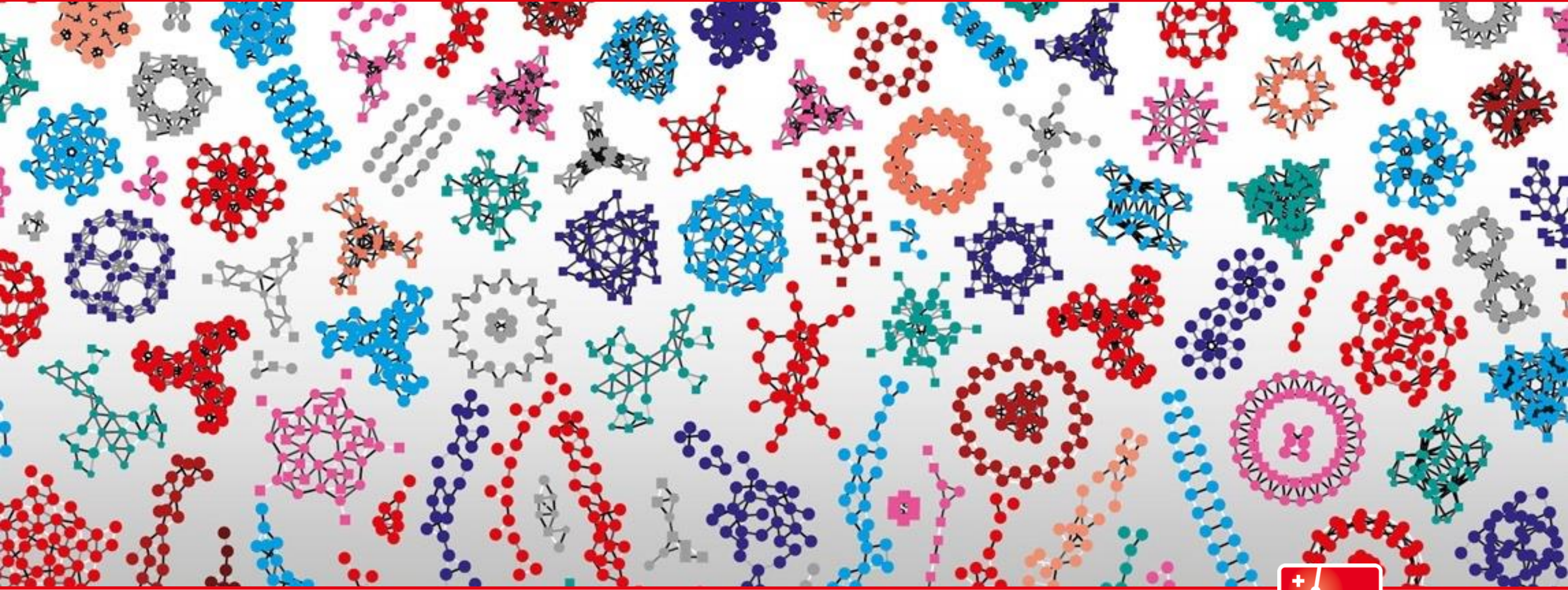


Deep learning techniques in Life sciences

Van Du Tran, Markus Müller, Wandrille Duchemin

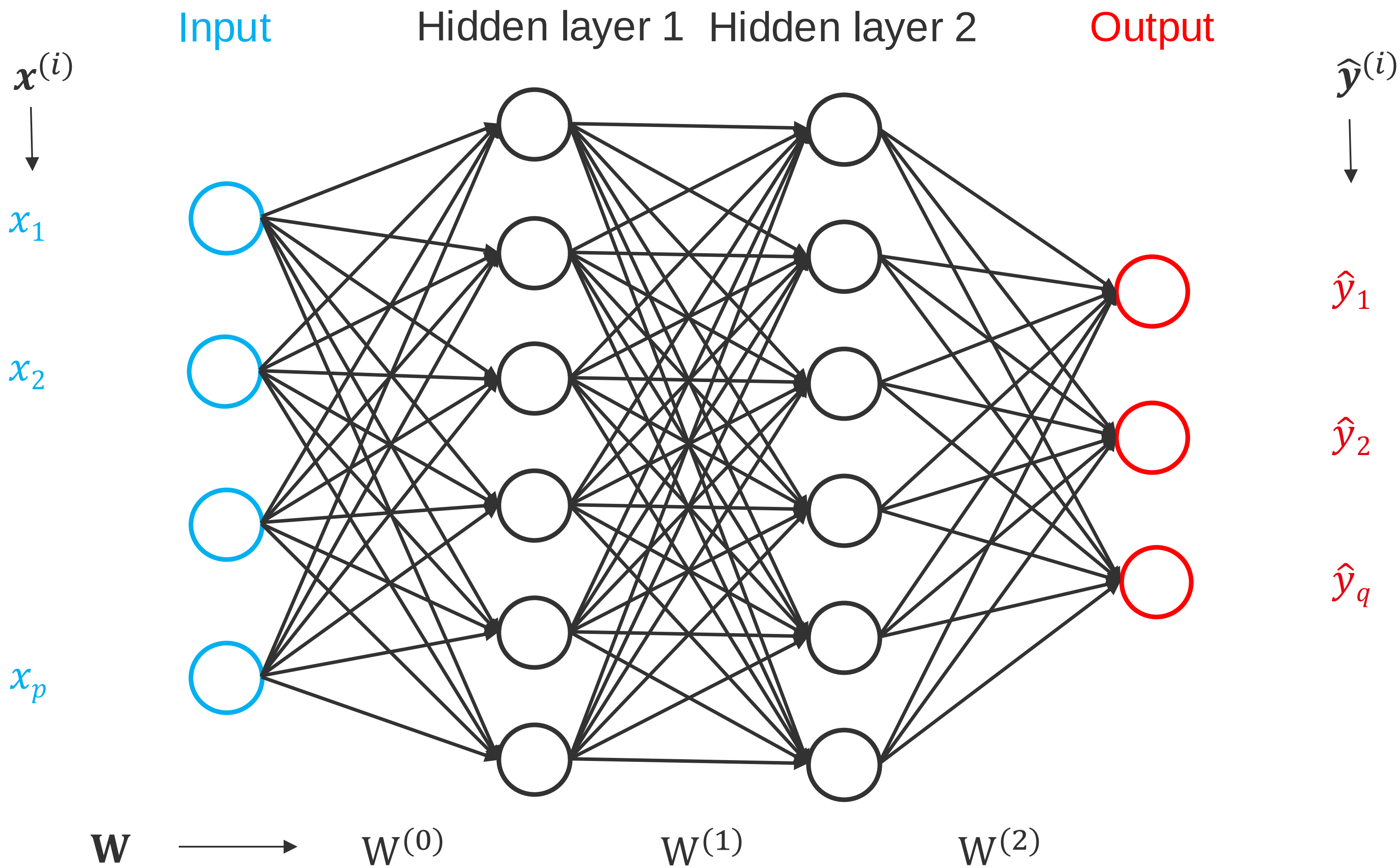
SIB Swiss Institute of Bioinformatics

November 2024

Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Deep reinforcement learning
- Generative adversarial networks

Neural Network revisited

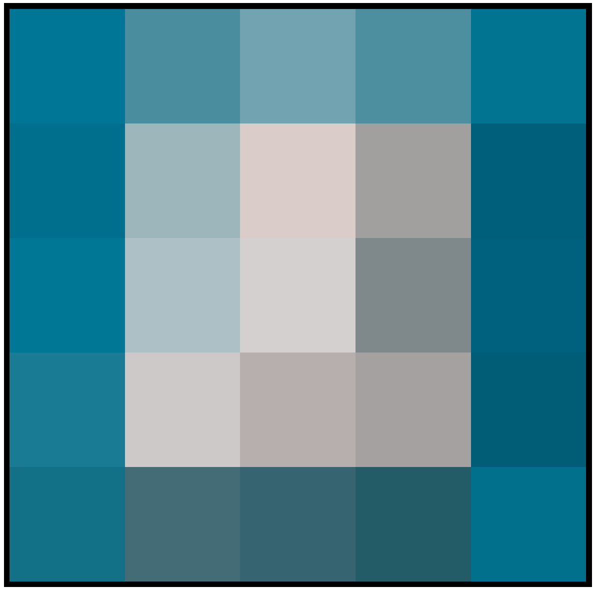


Activation functions on linear regressions

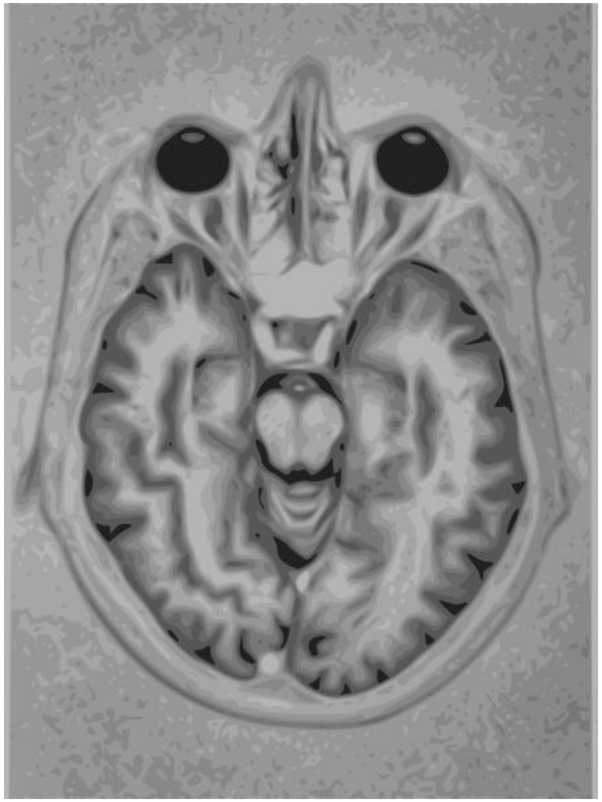
Loss optimization $\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(f\left(\boldsymbol{x}^{(i)}, \mathbf{W}\right), \boldsymbol{y}^{(i)}\right)$

