Artificial Intelligent Systems

(BMA-EL-IZB-LJ-RE 1. year 2024/2025)

Artificial Neural Networks

Simon Dobrišek

Lecture Topics

- Knowledge from experimental data
- Artificial neural networks
- Feedforward multilayer perceptron
- Convolutional neural networks
- Relation to Bayesian networks
- Applications of artificial neural networks

Knowledge from experimental data

- Data-driven knowledge discovery is based on the analysis of empirical/experimental data or the input-output behaviour of a real system without an explicitly given theory.
- Learning from experimental data is building a computational model that maps the input data to the output data.
- Such approaches are based on searching for regularities and patterns in data.
- For example, input data may be sensorial data and output data the identity of the perceived objects or even the identity of the action that should be taken when the object is perceived.

KNOWLEDGE FROM EXPERIMENTAL DATA

- The precise form of the function that maps the input data to output data is determined during the training/learning phase on the basis of the training data.
- Once the model is trained it can map new input data to output data that are said to comprise a test set.
- The ability to map correctly new data that differ from those used for training is known as generalization.
- Applications in which the training data comprises examples of the input data along with their corresponding target data are known as supervised learning problems.
- For most practical applications, the original input data are typically preprocessed to transform them in to some new space where, it is hoped, the problem is easier to solve - this pre-processing stage is also called feature extraction.

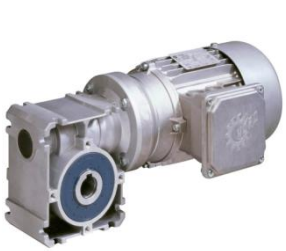
KNOWLEDGE FROM EXPERIMENTAL DATA

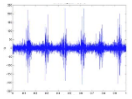
- Cases in which the aim is to assign each input data to one of a finite number of discrete categories are called the *classification* problems.
- If the desired output consists of one or more continuous variables, then the task is called *regression*.
- In other problems, the training data consist of input data without any corresponding target data.
- The goal in such unsupervised learning problems may be
  - to discover groups of similar examples within the data, known as *clustering*,
  - or to determine the distribution of the data within the input space, known as *density estimation*,
  - or to project the data from a high-dimensional space down to less dimensions for the purpose of better data representation.

Knowledge from experimental data

- In the field of *pattern recognition* and *data mining,* many different computational models have been developed for solving the mentioned tasks.
- The models range from the polynomial and probabilistic decision functions, decision trees, as well as non-parametric methods based on nearest neighbor rules and similarity measures.
- One of the most successful biology-inspired pattern recognition models are based on simulating biological neural systems.
- Such models are know as artificial neural networks and can be seen as artificial representations of the human brain that try to simulate its learning abilities.

Pattern Recognition System





Sensors

Samples
f(x)

Feature extraction

0.9165
0.4326
0.4270
0.0293
0.5104
0.9637
0.2754
0.7434
0.1168
0.7693

Features / attributes
$\{x_1, \ldots, x_n\}$

2

1

Classification

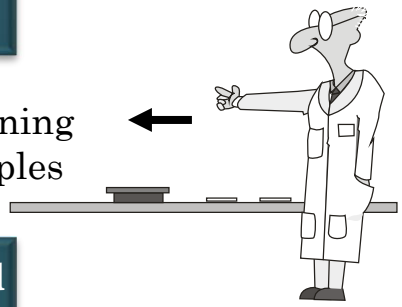
"Gear malfunction"

Prototypes
Decision functions
Decision trees
Neural networks
Knowledge repr. forms

Supervised learning

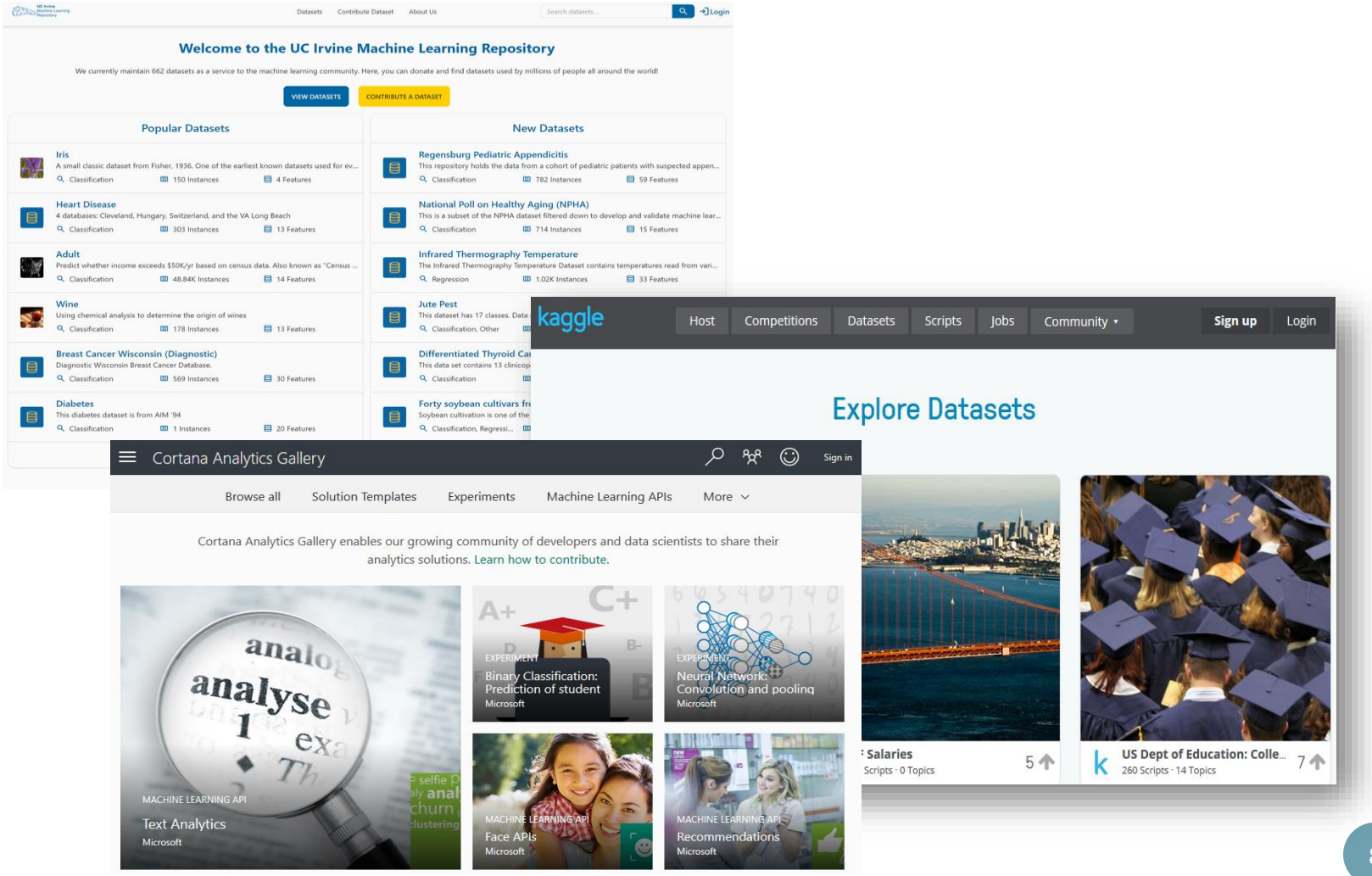
Unsupervised learning

Training samples



Experimental/Empirical Data

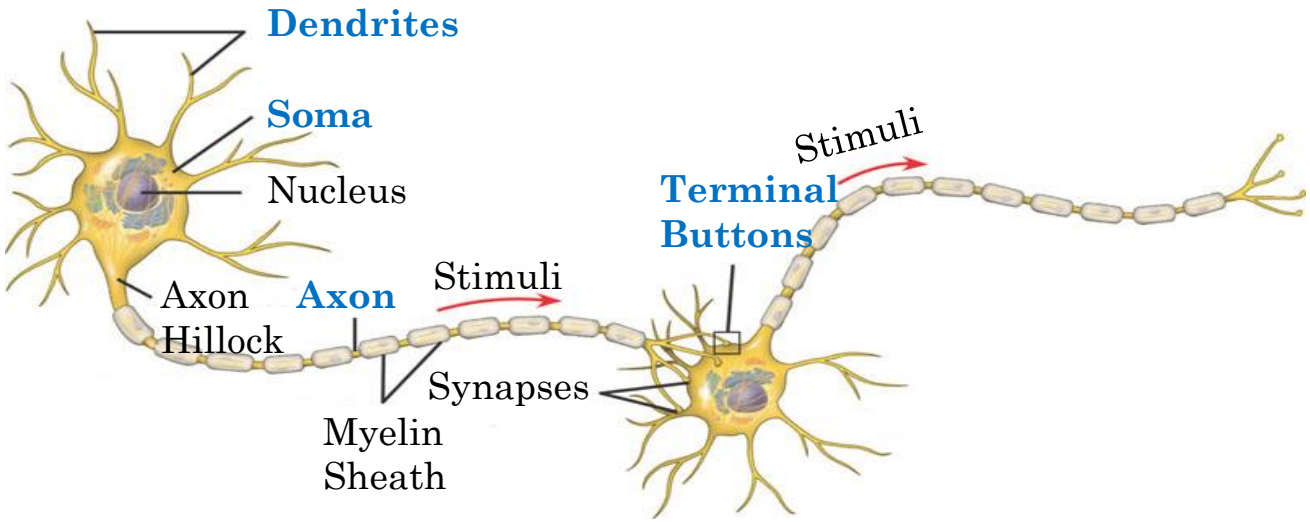
- Many different kinds of empirical/experimental data is available for the research in this field



Artificial Neural Networks

- Artificial neural networks (ANNs) model the natural biological neural systems
- They are based on the parallel processing of data
- They have the ability to learn and adapt
- With a very simple processing elements they form a very complex behaviour
- They are a powerful system for solving pattern recognition and other artificial intelligence problems

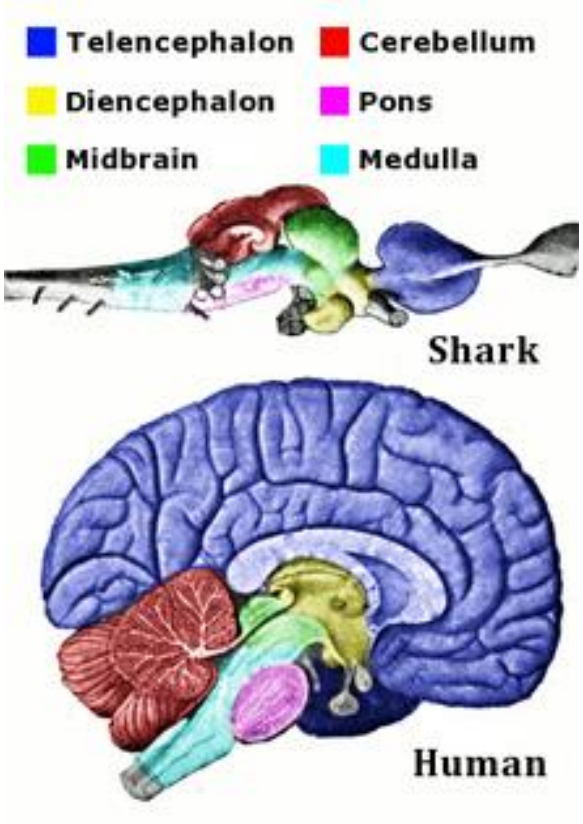
Model of a Neural System

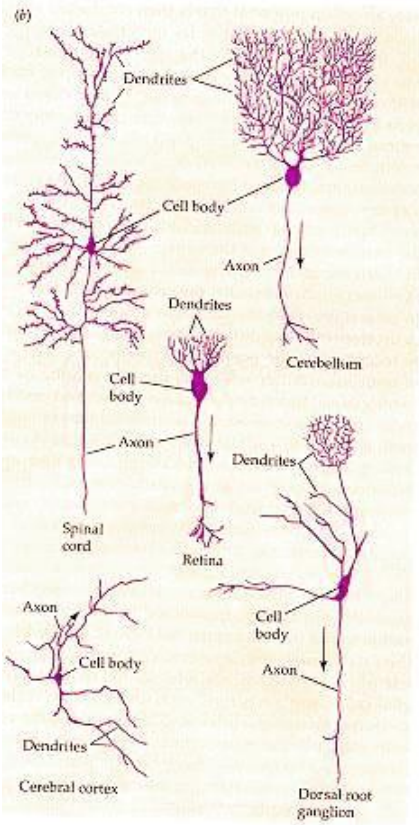


- A neural system is composed of simple processing elements (neurons) that are intricately connected with one another.
- A neuron is composed of input dendrites, the neuron body and the output axons.
- Stimuli from the axon of one neural cell travel through the synapses to the dendrites of other cells.

Neural Systems of Vertebrates

- Neuroanatomists divide the brains of vertebrates into the six main regions.
- Different neuronal/neural cells have evolved in various parts of the human brain and the spinal cord.





