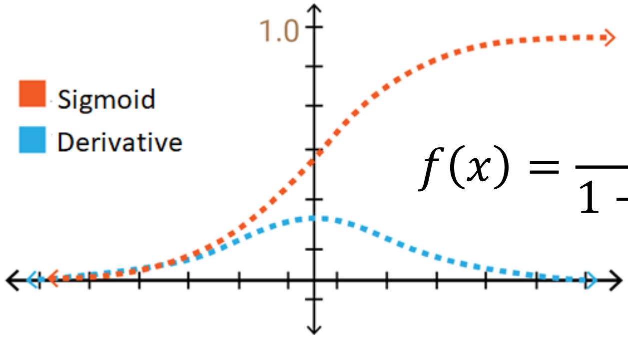
(Some) Activation Functions

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$$



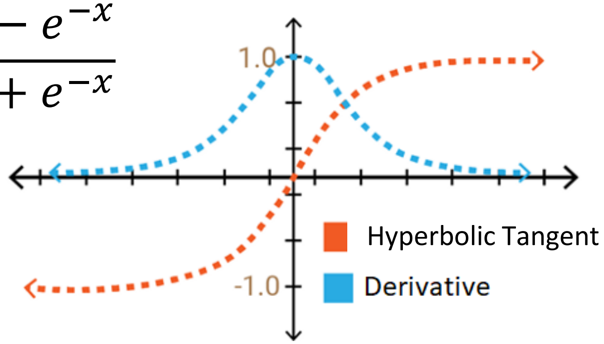


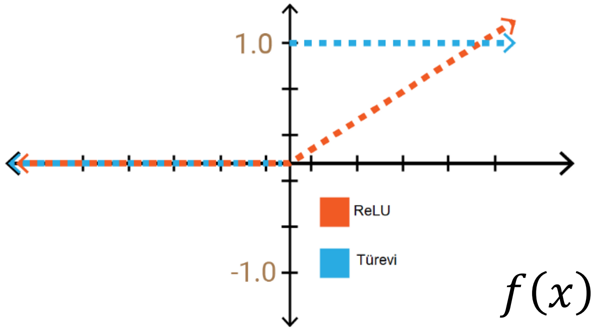
$$f(x) = x$$



$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



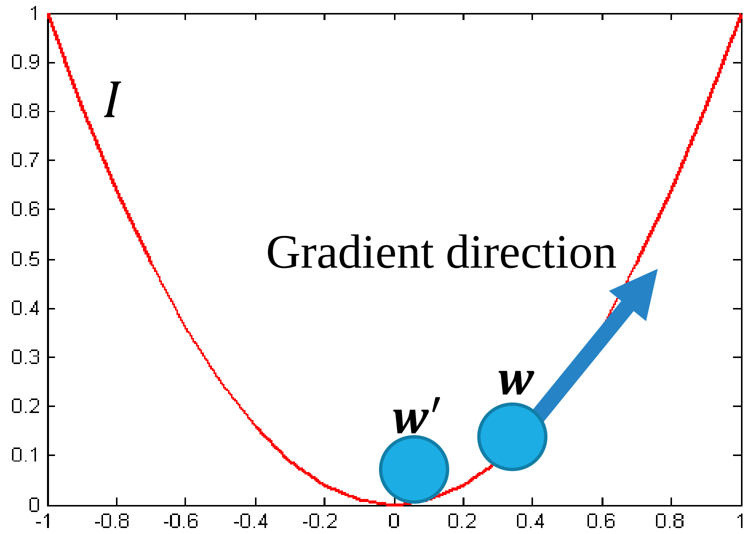


$$f(x) = \begin{cases} 0 \; (or \; \epsilon) \; for \; x < 0 \\ x \qquad\qquad for \; x \geq 0 \end{cases}$$

Training NNs – Cost minimization by Gradient Descent

Weights are updated in the opposite direction of the gradient of the loss function

$$w_i' = w_i - \alpha \frac{\partial I}{\partial w_i}$$



Gradient can be backpropagated by the chain rule

$$\begin{tikzcd}
\frac{\partial I}{\partial x} & \frac{\partial I}{\partial h_1} \arrow[l, "\frac{\partial h_1}{\partial x}"'] \arrow[d, "\frac{\partial h_1}{\partial w_1}"'] & \cdots \arrow[l, "\frac{\partial h_2}{\partial h_1}"'] & \frac{\partial I}{\partial h_n} \arrow[l, "\frac{\partial h_n}{\partial h_{n-1}}"'] \arrow[d, "\frac{\partial h_n}{\partial w_n}"'] & \frac{\partial I}{\partial y} \arrow[l, "\frac{\partial y}{\partial h_n}"'] \\
 & \frac{\partial I}{\partial \mathbf{w}_1} & \cdots & \frac{\partial I}{\partial \mathbf{w}_n} &
\end{tikzcd}$$

Loss Functions for NNs

Regression: A problem where you predict a real-value quantity.

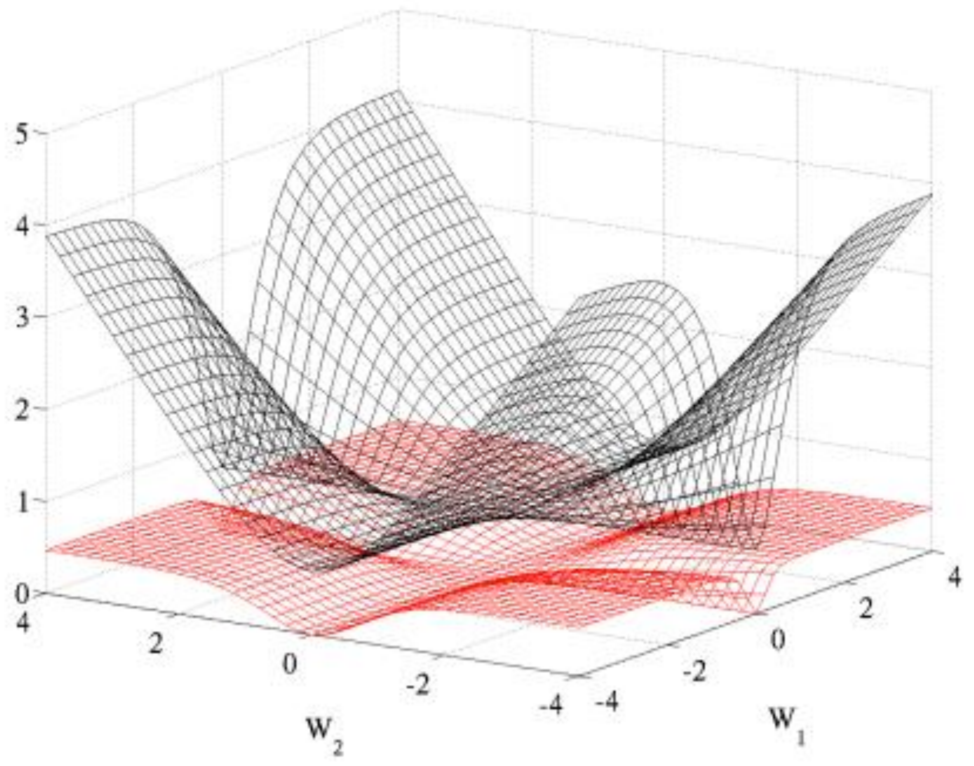
- Output Layer: One node with a linear activation unit.
- Loss Function: Quadratic Loss (Mean Squared Error (MSE))

Classification: Classify an example as belonging to one of K classes

- Output Layer: : One node with a sigmoid activation unit (K=2) or K output nodes in a softmax layer (K>2)
- Loss function: Cross-entropy (i.e. negative log likelihood)

	Forward	Backward
Quadratic	$J = \frac{1}{2}(y - y^*)^2$	$\frac{dJ}{dy} = y - y^*$
Cross Entropy	$J = y^* \log(y) + (1 - y^*) \log(1 - y)$	$\frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*) \frac{1}{y - 1}$

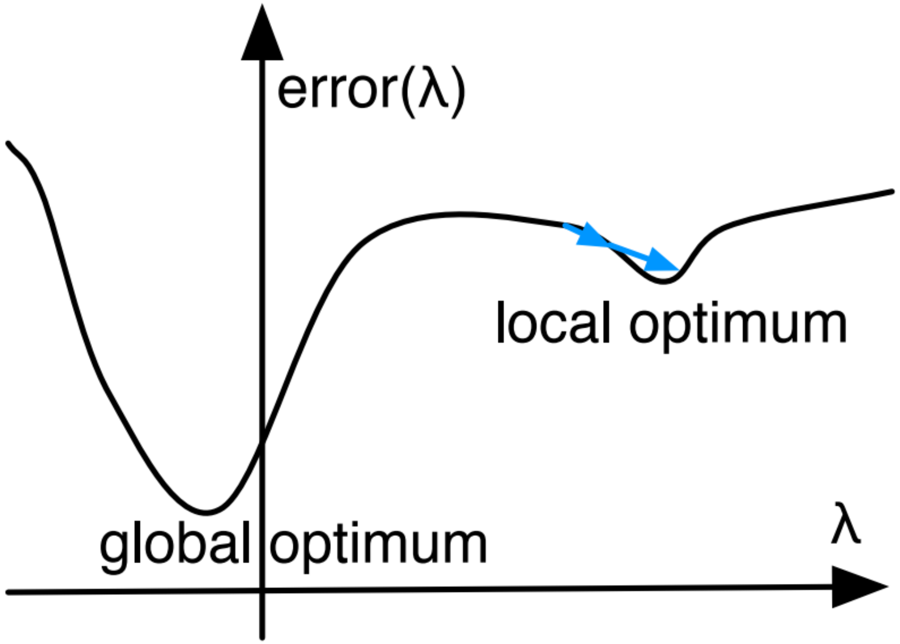
Cross-entropy vs. Quadratic loss



Glorot & Bentio (2010)

Figure 5: *Cross entropy (black, surface on top) and quadratic (red, bottom surface) cost as a function of two weights (one at each layer) of a network with two layers, W_1 respectively on the first layer and W_2 on the second, output layer.*

Cost functions are (unfortunately) more complex than simple convex functions



Local optimum

Optimization Algorithms

Standard Stochastic Gradient Descent (SGD)

- Easy and efficient but difficult to pick up the best learning rate
- Often used with momentum (exponentially weighted history of previous weights changes)

RMSprop

- Adaptive learning rate method (reduces it using a moving average of the squared gradient)
- Fastens convergence by having quicker gradients when necessary

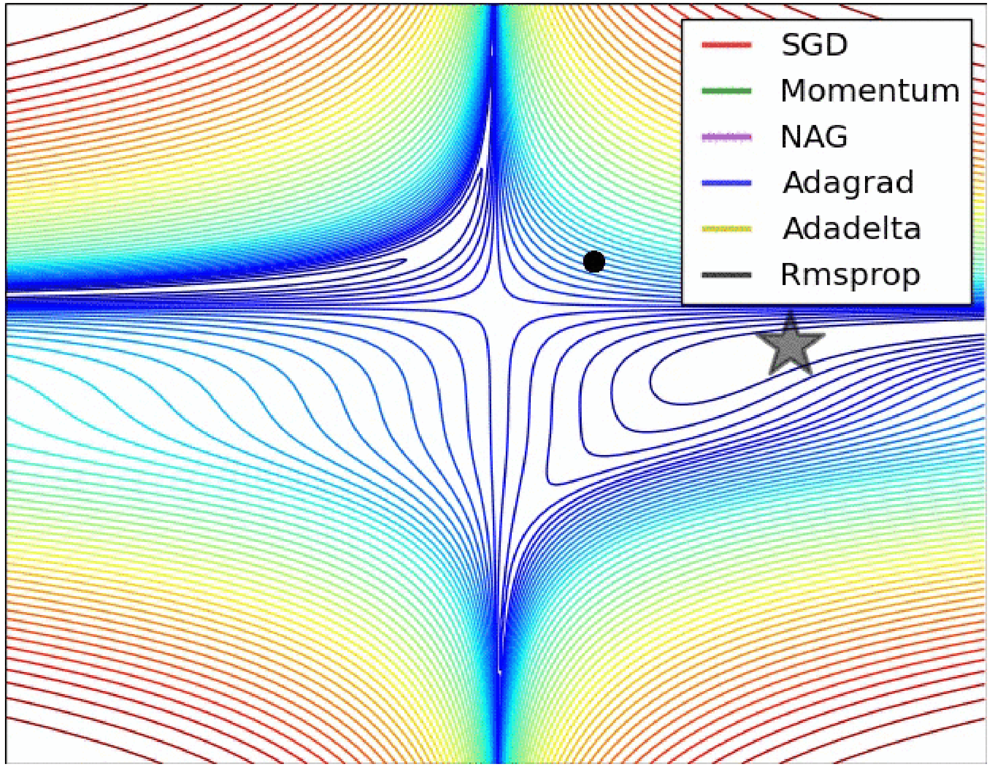
Adagrad

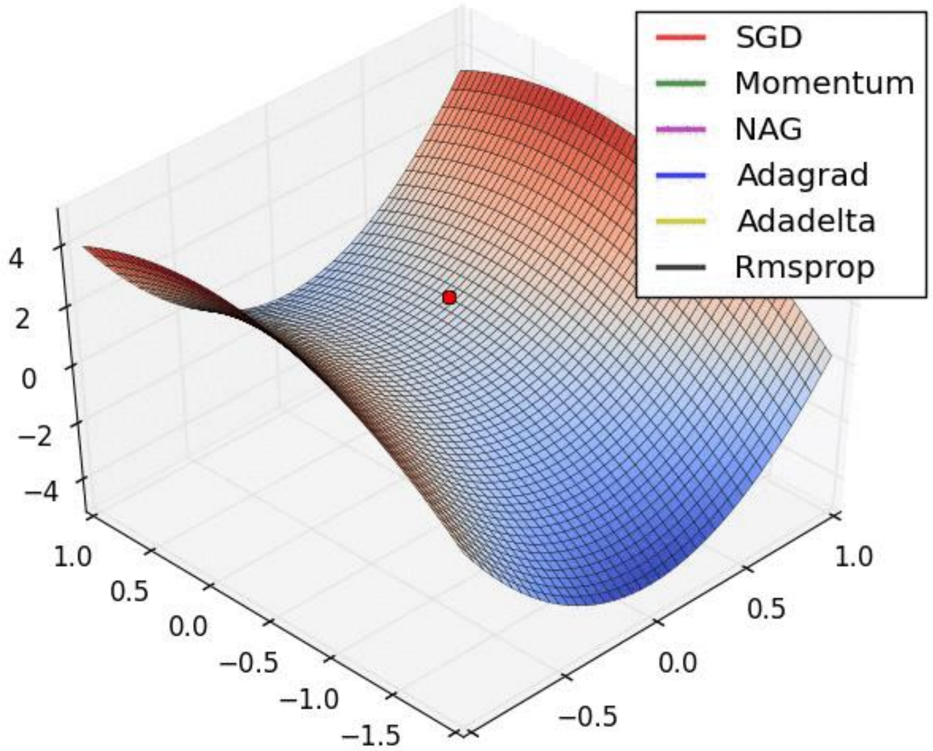
- Like RMSprop with element-wise scaling of the gradient