Dimensionality issue

5x5 pixels



1524 x 2048 pixels



Sparsity

Closeness

Full connectivity => curse of dimensionality!

https://commons.wikimedia.org/w/index.php?curid=1078380

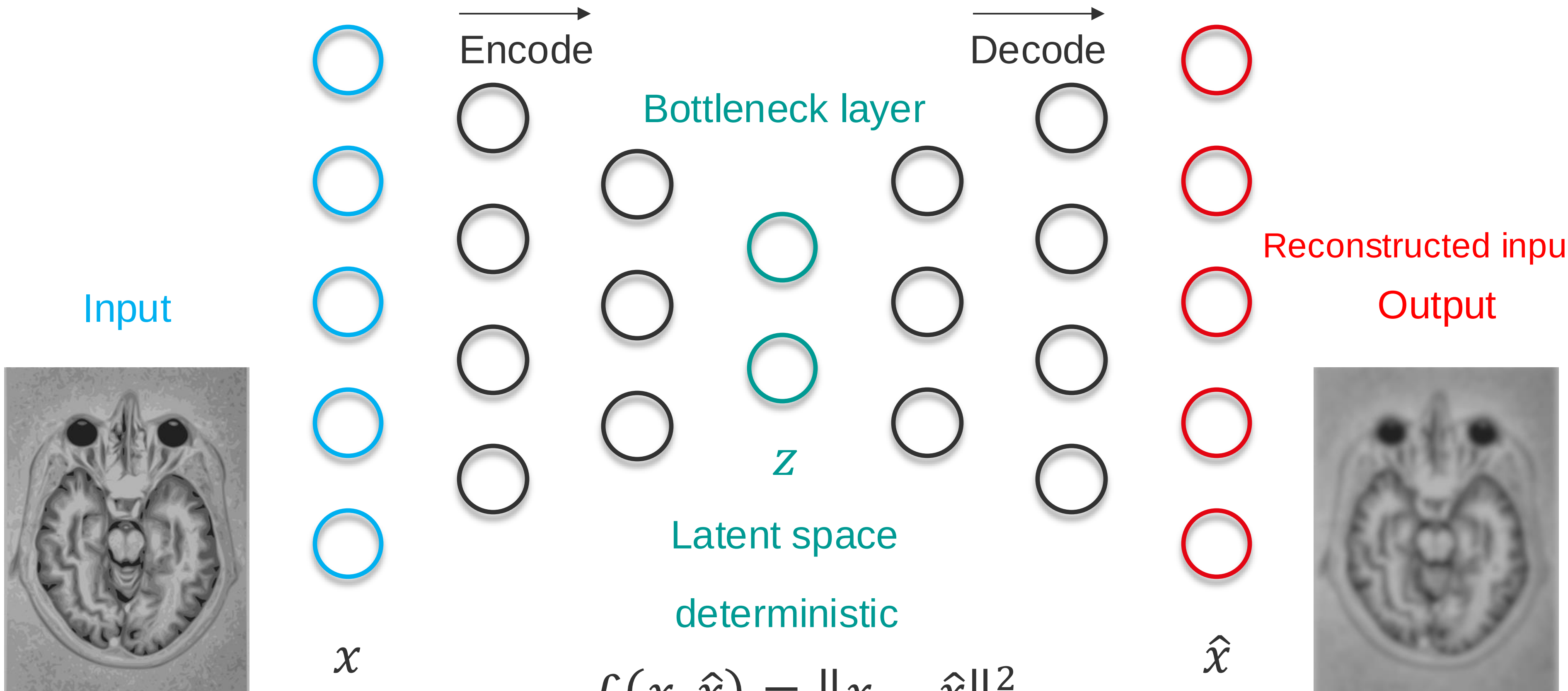
https://commons.wikimedia.org/wiki/File:Brain_Mri_nevit.svg

Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Deep reinforcement learning
- Generative adversarial networks

Autoencoder

- Learning a lower-dimensional feature representation (compression) from unlabeled training data and learning a reconstruction back



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

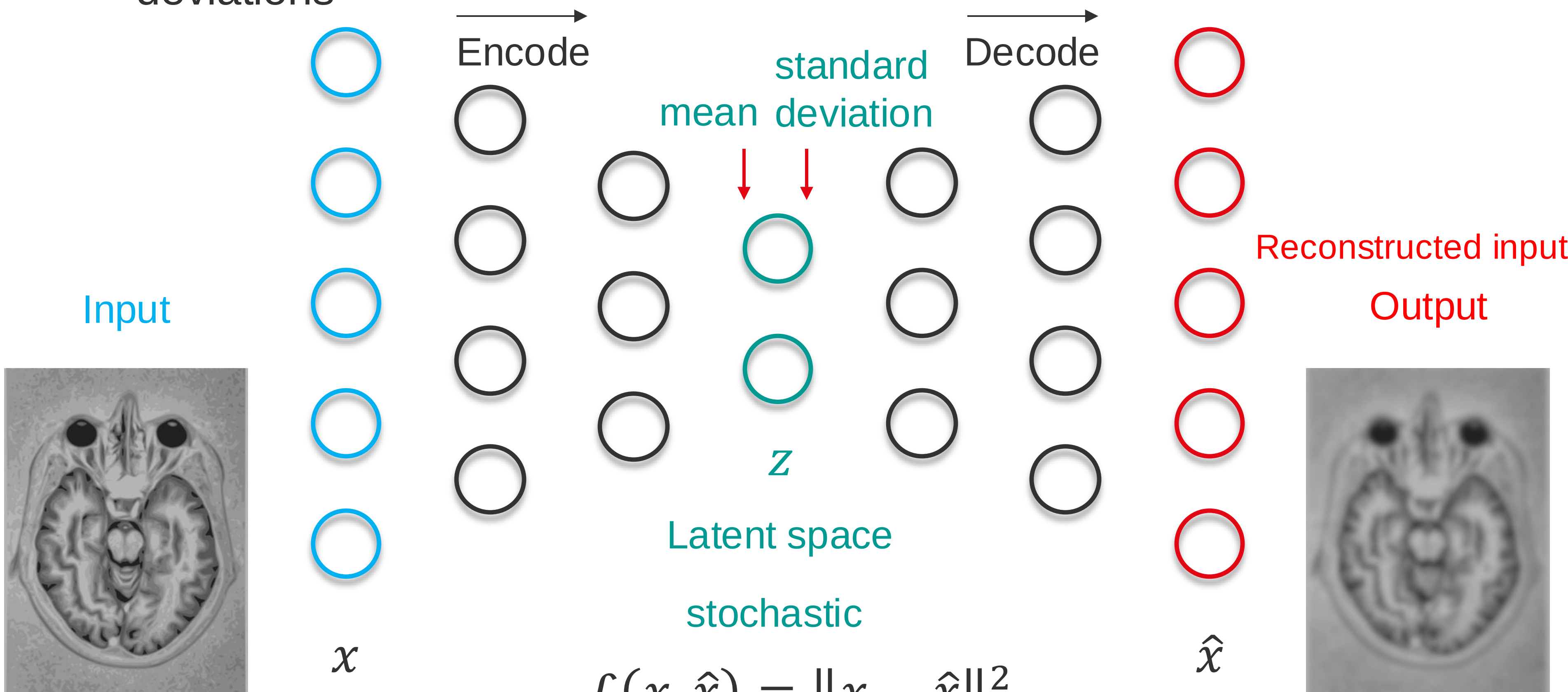
```
tensorflow.keras.Model(name='autoencoder')
```

Autoencoder

- Generalization of PCA: non-linear
- Challenges: models that learn a meaningful and generalizable latent space representation

Variational autoencoder

- Probabilistic twist on autoencoder: map the input into a distribution instead of a fixed vector => two vectors of means and standard deviations



$$\mathcal{L}(x,\hat{x}) = \|x-\hat{x}\|^2$$

```
tensorflow.keras.Model(name='vae')
```