Comparison of brain and computer

- The key difference between computers and the human brain is in the way of processing data

	Processing units	Size of basic units	Energy consumption	Processing speed	Processing method	Fault tolerance	Learning	Intelligence and consciousness
	10^{11} Cells 10^{14} Synapses	10^{-6} m	30 W	100 Hz	Parallel Distributed	Yes	Yes	Usually
	10^{10} Transistors	10^{-8} m	50-200W	10^{9} Hz	Sequential Centralized Multi-cores	Little	Little	Little and not yet

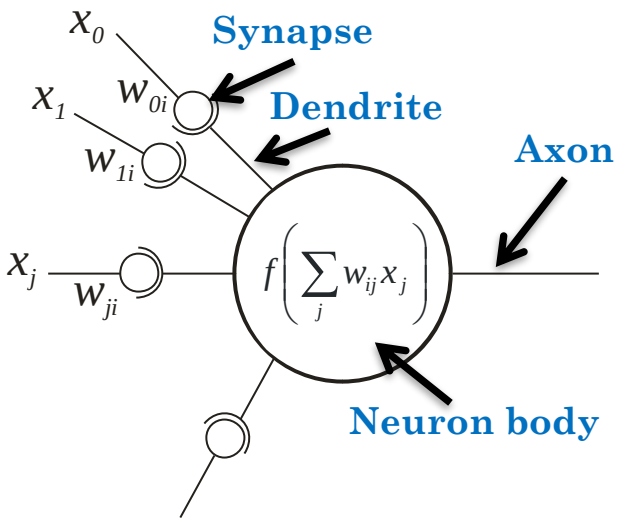
- During resting/sleeping the human brain require about 20% of daily energy intake (usually about 1,500 kilocalories), i.e.,

$$1500\ \frac{kcal}{day} = 62{,}5\ \frac{kcal}{hour} = 17{,}4\frac{cal}{sec} = 72{,}4\ \frac{jouls}{sec} = 62\ W$$

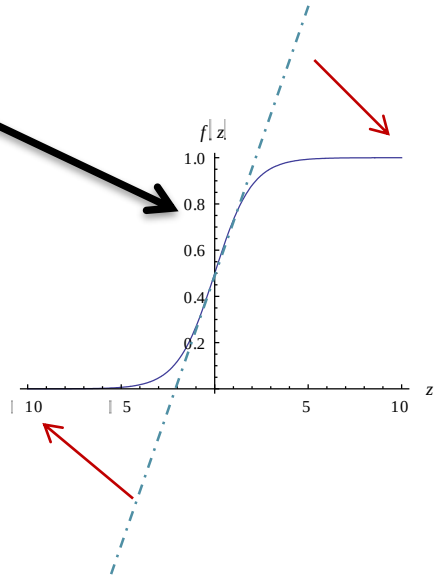
$$62W \cdot 20\% = 12{,}5W$$

McCulloch-Pitts' Model of Neuron

- Output stimuli is a "saturated" linear function of input stimuli.

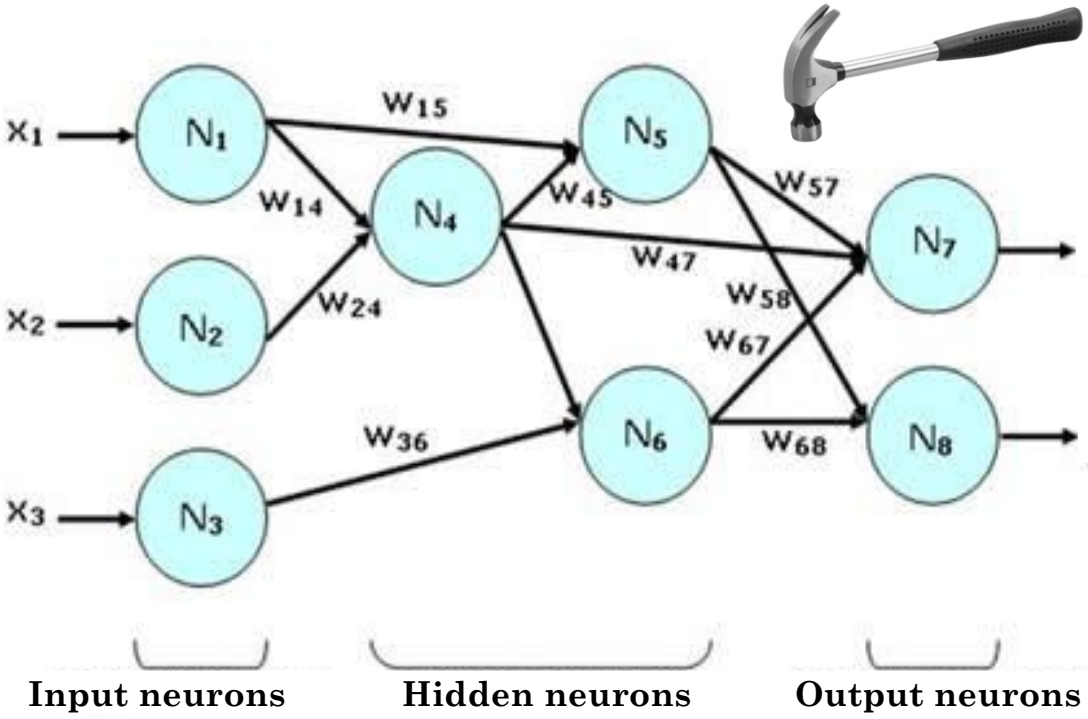


$$o_i = f\left(\sum_j w_{ij} x_j\right)$$



- This model is a simplification that allows the development of a simple mathematical model of artificial neural networks.

Artificial Neural Network



$y_1 = f_1(x_1, x_2, x_3)$

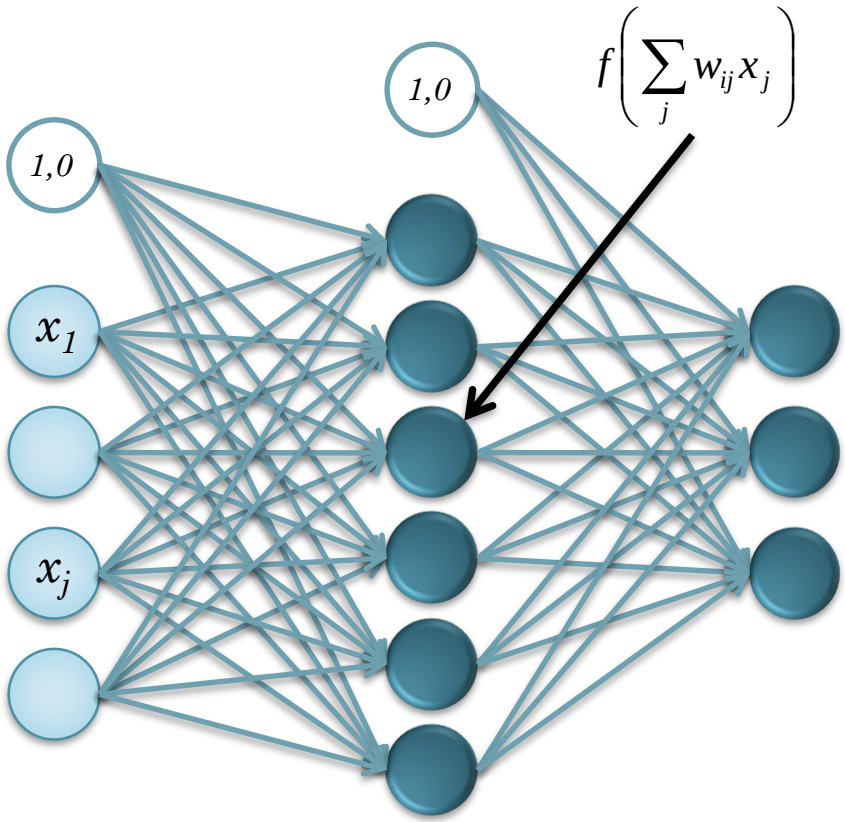
$y_2 = f_2(x_1, x_2, x_3)$

A multivariate function model of a neural network.

Types of artificial neural networks

- Multilayer feedforward neural networks.
- Self-organizing neural networks
- Recurrent neural networks.
- Hopfield, Boltzmann, and Hamming neural networks
- Hierarchical Bayesian network
- Long short-term memory recurrent neural networks
- Stacked auto-encoders
- Convolutional deep neural networks
- …

Feedforward multilayer perceptron



$$y_k(\mathbf{x}) = f\left(\sum_i w_{ki} f\left(\sum_j w_{ij} x_j\right)\right)$$

Mapping of random vector variables

$$g : X \rightarrow Y$$

- The network outputs are non-linear functions of linear combinations of nonlinear functions of linear combinations of …

Compact representation of neuron's function

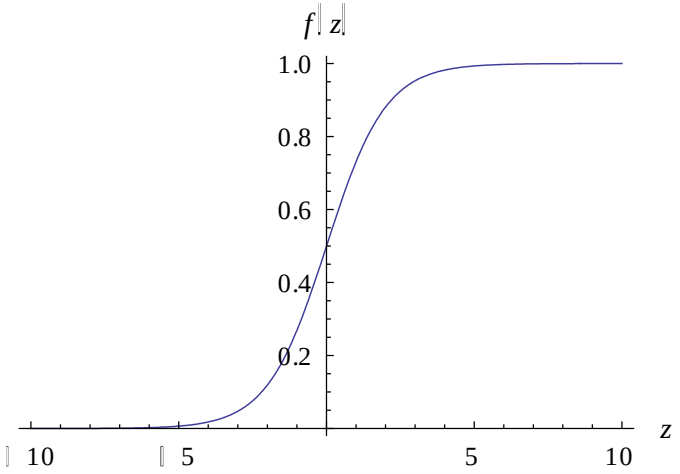
- The function of the i-th neuron can be given in a compact vector form as follows:

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_j \\ \vdots \end{bmatrix} \qquad \mathbf{w}_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{ij} \\ \vdots \end{bmatrix} \qquad z_i = \mathbf{w}_i^T \mathbf{x} = \sum_j w_{ij} x_j$$

$$f(z_i) = f(\mathbf{w}_i^T \mathbf{x}) = f\left(\sum_j w_{ij} x_j\right)$$

Activation functions

- The nonlinear activation function is normally monotonically increasing and differentiable, such as the sigmoidal function.

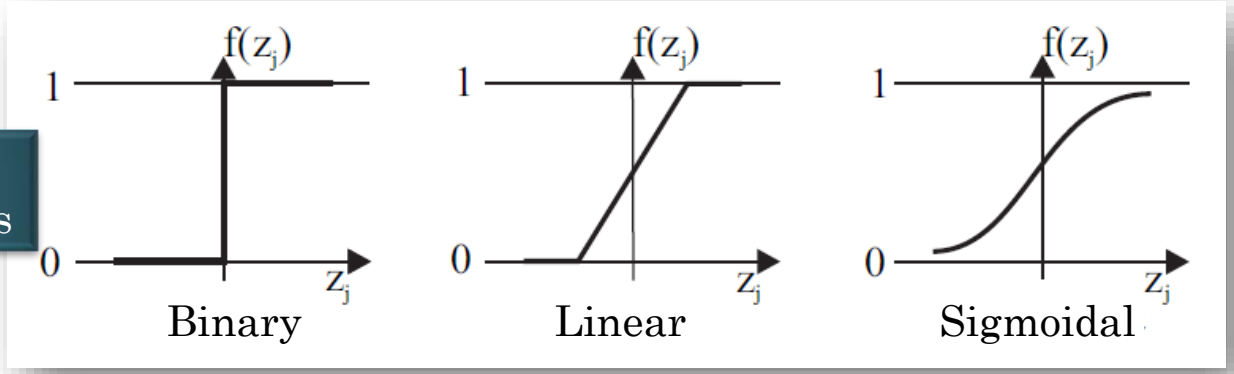


$$f(z) = \frac{1}{1+e^{-z}}$$

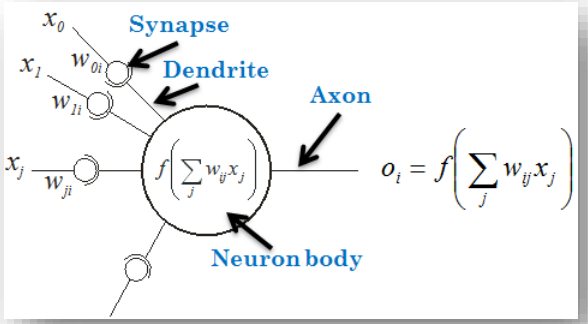
$$f'(z) = f(z)(1-f(z))$$

$$f(z_j) = f(\mathbf{w}_j^T \mathbf{x}) = \frac{1}{1+e^{\mathbf{w}_j^T \mathbf{x}}}$$

Examples of activation functions



PROPERTIES OF THE NEURON MODEL



- Synapses can be excitatory (+) and inhibitory (-).
- Neuron receives the weighted stimuli from other neurons, it integrates them and generates the output stimulus that travels along the axon from the cell body.
- The generation of the stimulus in the cell body of a neuron is called “ignition”.
- When a neuron ignites (triggers/fires) depends on the number and level of its excitatory and inhibitory inputs and its threshold.
- The higher is the threshold of a neuron, the higher excitation stimuli is needed to ignite.