ADAM

- Like Adagrad but adds an exponentially decaying average of past gradients like momentum

Optimization Algorithms



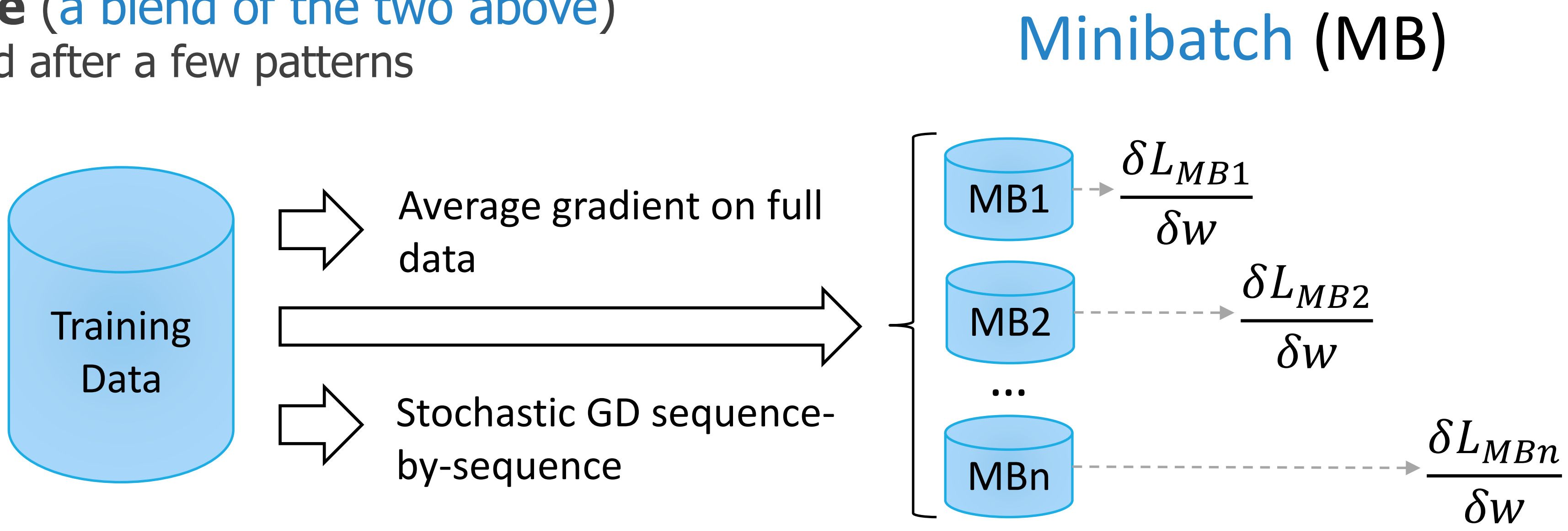


Learning fashions

Sequential mode (on-line, stochastic, or per-pattern)
- Weights updated after each pattern is presented

Batch mode (off-line or per-epoch)
- Weights updated after all patterns are presented

Minibatch mode (a blend of the two above)
- Weights updated after a few patterns



Convergence Criteria

Learning is obtained by repeatedly supplying training data and adjusting by backpropagation

- Typically 1 training set presentation = 1 epoch

We need a stopping criteria to define convergence

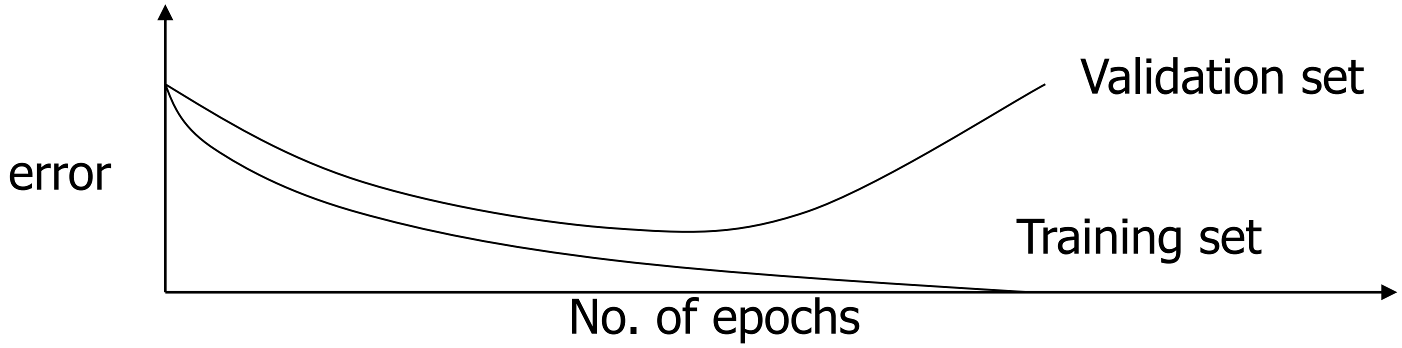
- Euclidean norm of the gradient vector reaches a sufficiently small value
- Absolute rate of change in the average squared error per epoch is sufficiently small
- Validation for generalization performance : stop when generalization performance reaches a peak

Early Stopping

Keep a hold-out validation set and assess accuracy after (every/some) epoch.

Maintain weights for best performing network on the validation set and stop training when error increases beyond this

Always let the network run for some epochs before deciding to stop (patience parameter), then backtrack to best result



Regularization

Constrain the learning model to avoid overfitting and help improving generalization

Add penalization terms to the loss function that *punish* the model for excessive use of resources

- Limit the amount of weights that is used to learn a task
- Limit the total activation of neurons in the network

$$J' = J(y, y^*) + \lambda R(\cdot)$$

Hyperparameter to be chosen in model selection

$R(W_\theta)$ Penalty on parameters

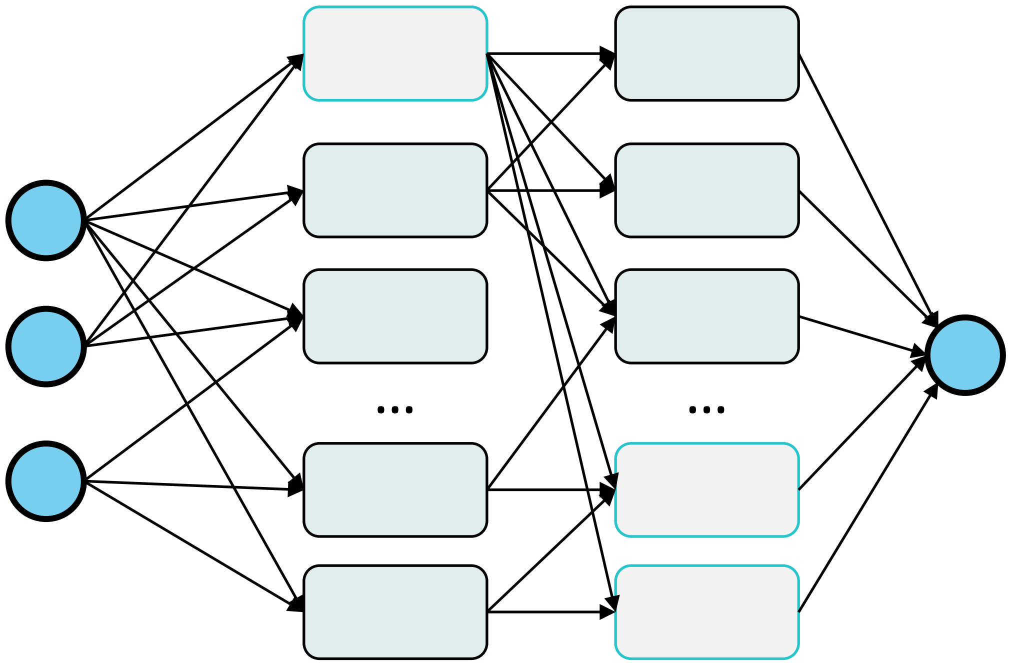
$R(Z)$ Penalty on activations

Common penalty terms (norms)

- ✓ 1-norm $||A||_1 = \sum_{ij} |a_{ij}|$
  - Parameters: $R(W_\theta) = ||W_\theta||_1^2$
  - Activations: $R(Z(X)) = ||Z(X)||_1^2$ (Z hidden unit activation)
- ✓ 2-norm $||A||_2 = \sqrt{\sum_{ij} a_{ij}^2}$
  - Parameters: $R(W_\theta) = ||W_\theta||_2^2$
  - Activations: $R(Z(X)) = ||Z(X)||_2^2$ (Z hidden unit activation)
- ✓ Any p-norm and more…

Dropout Regularization

Randomly disconnect units from the network during training

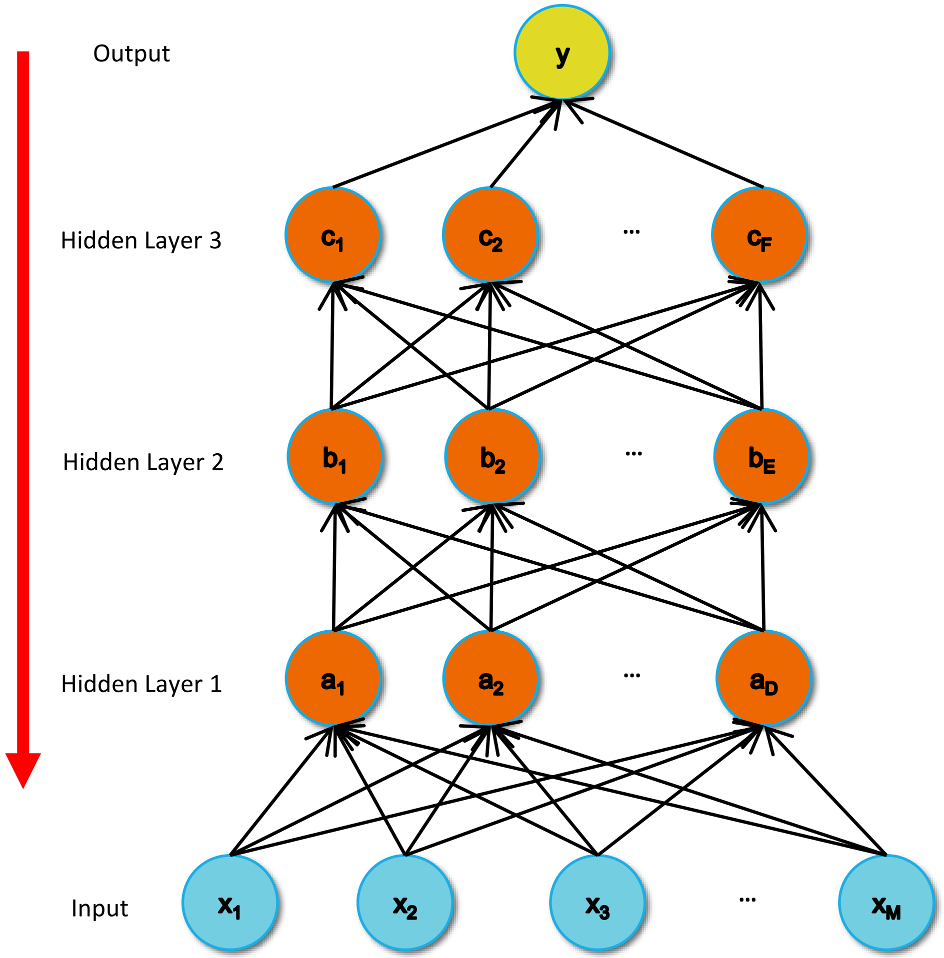


- ✓ Regulated by unit dropping hyperparameter
- ✓ Prevents unit coadaptation
- ✓ Committee machine effect
- ✓ Need to adapt prediction phase
- ✓ Used at prediction time gives predictions with confidence intervals
- ✓ Dropconnect: drops single connections