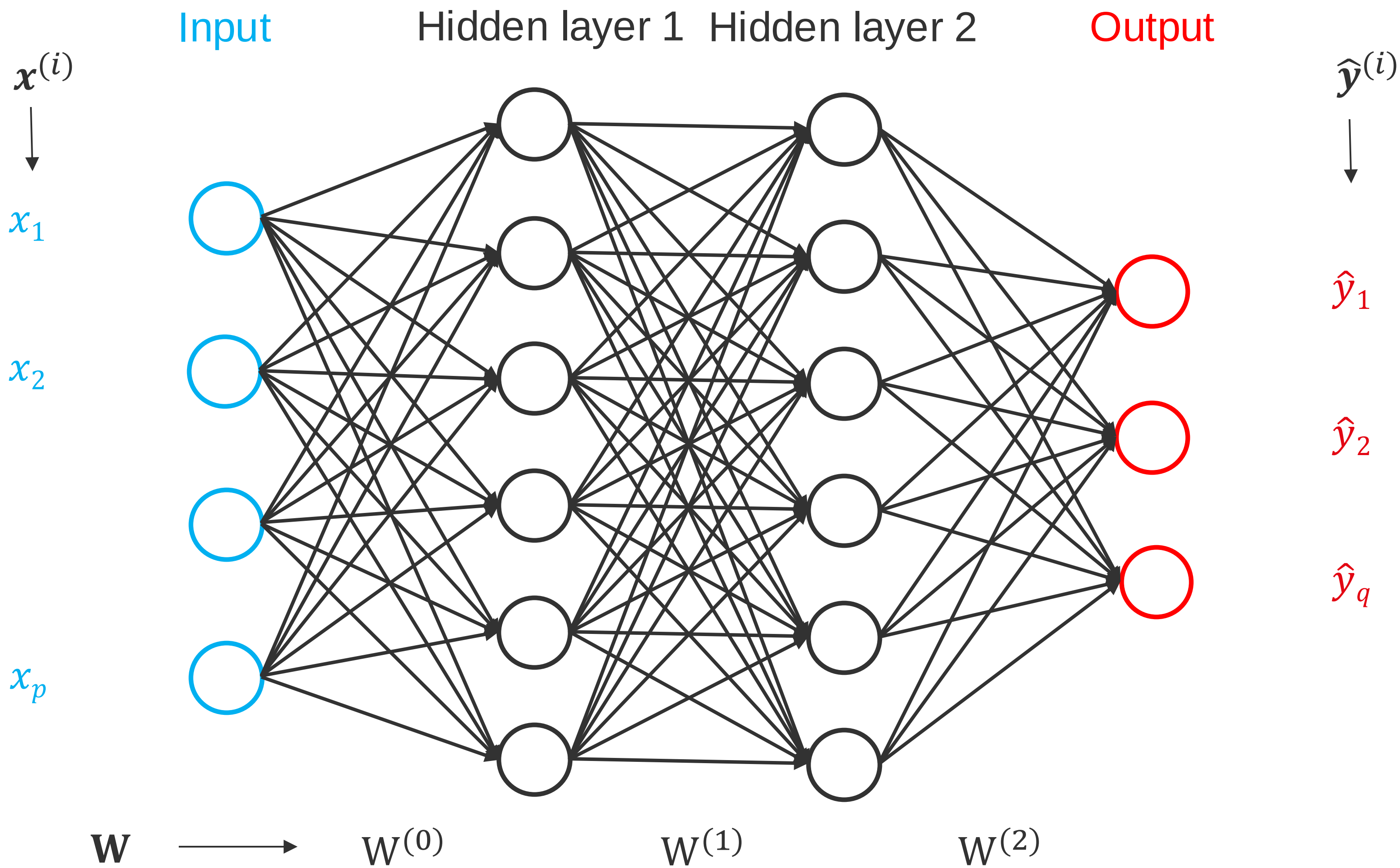
Autoencoders: When?

- Image compression, denoising and generation, recommendation system, anomaly detection, feature extraction
- Life sciences
  - dimensionality reduction (clustering) in sequencing data
  - multi-omics and biomedical data integration

Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Deep reinforcement learning
- Generative adversarial networks

Neural Network revisited



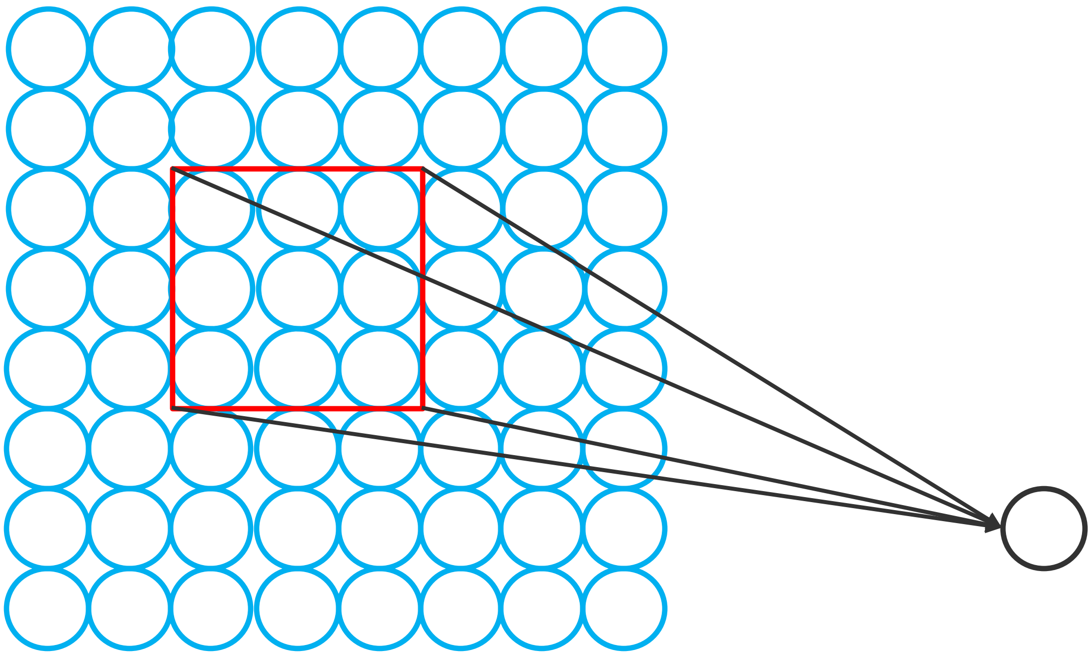
Activation functions on linear regressions

Loss optimization $\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{x}^{(i)}, \mathbf{W}), \boldsymbol{y}^{(i)})$

Convolutional Neural Network (CNN, ConvNet)

- Inspired by the organization of the visual cortex
- Image processing: take in image input => assign importance to features/objects => differentiate
- Each neuron receives connections only from a subset of neurons in the previous layer
- Spatial and temporal dependency

CNN: How? Convolutional layer



CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3		

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0		

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0	5	

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0	5	0

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0	5	0
3		

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0	5	0
3	0	

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0	5	0
3	0	3

Convolution operation: sum of element-wise products

CNN: How? Convolutional layer

Input image

1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1

Filter

1	0	1
0	1	0
1	0	1

Feature map

3	0	3
0	5	0
3	0	3

1	0	0
0	1	0
0	0	1

0	0	1
0	1	0
1	0	0

Convolutional layers:

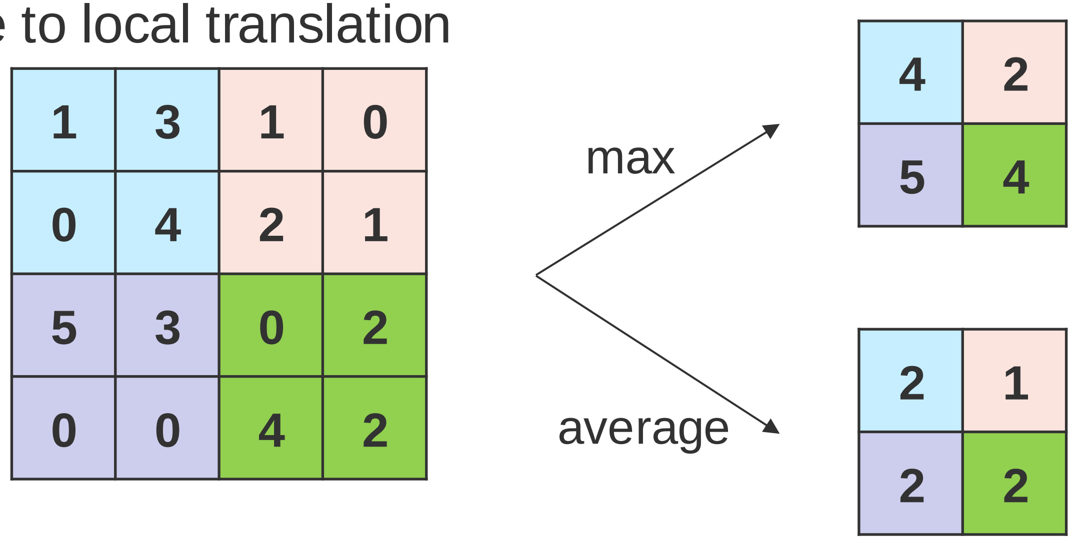
- Extract low to high-level features
- Reduce spatial size

