

Introduction to Deep Learning

Giorgio Bar – giorgio.bar@to.infn.it

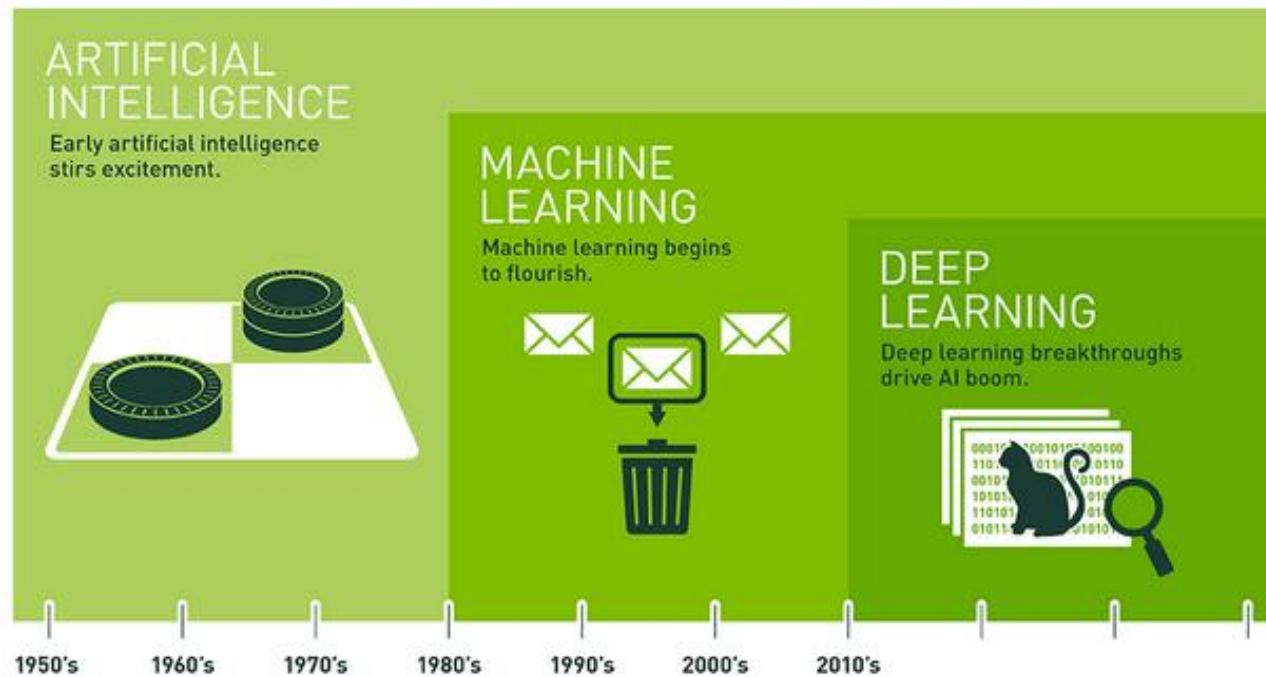
Scuola di Specializzazione in Fisica Medica – Università di Torino

Anno accademico 2023/2024 (anno solare 2024/2025)



Deep Learning

- A subset of AI and machine learning that uses multi-layered artificial neural networks. It can **automatically learn representations from data**, without introducing hand-coded rules or human domain knowledge.



<https://developer.nvidia.com/deep-learning>

What is AI?

- Thinking humanly. The cognitive modeling approach
- Acting humanly. The Turing Test approach
- Thinking rationally. The “laws of thought” approach
- Acting rationally. The rational agent approach

<https://elizaycayilmaz.medium.com/definitions-of-ai-5efb8989db09>

How do we define “Learning”?

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .
- T : Recognizing hand-written words
- E : Database of human-labeled images of handwritten words
- P : Percentage of words correctly classified

Mitchell, T. (1997). Machine Learning. McGraw Hill.

Machine Learning Recap

- Machine learning: a family of algorithms with ability to automatically learn and improve from experience without being explicitly programmed.
Potential to approximate linear and non-linear relationships
- For a given problem:
 - Find algorithm that will give the best results
 - Train model
 - Tune hyperparameters
 - Do cross-validation
 - Do inference

Machine Learning Recap

Supervised Learning Predicting values. **Known targets.**

User inputs correct answers to learn from.

Machine uses the information to guess new answers.

Regression

Estimate continuous values.

Classification

Identify a unique class

Unsupervised Learning Search for structure in data. **Unknown targets.**

User inputs data with undefined answers.

Machine finds useful information hidden in data.

Cluster Analysis

Group into sets

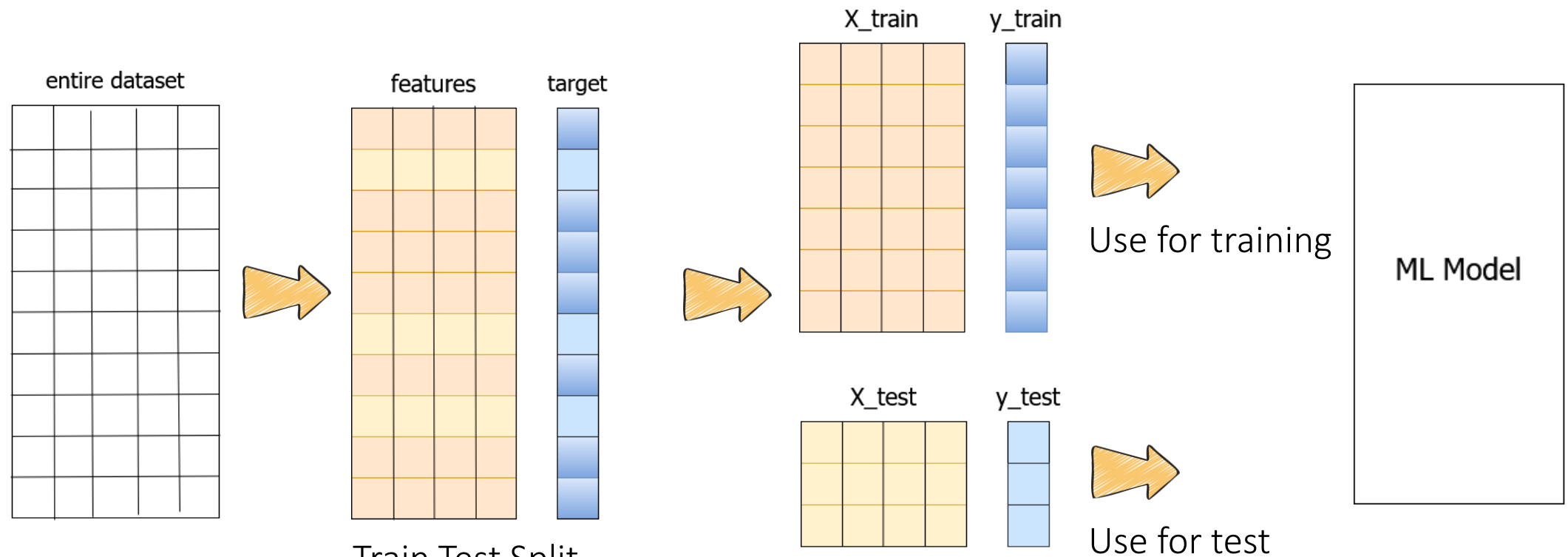
Density Estimation

Approximate distribution

Dimension Reduction

Select relevant variables

Training and test set

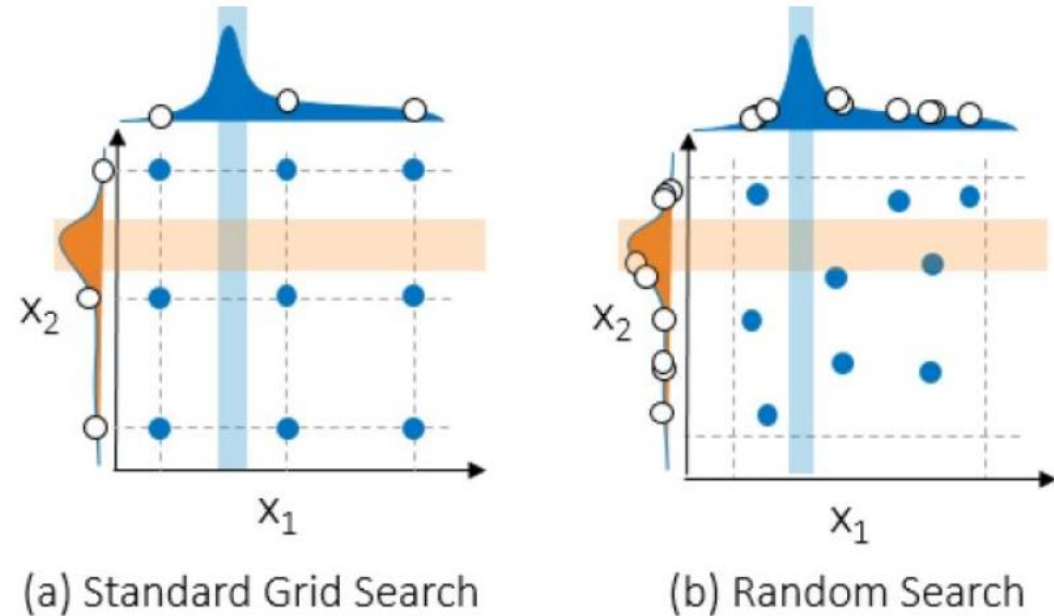


Train Test Split.

- Random sampling / Stratified sampling
- Split into training and test sets.

Parameters and hyperparameters

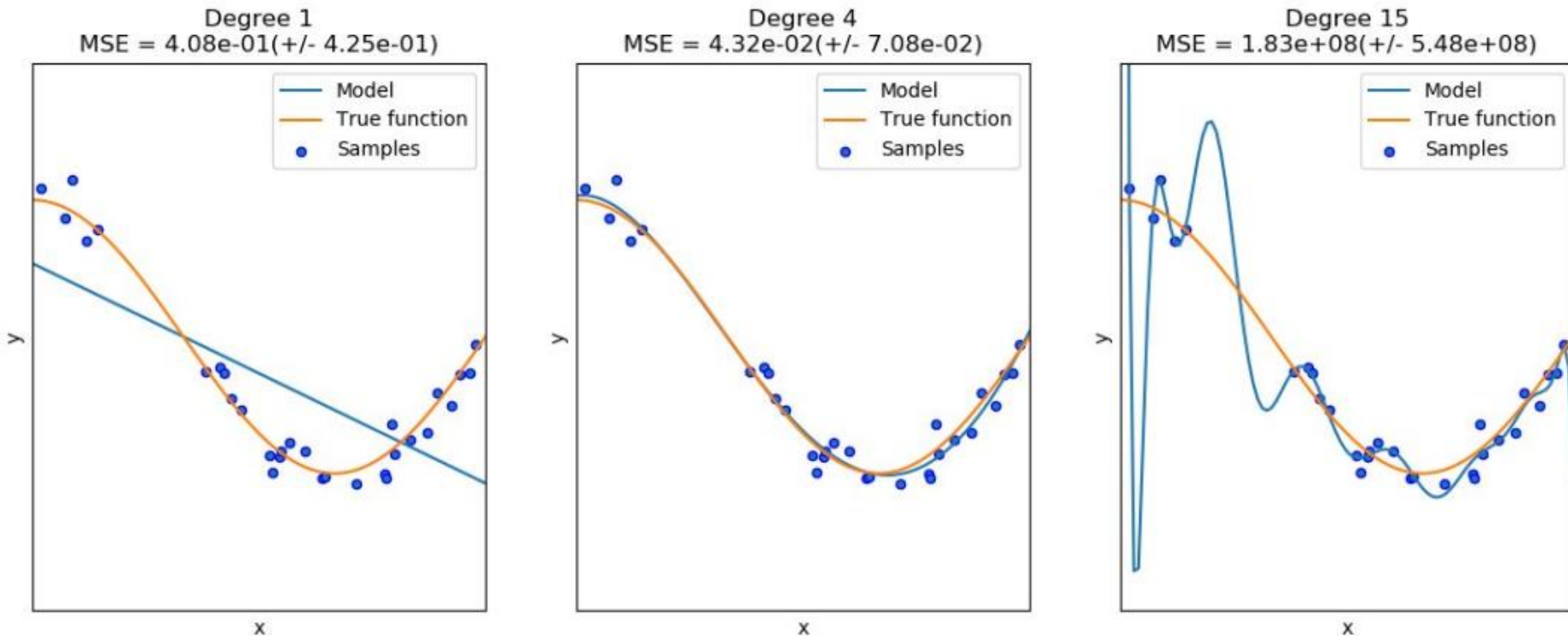
- Model **parameter**: a configuration variable that is internal to the model and whose value can be learned from data during training.
- **Hyperparameter**: a configuration that is external to the model and whose value cannot be directly trained from the data.
- Hyperparameter tuning