Fundamental Deep Learning Models

Deep Neural Networks

Backpropagation through many layers has numerical problems that makes learning not-straightforward (Gradient Vanish/Esplosion)



Actually deep learning is way more than having neural networks with a lot of layers

Representation learning

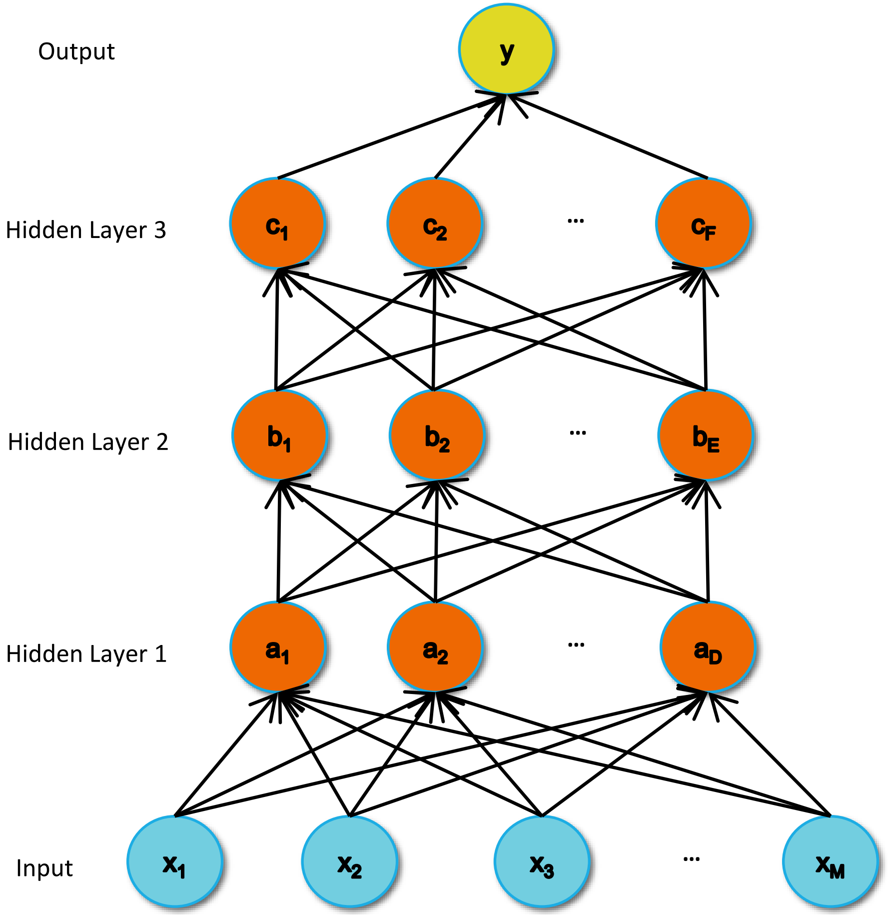
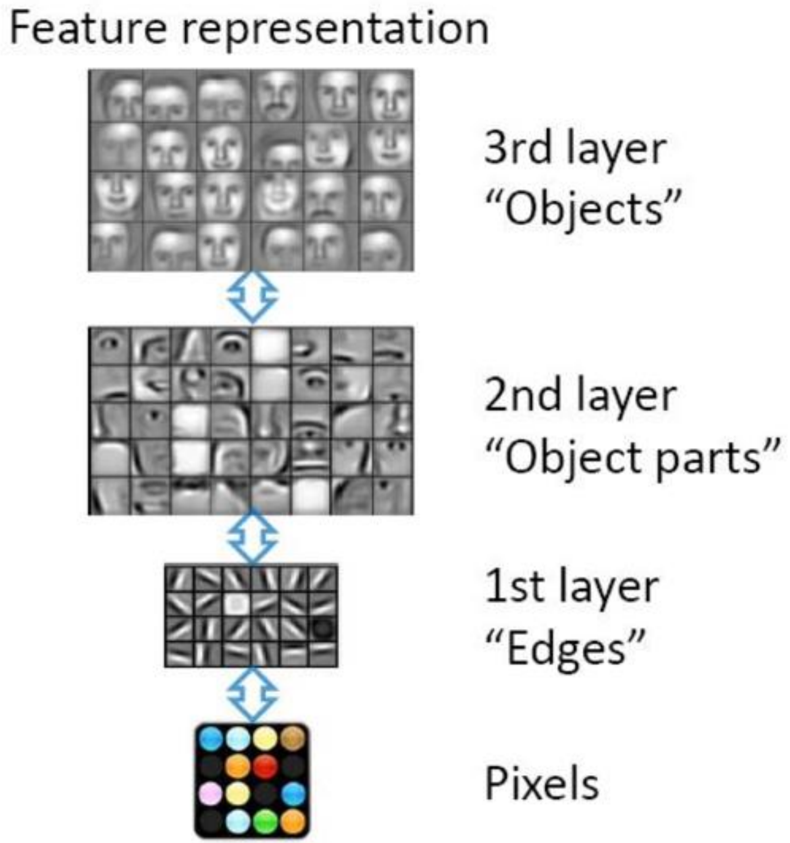


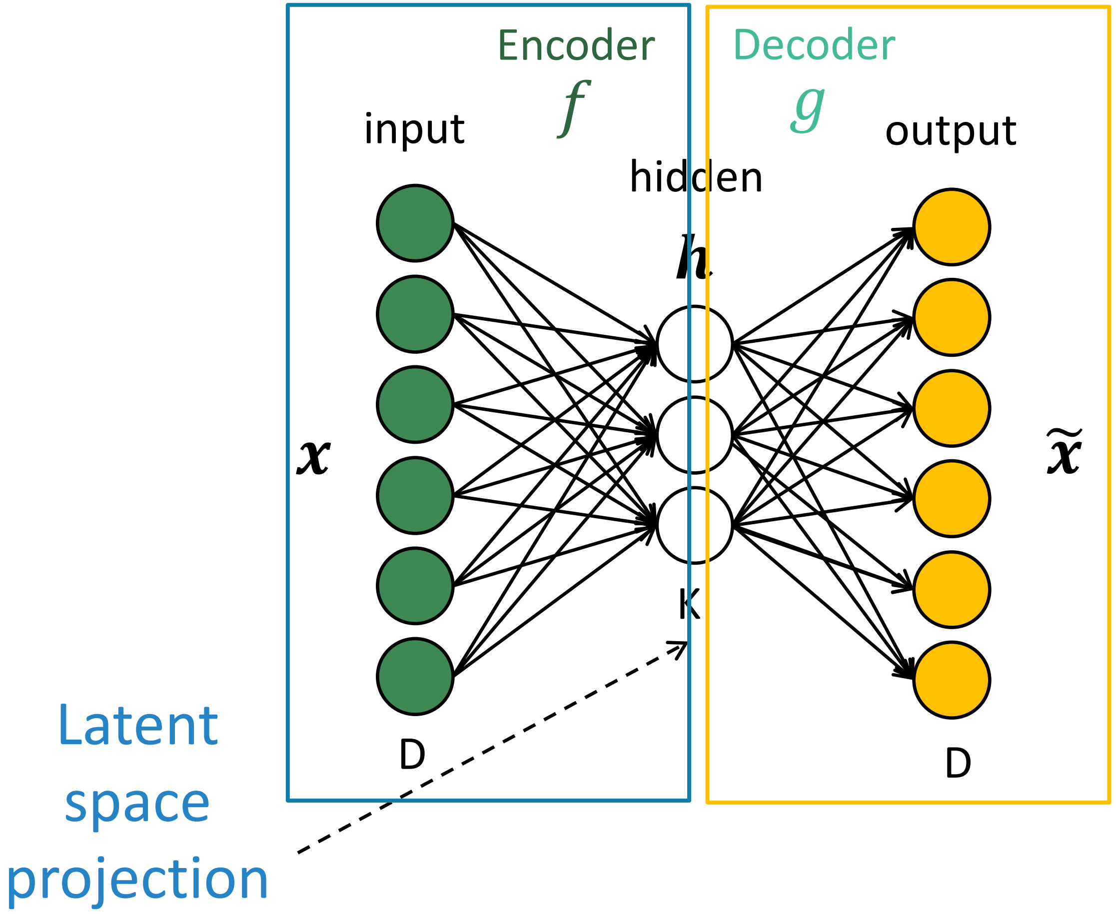
Figure from Honglak Lee (NIPS 2010)



Autoencoders

VECTORIAL DATA

Basic Autoencoder (AE)



Train a model to reconstruct the input

Passing through some form of information bottleneck

- K << D, or?
- **h** sparsely active

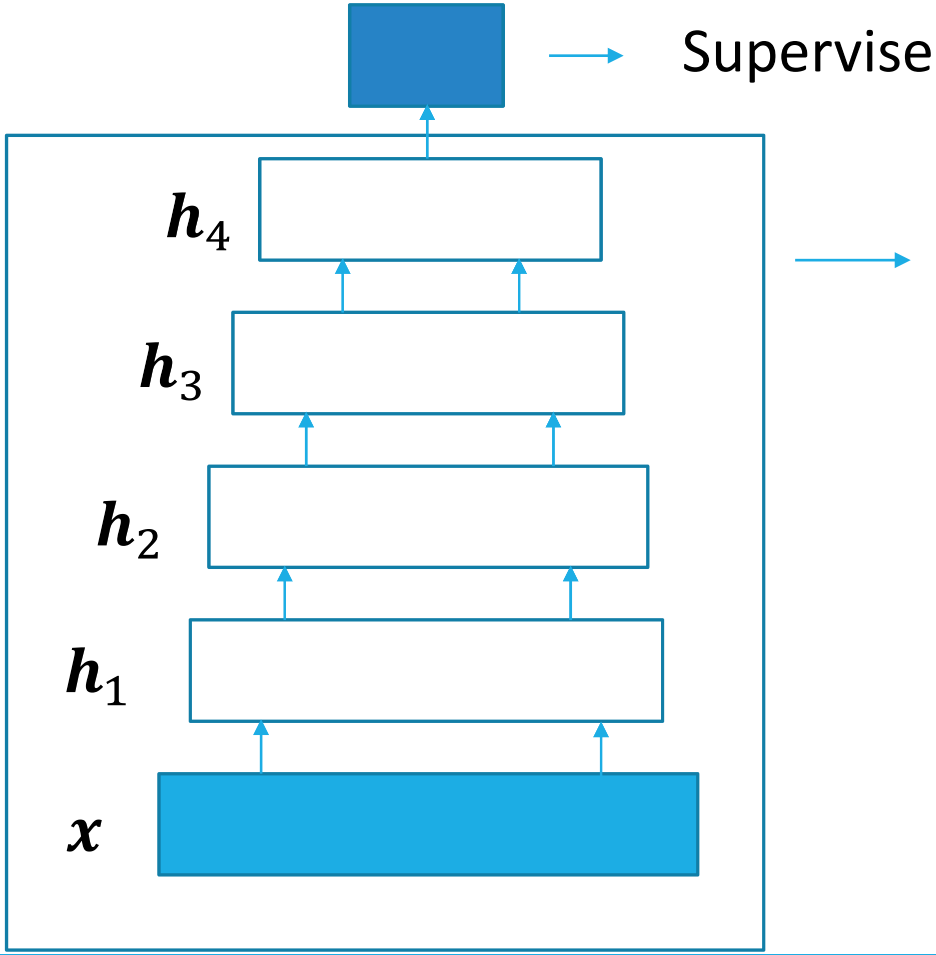
Train by loss minimization + penalization

$$J_{SAE}(\theta) = \sum_{\boldsymbol{x} \in S}(L(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \lambda\Omega(\boldsymbol{h}))$$

$$\Omega(\boldsymbol{h}) = \Omega(f(\boldsymbol{x})) = \sum_j |h_j(\boldsymbol{x}))|$$

$$\Omega(\boldsymbol{h}) = \Omega(f(\boldsymbol{x})) = \left\|\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\right\|_F^2$$