Limitation: sensitive to position of features

`tensorflow.keras.layers.Conv2D`

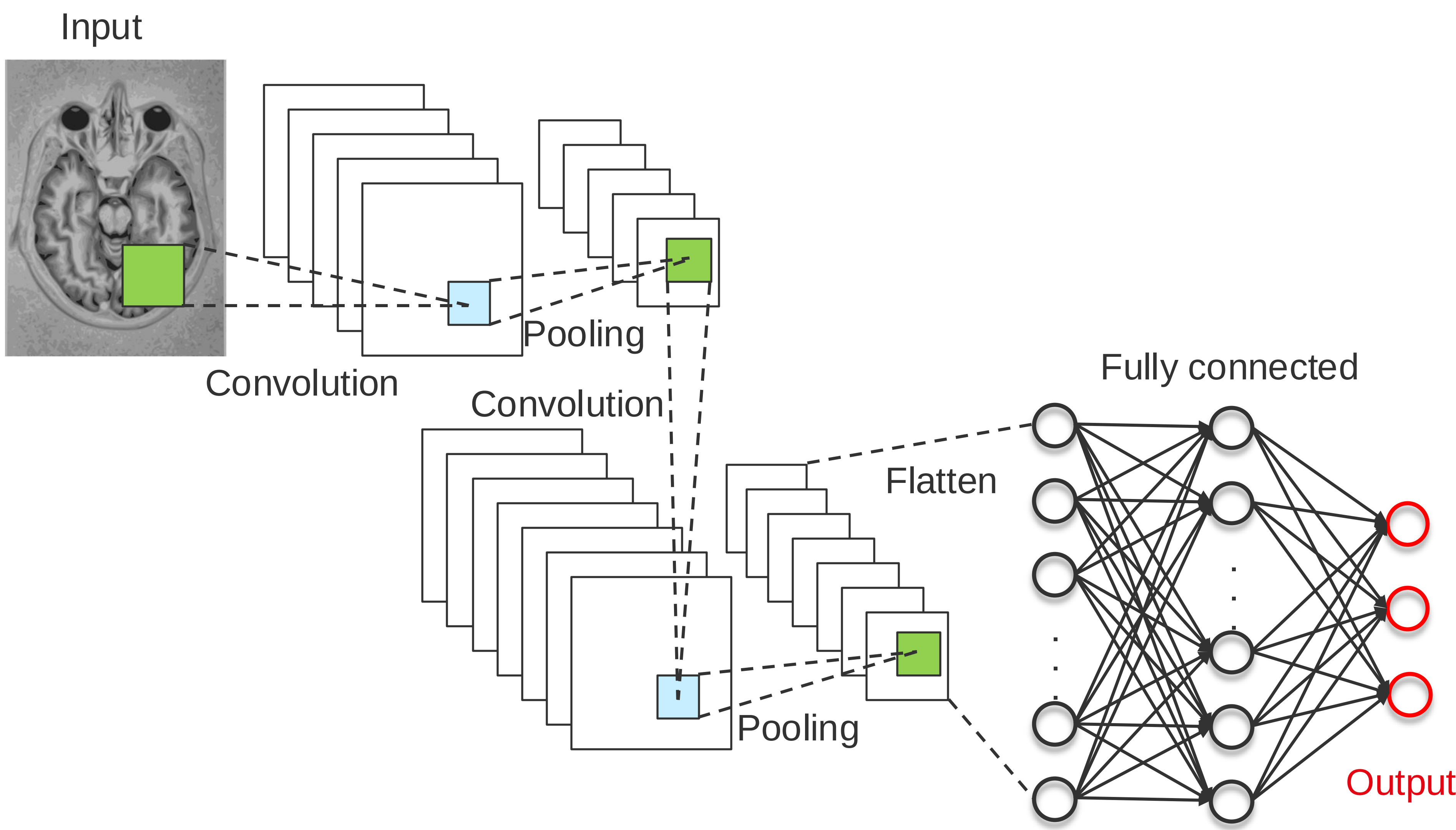
CNN: How? Pooling layer

- Down sampling: lower resolution + important features
- After Convolutional layer + Nonlinearity (ReLU) for each feature map
- Usually 2x2 pixels with stride of 2 pixels => reduce to ¼ feature map
- Max or average pooling
- Invariance to local translation



`tensorflow.keras.layers.MaxPool2D`

CNN: How? Fully connected layer



CNN: When?

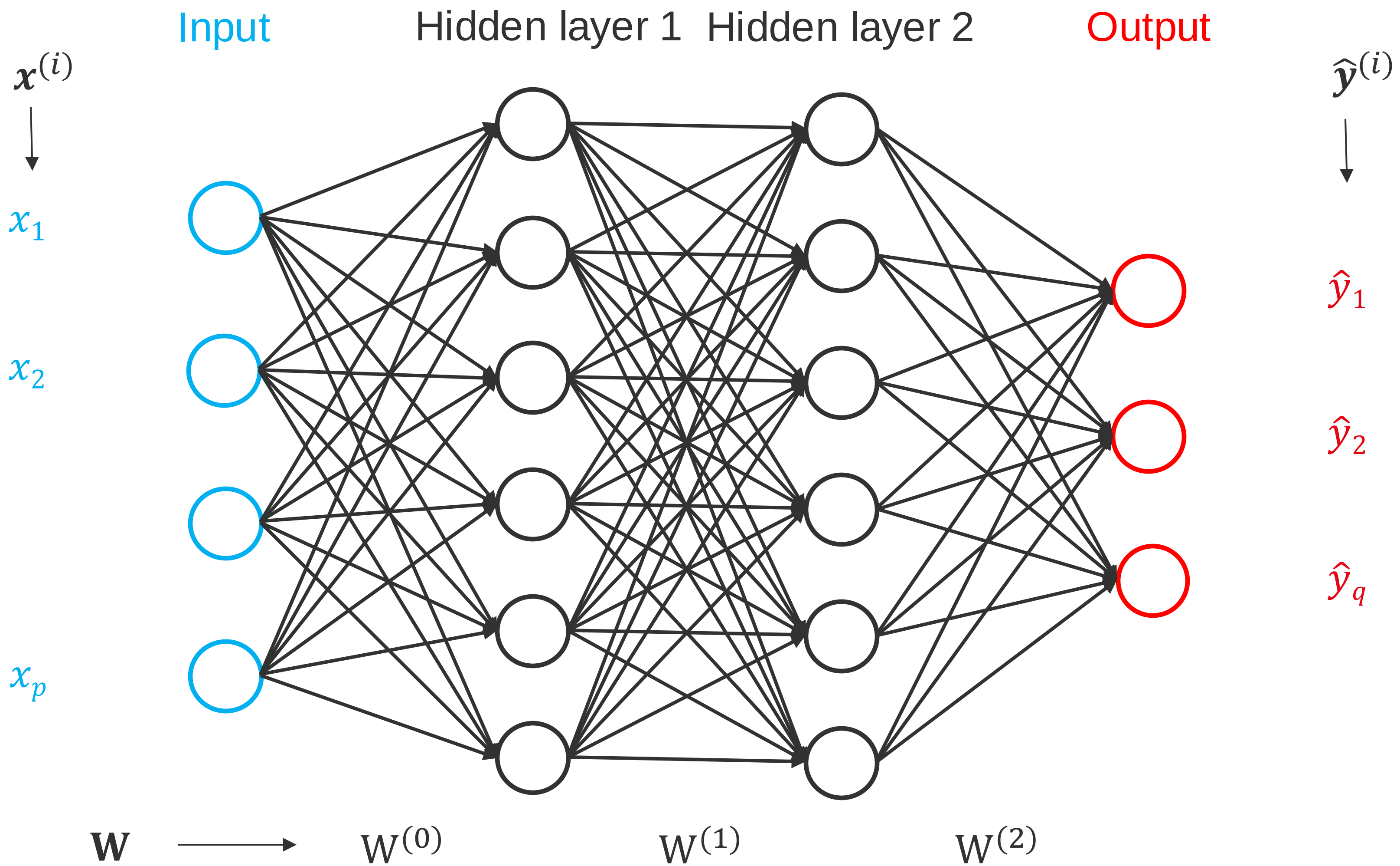
- Image recognition, video analysis, natural language processing

- Life sciences:
  - sequence analysis (DNA, RNA sequence data),
  - structure prediction (imaging data),
  - biomedical image processing and diagnosis (imaging data),
  - biomolecular function prediction (microarray, sequencing data, structure properties), protein variant impact prediction
  - biomolecule interaction prediction and system biology (microarray, interaction data)

Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Deep reinforcement learning
- Generative adversarial networks

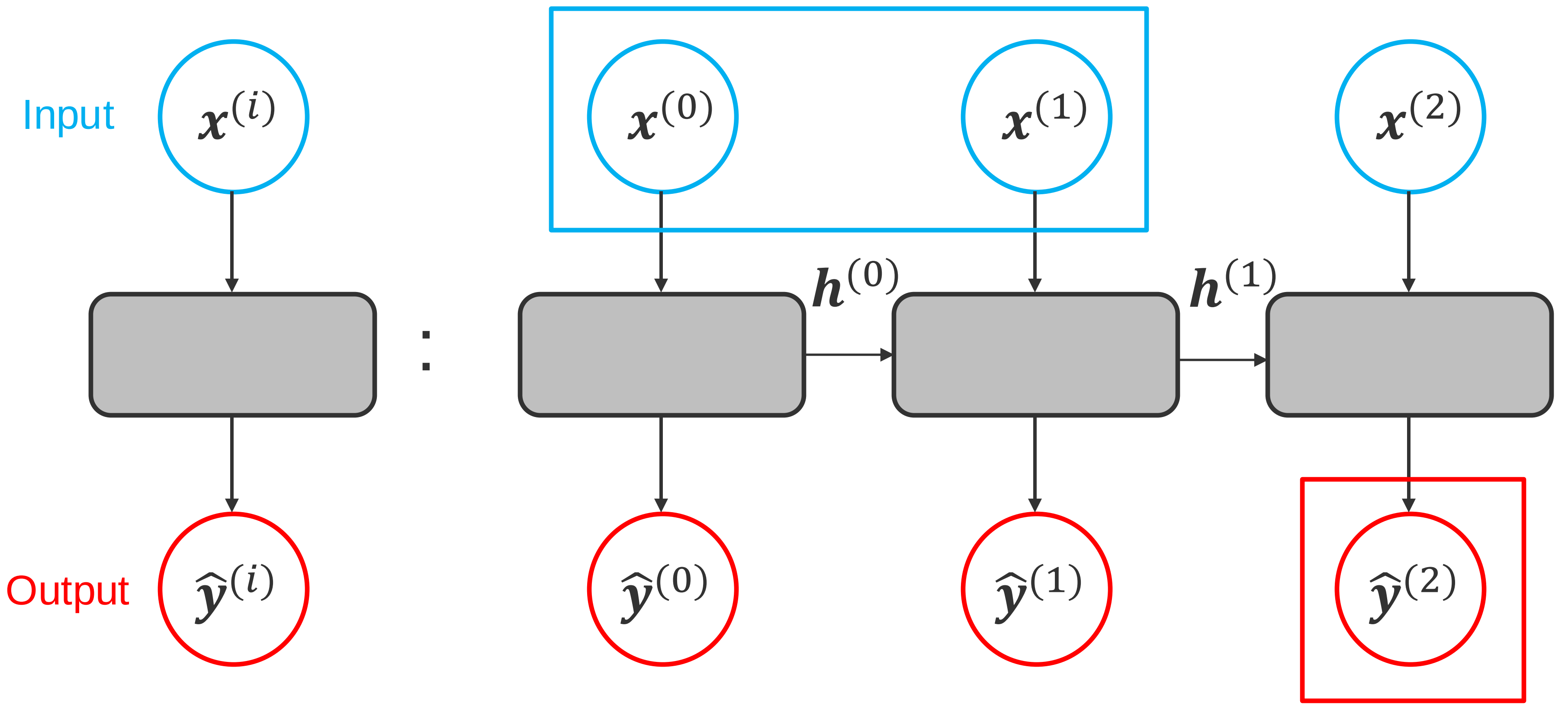
Neural Network revisited



Activation functions on linear regressions

Loss optimization $\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{x}^{(i)}, \mathbf{W}), \boldsymbol{y}^{(i)})$

