Univerzita Pardubice
Fakulta elektrotechniky a informatiky

# Artificial intelligence in automation

Study material

Ing. Dominik Štursa

UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# Artificial intelligence in automation

## Topic 1: Artificial neural networks, division, definition of terms

**Study Objective**

The objective of this topic is to familiarize students with the fundamental concepts and terminology related to artificial neural networks, highlighting their structure and functional mechanisms. Students will explore various types of neural networks, including their specific applications in automation, enabling a deeper understanding of how these networks are differentiated and applied across different scenarios.

**Time Required for Study**

2 hours

**Keywords**

Neural Networks, Deep Learning, Backpropagation, Supervised Learning

## 1 Introduction to Neural Networks

### 1.1 Definition and Basic Concepts

Artificial neural networks (ANNs) are computing systems inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one neuron to another. The receiving neuron processes the signal and signals downstream neurons connected to it. The visualization of the neural network is at figure 1.

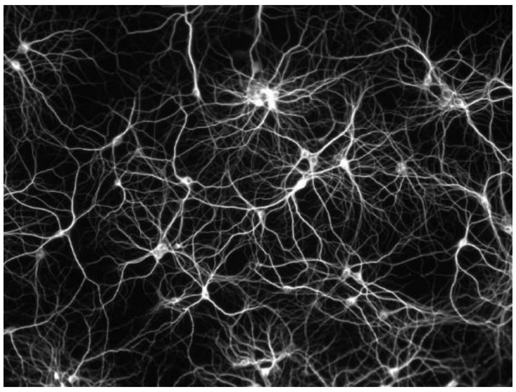


**Figure 1 - Visualization of the neural network. (Doležel, 2016)**

Neural networks rely on a process known as learning, which involves adjusting the weights of connections based on the input and output data they receive, a process that mimics the way humans learn.

## 1.2 Historical Background

The concept of neural networks dates back to the 1940s with the development of simple neural network models by Warren McCulloch and Walter Pitts. The real momentum, however, picked up in the 1980s with the backpropagation algorithm which allowed neural networks to adjust hidden layers of neurons in an efficient manner. This period also saw the development of the concept of deep learning, where neural networks with many layers can learn from a vast amount of data. The model of McCulloch-Pitts artificial neuron is showed on figure 2.

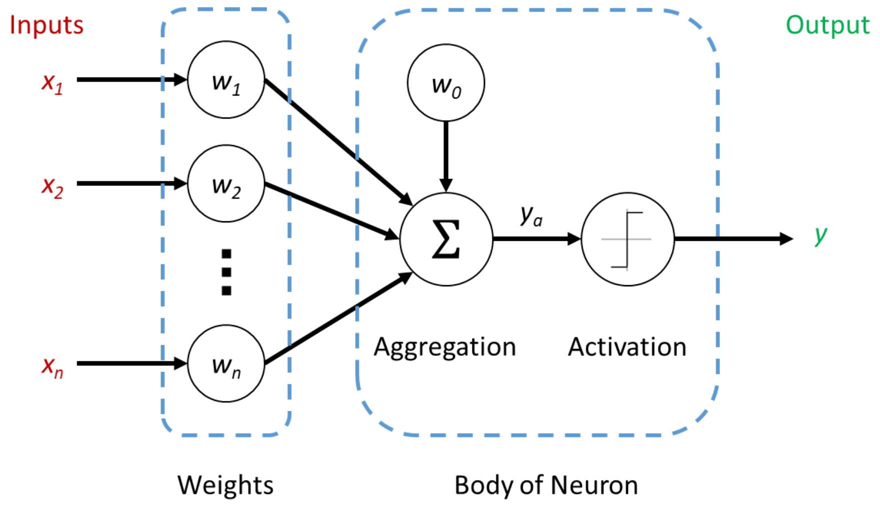


**Figure 2 - Model of McCulloch-Pitts artificial neuron.**

## 1.3 AI Importance in Automation

In the field of automation, neural networks have become indispensable due to their ability to learn and make intelligent decisions based on data. For example, neural networks are used in robotic process automation to handle tasks that require visual recognition, decision-making, and speech recognition. The adaptive learning feature of neural networks enables automated systems to optimize operations based on variable data inputs, which significantly enhances efficiency and effectiveness in industrial applications.