<https://blog.roboflow.com/what-is-hyperparameter-tuning/>

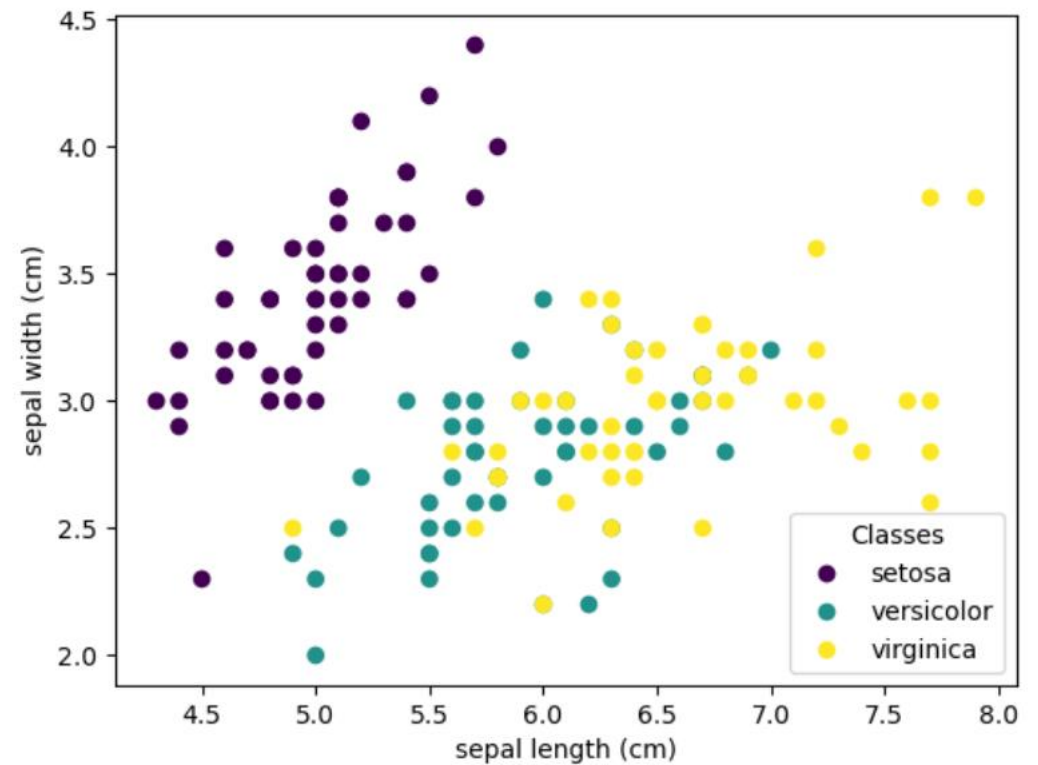
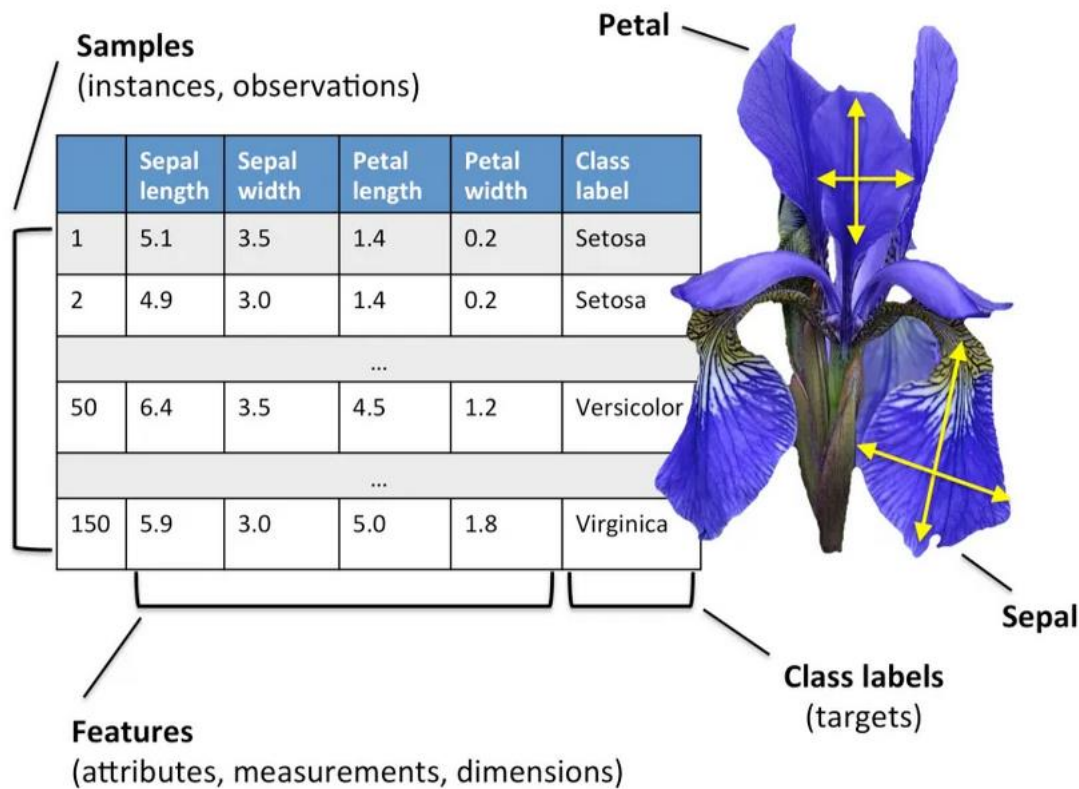
Overfitting / Underfitting

- **Underfitting**: the model doesn't have enough parameters to describe data.
- **Overfitting**: the model has too many parameters and **cannot generalize**.



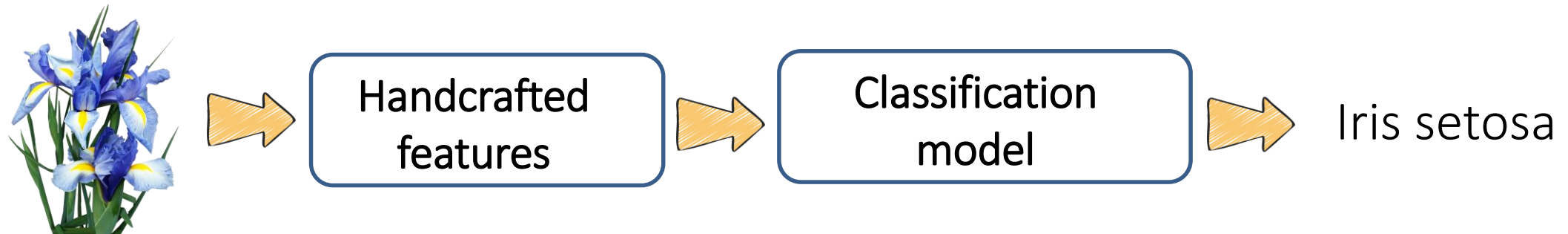
Choosing a representation

- Find a solution to learn both the representation and the model for the data by exploiting advanced computing capabilities