Neural Network with recurrence



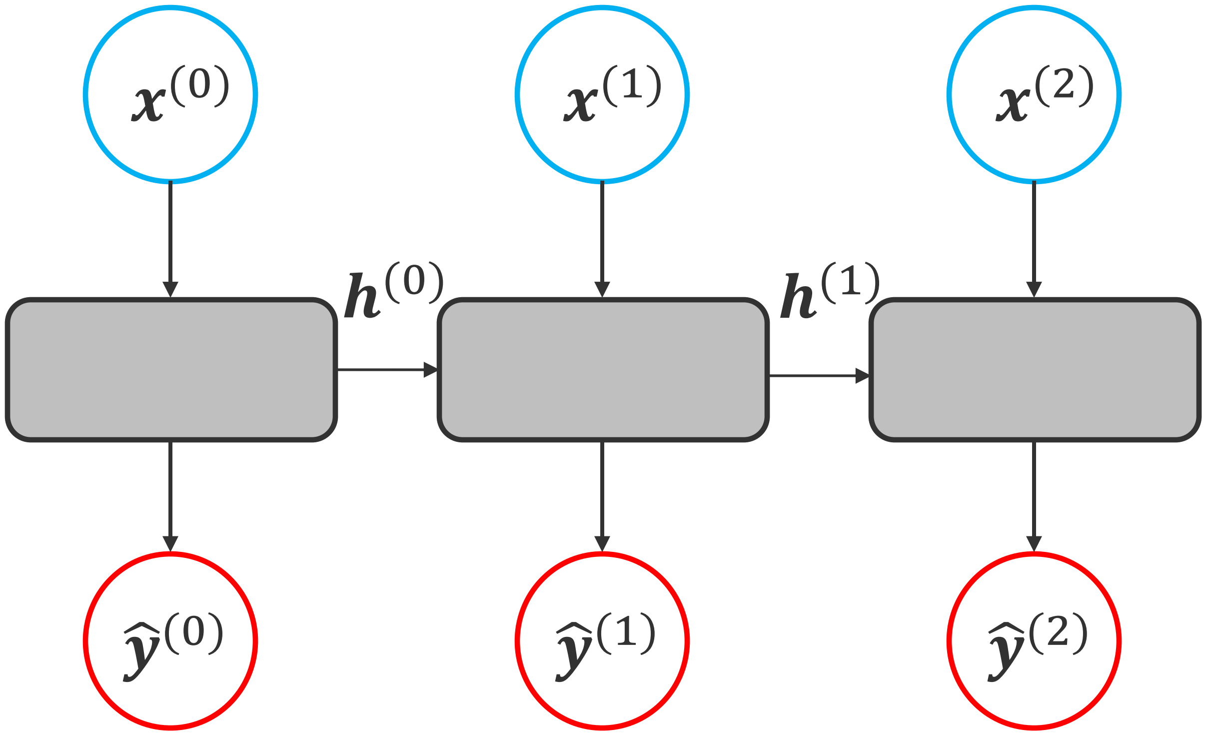
$\hat{\boldsymbol{y}}^{(i)} = f(\boldsymbol{x}^{(i)})$

$\hat{\boldsymbol{y}}^{(i)} = f(\boldsymbol{x}^{(i)}, h_{i-1})$

past memory

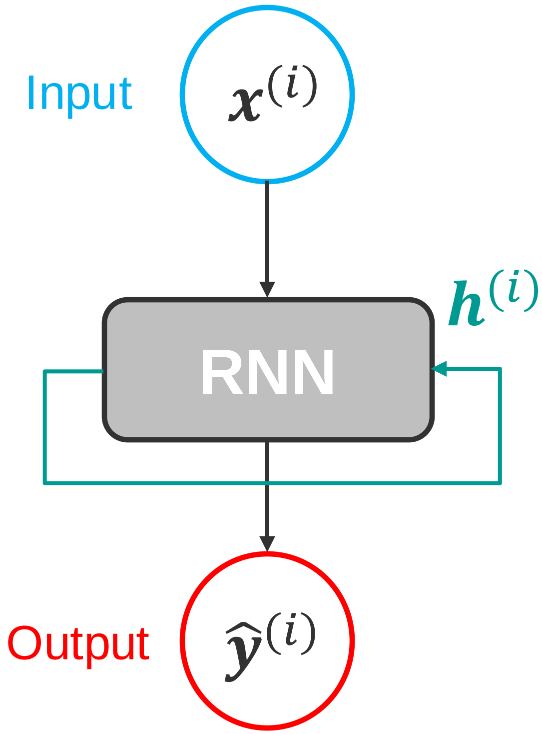
Recurrent Neural Network (RNN)

- Sequential or time series data: data points depend upon previous data points
- Memory: store historical information to forecast future values

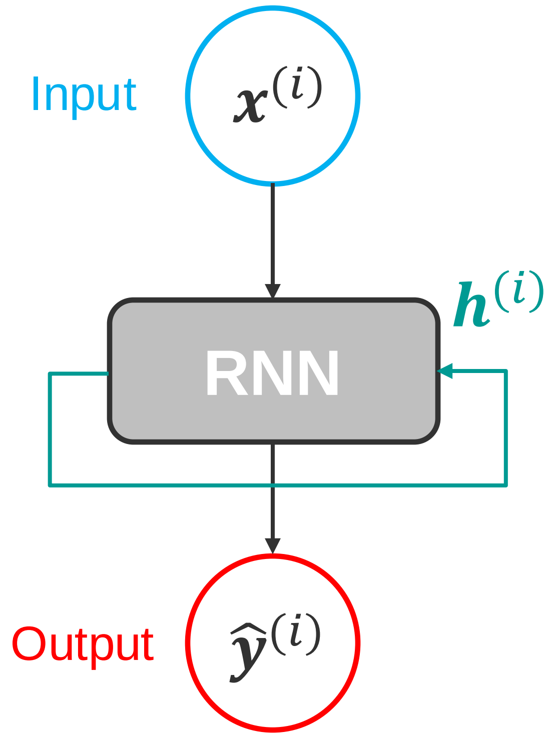


Recurrent Neural Network (RNN)

- Sequential or time series data: data points depend upon previous data points
- Memory: store historic information to forecast future values



RNN: How?



State $\boldsymbol{h}^{(i)}$ is updated at each time step with a recurrence relation:

$$\boldsymbol{h}^{(i)} = \underbrace{f_{\mathbf{W}}}(\boldsymbol{x}^{(i)}, \boldsymbol{h}^{(i-1)})$$

Same function (sigmoid, tanh, ReLU) and parameters at every time step!

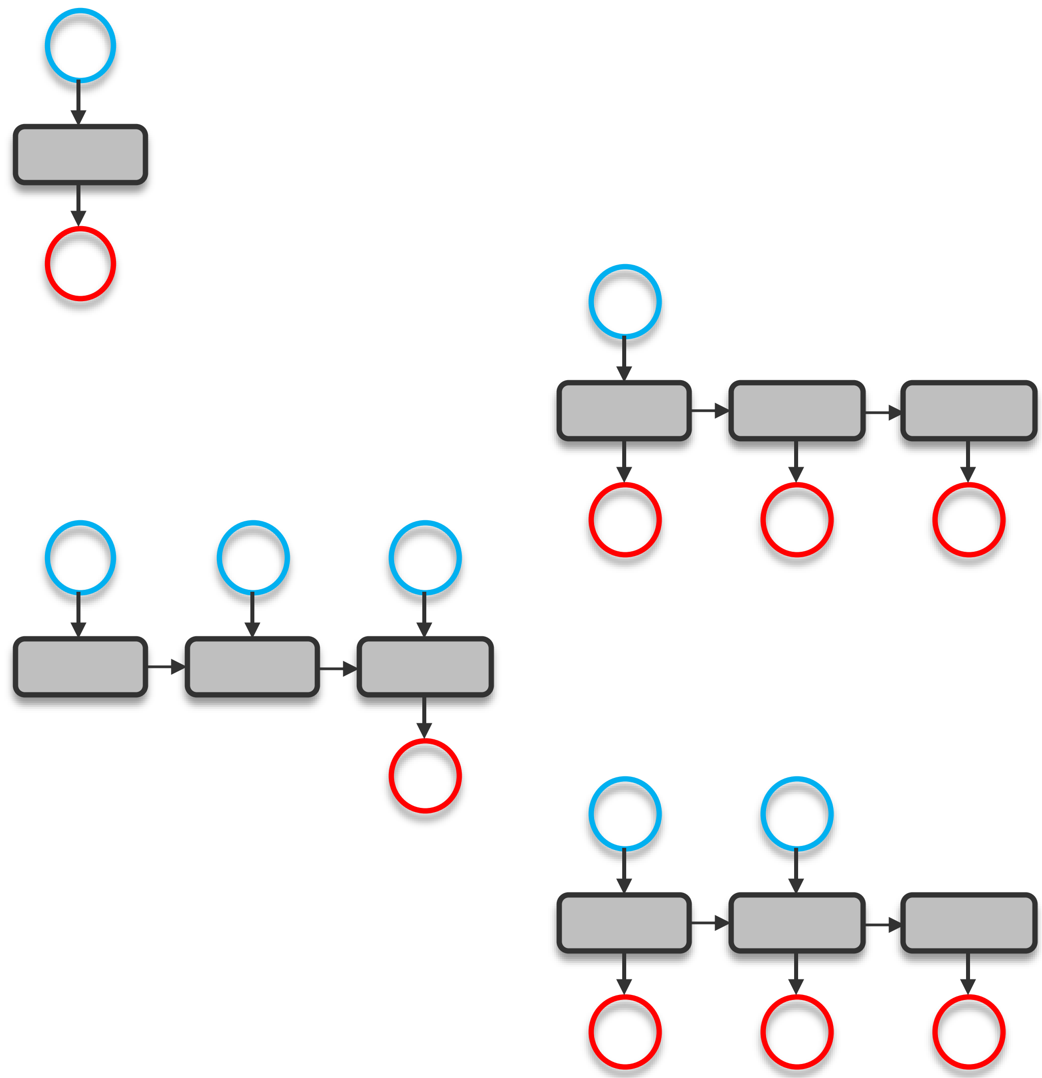
$$\boldsymbol{h}^{(i)} = f(\mathrm{W}_x^{\mathrm{T}} \boldsymbol{x}^{(i)} + \mathrm{W}_h^{\mathrm{T}} \boldsymbol{h}^{(i-1)} + b_h)$$

$$\hat{\boldsymbol{y}}^{(i)} = f(\mathrm{W}_y^{\mathrm{T}} \boldsymbol{h}^{(i)} + by)$$