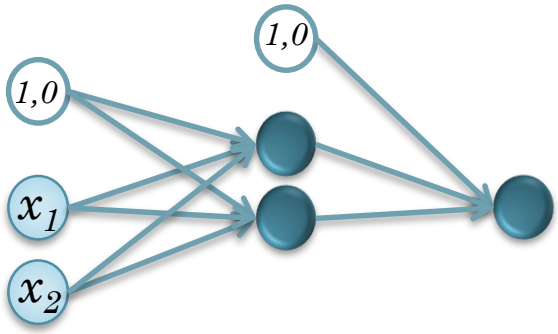
Neural Networks as Multivariate Function Approximations

- A neural network can be seen as a parametric multivariate function approximation, where the parameters are the synapse weight coefficients of the neurons.

$$y_k(\mathbf{x}) = f\left(\cdots + w_{ki} f(\mathbf{w}_i, \mathbf{x}) + \cdots + w_{k2} f(\mathbf{w}_2, \mathbf{x}) + w_{k1} f(\mathbf{w}_1, \mathbf{x}) + w_{k0}\right)$$

- The function being modelled is not known - available is only a limited number of training samples in the input space.
- As the input data is considered to comprise noise the approximation problem can also be seen as a multivariate regression problem.

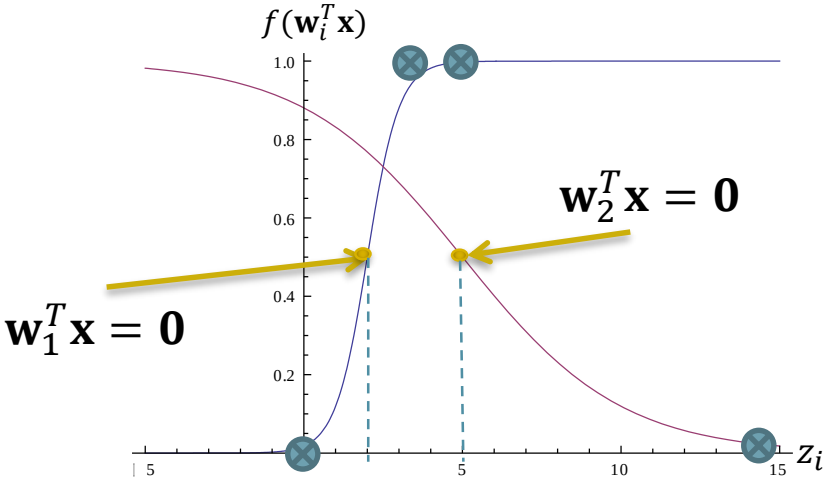
EXAMPLE



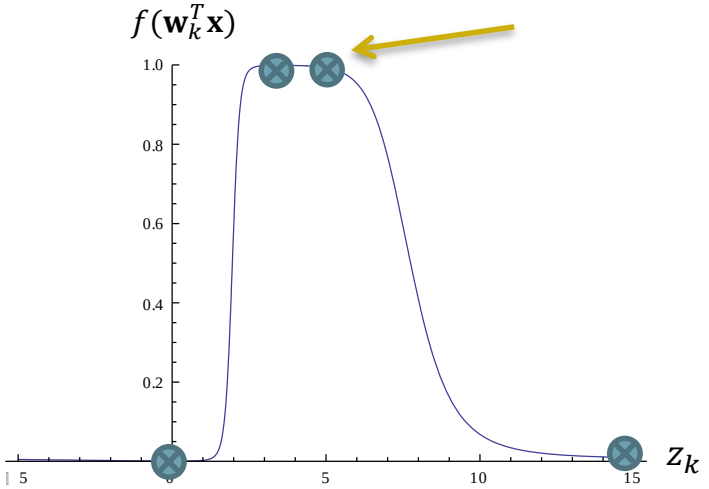
- A decision boundary of a neuron is defined at

$$\mathbf{w}^T\mathbf{x} = \mathbf{0} \implies f(\mathbf{w}^T\mathbf{x}) = 0.5$$

- The initial position of the decision boundaries of all the neurons in a neural network can be defined using some suitable unsupervised clustering technique.



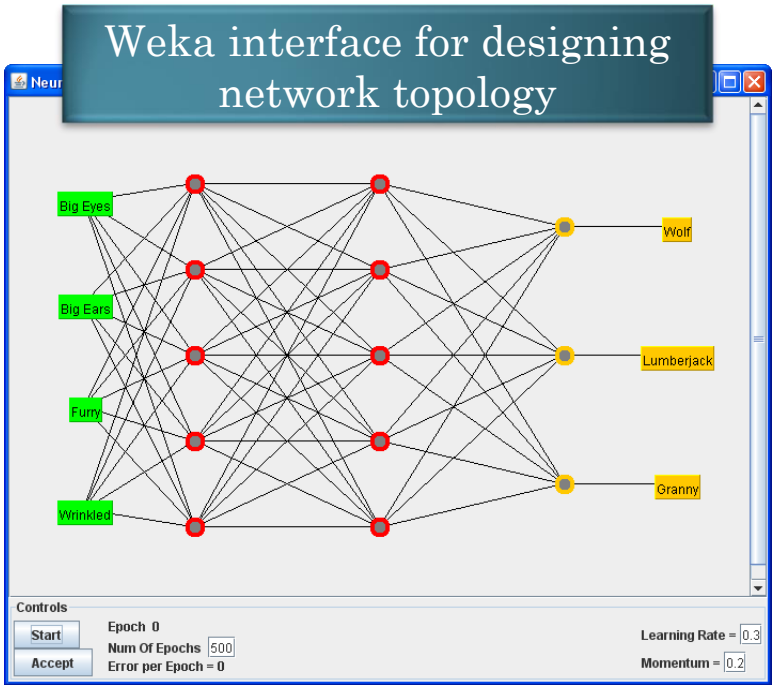
$\mathbf{w}_1^T = [2\,, -4, 0]$ $\mathbf{w}_2^T = [-0{,}4\,, -2]$

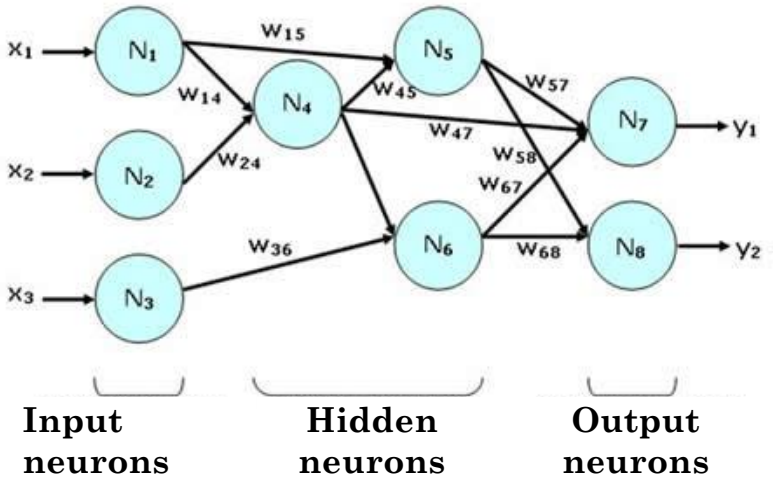


$\mathbf{w}_k^T = [20\,, 20\,, 25]$

Neural Network Structure

- The structure of a neural network is flexible and depends on the given task.
- Its structure is usually designed be the user, and the most common is the multilayered structure.

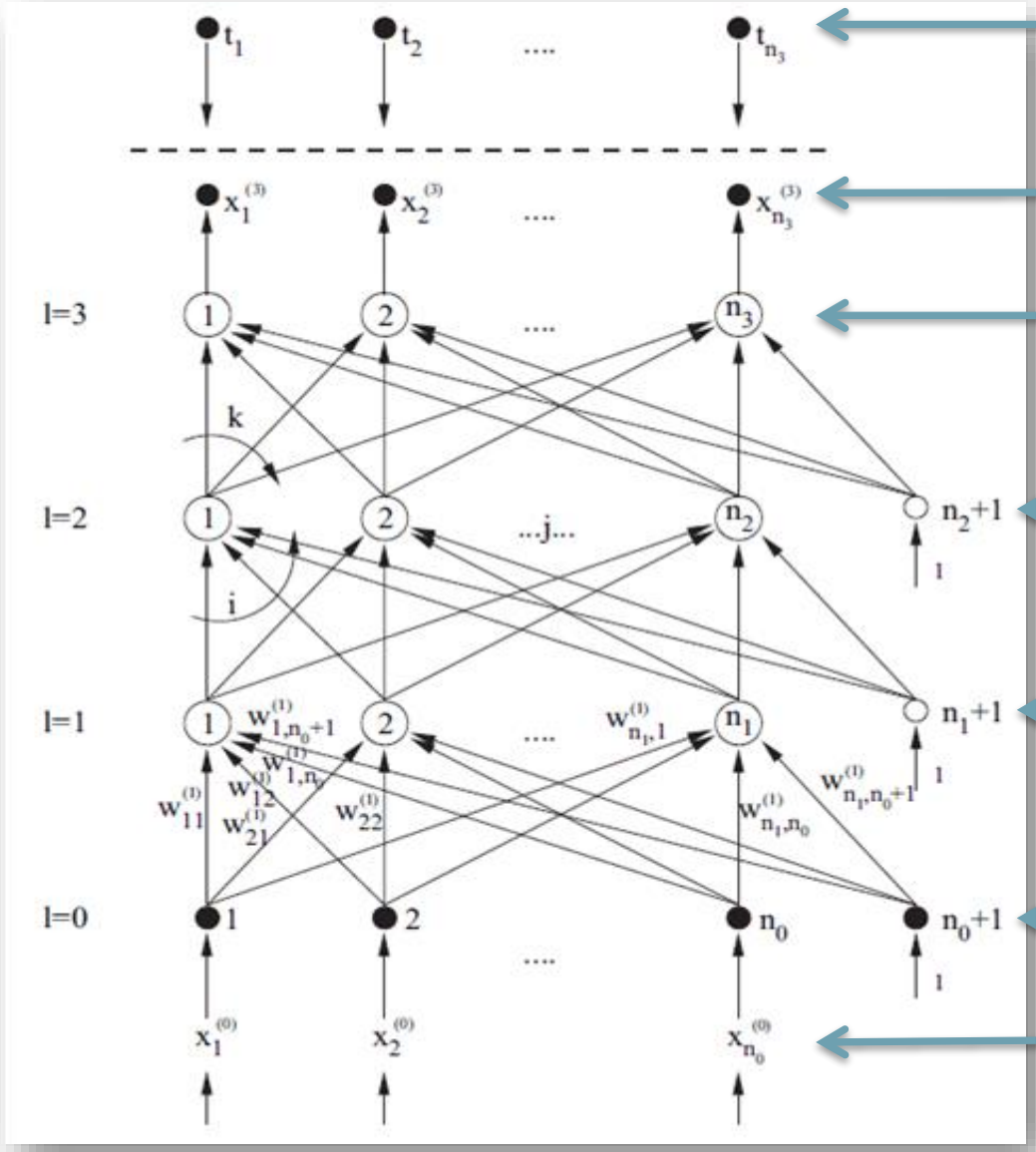




Multilayer Perceptrons

- A multilayer perceptron (MLP) consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one.
- The multilayer perceptron consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes.
- The input layer is layer 0, and when we talk of an N-layer network we mean there are N-layers of weights and N-non-input layers of processing units.
- Multilayer perceptrons using a backpropagation training algorithm are the standard algorithm for any supervised learning pattern recognition process.

Three-layer Perceptron



Target outputs

Network outputs

Output layer

Second hidden layer

First hidden layer

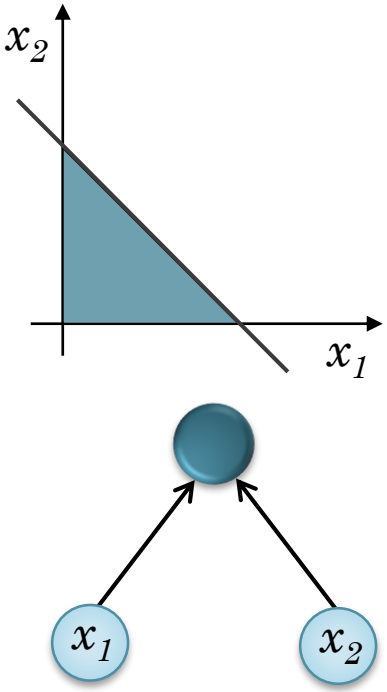
Zeroth input layer

Input feature vector

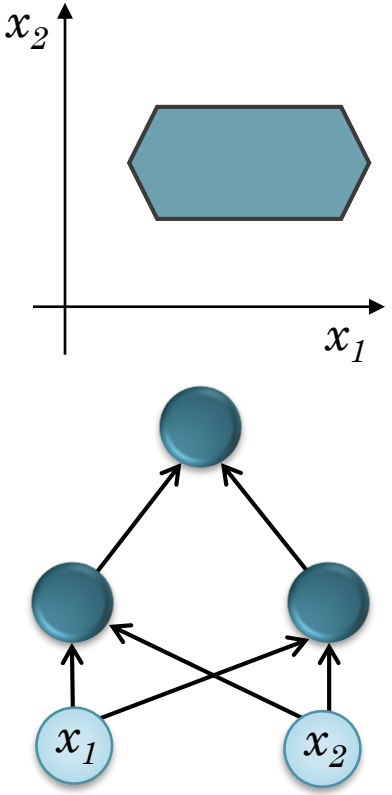
Multilayer Perceptron Structure

- The best number of hidden layers and hidden units of a MLP depends on many factors, including:
  - The numbers of input and output units
  - The complexity of the function or classification to be learned
  - The amount of noise in the training data
  - The number and distribution of training data patterns
  - The type of hidden unit connectivity and activation functions
  - The training algorithm used