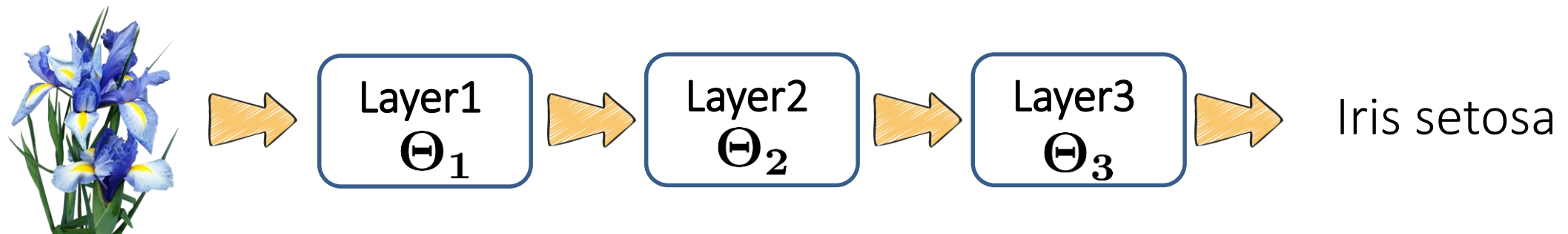


Choosing a representation

- Traditional framework

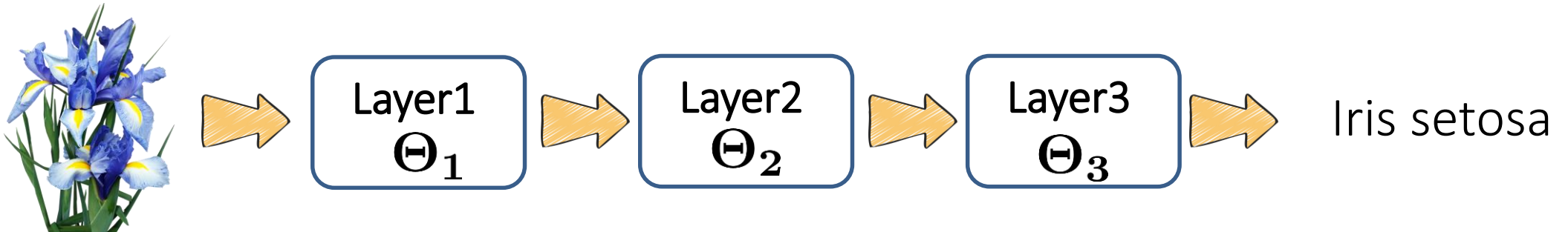


- Deep Learning



Deep Learning

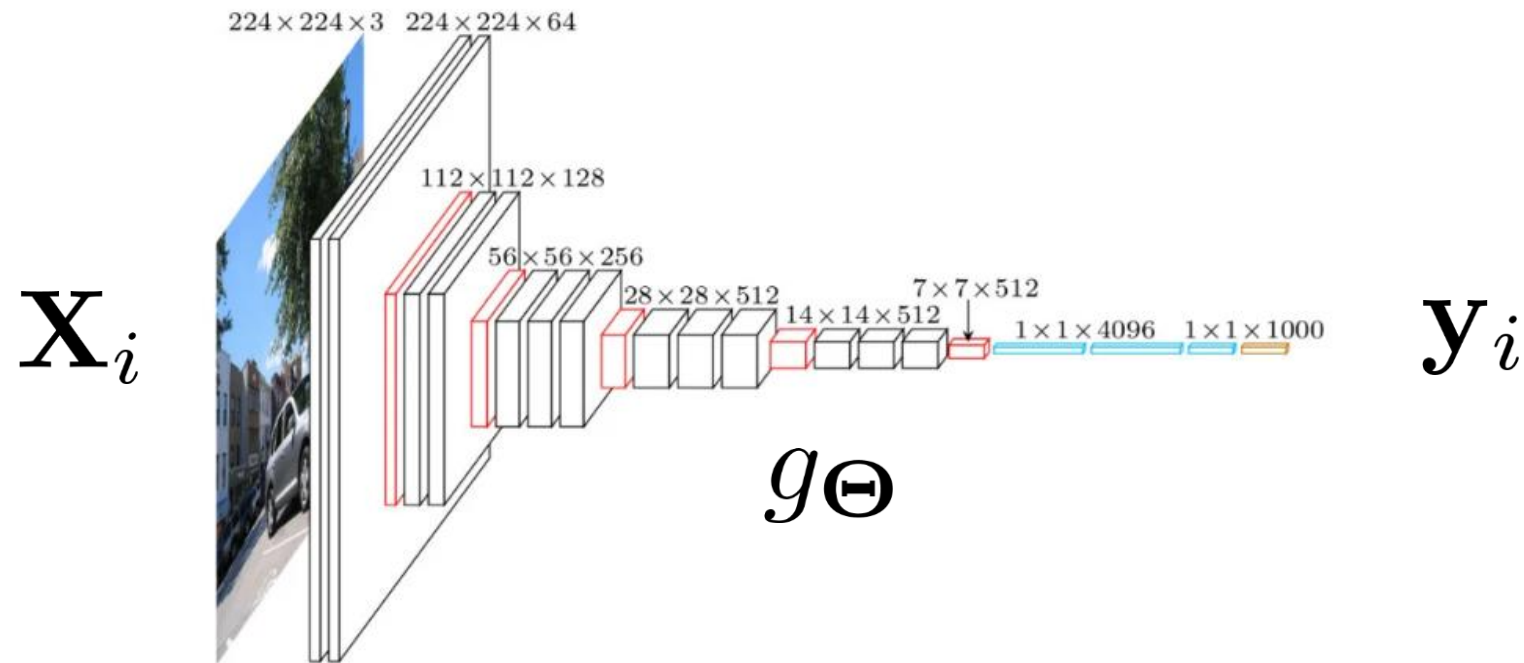
- Deep Learning means using an artificial **neural network** with **several layers** of nodes between input and output.
- A family of **parametric models** which learn **nonlinear hierarchical** representations.



- Layers between input and output compute relevant features **automatically** in a series of stages.

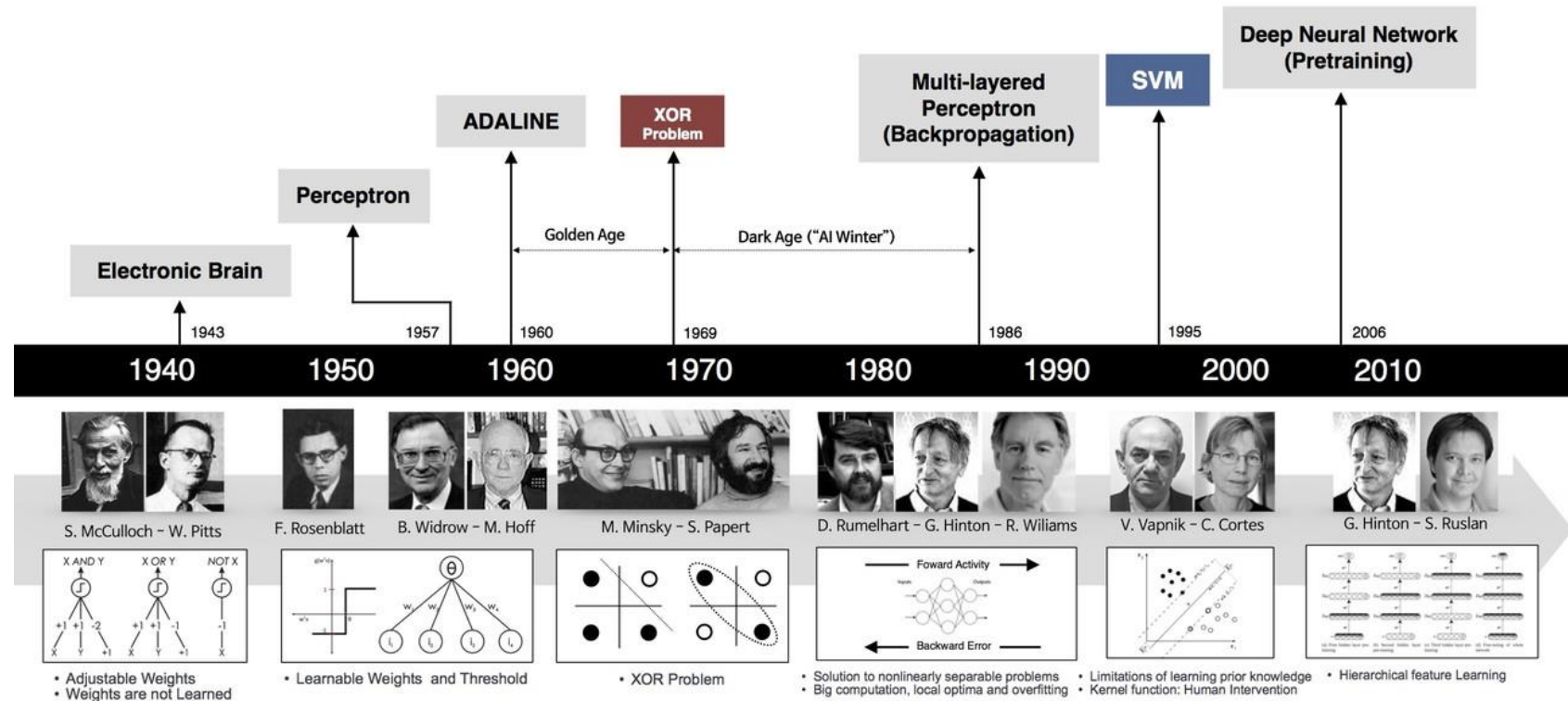
Deep neural networks

We deal with **highly parametrized models** called deep neural networks



Brief history of neural networks

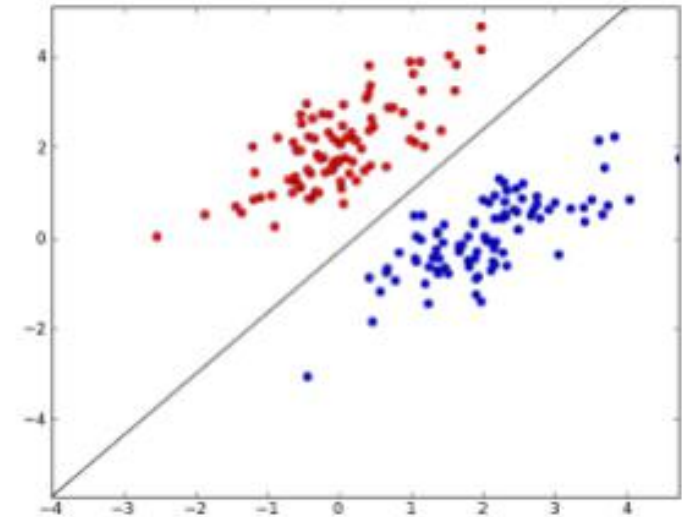
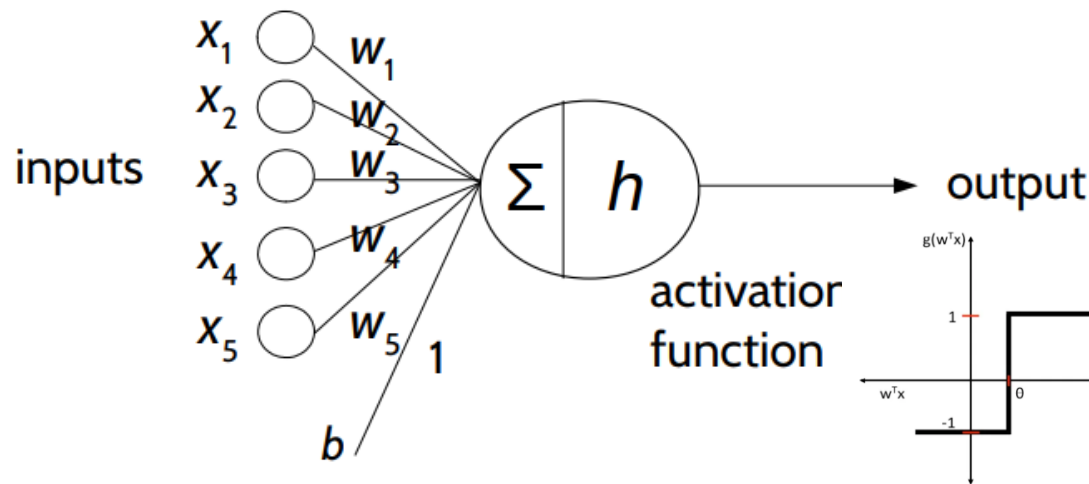
- Neural Networks have been around for many years, their development occurred in a series of stepwise improvements, interspersed with periods of stasis.



<https://medium.com/@sanatbhandarkar75/deep-learning-demystified-4a7554a5c3ba>

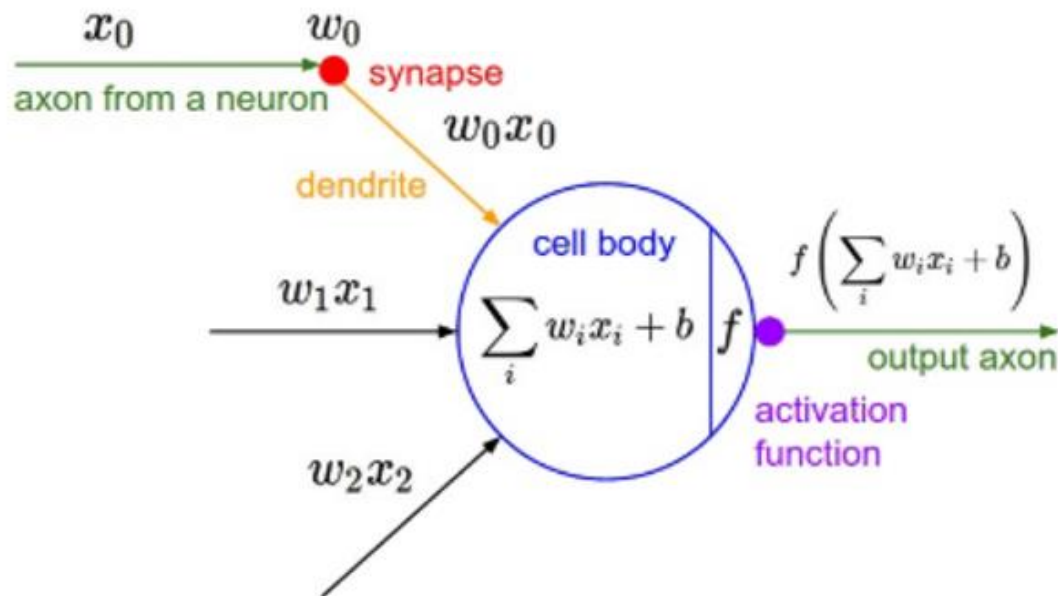
Rosenblatt's Perceptron (1958)

- A machine for **linear** classification.
 - The output is determined by a weighted sum of the inputs, which is then passed through an activation function (a step function).
 - An iterative method is used to adjust the weights of the perceptron based on the errors made in predictions.