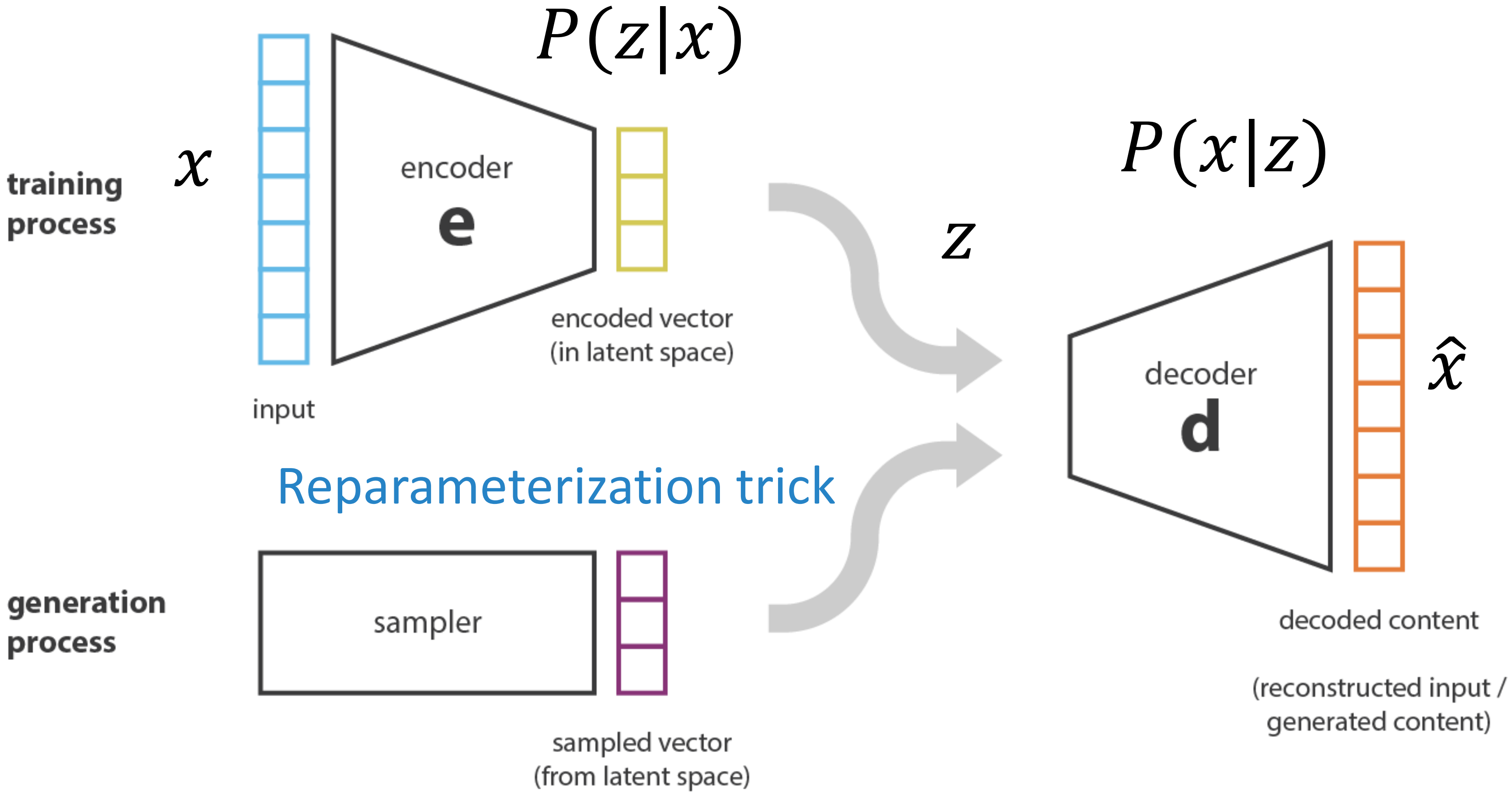
Deep Autoencoder



Supervised learning

- Unsupervised training
- Hierarchical autoencoder
- Extracts a representation of inputs that facilitates
  - Data visualization, exploration, indexing,…
  - Realization of a supervised task

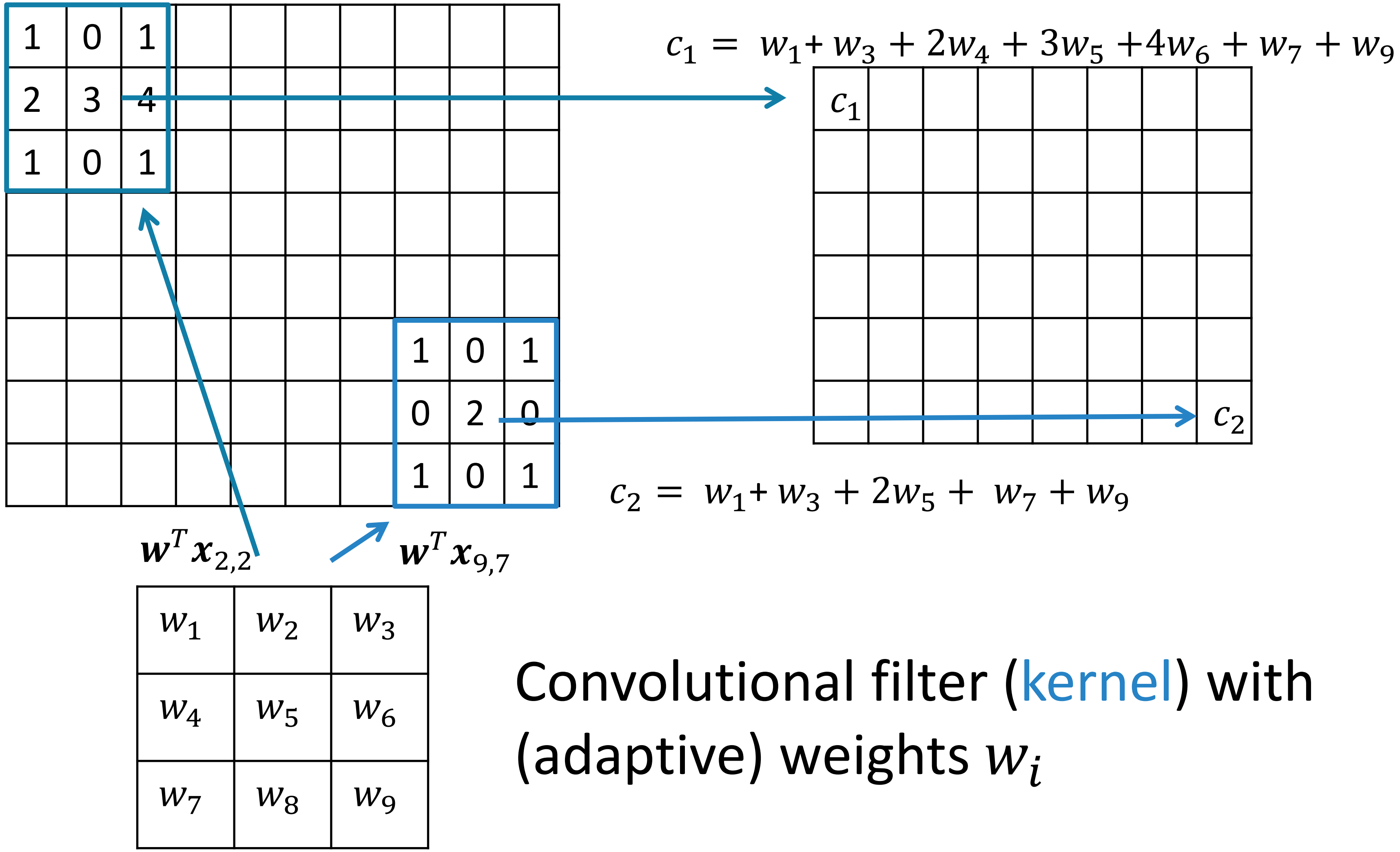
Variational Autoencoder (VAE)



Convolutional Neural Networks

IMAGE DATA

Adaptive Convolution Operator

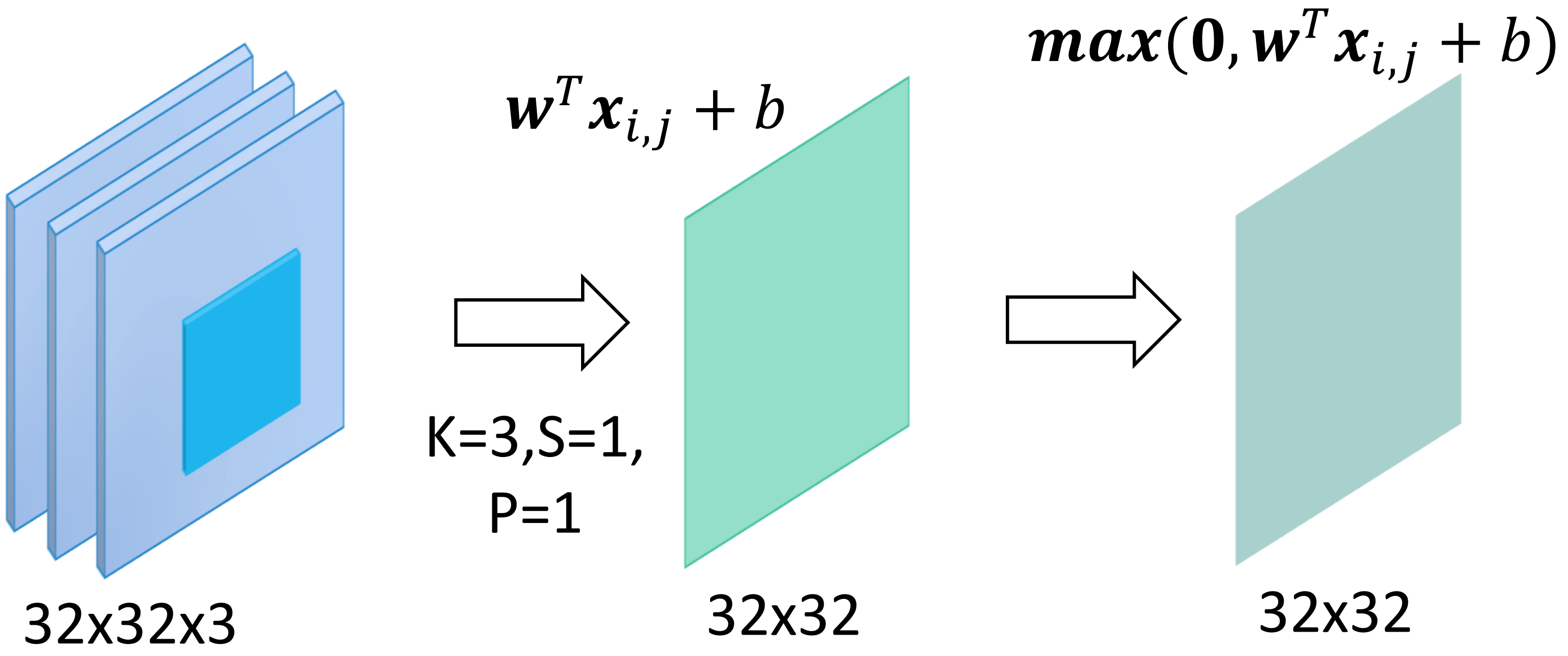


Convolutional filter (kernel) with (adaptive) weights w_i

Feature Map Transformation

Convolution is a linear operator

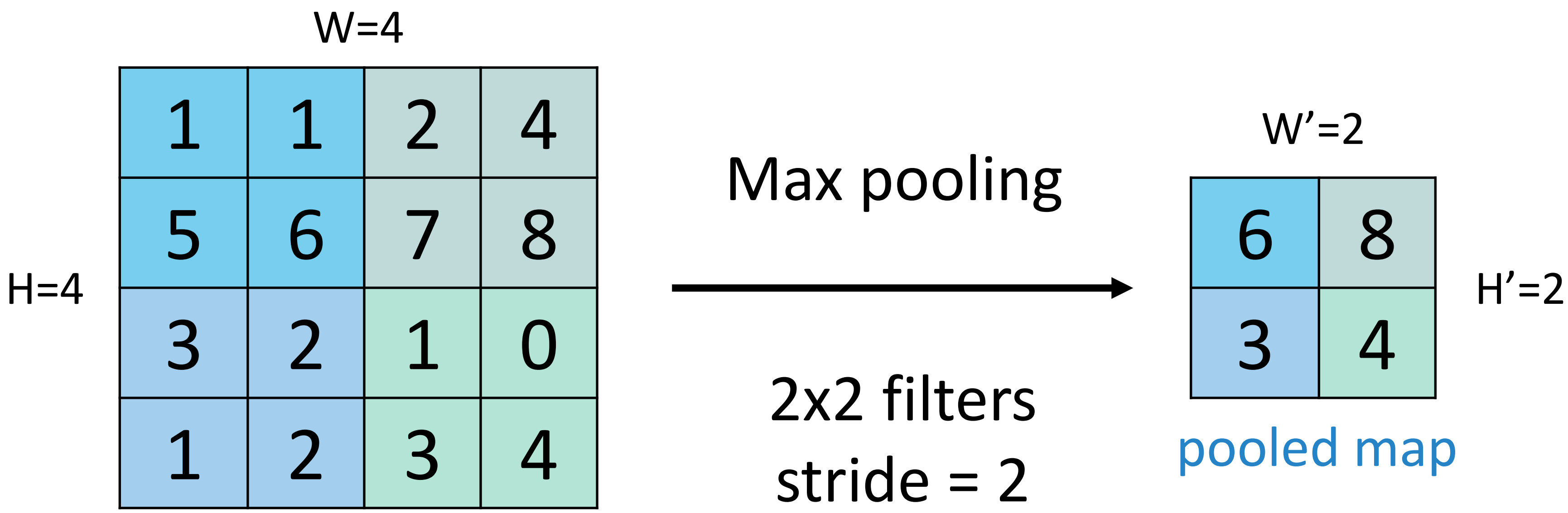
Apply an element-wise nonlinearity to obtain a transformed feature map