# 2 Types of Neural Networks

Neural networks can be broadly classified based on their architecture and the specific tasks they are designed to perform. This section introduces the primary types of neural networks including feedforward, convolutional, and recurrent neural networks, each suited to different kinds of problems and data types.

## 2.1 Feedforward Neural Networks

Feedforward Neural Networks (FNNs) are the simplest type of artificial neural network architecture. In this network, information moves in only one direction – forward – from the input nodes, through the hidden nodes (if any), and to the output nodes. There are no cycles or loops in the network. FNNs are widely used in approximations, basic pattern recognition and classification tasks.

## 2.2 Perceptrons

The perceptron is one of the simplest types of feedforward neural networks and forms the foundational building block for many neural network models. It consists of a single neuron with adjustable weights and a bias, modeling decisions by weighing input signals. Perceptrons are particularly significant for their role in the history of neural networks, serving as a basic example of binary classification.

## 2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are specialized to process data that comes in the form of multiple arrays, such as images. CNNs employ a mathematical operation called convolution which allows the network to focus on small regions of the input data, making them extremely efficient for tasks such as image classification, detection, recognition or segmantation.

## 2.4 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are designed to recognize sequences of data. Unlike feedforward neural networks, RNNs have connections that loop back on themselves, allowing them to maintain a 'memory' of previous inputs in their internal state. This makes them ideal for tasks where context from earlier in the sequence is crucial, such as language translation or speech recognition.

## 2.5 Self-Organizing Maps (SOMs)

Self-Organizing Maps are a type of unsupervised learning neural network that is used for data visualization and dimensional reduction. SOMs operate by organizing data into a map in a way that groups similar data items together. They are widely used for feature detection and complex pattern recognition, helping in the visualization of high-dimensional data.

### 2.6 Other Variants

Apart from the main types, there are other specialized neural networks such as Autoencoders, which are used for data compression and feature learning, and Generative Adversarial Networks (GANs), which consist of two neural networks contesting with each other to generate new, synthetic instances of data that can pass for real data.

## 3 Components of Neural Networks

### 3.1 Neurons

A neuron in an artificial neural network is a computational unit that receives inputs (either from original data or from the output of other neurons in the network), processes these inputs using a weighted sum, and then passes the result through an activation function to produce an output. Each neuron's output can then serve as an input to the next layer of neurons in the network, allowing complex functions to be modeled.

### 3.2 Weights and Biases

Weights in a neural network are parameters that transform input data within the network’s architecture. They are adjusted during training to minimize the difference between the actual output of the network and the desired output, effectively 'learning' from the data. Biases are additional parameters which allow the model to fit better with the data by shifting the activation function to the left or right, which can be crucial for learning patterns.

### 3.3 Activation Functions

Activation functions are mathematical equations that determine the output of a neural network. They introduce non-linear properties to the network which allows them to learn more complex patterns. Some common activation functions include the sigmoid, which squashes the output between 0 and 1; the hyperbolic tangent (tanh), which scales the output to between -1 and 1; and the ReLU (Rectified Linear Unit), which outputs the input directly if it is positive, otherwise, it will output zero.

### 3.4 Architecture of Neural Networks

The architecture of a neural network refers to the arrangement of neurons and layers. This includes the number of layers, the number of neurons in each layer, and how these neurons are interconnected. Different architectures are suited to different tasks, for example, CNNs for image tasks, RNNs for sequential data, and simple feedforward networks for general prediction tasks.