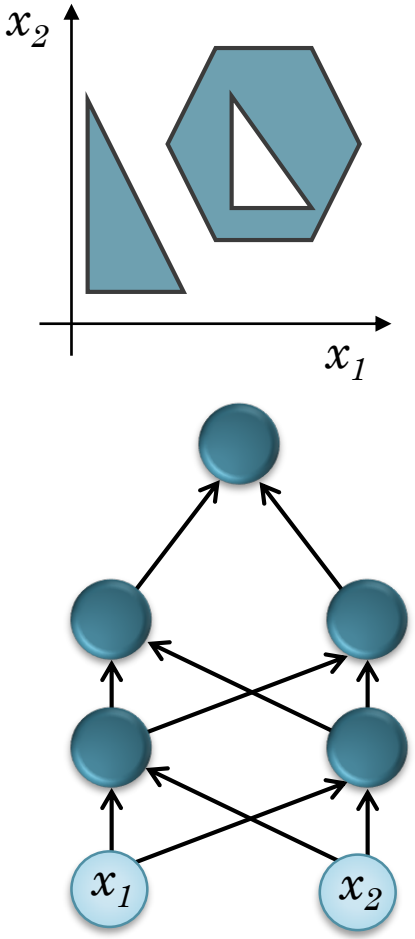
The role of MLP layers



1st layer defines linear boundaries

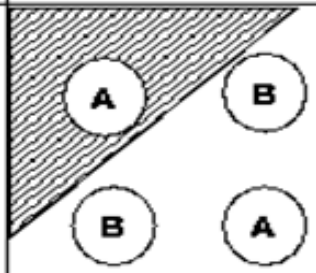
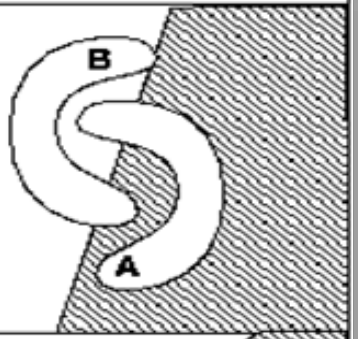
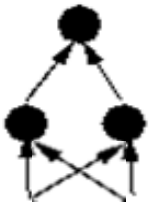
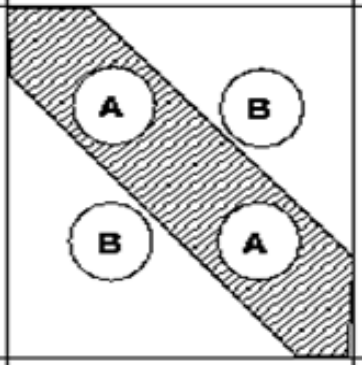
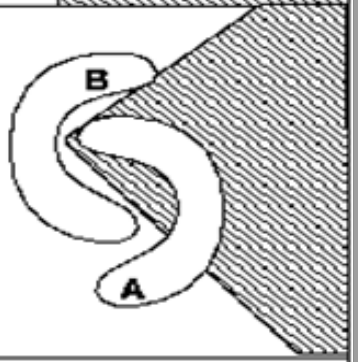
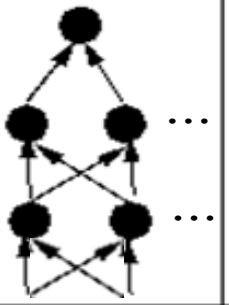
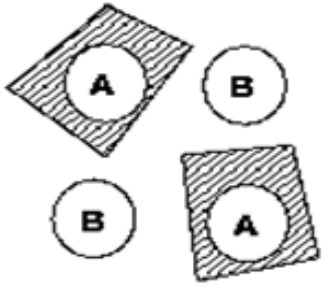
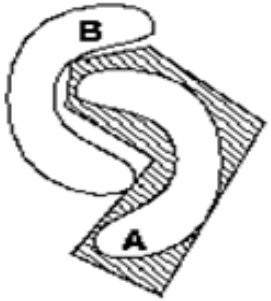


2nd layer combines the boundaries



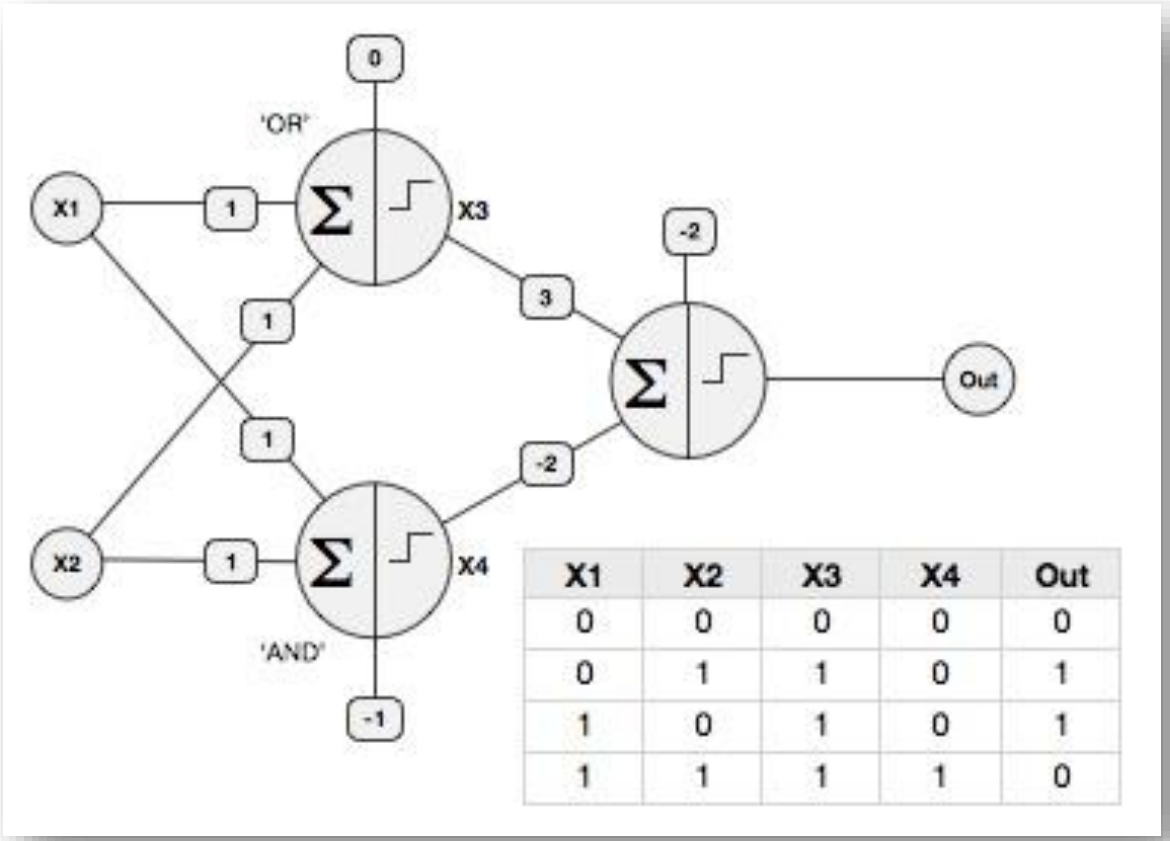
3rd layer can generate arbitrarily complex boundaries

THE ROLE OF MLP LAYERS

Structure	Regions	XOR	Meshed regions
single layer 	Half plane bounded by hyper-plane		
two layer 	Convex open or closed regions		
three layer 	Arbitrary (limited by No. of nodes)		

EXAMPLE

- A two-layer perceptron with the binary activation function and weights that model the XOR function.



X1	X2	X3	X4	Out
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Learning in Multi-Layer Perceptrons

- Weights (and thresholds) of the MLP neurons are determined using the error back-propagation algorithm and using an appropriate cost (loss) function.
- The most common training criteria, i.e., the cost function, is a sum squared error between the target and the actual outputs of the network given the input vectors.
- Other costs functions such as absolute loss and cross-entropy loss are used as well.
- The training algorithm is based on gradient descent on the error function of the network over the weights and thresholds of the MLP neurons.

Learning in Multi-Layer Perceptrons

- The training algorithm assigns the 'credit' or 'blame' to individual neurons involved in forming the overall response of the network, and alters their weights to bring the network closer to the desired behavior (by minimizing the cost/loss function).

- The weights $\mathbf{w}_j^{(l)}$ of the j-th neuron in the l-th layer of a MLP is adjusted for the given input vector $\mathbf{x}$ in the direction of reducing the network outputs error e, i.e.,

$$\mathbf{w}_j^{(l)}(k+1) = \mathbf{w}_j^{(l)}(k) - \beta \left(\frac{\partial e(\mathbf{w}, \mathbf{x})}{\partial \mathbf{w}} \right)_{\mathbf{w}=\mathbf{w}_j^{(l)}(k)}$$

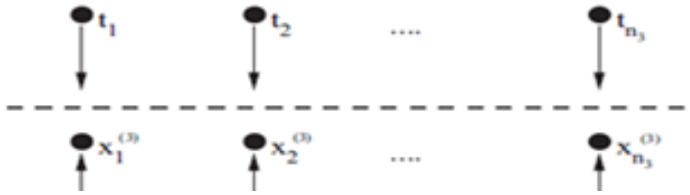
- The rule for modifying the weights $\mathbf{w}_j^{(l)}$ of the j-th neuron in the l-th layer is then shortly denoted as

$$\mathbf{w}_j^{(l)}(k+1) = \mathbf{w}_j^{(l)}(k) + \Delta\mathbf{w}_j^{(l)}(k)$$

Learning in Multi-Layer Perceptrons

- The weights update $\Delta \mathbf{w}_j^{(l)}(k)$ is obtained by taking take the partial derivative of the error function $e(\mathbf{w}(k), \mathbf{x})$ with respect to the neuron weights $\mathbf{w}(k)$ given the input vector $\mathbf{x}$.

- When the sum squared error (SSE) is used for the loss/cost function



$$e(\mathbf{w}(k), \mathbf{x}) = \frac{1}{2} \sum_{\boldsymbol{p}} \left(t_p(k) - x_p^{(3)}(k) \right)^2$$

where $t_p(k)$ denote the target output values and $x_p^{(3)}(k)$ the actual output values of the p-th neuron of the output layer ($l = 3$) of a tree-layer MLP, then the weights updates $\Delta \mathbf{w}_j^{(l)}(k)$ can be derived and are defined as follows.

Learning in Multi-Layer Perceptrons

- The weight updates of the j-th neuron of the output layer ($l = 3$) given the input vector $\mathbf{x}(k)$ in the k-th step is:

$$\Delta w_{ji}^{(3)}(k) = \beta d_j^{(3)}(k) x_i^{(2)}(k); \qquad i = 1, 2, \ldots, (n_2 + 1), \quad j = 1, 2, \ldots, n_3.$$

where for the sigmoid activation function

$$d_j^{(3)}(k) = \left(t_j(k) - x_j^{(3)}(k)\right)\left(1 - x_j^{(3)}(k)\right) x_j^{(3)}(k).$$

- The weight updates of the j-th neuron of the hidden layers ($l = 2, 1$) given the input vector $\mathbf{x}(k)$ in the k-th step is :

$$\Delta w_{ji}^{(l)}(k) = \beta d_j^{(l)}(k) x_i^{(l-1)}(k); \qquad j = 1, 2, \ldots, n_l, \quad i = 1, 2, \ldots, (n_{l-1} + 1), \quad l = 2, 1$$

where

$$d_j^{(l)}(k) = \left(1 - x_j^{(l)}(k)\right) x_j^{(l)}(k) \sum_{p=1}^{n_{l+1}} d_p^{(l+1)}(k) w_{pj}^{(l+1)}(k).$$

Choosing Appropriate Cost function and Activation Function

- The selection of the cost function for the network outputs and the activation function of the neurons depend on the given task:
  - Regression/ Function Approximation Problems

    Sum Squared Error (SSE) cost function,
    linear output activations, sigmoid hidden activations

  - Classification Problems (2 classes, 1 output)

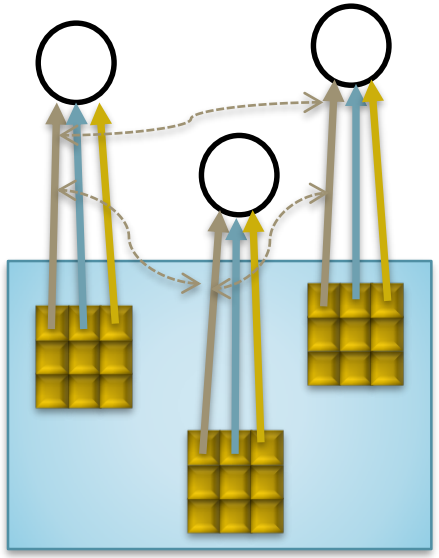
    Cross-Entropy (CE) cost function,
    sigmoid output and hidden activations

  - Classification Problems (multiple-classes, 1 output per class)

    Cross-Entropy (CE) cost function,
    softmax sigmoid outputs, sigmoid hidden activations

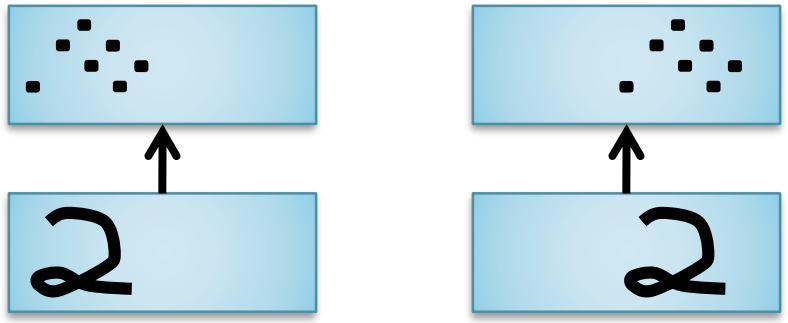
Example: Convolutional Neural Networks

- In recent years in the field of image recognition, a considerable progress has been made using the so-called convolutional neural networks (Hinton, Y. LeCun).
- Neurons in the layers of convolutional networks share the same weights, regardless of to which neurons in the previous layer are connected.
- In image recognition neural networks, each neuron of a convolutional layer is connected to several neurons in a square filed in the previous layer and share their weight with all other neurons in their layer.
- It's easy to modify the backpropagation algorithm to incorporate linear constraints between the weights



Example: Convolutional Neural Networks

- Convolutional neural networks use many different copies of the same feature detector with different positions.
- Replication of the detector greatly reduces the number of free parameters to be learned.
- They are based on using several different feature types, each with its own map of replicated detectors.
- Replicated features do not make the neural activities invariant to translation - the activities are equivariant.