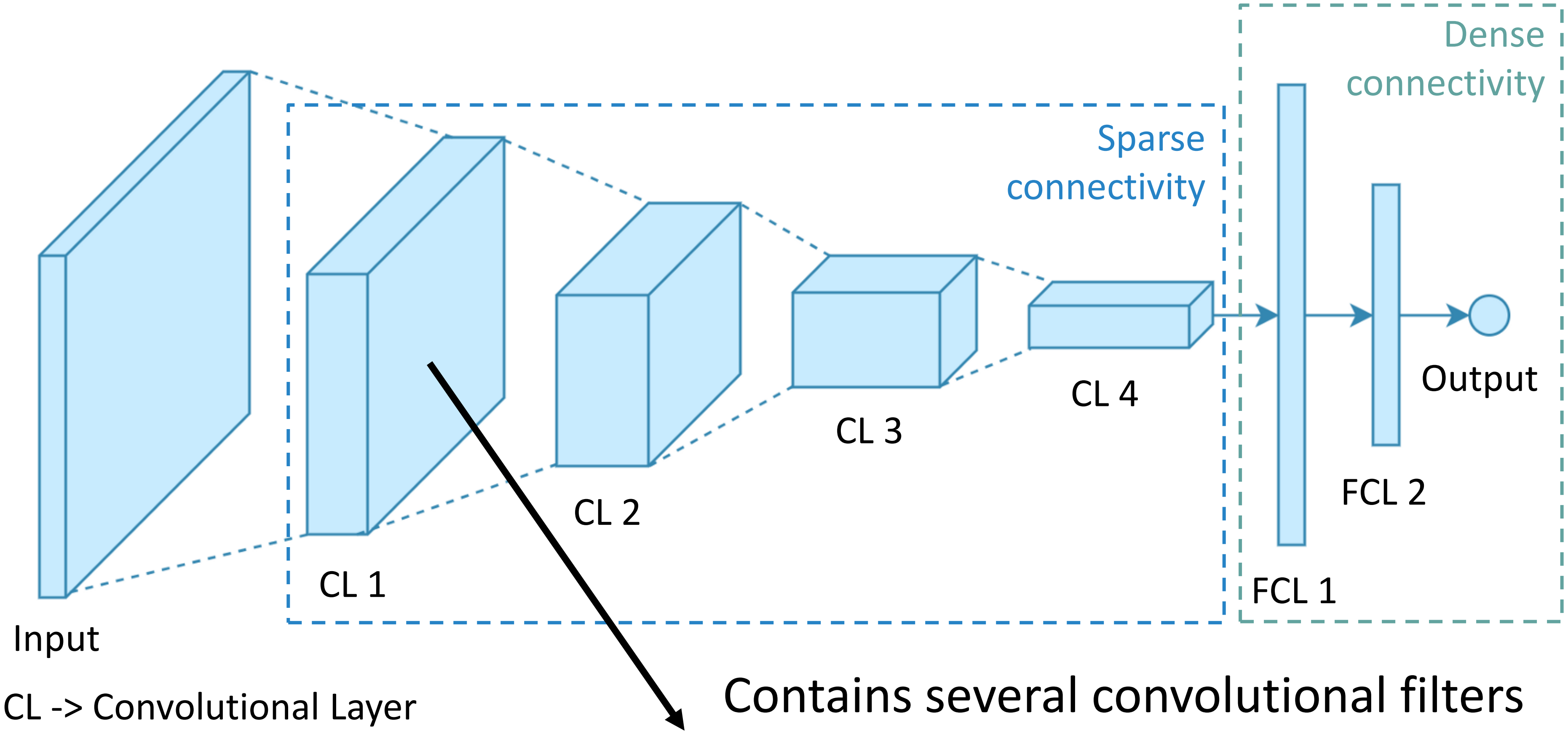
Pooling

Operates on the feature map to make the representation

- Smaller (subsampling)
- Robust to (some) transformations



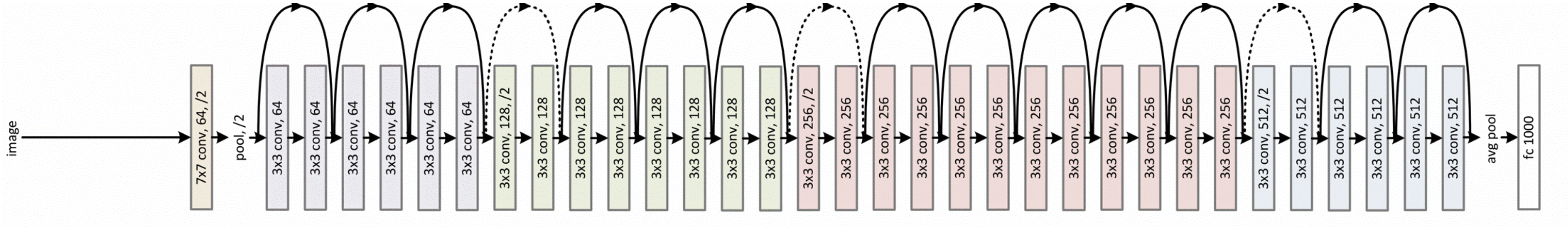
The Bigger Picture

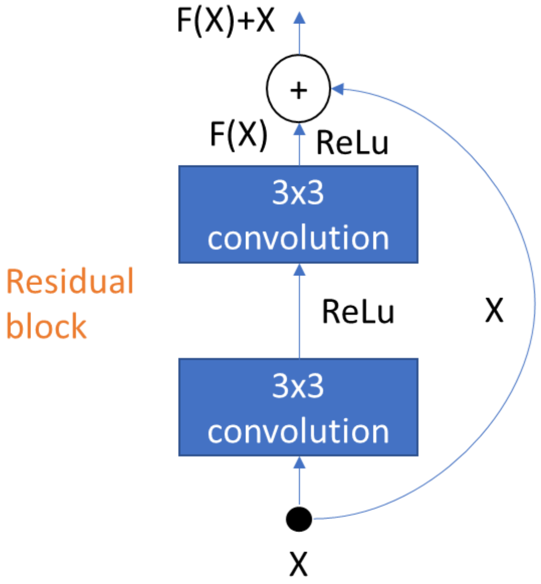


CL -> Convolutional Layer
FCL -> Fully Connected Layer

Contains several convolutional filters with different size and stride

Residual Blocks

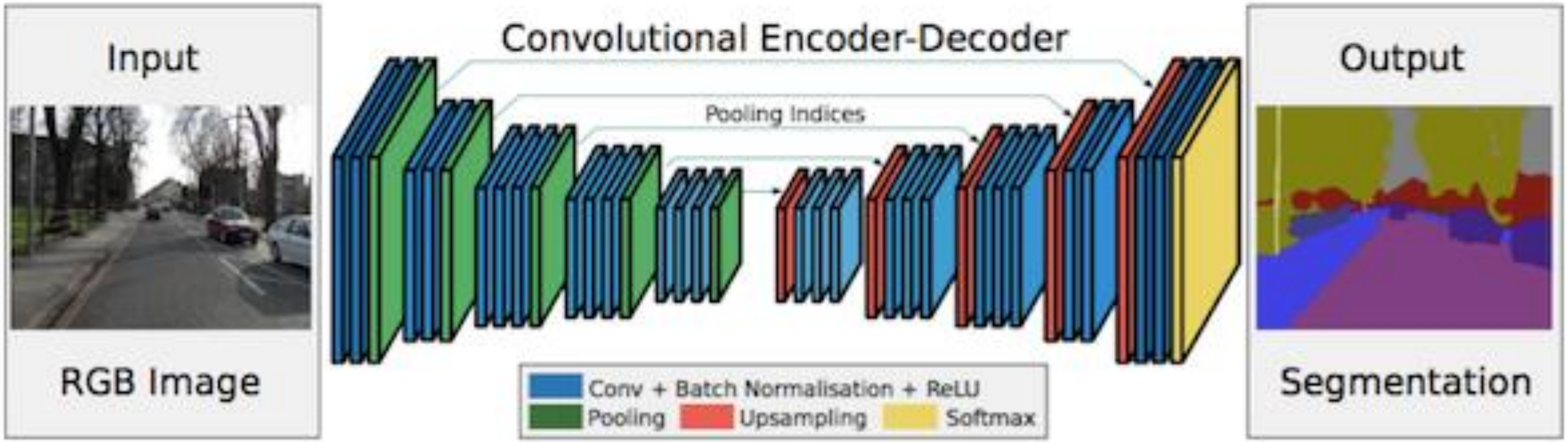




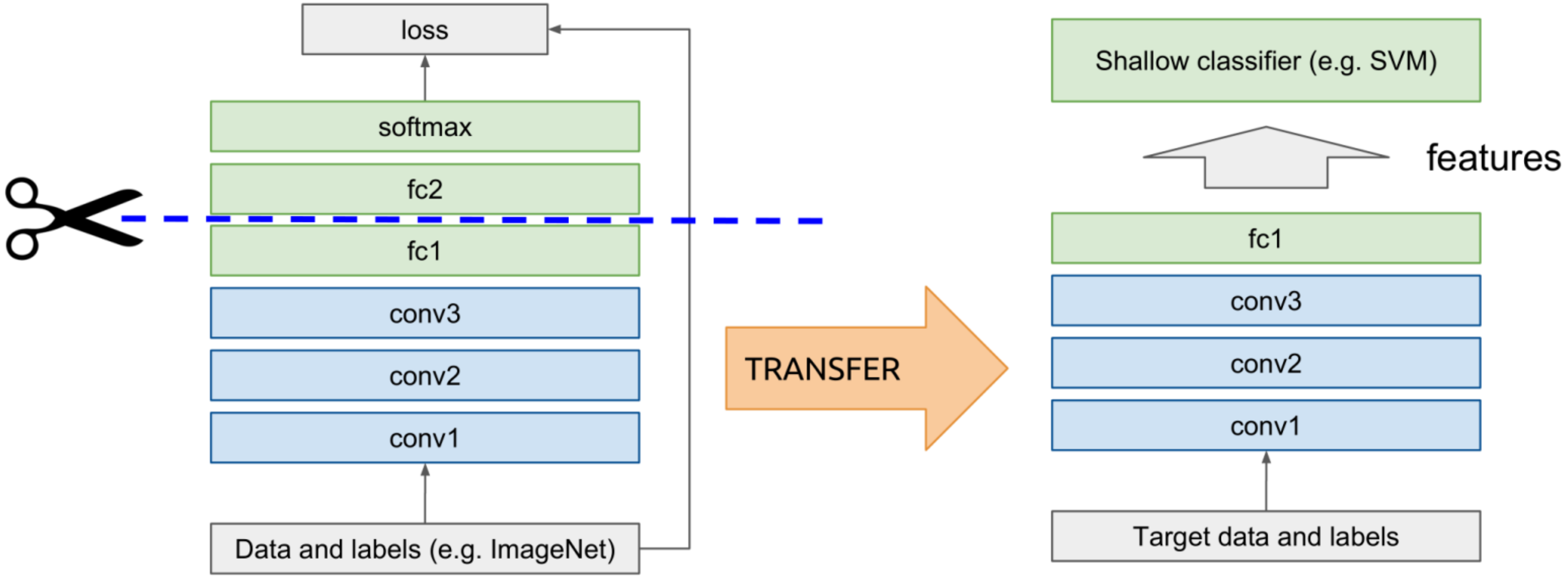
The input to the block X bypasses the convolution and is then combined with its residual F(X) resulting from the convolutions

When backpropagating the gradient flows in full through these bypass connections

Convolutions and Deconvolutions



Fine Tuning and Transfer Learning

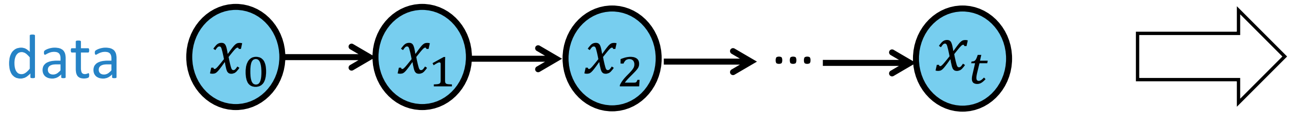


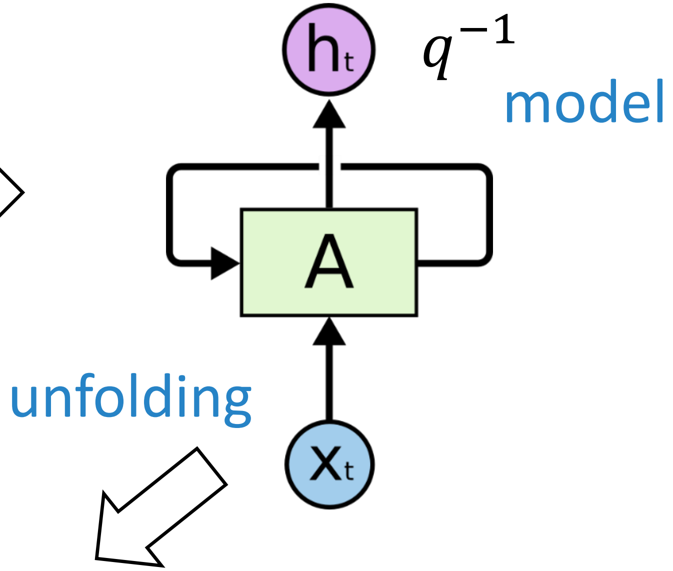
- ✓ Fine tuning a pre-trained model is the simplest case
- ✓ General transfer learning is much more: domain adaptation, multi-task learning, …