Artificial neuron

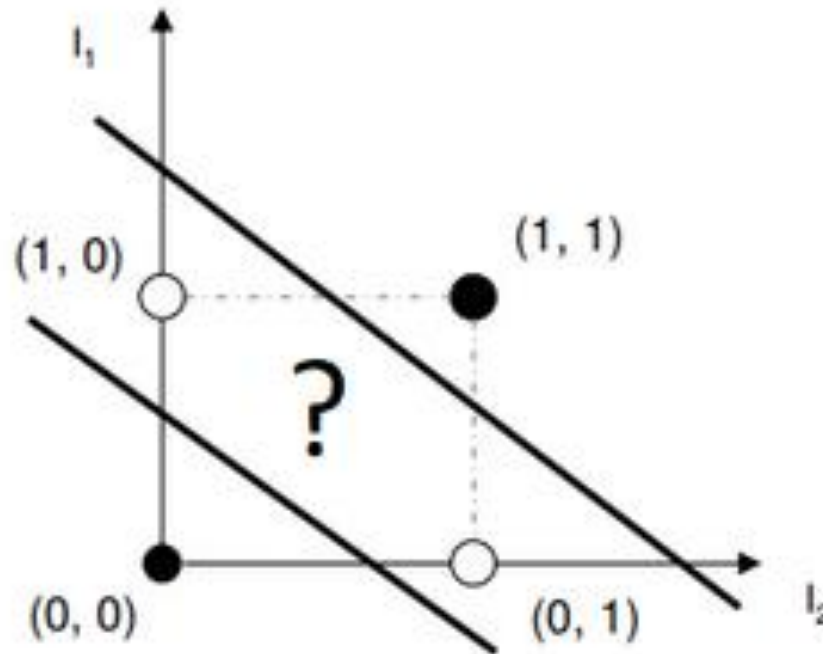
- A very rough mathematical approximation of a biological neuron.



First neural networks winter (1970 -Minsky)

- The original perceptron has trouble with simple non-linear tasks.
- XOR cannot be solved by a **1-layer** perceptron.

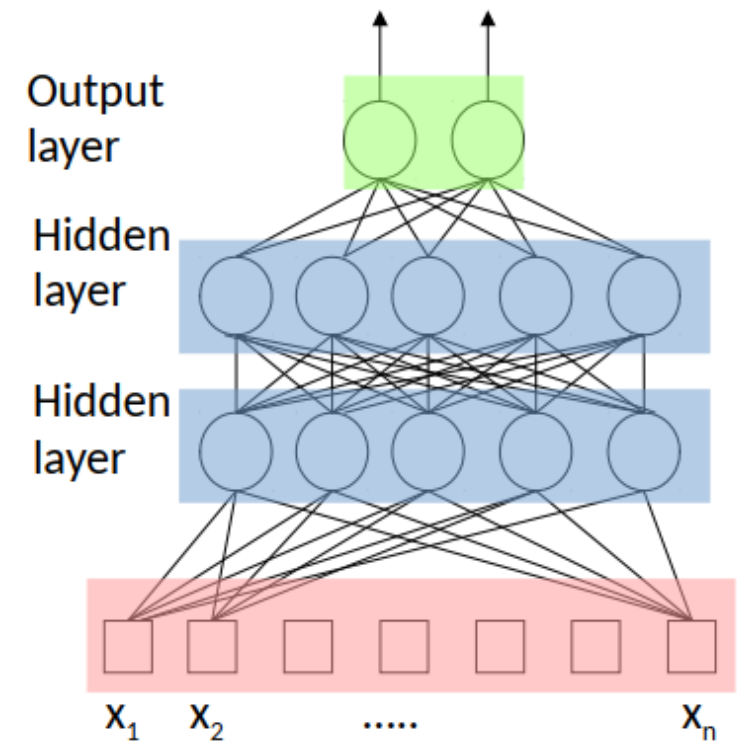
XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



No **line** can separate the white from the black

Feed Forward Neural Network

- Multi-Layer Perceptron (MLP) can solve XOR.
- Densely connect artificial neurons to realize compositions of **non-linear** functions.
- The information is propagated from the inputs (n-dimensional feature vectors) to the outputs.
- **Backpropagation** algorithm (1986) to train complex NN which processes “large” training sets.
- Backpropagation still plays a core role in training neural networks.




<https://www.nature.com/articles/323533a0>

Model training

- Training occurs through the process of modifying the weights of each layer (i.e. the parameters of the model) to produce a network that performs some function.

$$g_{\Theta}(\mathbf{x}) = (\sigma \circ f_{\Theta_n}) \circ (\sigma \circ f_{\Theta_{n-1}}) \circ \cdots \circ (\sigma \circ f_{\Theta_1})(\mathbf{x})$$


↑ parameters of the network ↑ parameters of layer n non-linear activation function ↑ input

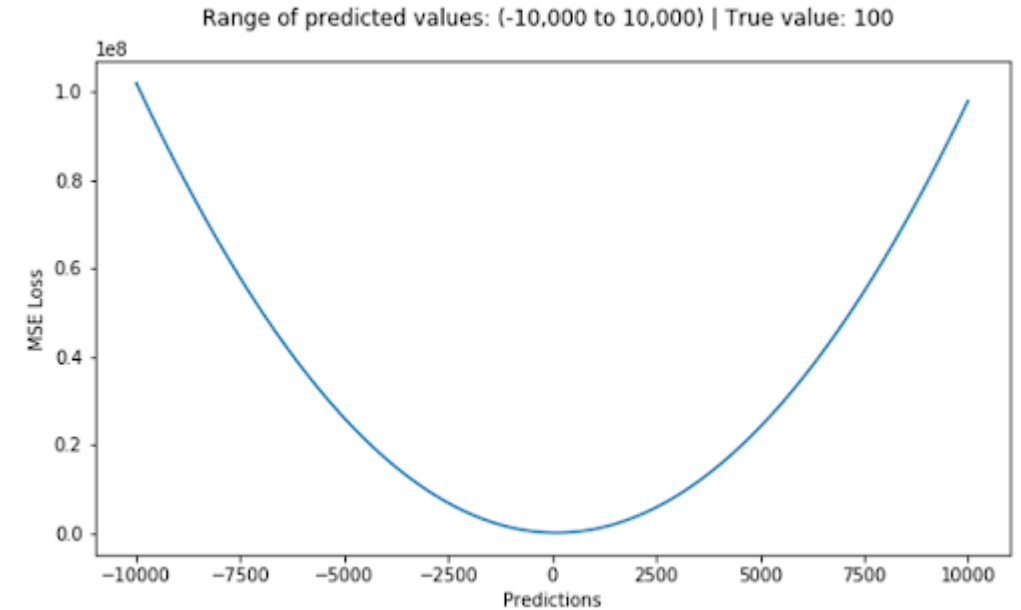
- The goal of backpropagation is to optimize the weights (bias terms correspond to a weight with a fixed input of 1) of a neural network to **minimize a loss function**.
- The loss function is a mathematical function that quantifies the difference between the predicted output of the network and the actual target values.

Loss functions

- **Mean Squared Error / L2 Loss**

A widely used and straightforward loss function for regression tasks. It measures the average squared difference between the predicted values and the actual values.

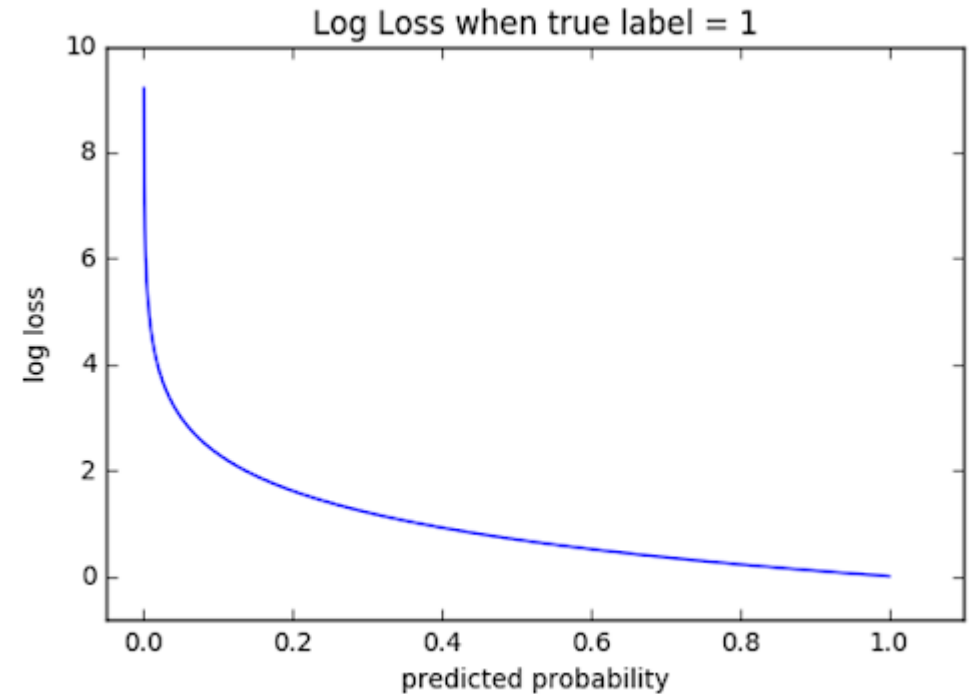
$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (y_{\text{true}} - y_{\text{pred}})^2$$



Loss functions

- **Binary Cross-Entropy Loss / Log Loss**

A common loss function used in classification problems. The cross-entropy loss decreases as the predicted probability converges to the actual label.