`tensorflow.keras.layers.SimpleRNN`

Types of RNNs

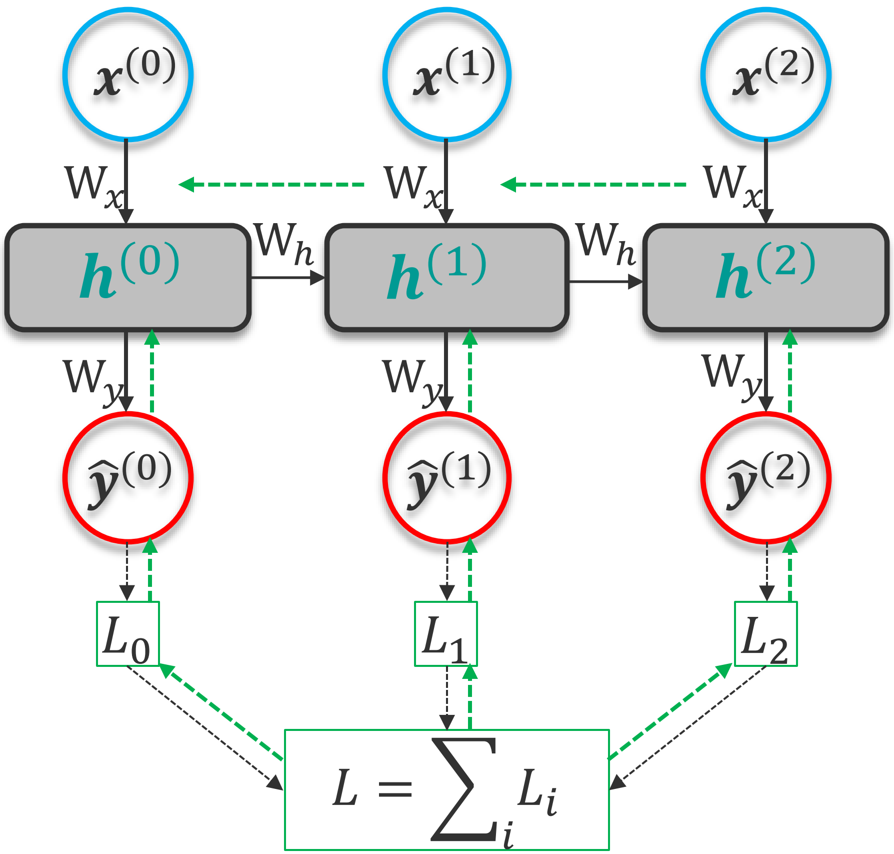
- One to One
- One to Many
- Many to One
- Many to Many



Back propagation through time

- Calculate gradient of the loss function w.r.t. each parameter backward through the network

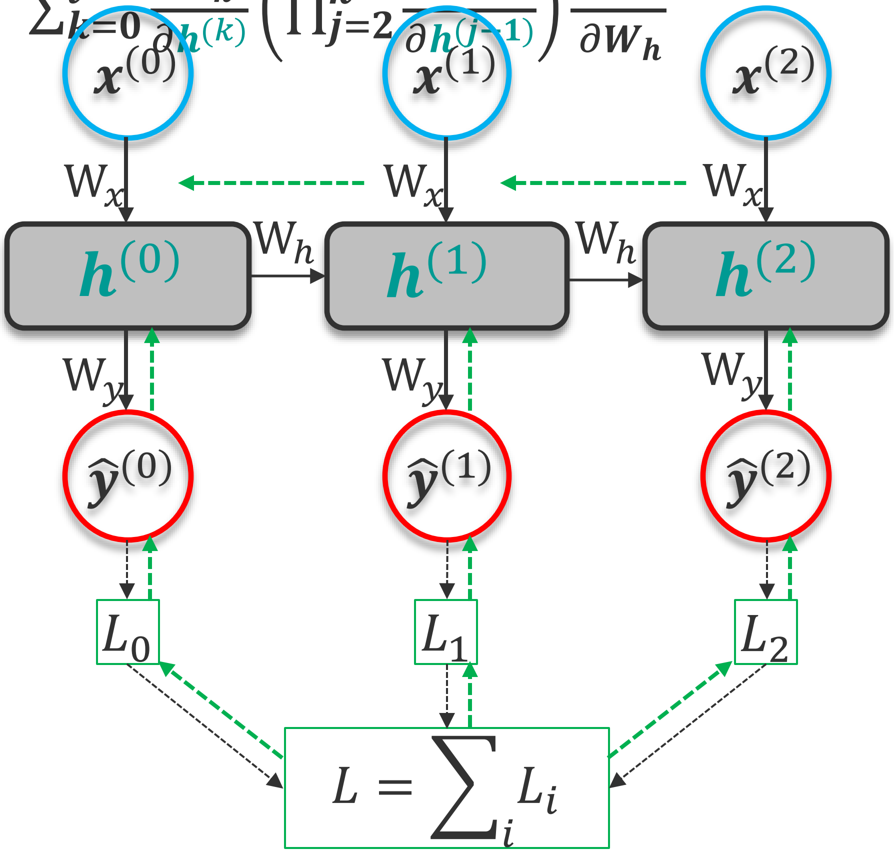
- Adapt parameter to minimize loss



Gradient issues

- Gradient w.r.t. $\boldsymbol{W_h}$ involves repeated computation of gradients w.r.t. $\boldsymbol{h}^{(i)}, \boldsymbol{h}^{(i-1)}, \dots, \boldsymbol{h}^{(1)}$

  => many factors of W_h: $\frac{\partial L}{\partial W_h} = \sum_{k=0}^{i} \frac{\partial L_k}{\partial W_h} \propto \sum_{k=0}^{i} \frac{\partial L_k}{\partial \boldsymbol{h}^{(k)}} \left(\prod_{j=2}^{k} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{h}^{(j-1)}}\right) \frac{\partial \boldsymbol{h}^{(1)}}{\partial W_h}$



$1.1^{100} \sim 1e+5$

Exploding gradient:
when many values > 1
$\Rightarrow$ Quickly reach infinity
$\Rightarrow$ Gradient clipping to scale large gradients

$0.9^{100} \sim 2e-5$

Vanishing gradient:
when many values < 1
$\Rightarrow$ Difficult to learn long period dependencies
$\Rightarrow$ Activation function, weight initialization, network structure

Adapted network structure

- "I grew up in France… I speak fluently (?)"

- Long Short-Term Memory Network: track information throughout many timesteps => memorize long-term dependencies
  - Memory cell: store information
  - Forget gate: what information to ignore
  - Input gate: which values from input to update memory state
  - Output gate: what to output based on input and memory state
  - Gate ~ neuron: activation of a weighted sum

`tensorflow.keras.layers.LSTM`

RNN: When?

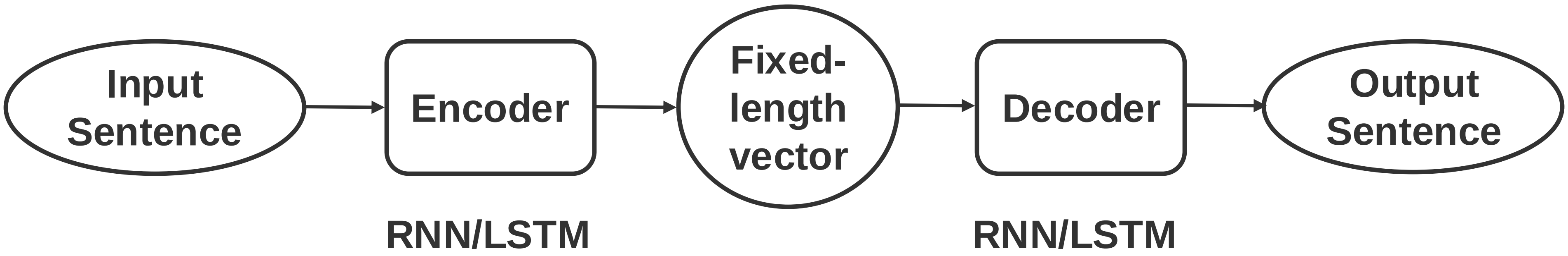
- Language translation, speech recognition, natural language processing
- Life sciences:
  - sequence analysis, genomic sequence evolution,
  - biomolecular function prediction
  - human-virus protein-protein interaction prediction,
  - biomedical ontologies,
  - forecast the spread of virus,
  - pharmaceutical development

Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Deep reinforcement learning
- Generative adversarial networks

Attention

- Machine translation



I do deep learning

I often find myself daydreaming about what I could do with deep pockets full of knowledge, learning from the depths of experience.