## 3.5 Learning Mechanisms

Neural networks learn through a process called training, where the network adjusts its weights and biases based on the error of its output compared to the desired outcome. This adjustment is facilitated by algorithms such as gradient descent, where the gradient (or derivative) of the error is calculated in respect to all weights in the network, and the weights are adjusted to minimize the error.

# 4 Learning Processes

## 4.1 Forward Propagation

In forward propagation, input data is fed into the neural network and passes through each layer. At each layer, the data is processed by summing the weighted inputs and biases, and then applying an activation function. This process transforms the input data step-by-step until it reaches the output layer, which produces the final prediction.

## 4.2 Backpropagation

Backpropagation is the mechanism by which neural networks learn from the error in their predictions. After forward propagation, the output is compared to the true value, and the difference (error) is calculated using a loss function. This error is then propagated back through the network, layer by layer, updating the weights and biases according to their contribution to the error. This is typically done using the gradient of the loss function with respect to each weight and bias.

## 4.3 Loss Functions

Loss functions are critical in training neural networks as they provide a measure of how well the network's predictions match the expected outputs. Common loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy for classification tasks. These functions calculate the difference between the predicted values and the actual values, guiding the network's learning.

## 4.4 Optimization Algorithms

Optimization algorithms are used to minimize the loss function, effectively improving the network's accuracy over iterations. Gradient descent is a foundational method where weights are updated in the direction that most reduces the loss. More sophisticated methods like stochastic gradient descent and Adam optimizer offer improvements such as faster convergence and adaptability to different problems.

# 5 Applications

## 5.1 Industrial Automation

Neural networks play a crucial role in modern industrial automation, enhancing efficiency and reliability. They are used in predictive maintenance to forecast equipment failures before they occur, minimizing downtime and maintenance costs. Neural networks also improve quality control by analyzing production data in real time to detect anomalies or deviations from quality standards.

## 5.2 Autonomous Vehicles

In the realm of autonomous vehicles, neural networks are essential for processing vast amounts of sensory data to make real-time decisions. They enable functions such as object recognition, which allows vehicles to identify and react to other cars, pedestrians, and obstacles. Neural networks also handle complex decision-making processes required for navigating traffic and adjusting to changing road conditions.

## 5.3 Healthcare

Neural networks are transforming healthcare by automating diagnostic processes and treatment planning. They analyze medical images, such as X-rays or MRIs, to assist in diagnosing diseases with high accuracy. Additionally, neural networks automate administrative tasks such as scheduling and maintaining patient records, enhancing operational efficiency.

## 5.4 Finance and Trading

In finance, neural networks are utilized for algorithmic trading, using pattern recognition to predict market movements and execute trades at optimal times. They also assess credit risk and fraud detection by analyzing transaction patterns and customer data, leading to more secure and efficient financial operations.

## 5.5 Customer Service

Neural networks enhance customer service through the deployment of advanced chatbots and virtual assistants capable of understanding and responding to human language. These technologies provide personalized customer interactions based on previous data, improving satisfaction and engagement.

## 6 Literature

DOLEŽEL, Petr. Úvod do umělých neuronových sítí. Online. 1. Pardubice: Univerzita Pardubice, 2016. ISBN 978-80-7560-022-6.

## List of abbreviations

| | |
|---|---|
| ANN | Artificial Neural Network |
| CNN | Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| GAN | Generative Adversarial Network |
| SOM | Self-Organizing Map |
| MSE | Mean Squared Error |
| SGD | Stochastic Gradient Descent |
| ReLU | Rectified Linear Unit |
| AI | Artificial Intelligence |

## Index

# Artificial intelligence in automation

## Topic 2: A simple Perceptron as an Elementary Neural Network

**Study Objective**

The objective of this topic is to introduce the perceptron as a foundational concept in neural networks. Students will learn about the structure, function, and learning process of a simple perceptron. The course will cover how perceptron operate, their limitations, and their historical significance in the development of more advanced neural networks.