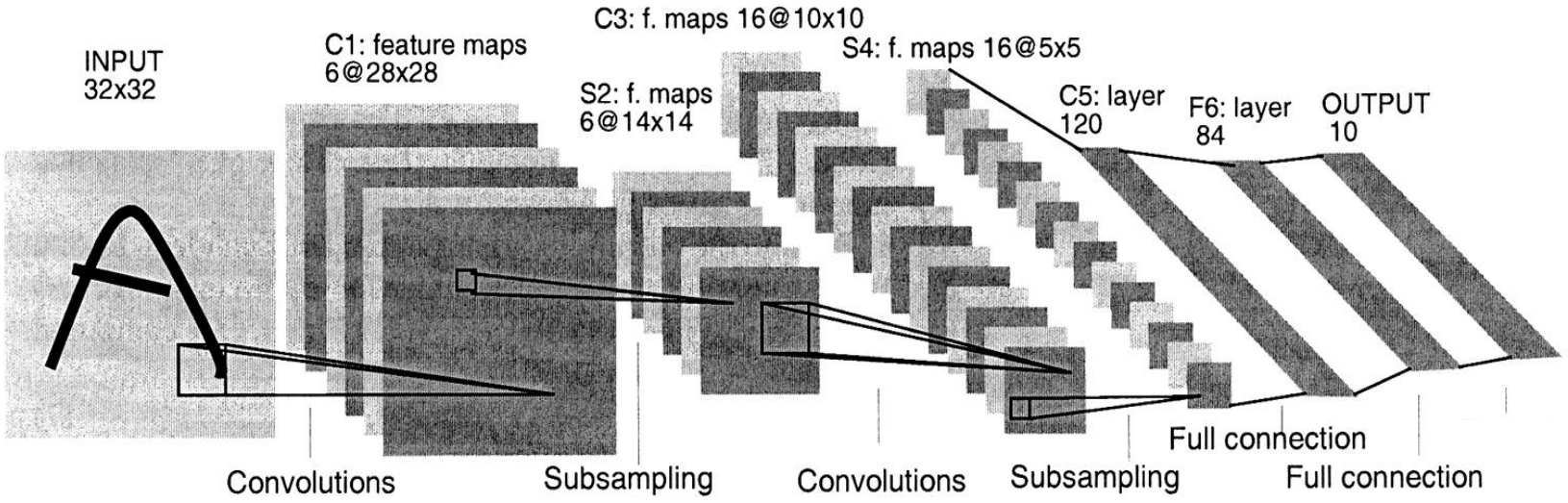
- If a feature is useful in some locations during training, detectors for that feature will be available in all locations during testing.

Example: Convolutional Neural Networks

- A small amount of translational invariance can be obtained by averaging the neighboring replicated detectors to give a single output to the next level.
- This reduces the number of inputs to the next layer of feature extraction, thus allowing us to have many more different feature maps.
- After several interchanging levels of convolutional feature detectors and pooling, the information about the precise positions of objects on the input image is lost.
- The highest layer of such a network is usually a common feed-forward multilayer neural network, in which each neuron in a given layer is connected to each neuron of the previous layer.

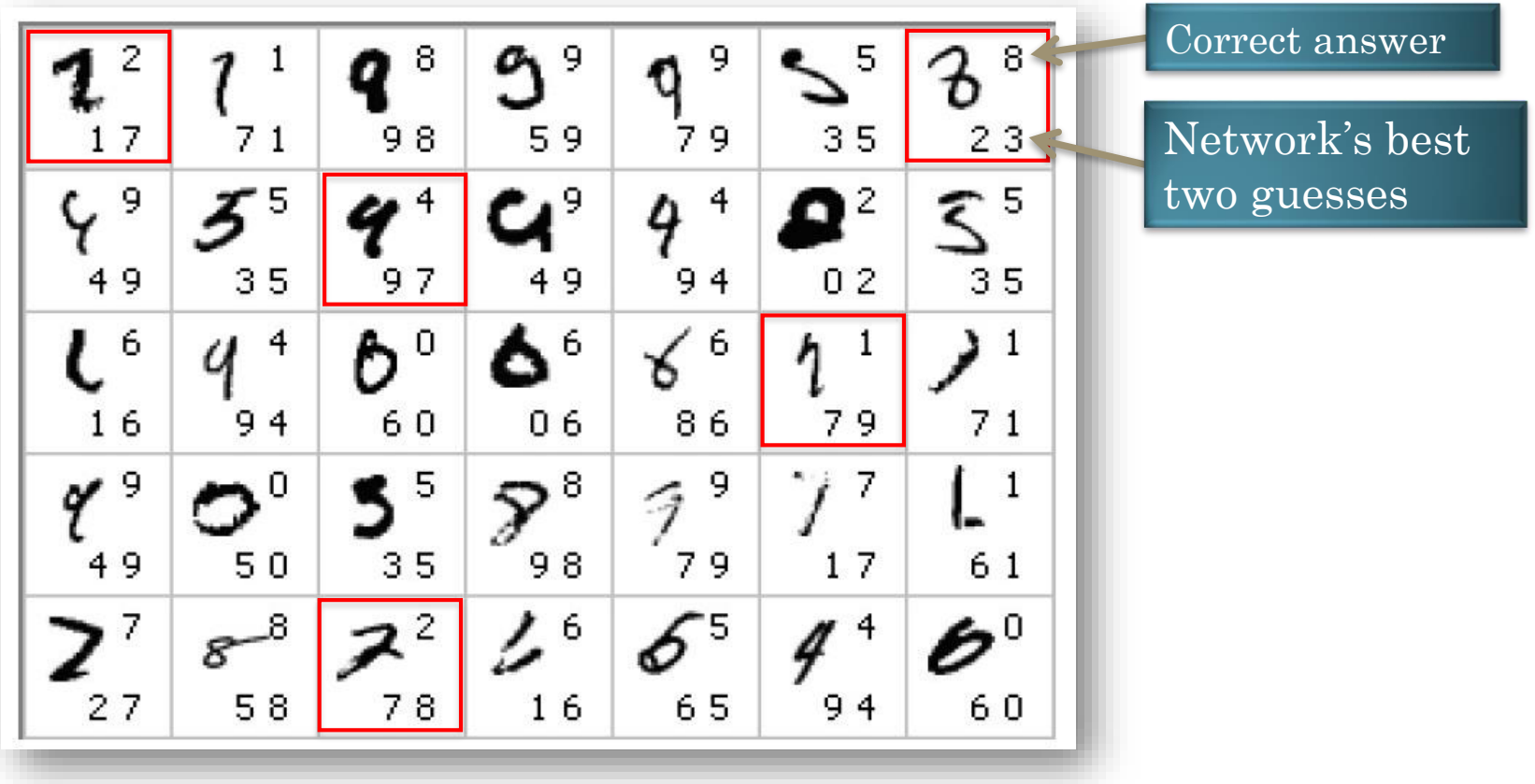
Example: Convolutional Neural Networks

- Depth of the convolutional neural networks depends on the complexity of the problem of automatic (image) pattern recognition task, the amount of the training data and the computing resources available.
- Yann LeCun and his collaborators developed a very reliable recognizer for handwritten digits by using backpropagation in a feedforward convolutional network.



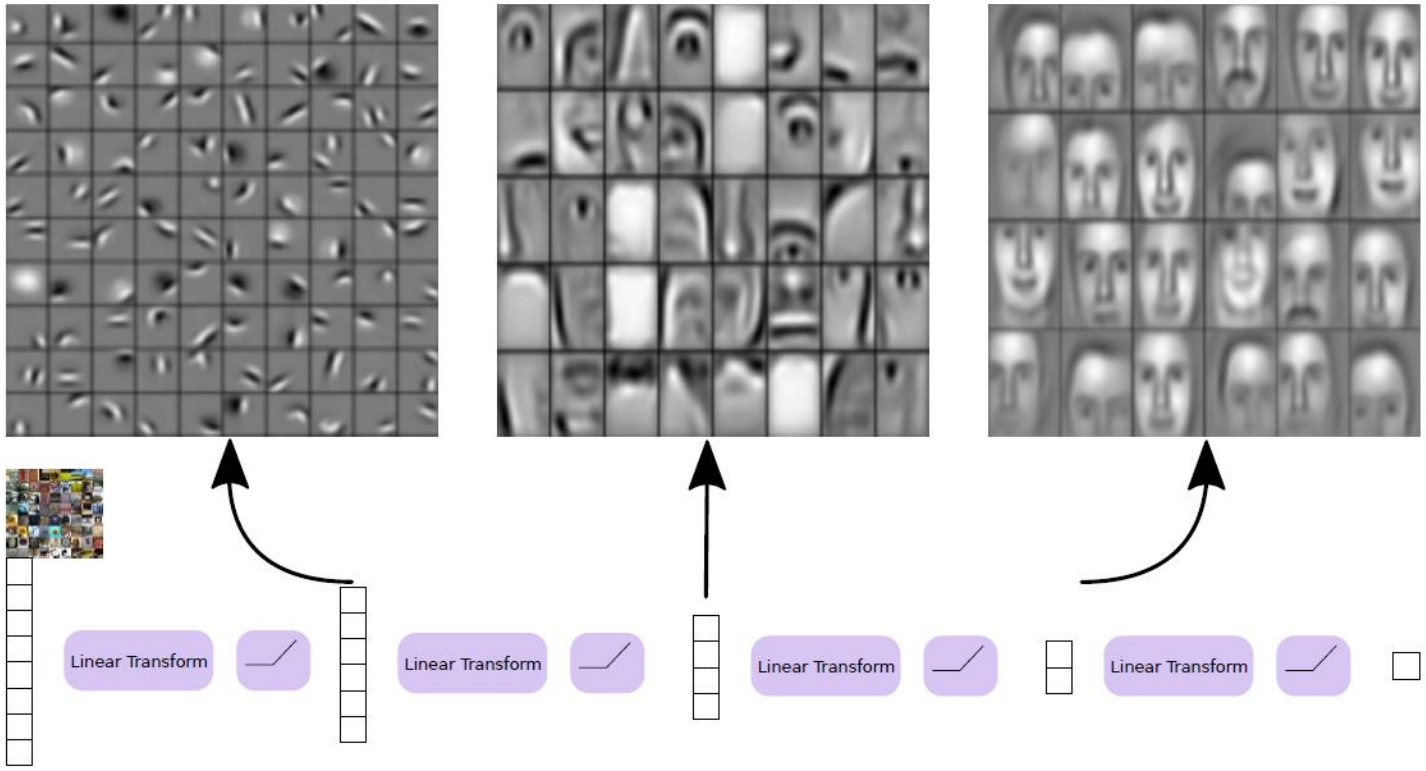
Example Results

- In handwritten numbers recognition (10,000 examples), the convolutional neural network made only a few individual errors.



Example: Convolutional Neural Networks

- Convolutional neural networks have been successfully used for solving many different image recognition problems, e.g., automatic face recognition problems.