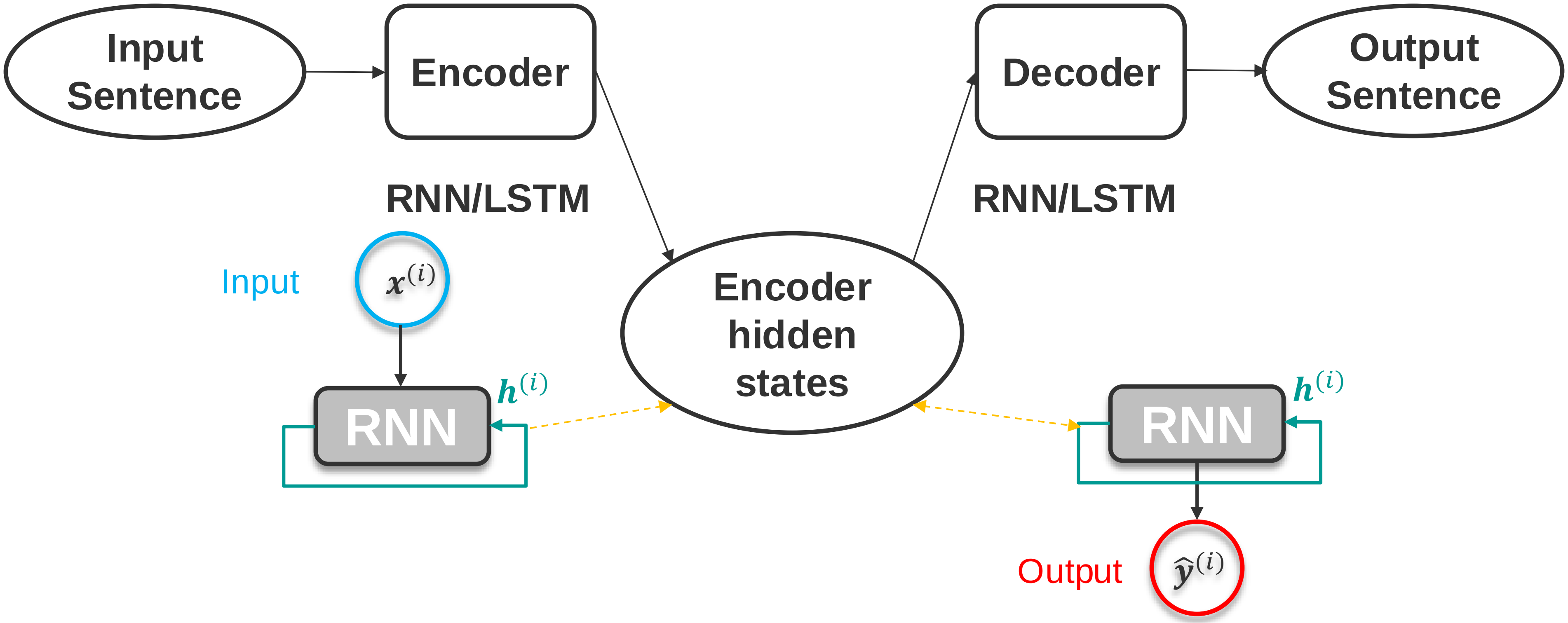
Generated by ChatGPT

Attention

- Mimic cognitive attention
- Improve RNN drawbacks: favor recent elements, fail to handle long sequences
- Model dependency without regard to the distance between elements in the sequences

Attention

- Machine translation



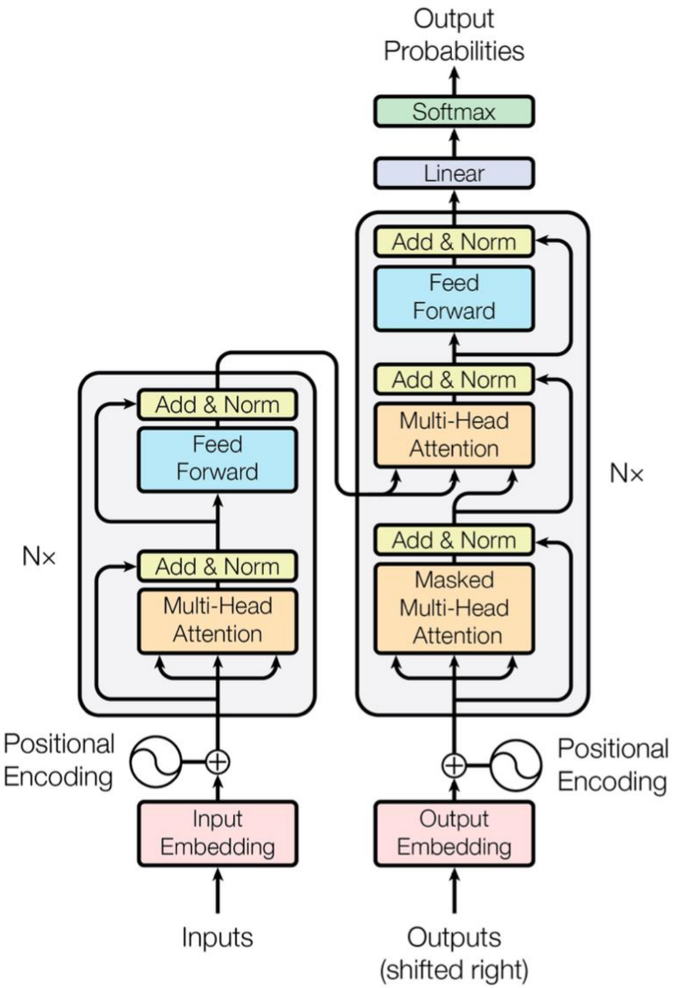
Global attention: all hidden states
Local attention: a few hidden states
Hierarchical-nested attention: both word and sentence levels

Transformer

- Encoder-Decoder using only Attention mechanism for global dependencies between input and output
- Self-attention
- Parallelized processing
- No recurrence or convolution

Large Language Model

- Pre-trained autoencoding models: BERT (Bidirectional Encoder Representations from Transformers)
- Autoregressive models: GPT (Generative Pretrained Transformer)



Vaswani et al. Attention Is All You Need. NIPS 2017

Application in life sciences

- Sequence analysis, genome analysis
- Multi-omics, spatial transcriptomics
- Biomedical informatics
- Drug discovery
- Protein structure prediction
- Foundation model for human single-cell transcriptomics, human genomics

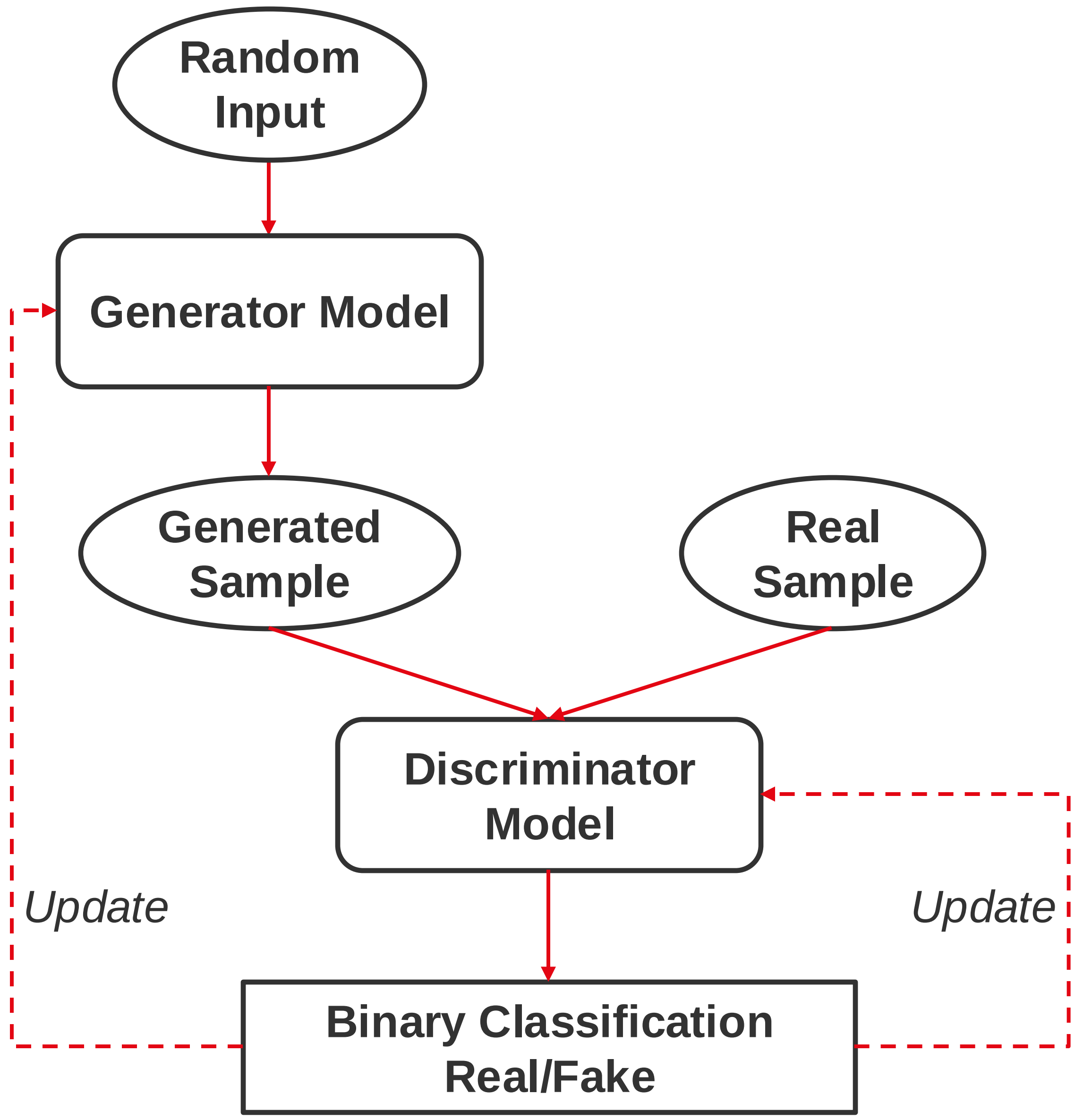
Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Generative adversarial networks
- Deep reinforcement learning

Generative Adversarial Network (GAN)

- Deep-learning-based generative model
- Data augmentation
- Generator: generate new plausible examples from the problem domain.
- Discriminator: classify examples as real (from the domain) or fake (generated).

GAN: How?


Random Input
Generator Model
Generated Sample
Real Sample
Discriminator Model
Update
Update
Binary Classification Real/Fake

GAN: When?