**Time Required for Study**

2 hours

**Keywords**

Perceptron, Hebbs Law, Supervised Learning, Binary Classification

## 1 Introduction to Perceptron

As one of the simplest neural networks, the perceptron is also one of the most well-known and widely used neural networks in practice. The topology consists of a single powerful element - a neuron (a simple perceptron), which will be discussed in this chapter.

### 1.1 Perceptron Model

As one of the simplest neural networks, the perceptron is also one of the most well-known and widely used neural networks in practice. The topology consists of a single powerful element - a neuron (a simple perceptron), which will be discussed in this chapter.

The basic executive of the perceptron is a neuron model with a linearly weighted aggregation function, and originally only a jump activation function was considered (Rosenblatt, 1958) - this is a special case of the basic McCulloch-Pitts neuron model. However, it should be noted that in the current literature the term perceptron is often used to refer to artificial neurons with a linear or sigmoidal activation function (Haykin, 1999).

The learning of a perceptron is supervised, and thus it is a network using supervised learning. The desired function of the neural network is specified by a training set in the form of pairs of input vector samples and a vector of desired (target) output values. The output of the perceptron is then a scalar quantity taking binary values 0, 1, or bipolar values -1, 1, respectively. The basic schematic of a simple perceptron is shown in Figure 1. The basic schematic of a simple perceptron is shown in figure 1.

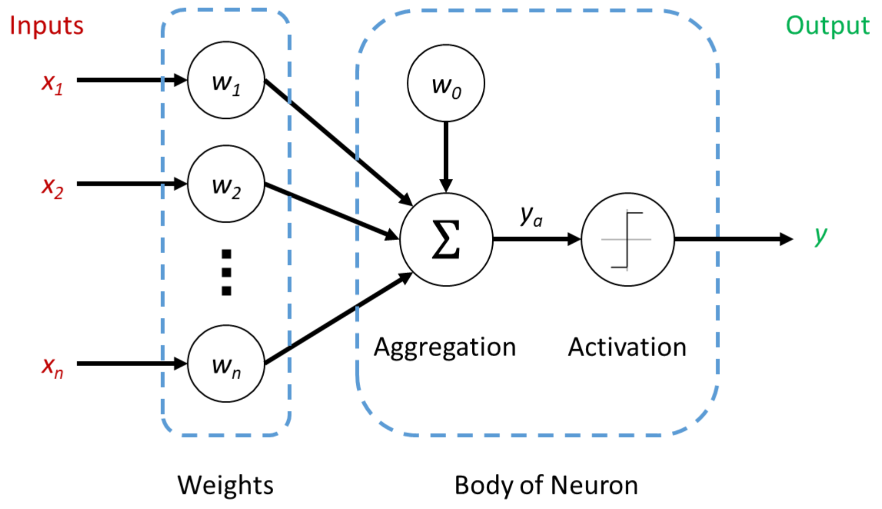


**Figure 1 – Model of perceptron.**

## 1.2 Perceptron Usage

Given the nature of the model using a jump or bipolar jump function as the activation function, it is evident that the output of the perceptron takes on a binary character. Thus, only problems whose solutions are linearly separable can be solved through the perceptron. An example is pattern recognition in data - using a simple perceptron we are only able to separate the data into two groups, moreover the data must be separable by a superplane as shown in Figure 2.