$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

Backpropagation

- By minimizing a loss, the network improves its performance on the given task

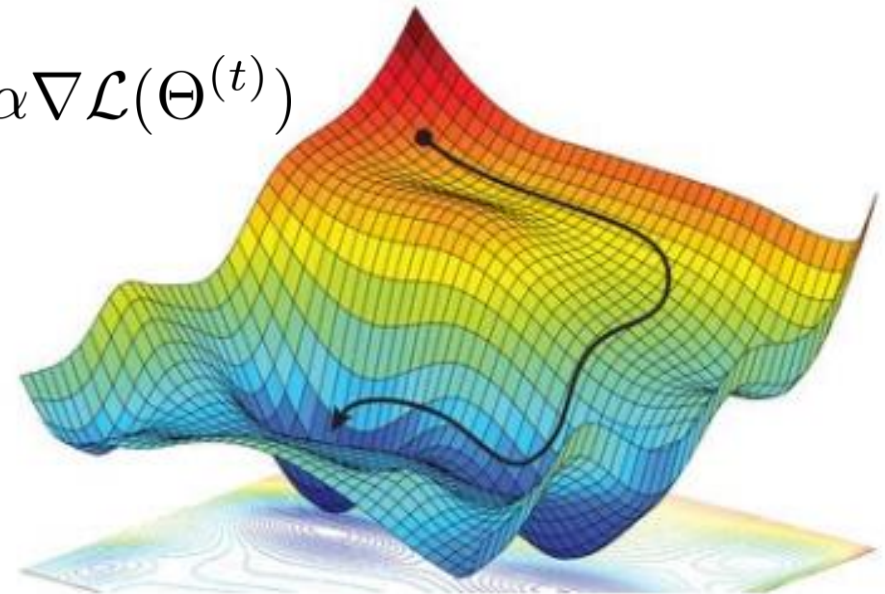
$$\Theta^* = \arg \min_{\Theta} \sum_{i=1}^m \mathcal{L}(y_i, g_{\Theta}(\mathbf{x}_i))$$

Where $\{\mathbf{x}_i, y_i\}$ is the single training sample

- Backpropagation achieves this by propagating errors backward through the network, starting from the output layer and moving towards the input layer

Optimization with gradient descent

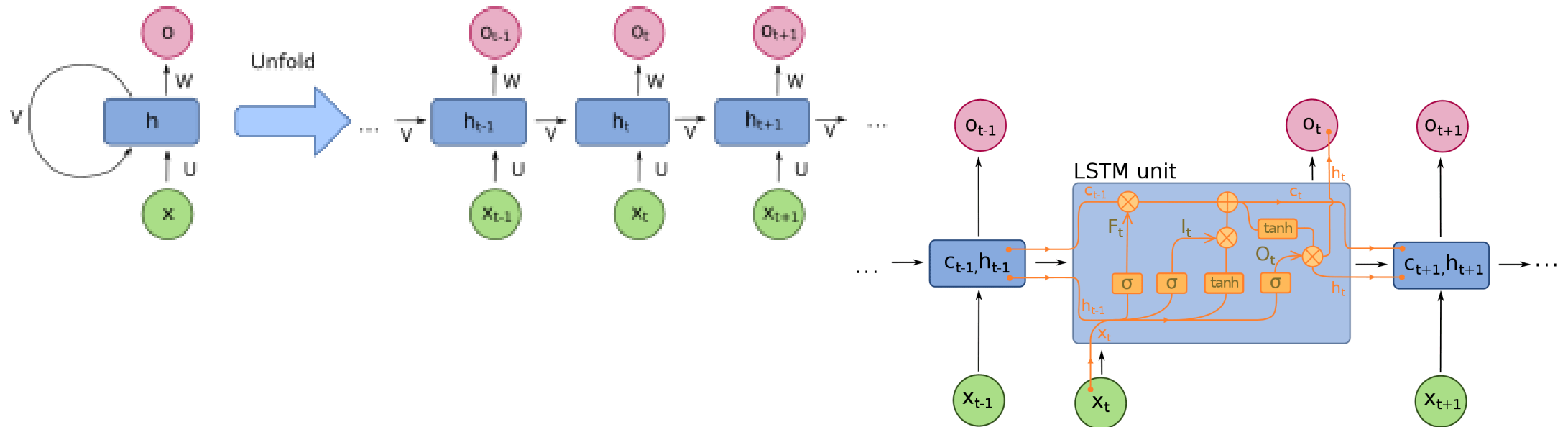
- Gradient Descent is a first-order iterative minimization algorithm.
- Overall idea: Move where the function decreases the most.
 1. Start from some point $\Theta^{(0)} \in \mathbb{R}^n$
 2. Iteratively compute $\Theta^{(t+1)} = \Theta^{(t)} - \alpha \nabla \mathcal{L}(\Theta^{(t)})$
 3. Stop when a minimum is reached
- The **learning rate hyperparameter** $\alpha > 0$ determines the step size at each iteration.



Example of a loss function $\mathcal{L}_{\Theta} : \mathbb{R}^2 \rightarrow \mathbb{R}$

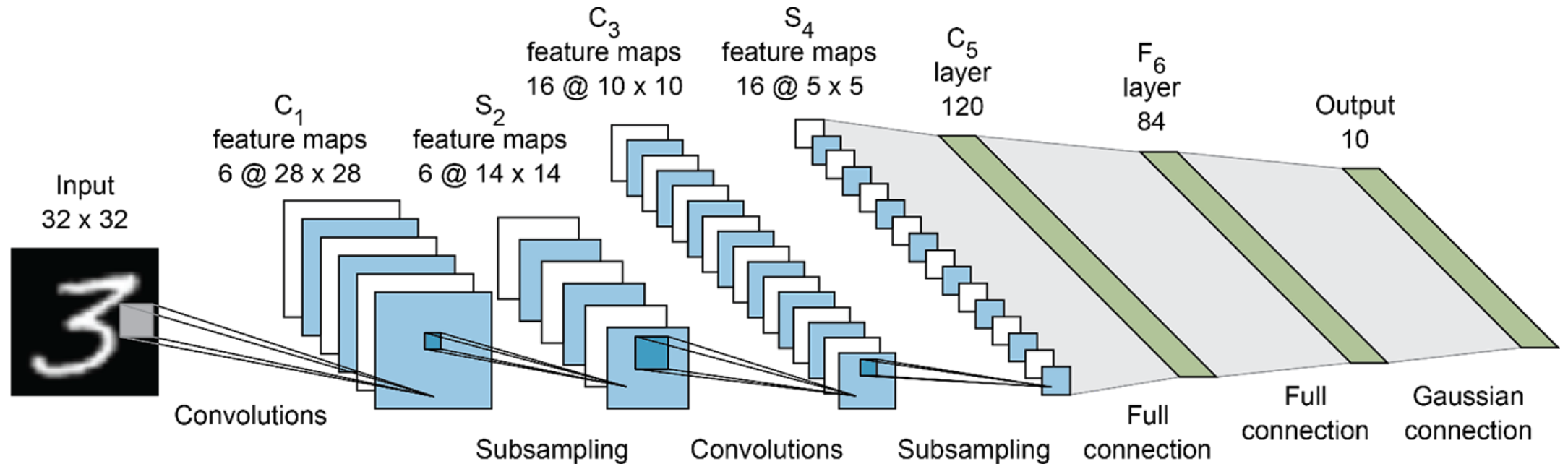
LSTM (1997 Hochreiter and Schmidhuber)

- Long Short-Term Memory Network.
- An advanced recurrent neural network (RNN) that can learn long-term dependencies between time steps of sequence data (more on this later).



CNN (1998 LeCun)

- Convolutional Neural Networks
- LeNet-5, specifically designed for optical character recognition (OCR).
- Successful demonstration of hierarchical feature learning.



The need for priors

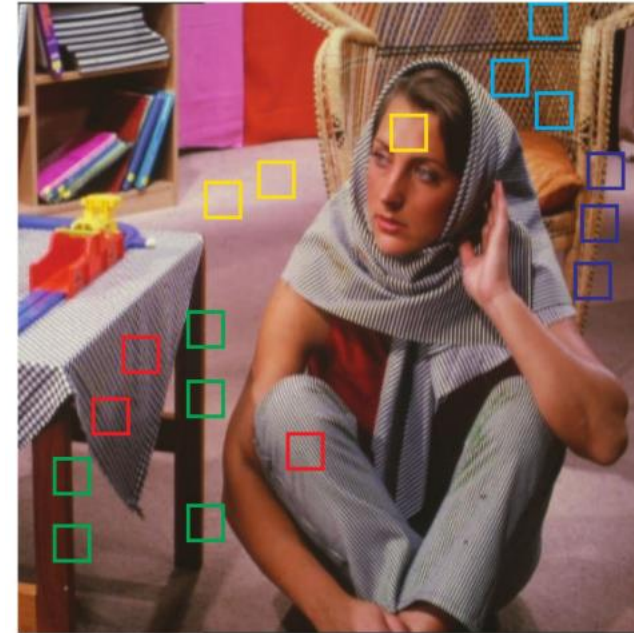
- Deep feed-forward networks are provably **universal**, however:
 - We can make them arbitrarily complex.
 - The number of parameters can be huge.
 - Very difficult to optimize.
 - Very difficult to achieve generalization.
- We need additional **priors** as a (partial) remedy to the above.
- Look for “universal” priors that are task-independent to some extent.
- Task-independent priors must come with the **data**.

Take advantage of the structure of the data

- Data is often composed of **hierarchical, local, shift-invariant patterns**.
- CNNs directly exploit this fact as a prior



Data often carries structural priors in terms of repeating patterns and compositionality

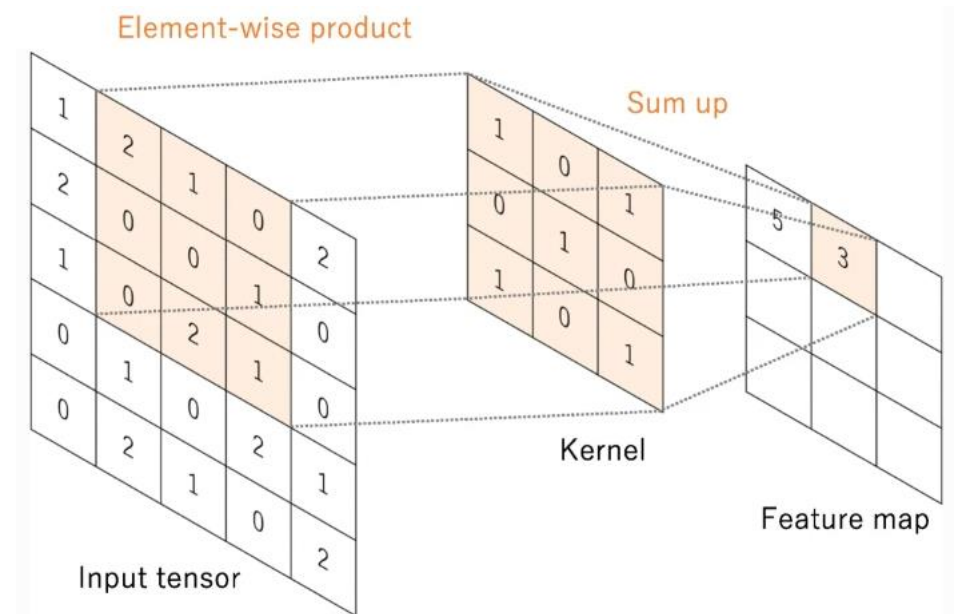
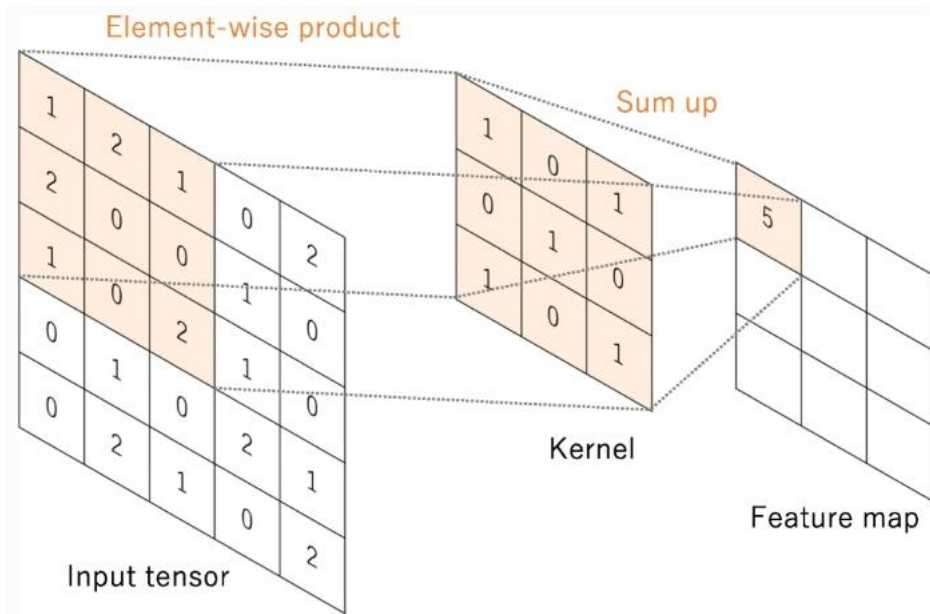