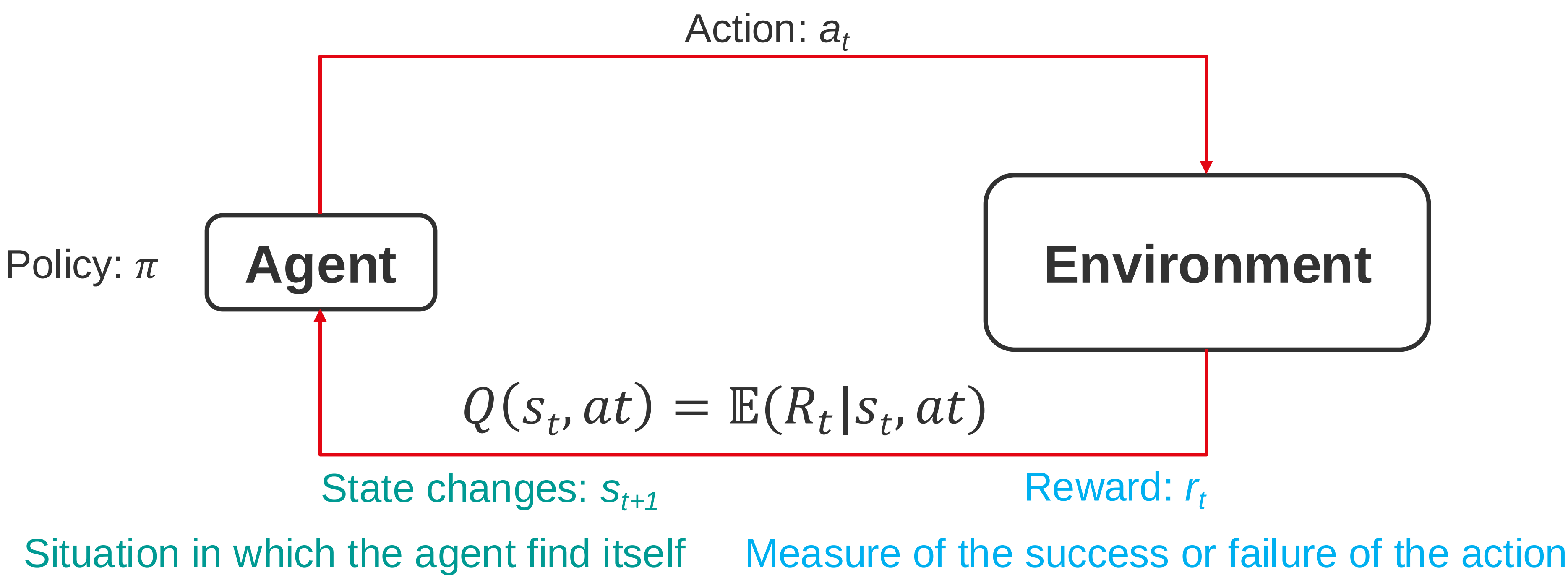
- Digital image processing
- Life sciences:
  - Medical imaging processing
  - Medical informatics: generating discrete data based on real medical data, addressing imbalanced samples problem
  - Generating scRNA-seq data, genomic sequences, compound and protein designs

Outline

- Autoencoders
- Convolutional neural networks
- Recurrent neural networks
- Attention mechanism and transformers
- Generative adversarial networks
- Deep reinforcement learning

Reinforcement Learning

- Reinforcement learning: learning to make decisions through trial and error → Goal: *to act*

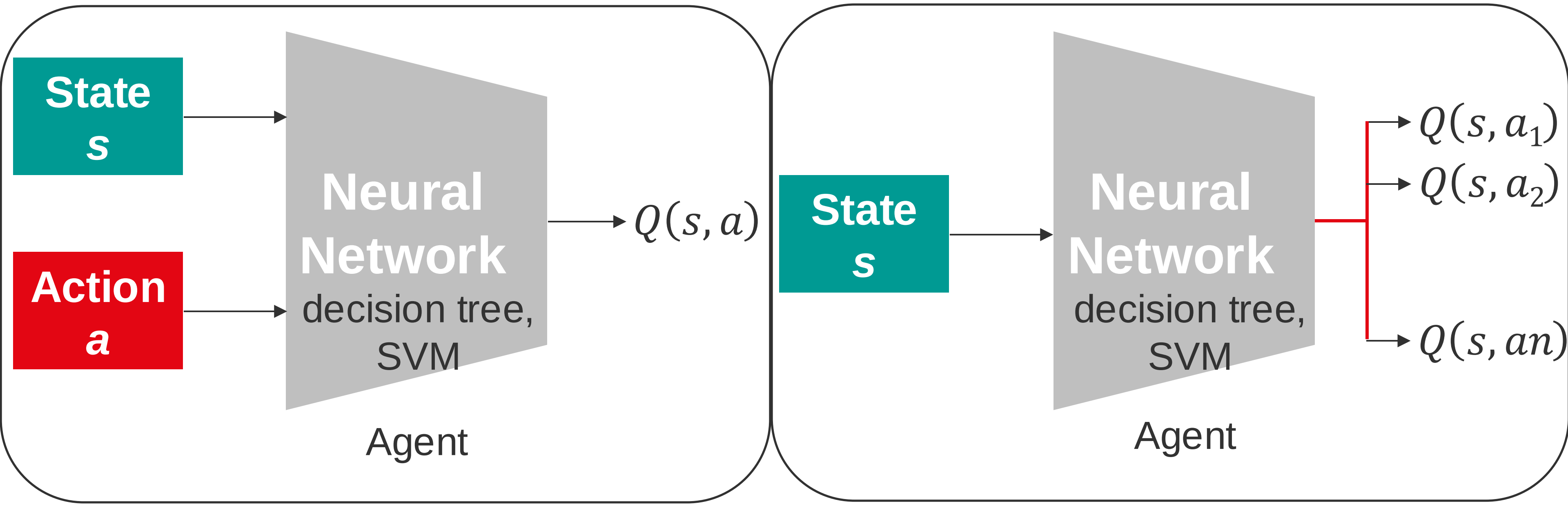


Total Reward (return): $R_t = \sum_{i=t}^{\infty} r_i$

Discounted Total Reward (return): $R_t = \sum_{i=t}^{\infty} \gamma^i r_i$

Deep Reinforcement Learning

- Large state and action spaces
- Map states and actions to Q values



Action + State → Expected Return

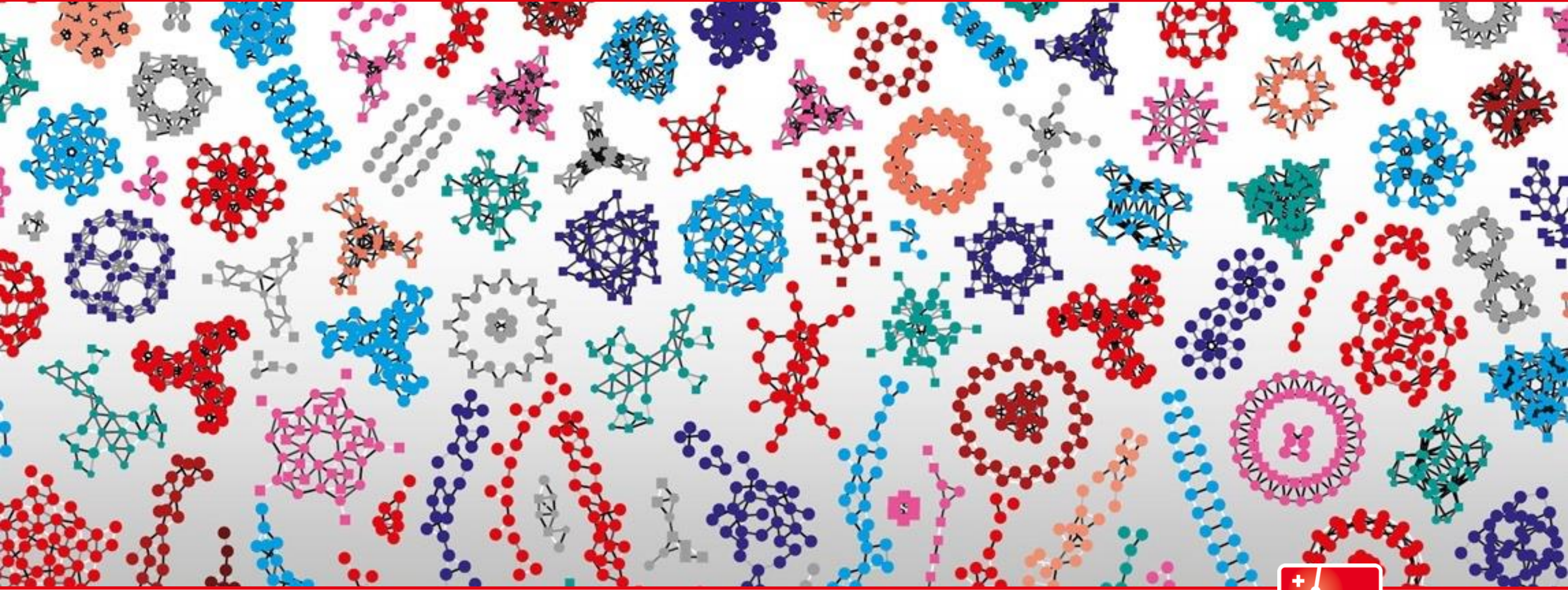
State → Expected Return for each action

Use neural networks to learn Q, then infer the optimal policy $\pi(s)$

Deep Reinforcement Learning: When?

- Recommendation systems, robotics, energy smart grids

- Life sciences
  - biological sequence annotation,
  - biological data mining,
  - prediction of bacterial genomes,
  - protein-protein interaction,
  - segmentation in medical imaging,
  - brain machine interface



SIB
Swiss Institute of Bioinformatics

Thank you!

elixir SWITZERLAND

www.sib.swiss