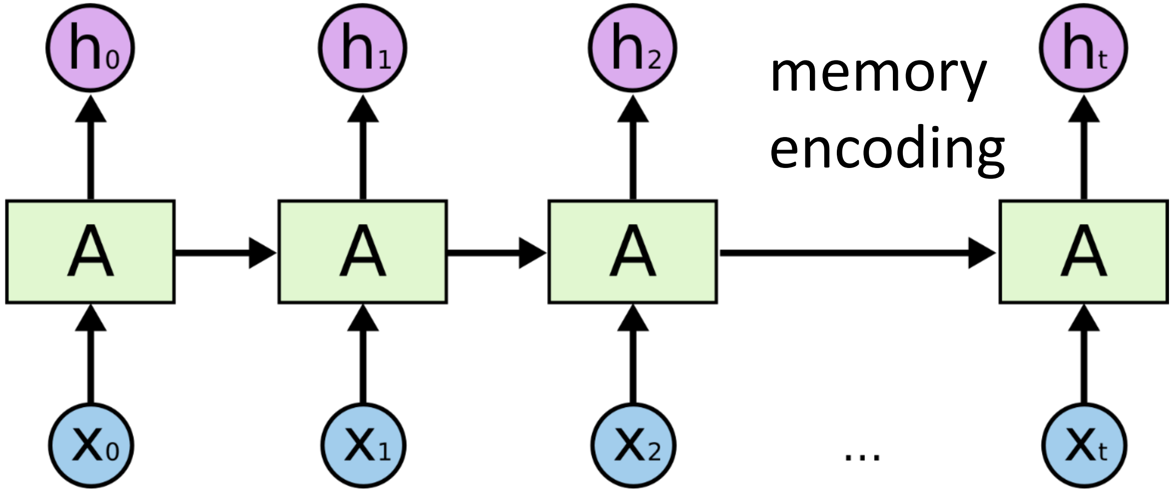
Recurrent Neural Networks (RNNs)

SEQUENTIAL DATA

Unfolding RNN (Forward Pass)



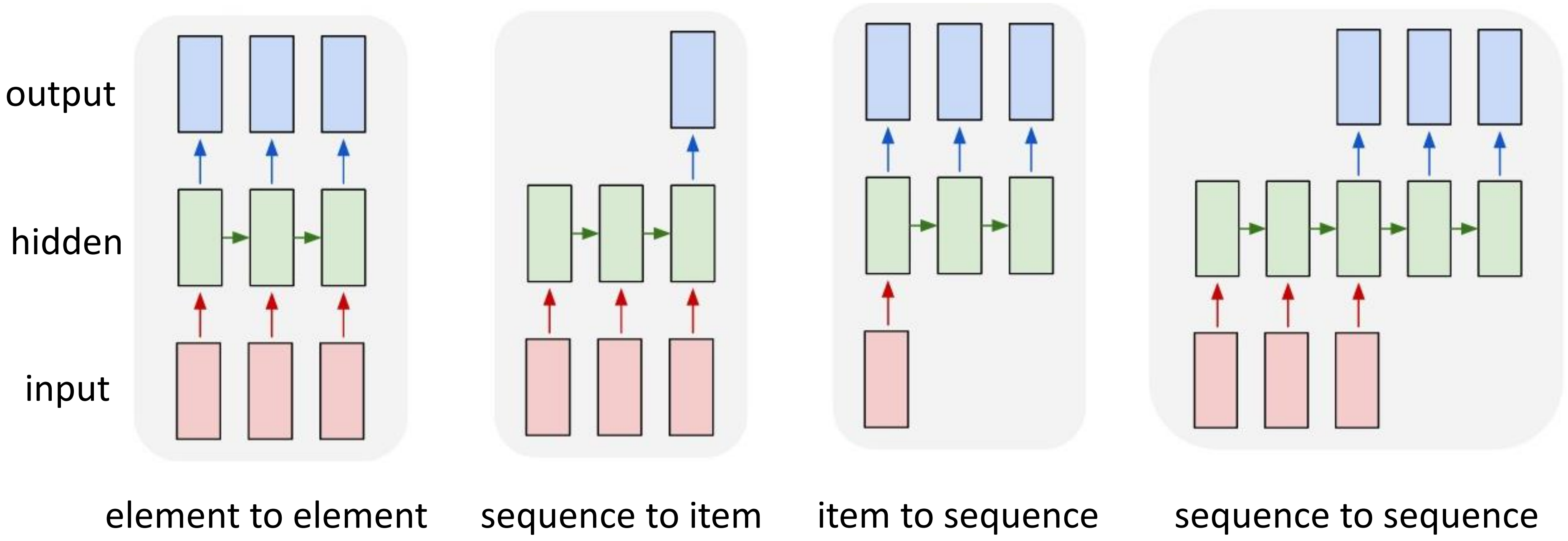




Map an arbitrary length sequence $x_0..x_t$ to fixed-length encoding $\boldsymbol{h}_t$

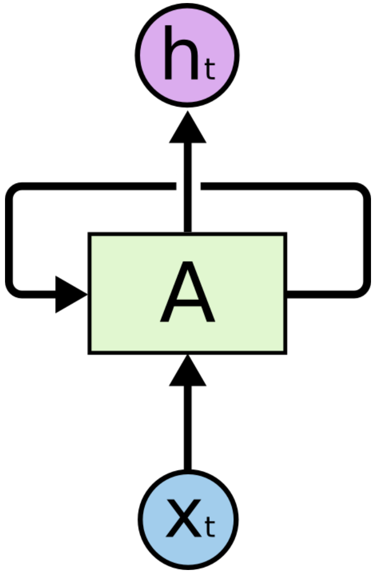
Supervised Recurrent Tasks

Graphics credit @ karpathy.github.io



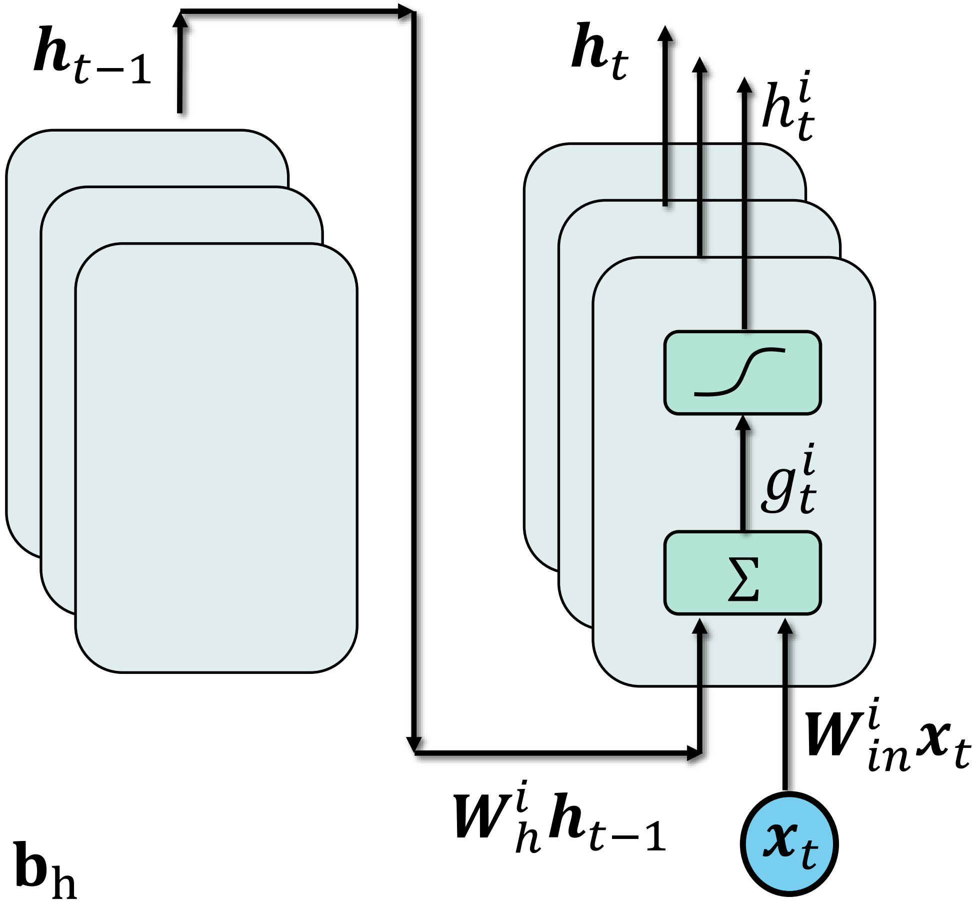
Recurrent Neural Network

$$\boldsymbol{y}_t = f(\boldsymbol{W}_{out}\boldsymbol{h}_t + \boldsymbol{b}_{out})$$

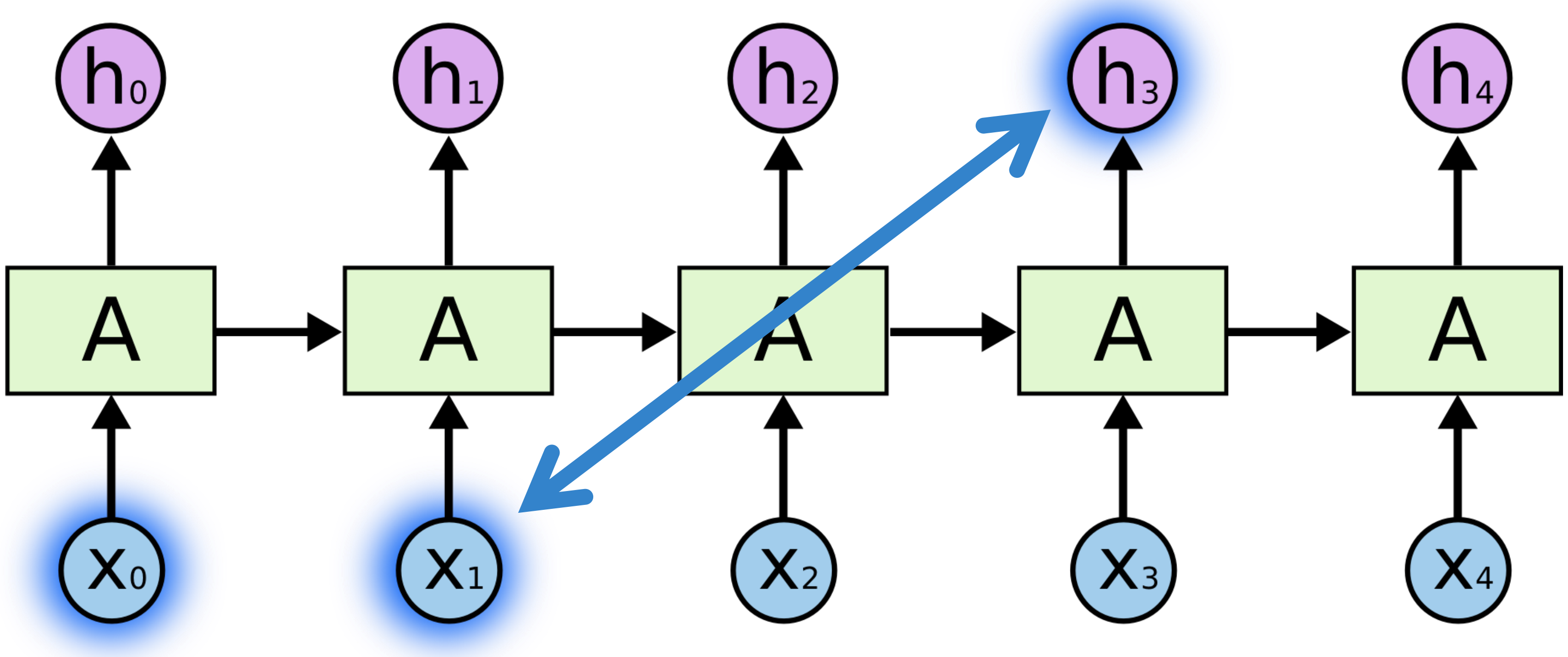


$$\boldsymbol{h}_t = tanh(\boldsymbol{g}_t)$$

$$\boldsymbol{g}_t(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) = \boldsymbol{W}_h\boldsymbol{h}_{t-1} + \boldsymbol{W}_{in}\boldsymbol{x}_t + \mathbf{b}_\mathrm{h}$$