Data tends to be self-similar across the domain

https://www.i-aida.org/wp-content/uploads/2022/07/lect2_generalized_convolution.pdf

Convolutional Neural Networks

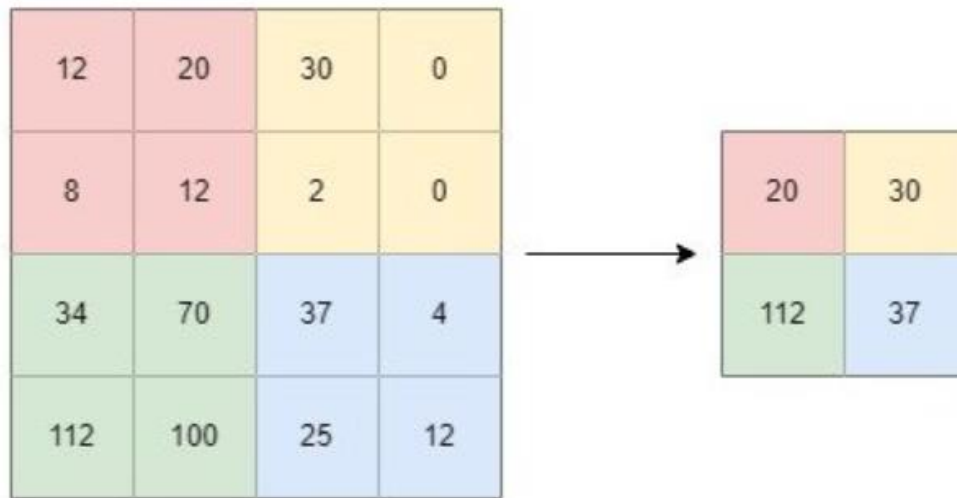
- **Sharing weight** parameters across spatial positions (shift-invariant kernels)



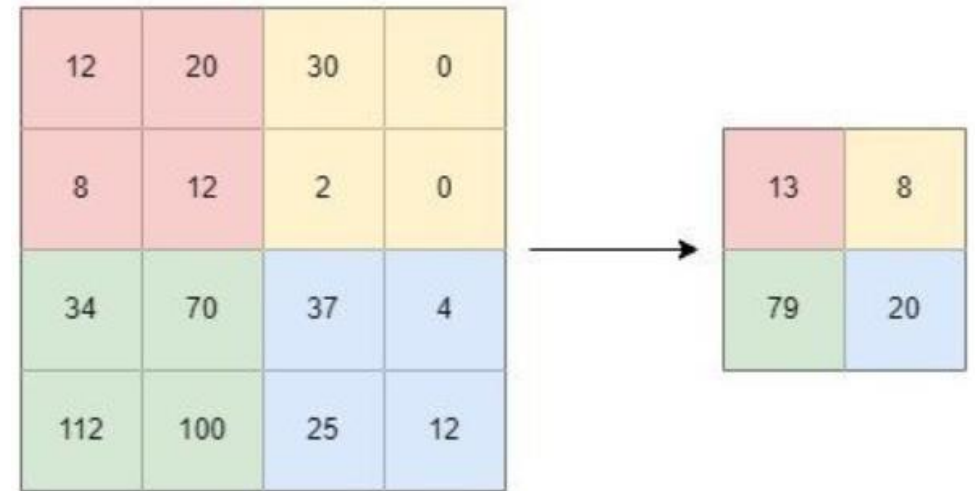
<https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9>

Pooling layers

- Pooling is a down-sampling operation that reduces the spatial dimensions of the feature maps while retaining important information.
- Two common pooling methods are average pooling and max pooling



Max Pooling



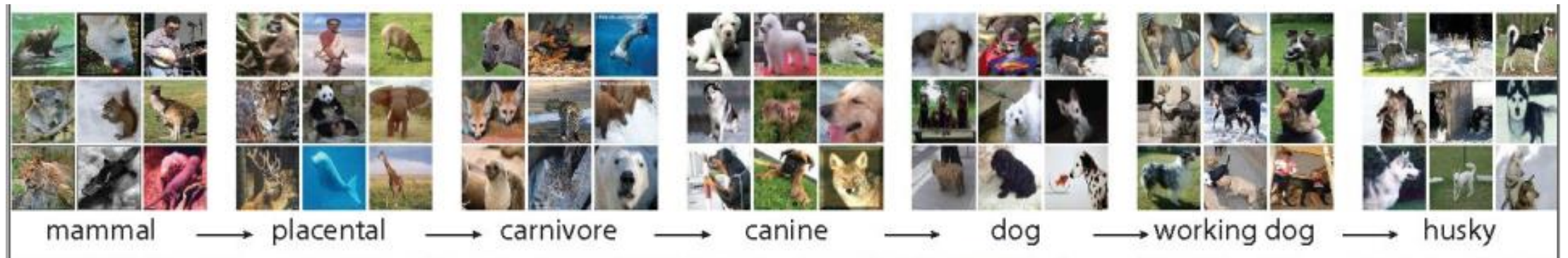
Average Pooling

Second neural networks winter

- General interest in AI declined as the expectations could not be met.
- Lack of processing power.
- Lack of data (large and well-annotated data sets)
- Neural networks could not improve beyond a few layers
 - The vanishing gradient problem
- New Machine Learning models e.g. Support Vector Machines (SVMs)
 - Similar accuracies with better math and proofs and fewer heuristics

Pre-trained Deep Neural Networks

- ImageNet: A Large-Scale Hierarchical Image Database (2009 Fei-Fei Li)



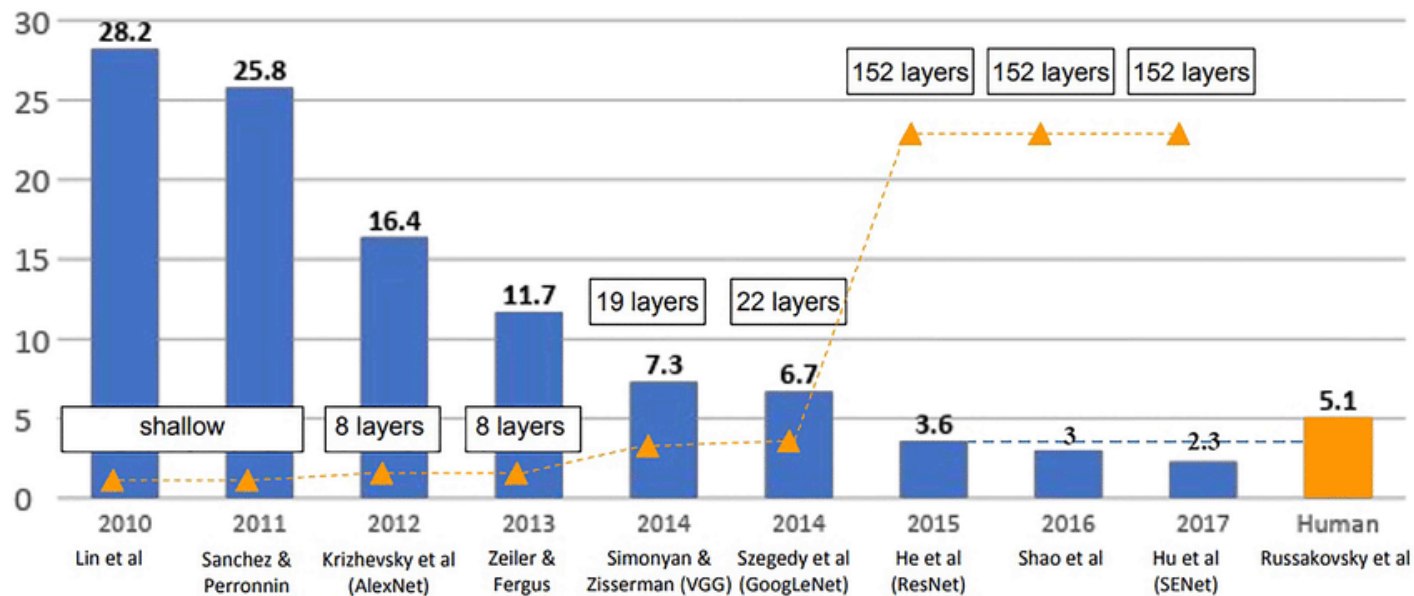
- Features extracted from a pre-trained image classification NN can be used as a starting point to learn a new task.
- Transfer learning** to address the issue of lack of training data.

<https://doi.org/10.3389/fonc.2024.1432188>

https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf

Deep Learning Renaissance AlexNet (2012)

- The main ingredients of modern Deep Learning:
 - a massive training data set (ImageNet),
 - graphics processing units (GPUs) for hardware acceleration,
 - a deep convolutional neural network, technical advances (more on this later)



Algorithms that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) . Top-5 error (probability that all top-5 classifications proposed by the algorithm for the image are wrong)

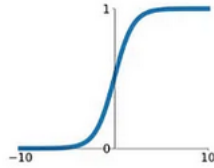
Technical advances

- Activation functions and losses (e.g. ReLU and cross-entropy)
- Data augmentation and pretraining
- Address overfitting (e.g. dropout)
- Architectures (e.g. ResNet)
- Weights initialization
- Batch normalization
- ...

Activation functions

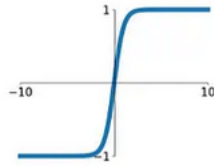
Sigmoid

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



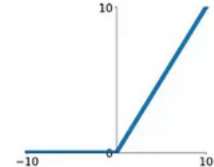
tanh

$$\tanh(x)$$



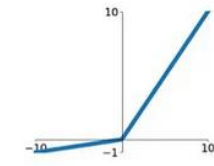
ReLU

$$\max(0, x)$$



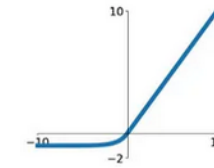
Leaky ReLU

$$\max(0.1x, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



- Sigmoid: commonly used in the output layer of binary classification models. Sigmoid and tanh face the vanishing gradient problem (they saturate in both directions).
- ReLU function is a general activation function
 - Better gradient propagation, fewer vanishing gradient problems
 - Sparse activation: e.g. in a randomly initialized network, only about 50% of hidden units are activated (having a non-zero output).
 - Dying ReLU: some neurons consistently output zero and become inactive. Leaky ReLU mitigates this issue.