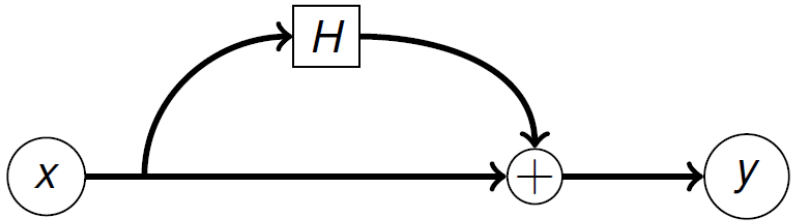


A. Maier, K. Breininger, N. Ravikumar, T. Würfl | Activation Functions and Convolutional Neural Networks

Example: Deep Residual Neural Networks

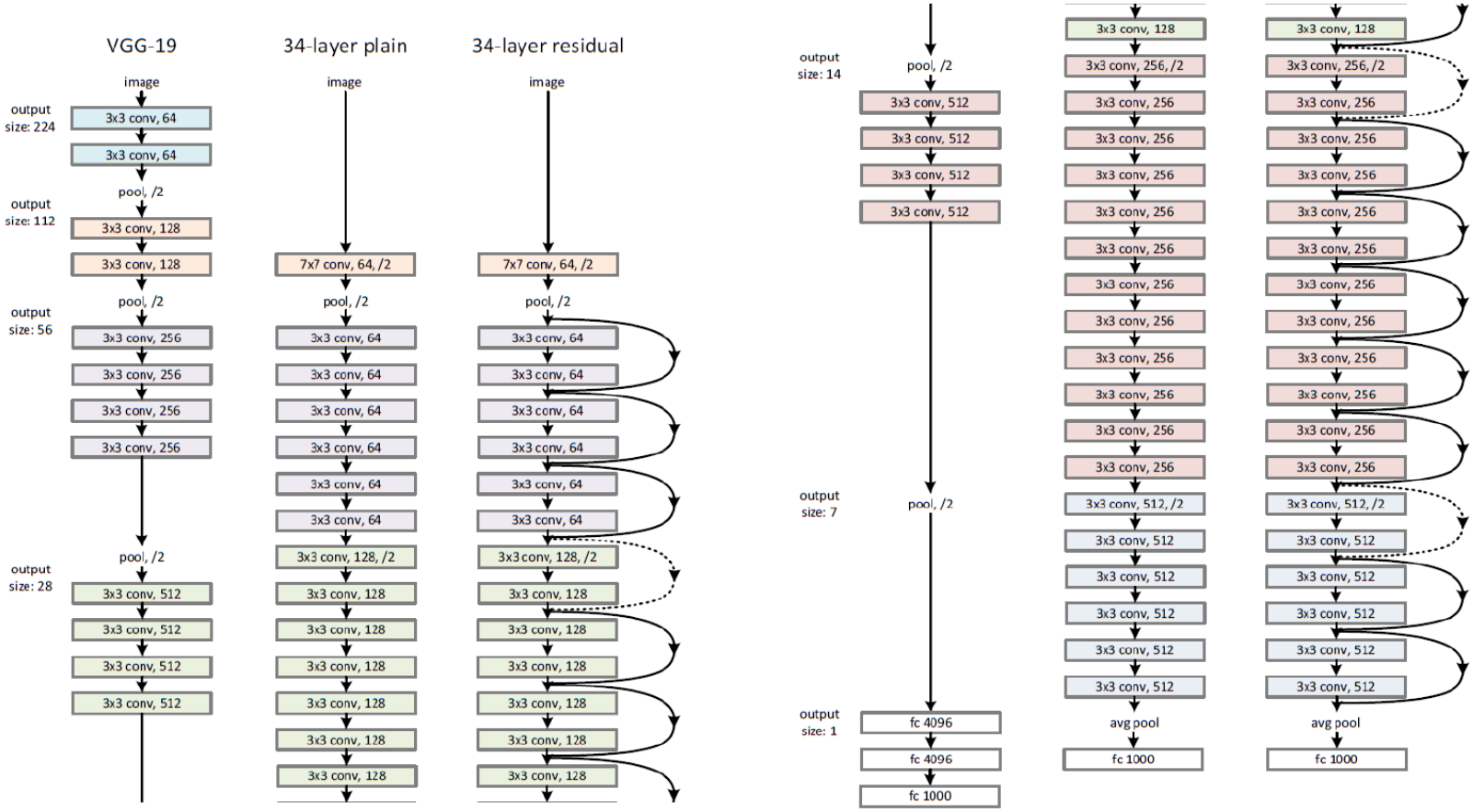
- The vanishing gradient problem has emerged by increasing the depth of neural networks.
- This problem was solved to some extend by introducing the residual units.
- Non-residual neural networks learn the function mapping $F(X)$.
- Residual network units learn the residual mapping $H(X)$.

$$H(x) = F(x) - x \Leftrightarrow F(x) = H(x) + x$$



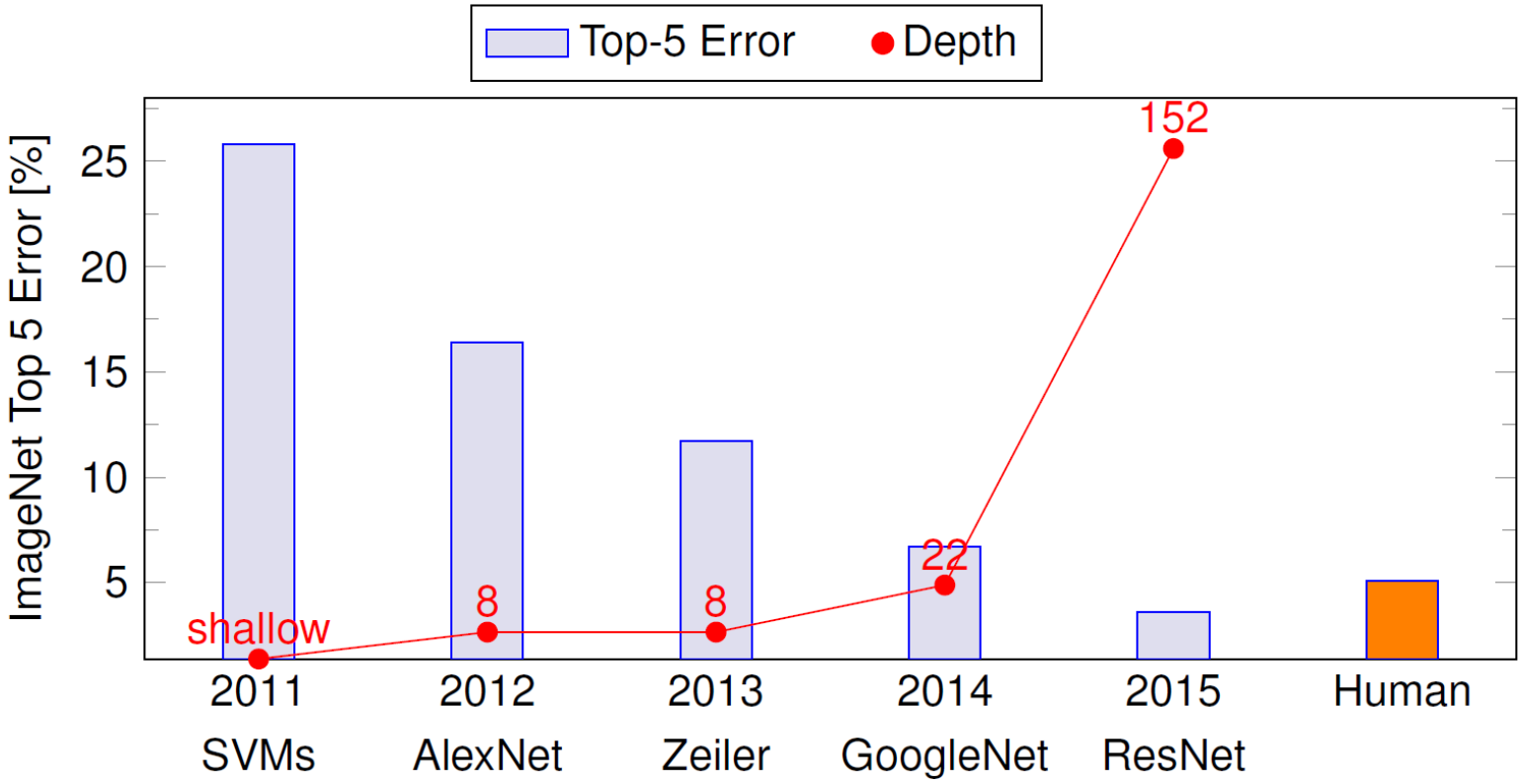
A. Maier, K. Breininger, N. Ravikumar, T. Würfl | Architectures

Example: Deep Residual Neural Networks



VGG: 19.6 billion FLOPs, Plain / ResNet: 3.6 billion FLOPs

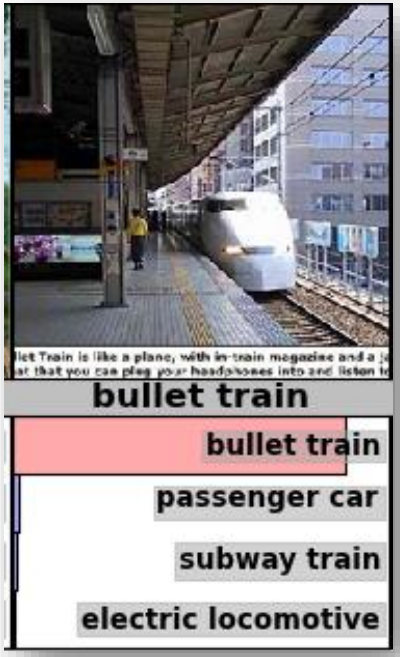
Example: Evolution of Depth of Convolutional Neural Networks



A. Maier, K. Breininger, N. Ravikumar, T. Würfl | Architectures

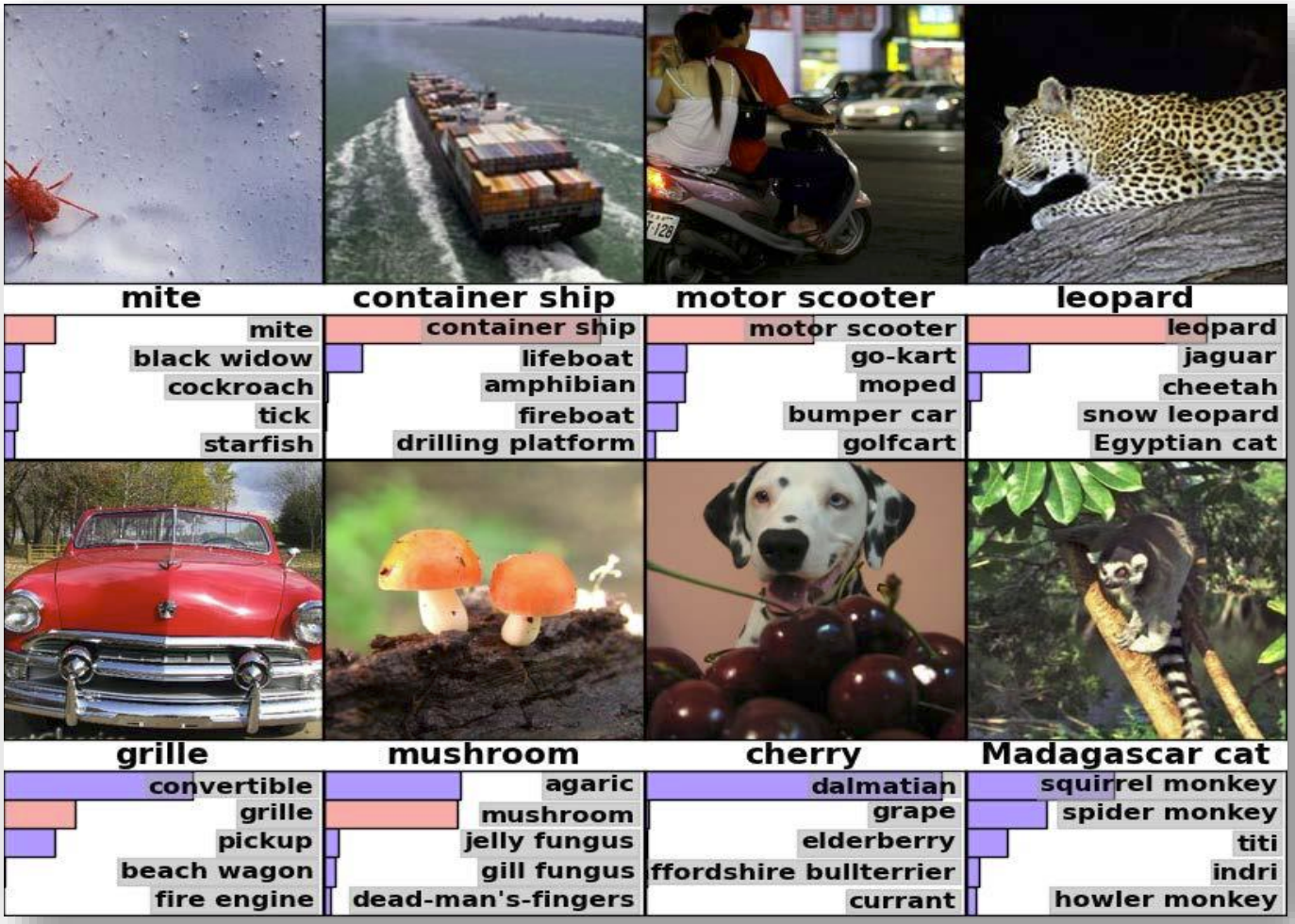
Example: The competition on ImageNet

- The dataset has 1.2 million high-resolution training images.
- The classification task: Get the "correct" class in your top 5 bets. There are 1000 classes.
- The localization task: For each bet, put a box around the object. Your box must have at least 50% overlap with the correct box.