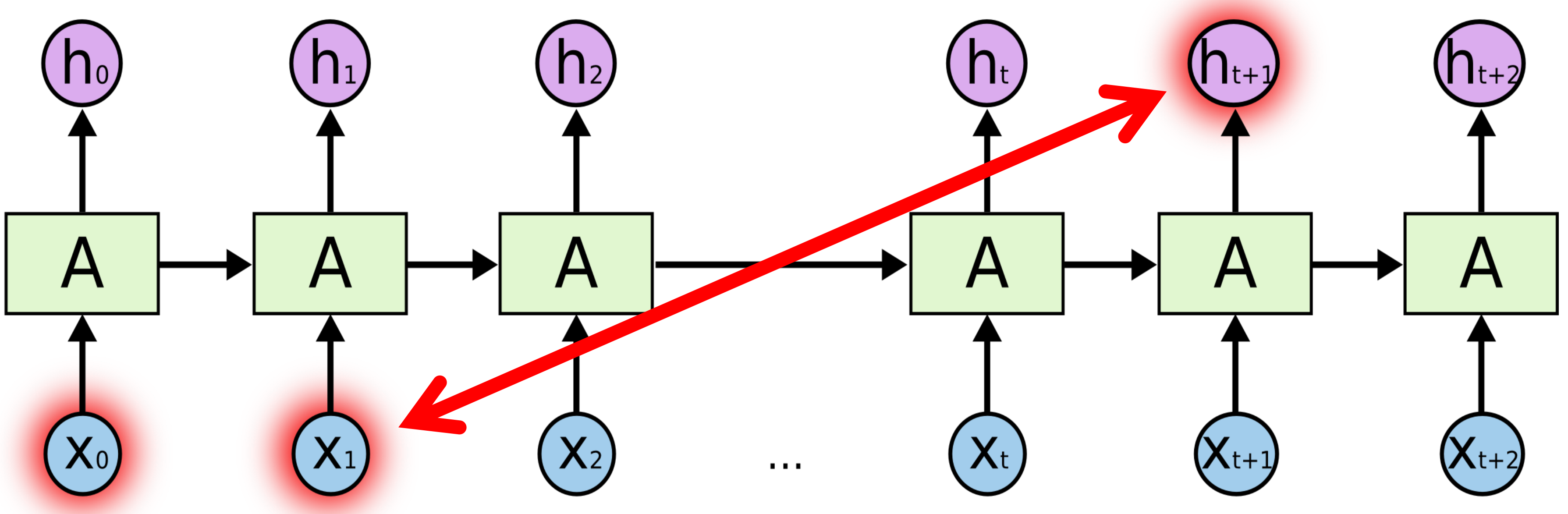
Learning to Encode Input History



Hidden state $\boldsymbol{h}_t$ summarizes information on the history of the input signal up to time t

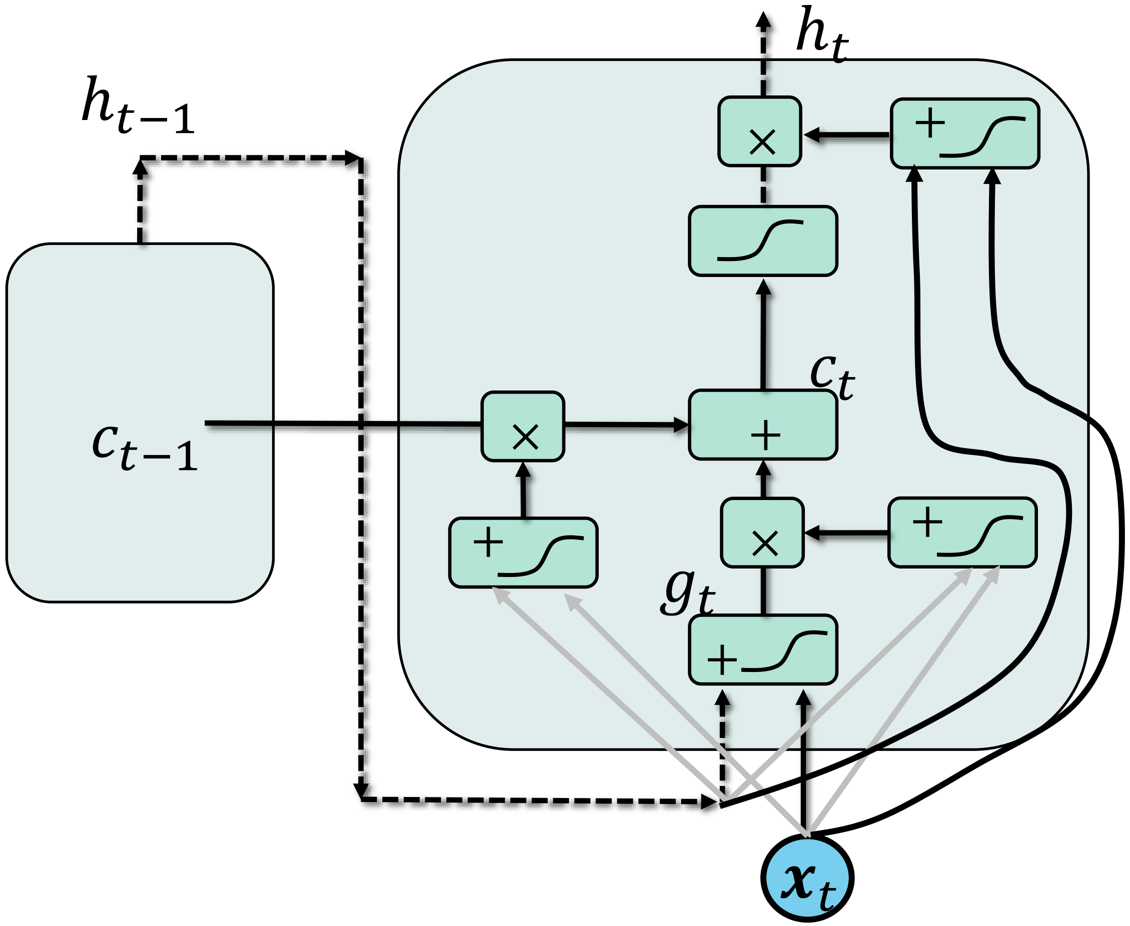
Learning Long-Term Dependencies is Difficult

When the time gap between the observation and the state grows there is little residual information of the input inside of the memory



Exploding/vanishing gradient

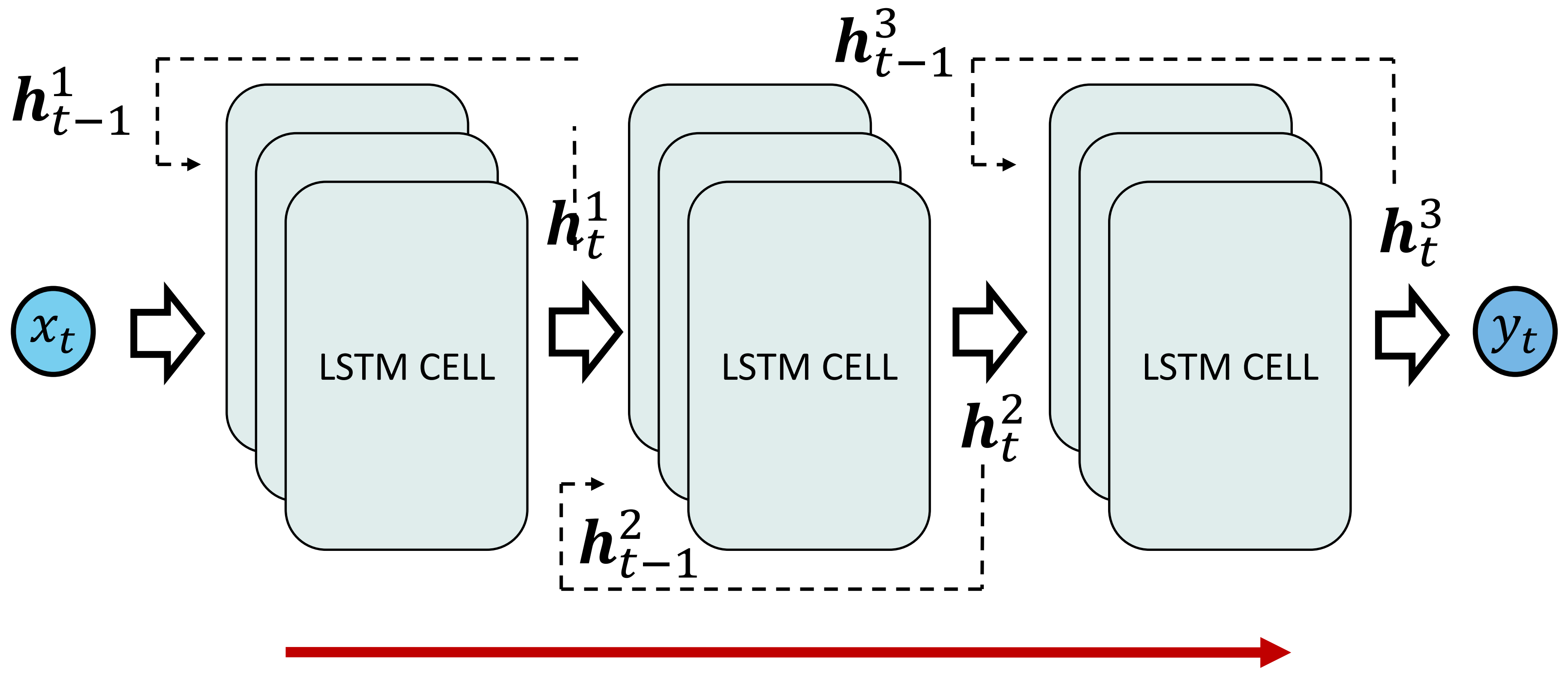
Gated Recurrent Networks



Using gates to control memory access

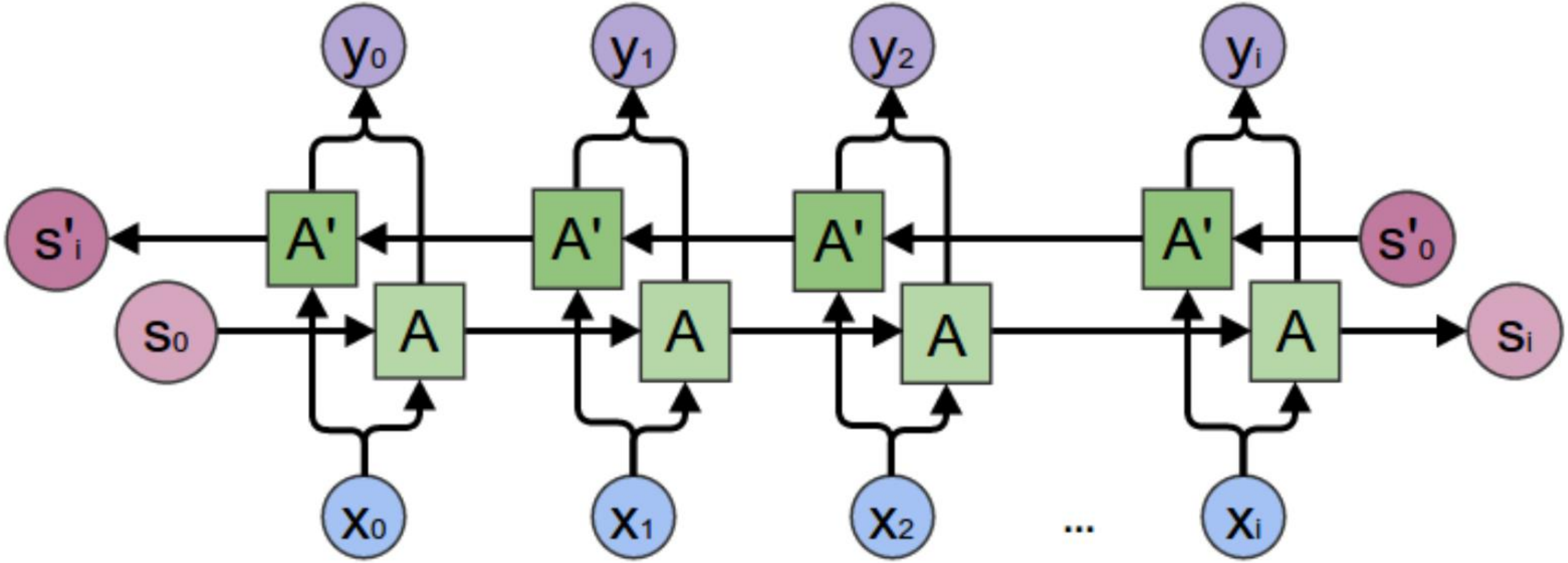
- ✓ Long-Short Term Memory
- ✓ Gated Recurrent Unit

Deep LSTM



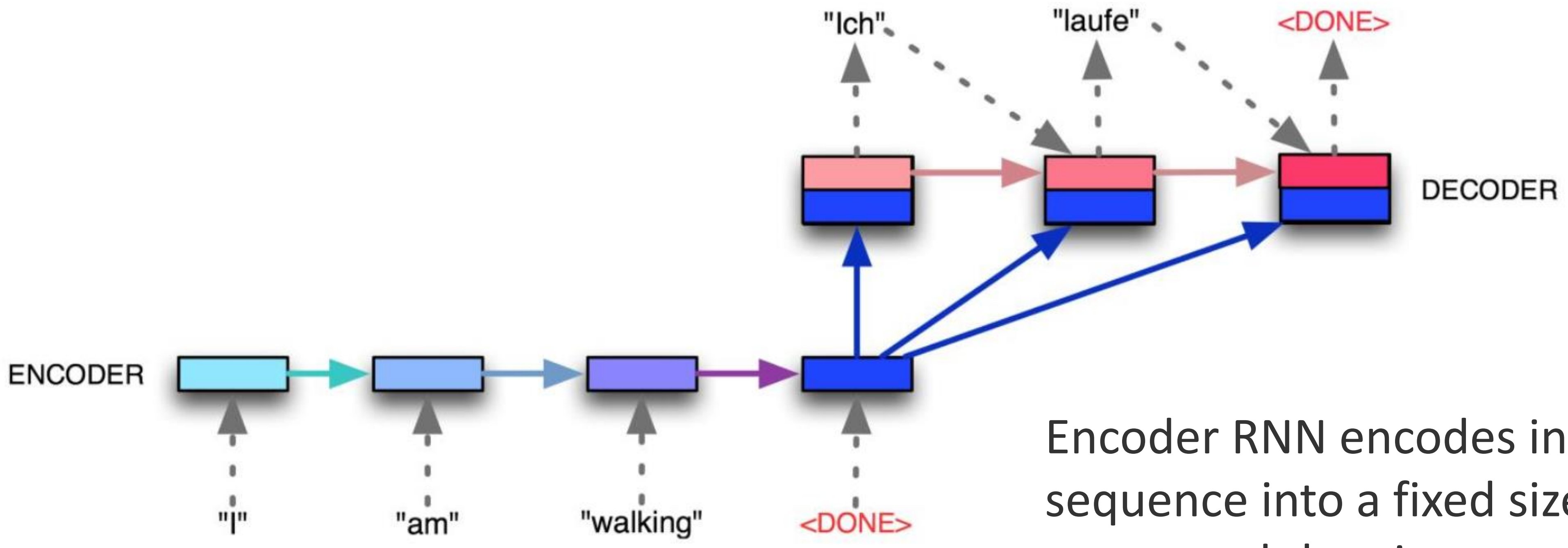
LSTM layers extract information at increasing levels of abstraction (enlarging context)