Stochastic Gradient Descent (SGD)

- Use a mini-batch **sampled** from the dataset for gradient estimate.
- ① Initialize Θ
- ② Pick a mini-batch \mathcal{B}
- ③ Update the model parameters $\Theta^{(t+1)} = \Theta^{(t)} - \alpha \nabla \mathcal{L}_{\Theta}(\mathcal{B})$
- ④ Go back to step ②
- When steps ② - ④ cover the entire training set we have an **epoch**.
- Like Gradient Descent, the algorithm proceeds for many epochs.
- The update cost is **constant** regardless of the size of the training set.
- SGD does not depend on the number of examples, implying **better generalization**.

Regularization

- One of the major aspects of training a model is addressing overfitting, where a model memorizes training data details but can't generalize to new data.
- **Early Stopping** is a regularization technique that stops training when the model performance on the validation set is getting worse.
- **L1 and L2 Regularization** helps the model to generalize by adding a penalty to the loss function to discourage the model from having large weight values.

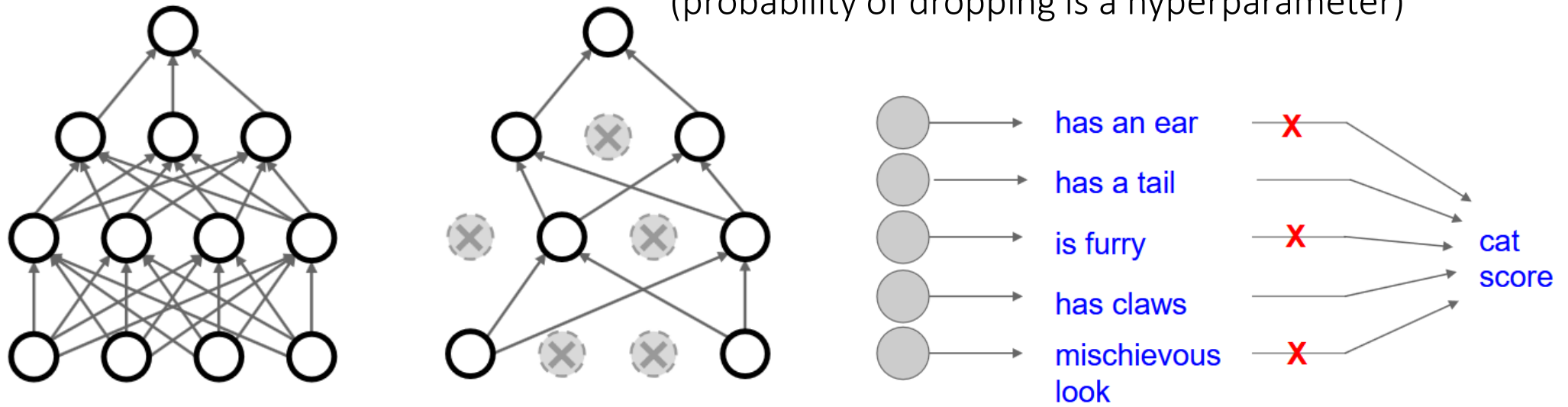
$$\text{L1 + MSE} \quad \text{Min} \left(\sum_{i=1}^m (y_i - w_i x_i)^2 + p \sum_{i=1}^m |w_i| \right)$$

$$\text{L2 + MSE} \quad \text{Min} \left(\sum_{i=1}^m (y_i - w_i x_i)^2 + p \sum_{i=1}^m (w_i)^2 \right)$$

Regularization: Dropout

- **Dropout**: in each forward pass, randomly set some neurons to zero.

(probability of dropping is a hyperparameter)

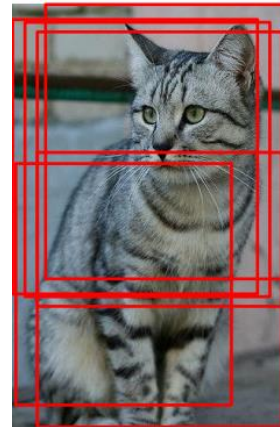


- Forces the network to have a redundant representation, prevents co-adaptation of features
- Another interpretation: Dropout is training a large ensemble of models (that share parameters).


Srivastava et al, "Dropout: A simple way to prevent neural networks from overfitting", JMLR 2014

Regularization: Data augmentation

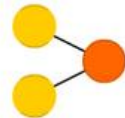
- Increase the diversity of data available for training, without collecting new data.
- Horizontal flips
- Random crops and scales
- Color jitter e.g. randomize contrast and brightness
- Random mix/combinations of translation, rotation, stretching ...



The Neural Network Zoo

-  Input Cell
-  Backfed Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Capsule Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Gated Memory Cell
-  Kernel
-  Convolution or Pool

Perceptron (P)



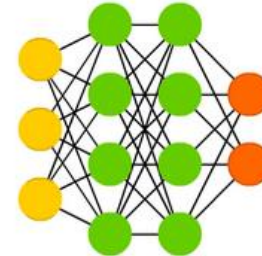
Feed Forward (FF)



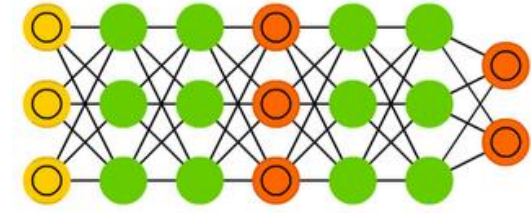
Radial Basis Network (RBF)



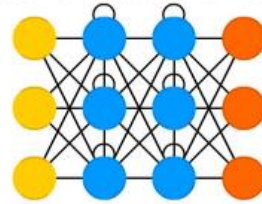
Deep Feed Forward (DFF)



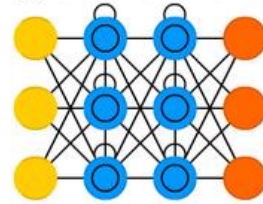
Generative Adversarial Network (GAN)



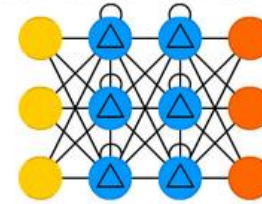
Recurrent Neural Network (RNN)



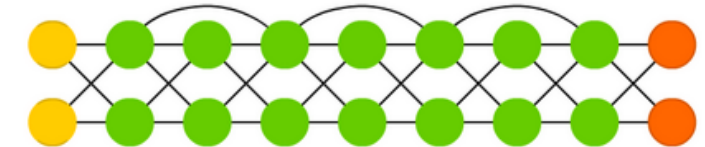
Long / Short Term Memory (LSTM)



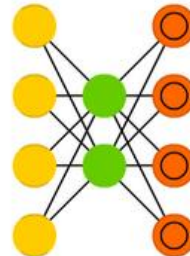
Gated Recurrent Unit (GRU)



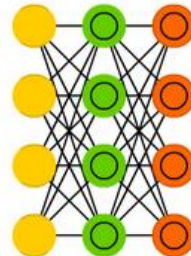
Deep Residual Network (DRN)



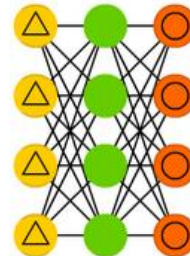
Auto Encoder (AE)



Variational AE (VAE)



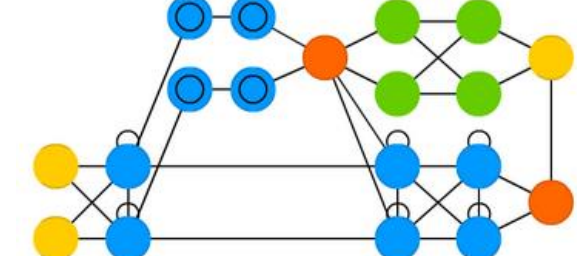
Denoising AE (DAE)



Sparse AE (SAE)



Attention Network (AN)

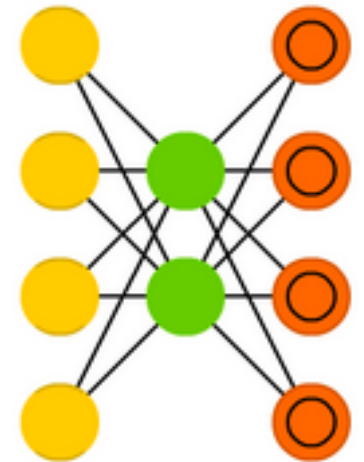


<https://www.asimovinstitute.org/neural-network-zoo/>

Autoencoders

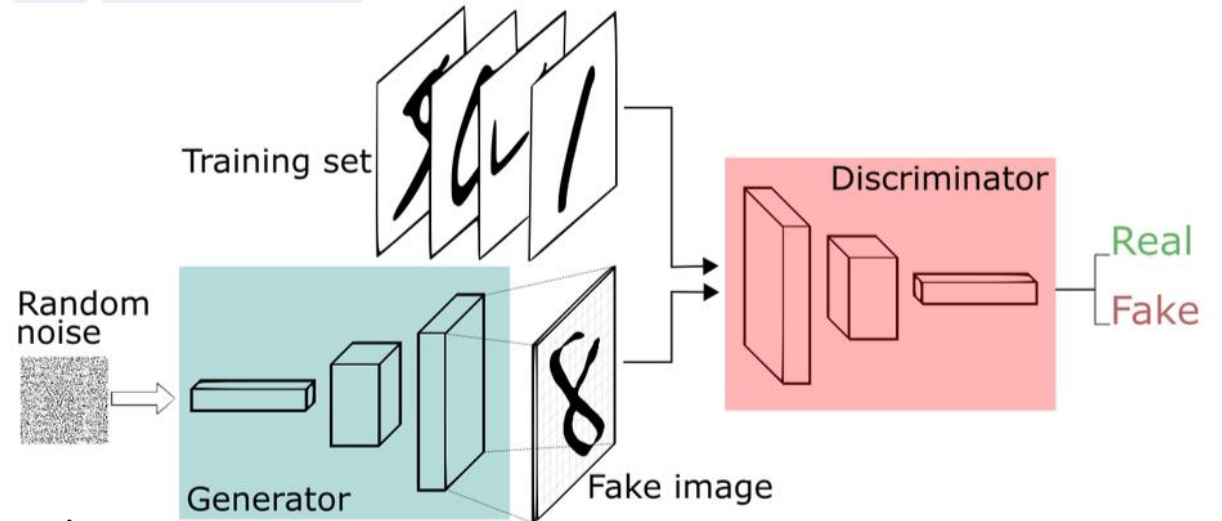
- **Unsupervised learning**, compress (encode) information automatically.
- An encoder is a deterministic mapping that transforms an input vector x into a **hidden representation**, while a decoder maps it back to a reconstructed input z via another mapping g .
- An autoencoder compares the reconstructed input z to the original input x and try to minimize the reconstruction error.
- Once the autoencoder is trained, the decoder is discarded and the encoder output can be used as a feature vector in a supervised learning model, for visualization, or more generally for dimensionality reduction.