Example: The competition on ImageNet

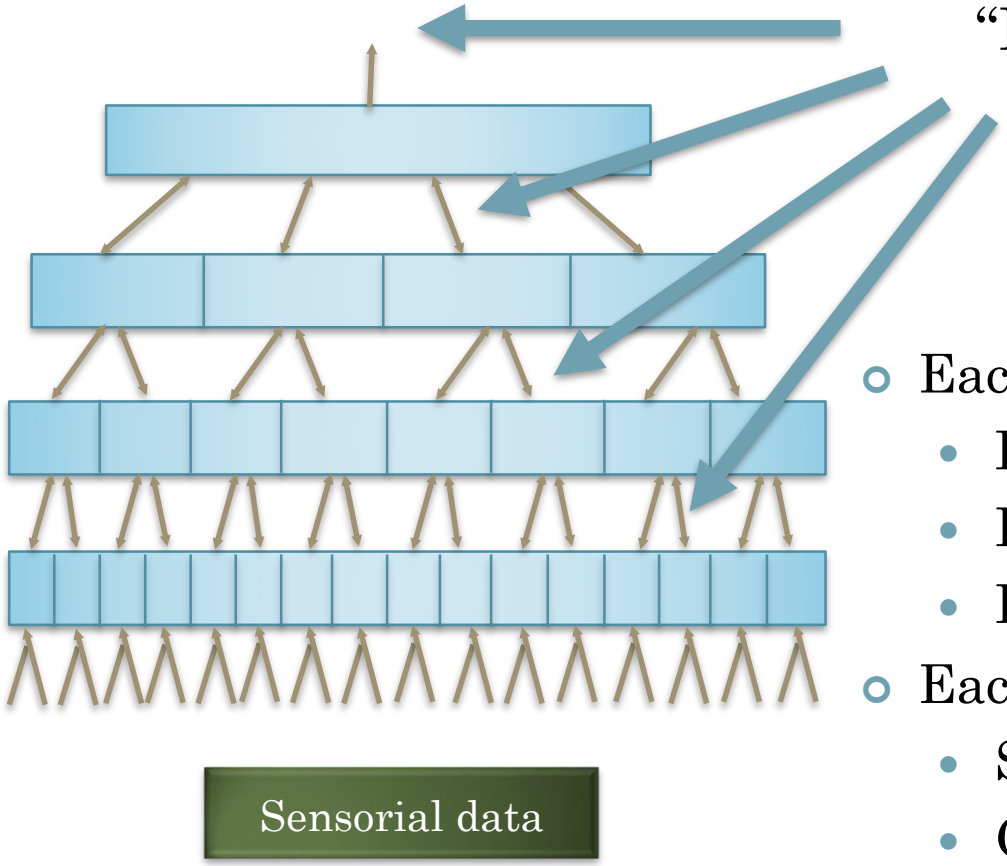
- More examples of how well the deep convolutional neural network works for object recognition.



Example: Hierarchical Temporal Memory

- The concept of Bayesian networks is related to the concept of neural networks and several neural/Bayesian network models have been proposed that combine these concepts.
- The concept of Hierarchical Temporal Memory (HTM) (Hawkins, Dileep) represents one such example.
- HTM combines and extends approaches used in Bayesian networks, spatial and temporal clustering algorithms, while using a tree-shaped hierarchy of nodes that is common in neural networks (Neocognitron etc).
- Each node in the HTM hierarchy discovers an array of causes in the input patterns and temporal sequences it receives.
- A Bayesian belief revision algorithm is used to propagate feed-forward and feedback beliefs from child to parent nodes and vice versa.

Example: Hierarchical Temporal Memory

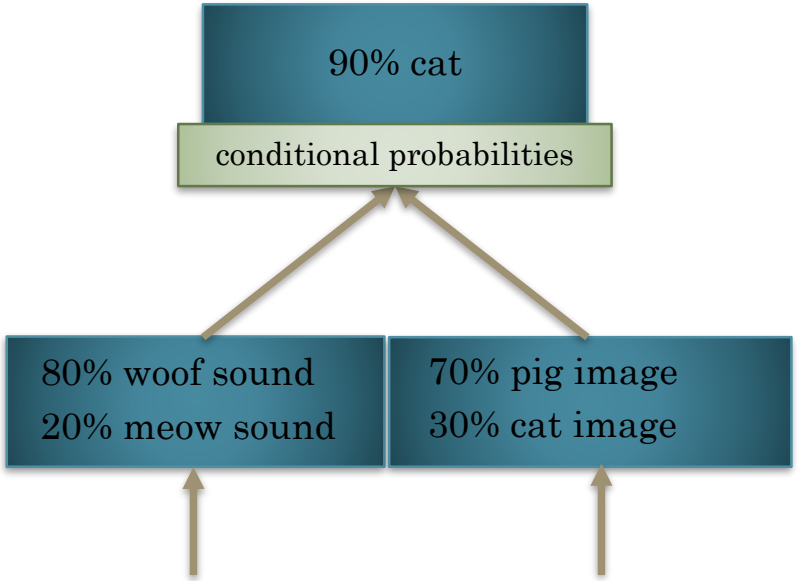


- Each node:
  - Discovers causes (of its input)
  - Passes beliefs up
  - Passes predictions down
- Each node
  - Stores common sequences
  - Changing sensory data forms stable beliefs at top
  - Stable beliefs at top form changing sensory predictions

Example: Hierarchical Temporal Memory

Why does hierarchy make a difference?

- Shared representations lead to generalization and efficiency
- HTM hierarchy matches spatial and temporal hierarchy of causes in world
- Belief propagation techniques ensure all nodes quickly reach mutually compatible beliefs

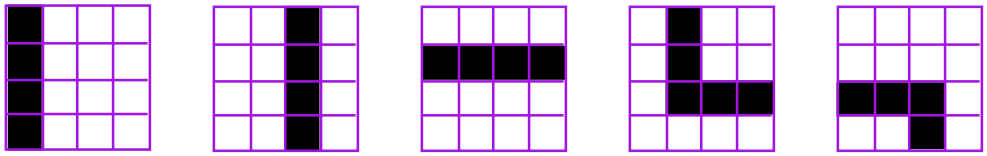


Example: Hierarchical Temporal Memory

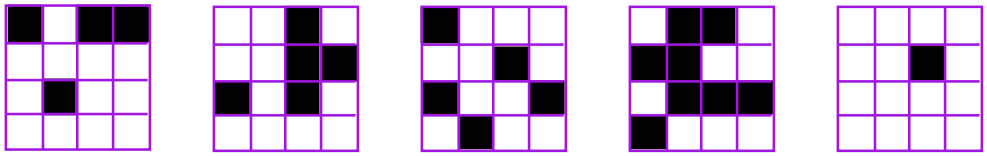
How does each HTM node discover causes?

- They learn common spatial patterns

Common patterns: remember



Uncommon patterns: ignore



Example: Hierarchical Temporal Memory

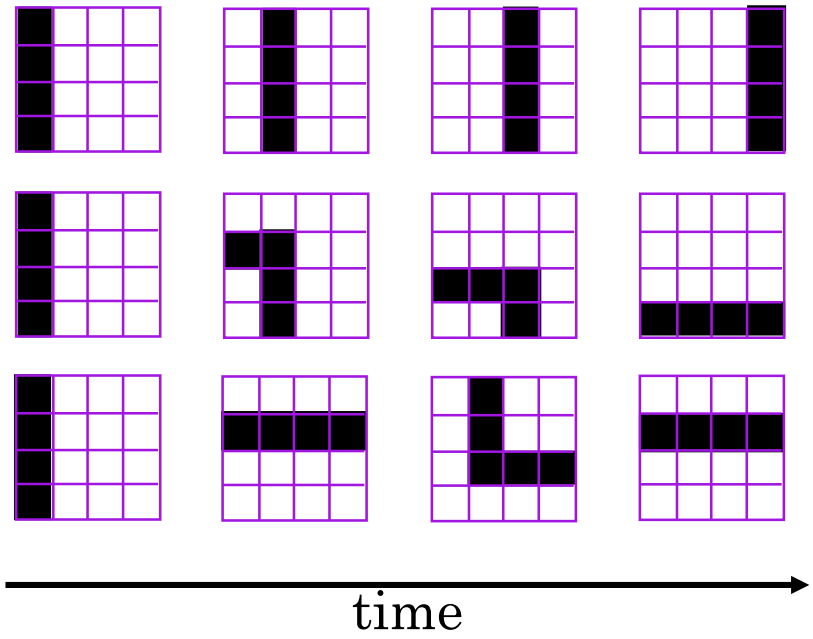
How does each HTM node discover causes?