Bidirectional RNNs



Learning representations from past as well as from future

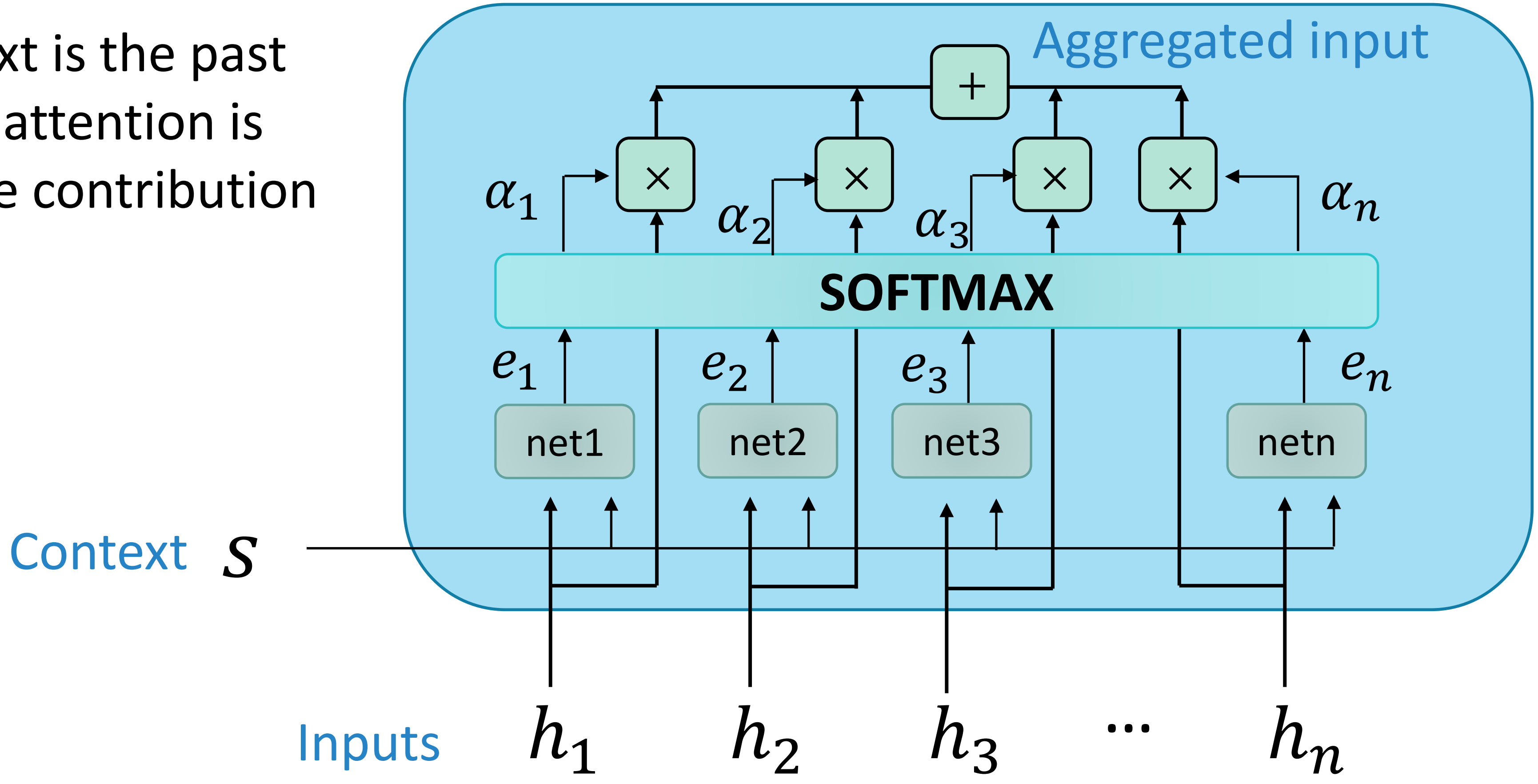
Encoder-Decoder Architectures



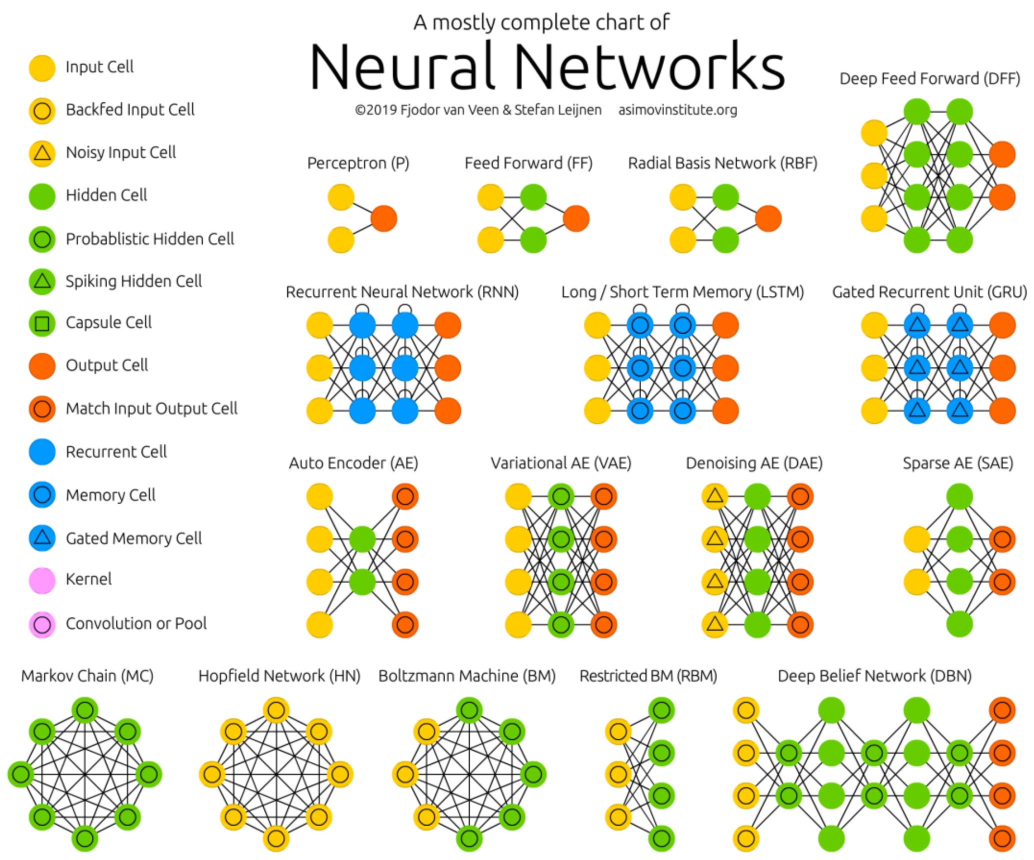
Encoder RNN encodes input sequence into a fixed size vector and then is passed to decoder RNN

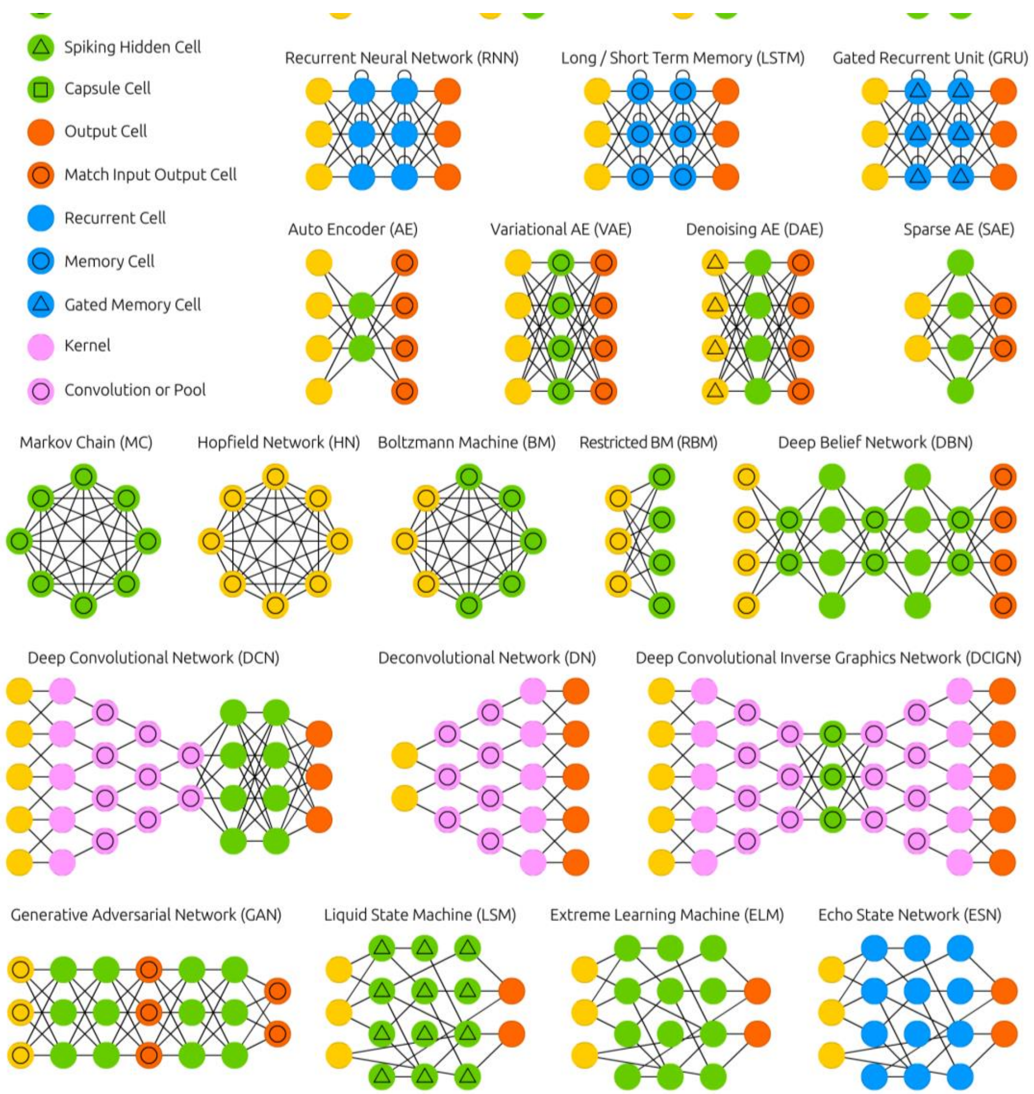
Attention

In seq2seq context is the past output state and attention is used to weigh the contribution from input states



Wrap-Up

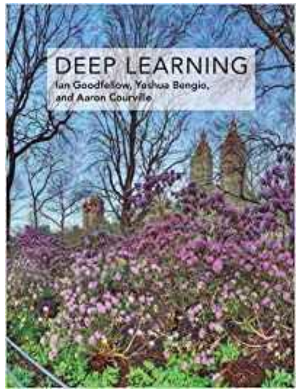




Largely a subset of the existing architectures

ML, DL and RL Frameworks

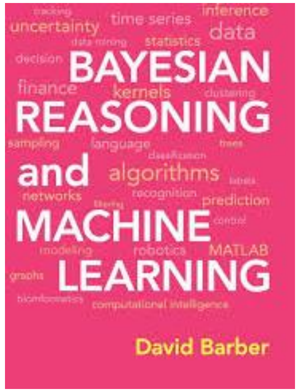
ML and Deep Learning References



Ian Goodfellow and Yoshua Bengio and Aaron Courville , Deep Learning, MIT Press

✓Reference book for deep learning

✓Also freely available online



David Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press

✓Reference book for Bayesian/probabilistic methods

✓Also freely available online

Advanced ML course @ UNIPI: bit.ly/2rzREqb

Next Lecture

Introduction to Reinforcement Learning

- ✓ Fundamentals of RL
  - ✓ Agent and environment
  - ✓ Actions and observations
  - ✓ History and state
- ✓ Components of a RL agent
  - ✓ Policy, value and model
- ✓ A preliminary taxonomy of models
- ✓ Notable problems within ML