Auto Encoder (AE)



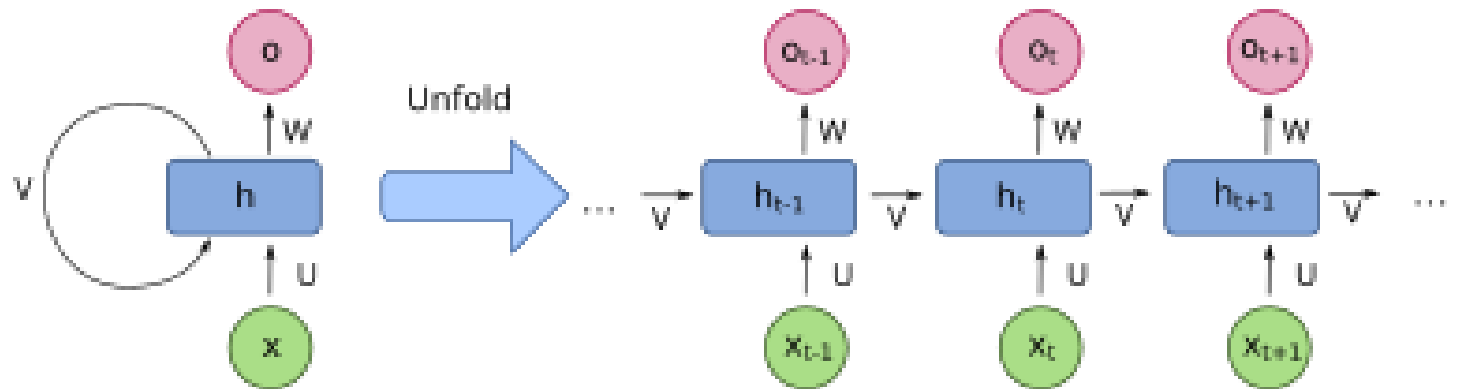
Generative Adversarial Networks (GANs)

- A technique for generating realistic synthetic data, two competing models that “work against” each other.
- The **generator** produces samples close to training samples.
- The **discriminator** (a classifier) tries to distinguish real data from the data created by the generator.
- Trained together until the generator is able to create realistic synthetic data that the discriminator can no longer determine is fake.



From RNNs to Transformers

- RNNs are a class of neural networks that are able to handle **sequential data**.
- A **hidden state** allows them to retain information from previous time steps.
- RNNs operate on sequential data by processing one input at a time, updating their hidden state based on the current and previous hidden states.
- Once unfolded in time, they can be seen as very deep FF networks in which all the layers share the same weights.

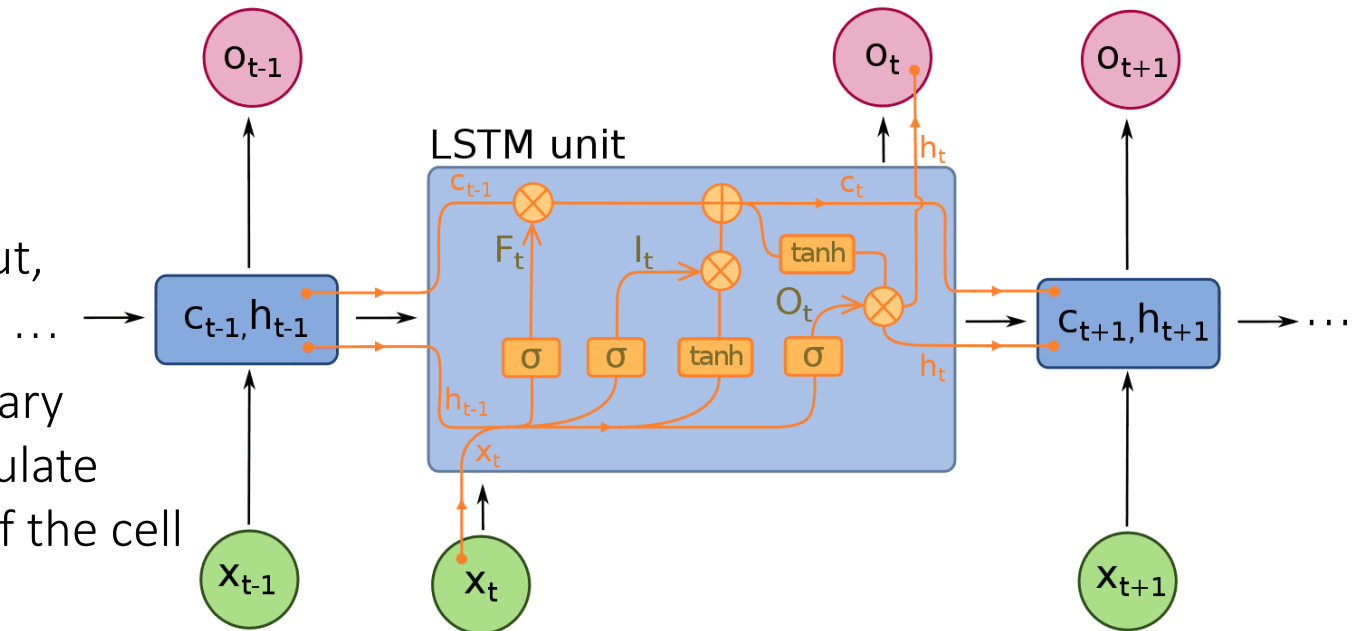


Long/Short Term Memory (LSTM)

- LSTM is a specific type of RNN architecture that addresses the vanishing gradient problem, where the influence of earlier inputs diminishes exponentially as the sequence progresses, making it difficult to capture long-term dependencies.

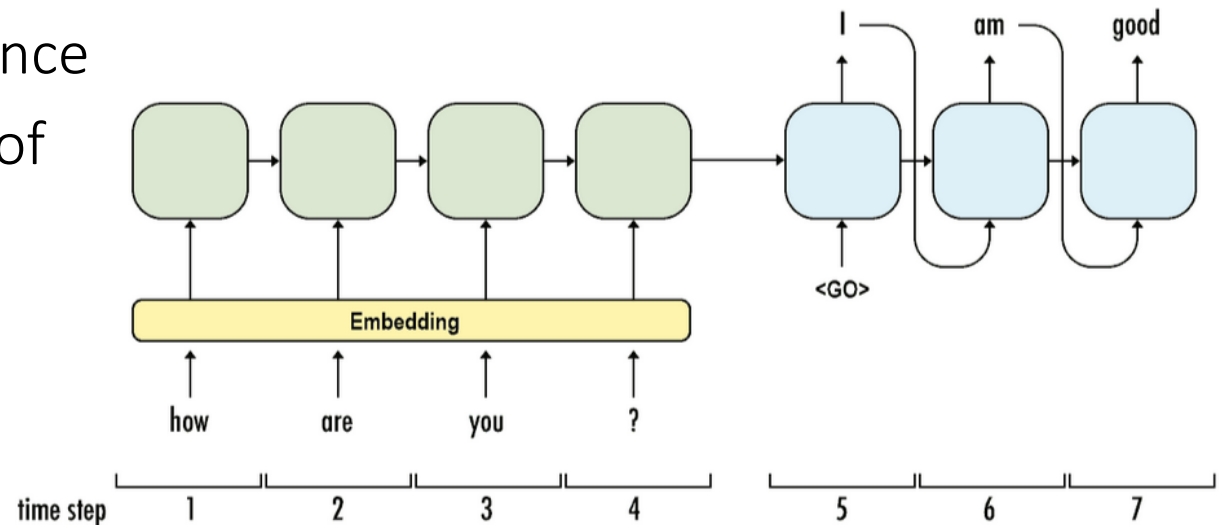
An LSTM unit is made of a cell, an input, an output and a forget gate.

The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell



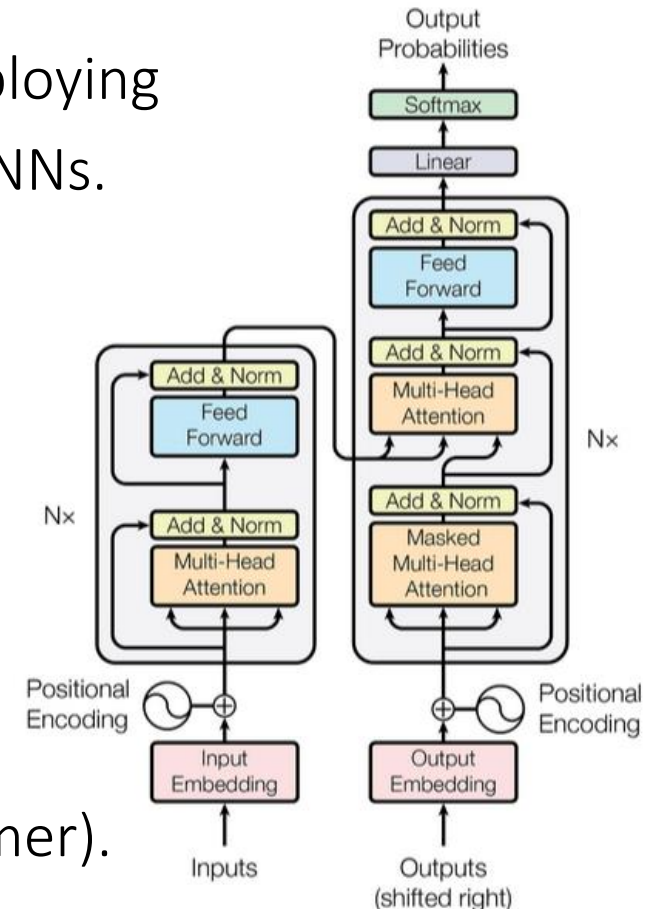
Sequence-to-Sequence model

- RNN/LSTM can be applied on NLP tasks such as machine translation using an encoder-decoder model: the input sequence is encoded into a fixed-length representation (context vector), the decoder generates an output sequence based on the encoded representation.
- Issues arise when the input sequence is quite long, and not every piece of the context is required at every decoding stage.



Transformers

- Transformers consist of encoder and decoder layers, employing multi-head self-attention mechanisms and feed-forward NNs.
- They process the entire input all at once, **self-attention** allows the model to weigh the importance of different input tokens when making predictions, long-range dependencies captured without sequential processing.
- State-of-the-art model in the field of NLP: BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer).



Attention Is All You Need by Vaswani et al. in 2017 <https://arxiv.org/abs/1706.03762>

Deep Learning with PyTorch

- An Open Source framework for Deep Learning (and Machine Learning).
- Why PyTorch?
 - Pythonic Nature: follows standard Python conventions
 - Easy to learn: intuitive syntax and similar to NumPy
 - Strong Community: find help on <https://discuss.pytorch.org>



several others ...

https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#what-is-pytorch

Deep Learning with PyTorch

- Deep Learning:
 - Mathematical computing on Multidimensional Arrays (aka Tensors)
 - Highly benefit from Parallel Computing
- PyTorch let us easily exploit the **parallelism offered by GPUs**
 - Strong performance
 - **Automatic Differentiation**
 - High flexibility



https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#what-is-pytorch

Tensors

- Tensors are the PyTorch counterpart of NumPy arrays
- Contain only numerical values, used to encode:
 - **Signal to process** (e.g. images, strings of text, videos, ...)
 - **Internal states and parameters** of neural networks
- All of PyTorch computation takes place on Tensors
- PyTorch provides simple methods to transfer tensors between CPU and GPU(s)



```
#           R     G     B
img = tensor([[[ 48,  80,  79],
                [175, 104, 207],
                [162,  24, 224],
                [ 97,  27,  28],
                [ 51, 137,  60],
                [124, 214, 249]],
              ...
              ]])
t.size() ==> [256, 256, 3]
t.device ==> gpu:0
t.dtype ==> torch.float32
```


Neural networks and Backpropagation

- Define the neural network architecture
- While training (repeat until convergence):
 - **Forward pass**: feed input data to the network to obtain the network prediction
 - **Compute loss**: compare network prediction with ground truth
 - **Backward pass**: compute the gradients of network parameters w.r.t to the loss function (PyTorch **Autograd**)
 - **Update** the network parameters (PyTorch **Optimizer**)