- Learn common sequences of spatial patterns as well

Common sequence: assign to cause

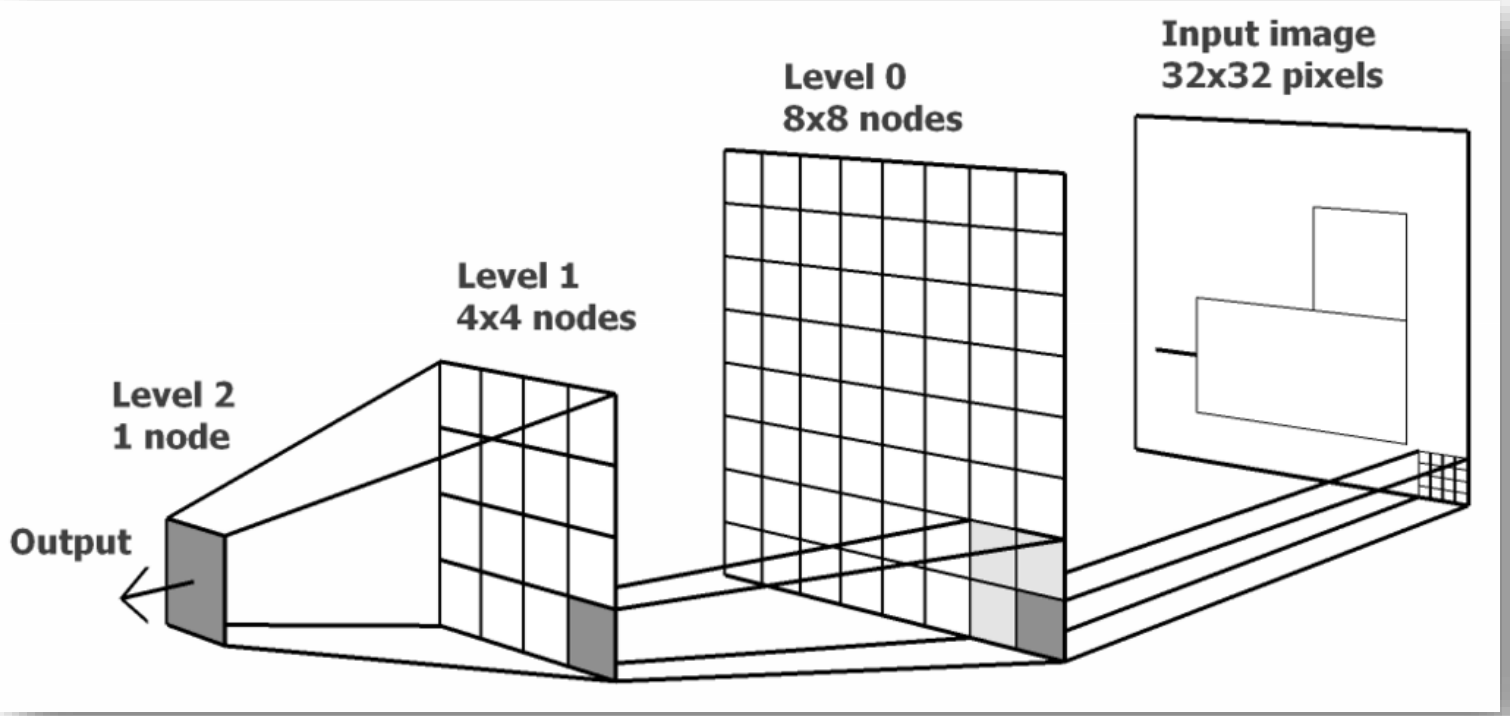
Common sequence: assign to cause

Uncommon sequence: ignore



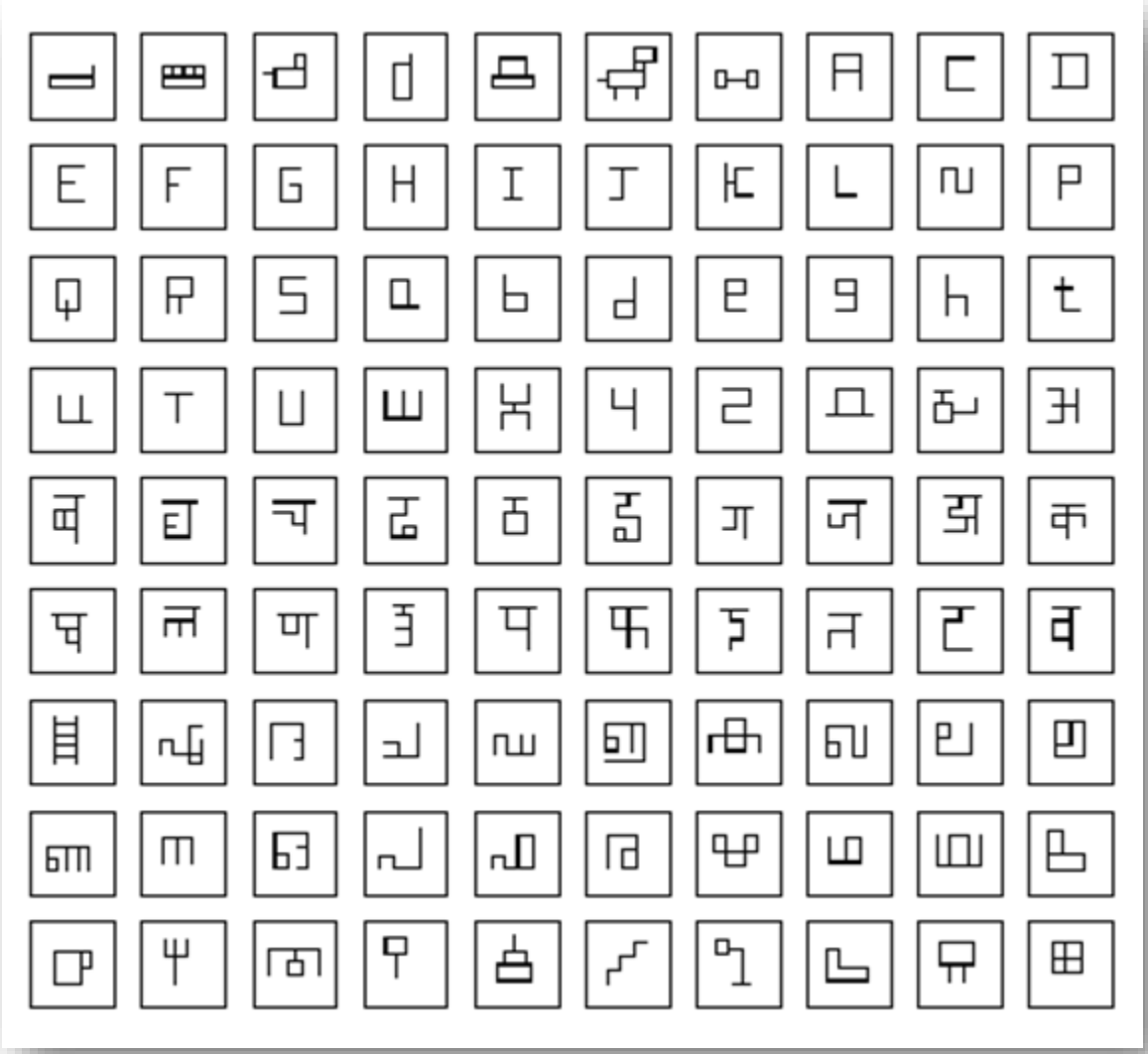
Example: Hierarchical Temporal Memory

An example of HTM hierarchy used for image recognition



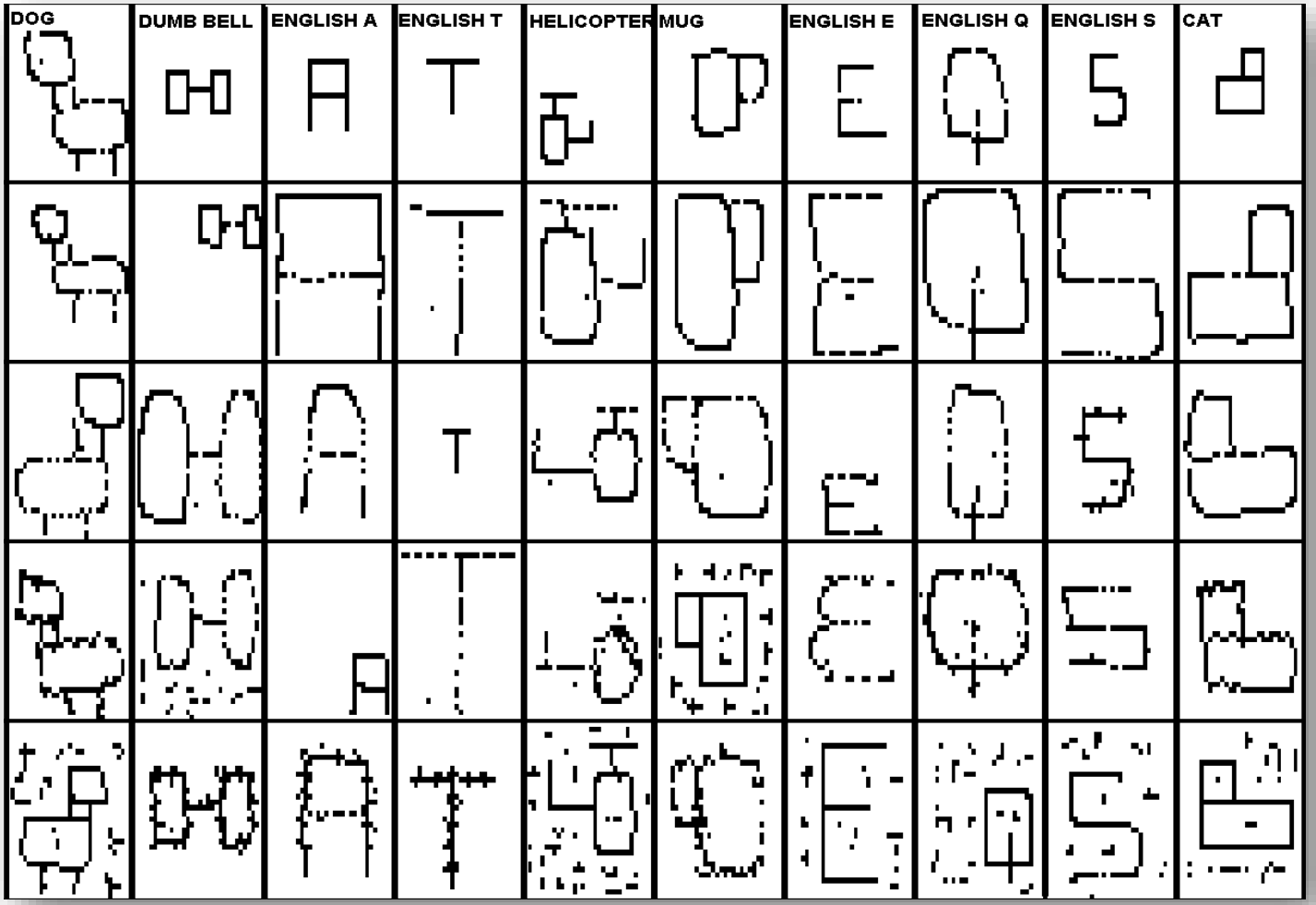
Example: Hierarchical Temporal Memory

Examples of training images



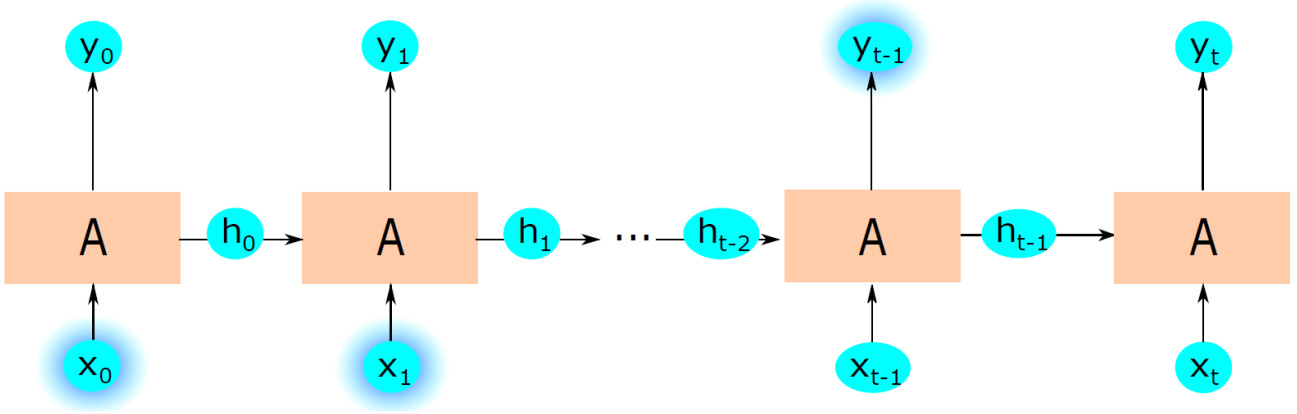
Example: Hierarchical Temporal Memory

Examples of correctly recognized images



Example: Recurrent Neural Networks

- Feed forward neural networks are normally used for modelling mappings of vector inputs to target vector outputs.
- However, there are many sequential percepts (speech, music, video, biosignals, etc) that one would like to map to some target outputs.
- The sequential nature of such problems can be modelled with Recurrent Neural Networks (RNNs).
- The basic structure of RNNs can be presented as a sequence of copies of the same units
- Each unit passes hidden state as additional input to successor



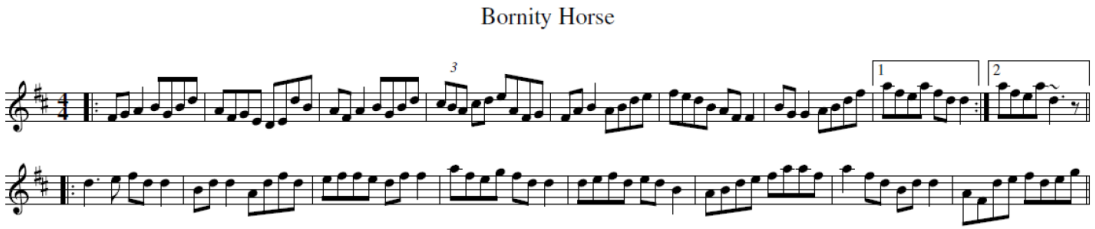
Example: Recurrent Neural Networks

- Recurrent Neural Networks have been successfully used for character-based language modelling, machine translation, text generation, music composition, etc.

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.
```

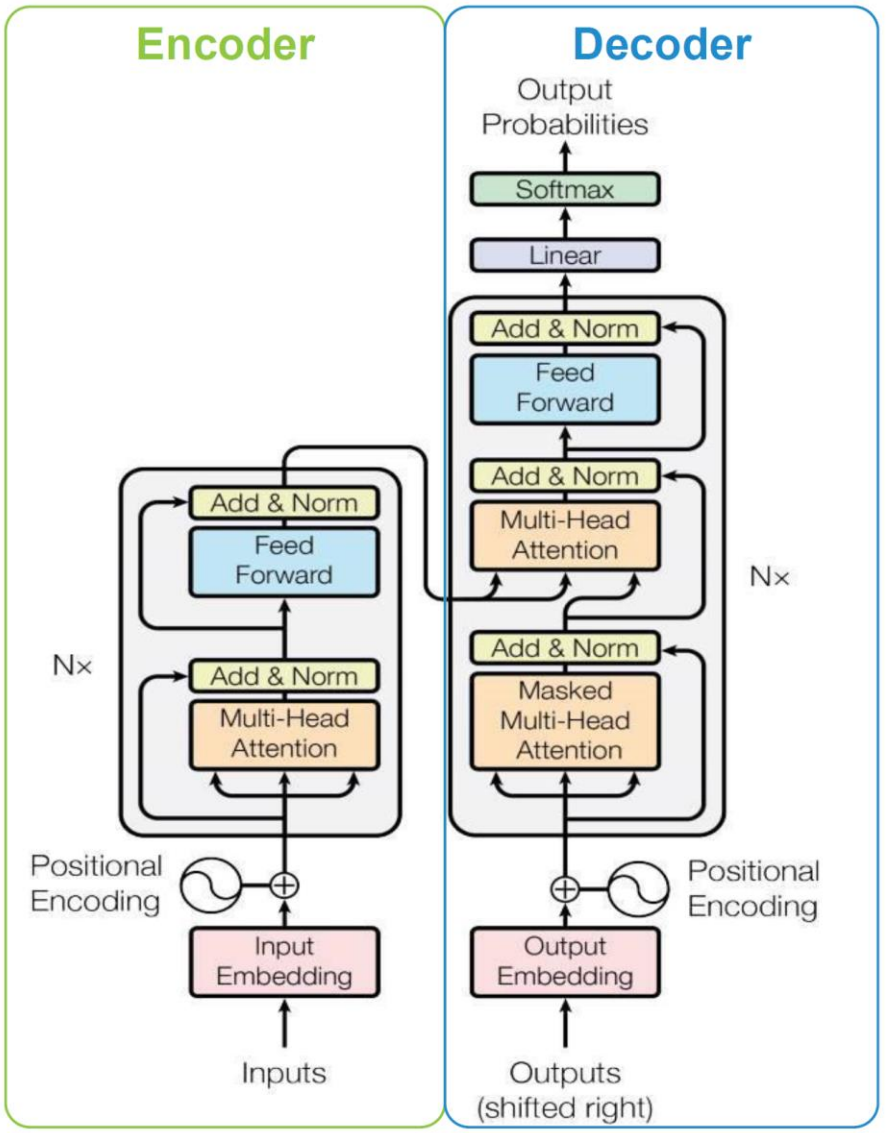


https://themachinefolksession.org/tune/31

https://youtu.be/6FxPJD0JZQo

Example: Transformer Networks

- Neural networks based on attention, replacing recurrent layers.
- Processes entire sequences in parallel,
  boosting speed and efficiency.
- Highly scalable with increased computational power and data size.
- Versatile across multiple domains, including text, vision, and speech.
- Encoder generates a representation encoding the entire input sequence.
- Decoder combines encoder output and previous decoder outputs to predict next symbol in sequence.



Applications of artificial neural networks

In general, the tasks artificial neural networks are applied fall within the following broad categories:

- Function approximation or regression analysis, including time series prediction.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators, prosthesis.
- Control, including Computer numerical control.
- Image registrations and noise removal.
- …

Questions

- Describe the basic model of neural networks.
- What types of artificial neural networks are known?
- Describe the basic learning method in multilayer perceptrons.
- What problems are solved by artificial neural networks?
- Describe the basic properties of deep convolutional neural networks .