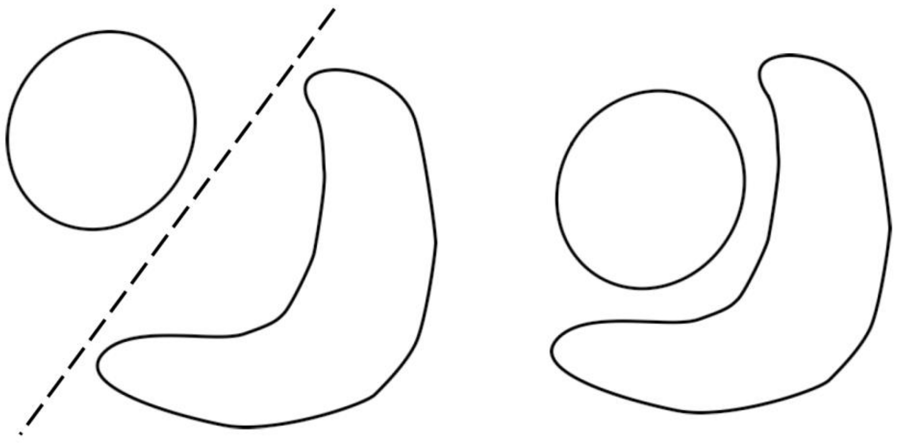


**Figure 2 - Linearly separable data, linearly non-separable data. (Doležel, 2016)**

## 2 Simple Perceptron Response

Like all artificial neural networks, a simple perceptron acts as a processor responding to excitation in the form of input xi. In the case of a perceptron, a single scalar value y is the response to a particular set of input. This value can be determined by the relation (1)

$$y = \phi(y_a). \tag{1}$$

Where $\phi$ is the activation fuction and $y_a$is the input potencial of the neuron for which the following formula applies (2)

$$y_a = \sum_{i=0}^{n} x_i w_i. \tag{2}$$

In which the number $n$ represents number of inputs; $w_i$ are weights of individual inputs; $x_i$ are individual inputs; and $x_0 = 1$ is an artificial input for the perceptron bias.

Two options can then be applied to the activation function:

- The first option is the step activation function for which the following applies (3)
$$y = \begin{cases} 1, & y_a \geq 0 \\ 0, & y_a < 0 \end{cases}. \tag{3}$$
- The second is a bipolar step function for which the following relation holds (4)
$$y = \begin{cases} 1, & y_a \geq 0 \\ -1, & y_a < 0 \end{cases}. \tag{4}$$

## 3 Training a Simple Perceptron

A simple perceptron uses inductive learning, where we talk about learning with a teacher who provides the correct information about the desired output of the network. The so-called error learning perceptron is built on this basis. However, due to the nature of the output, it is also possible to apply Hebb's law of learning to a simple perceptron.

### 3.1 Hebbian Learning

Hebbian learning rule is a precursor of all current learning algorithms. Its author is Canadian psychologist Donald Hebb and it was first published in 1949 (Hebb, 1949). This learning law describes an algorithm for modifying the values of synaptic connections in the nervous systems of an organism. It is based on the assumption that if two interconnected neurons are active at the same time, then their connections (represented by synaptic weights) are strengthened. In the opposite case (i.e. if both neurons are inactive), their mutual coupling is weakened, with each neuron under consideration having only two states - active and inactive.

Thus, in relation to the above, if the value of the input to the neuron is synchronous with the expected output, then the weight of the connection between the corresponding input and the neuron is strengthened; if it is asynchronous (the values are not the same), the weight is weakened. Mathematically, this can be written in the case of bipolar inputs and activation functions by the relation (5)

$$w_i = w_i + x_i \cdot t. \tag{5}$$

In which $t$ is a neuron target.

## 3.2 Error Learning

Error learning, also known as perceptron learning rule, is a fundamental algorithm for training neural networks and plays a key role in supervised learning scenarios where the challenge is to minimize the difference between the actual output and the desired output. This method is particularly important for training a simple perceptron, allowing it to learn to perform binary classification tasks.

The basic idea of error learning is simple. For each input to the perceptron, the actual output is compared with the desired output. If the actual output is equal to the desired output, no weight changes are made. However, if equality does not hold (the error is not zero), the weights are adjusted to reduce the error. The adjustment of the weights is based on the formula (6)

$$w_i = w_i + \alpha(t - y)x_i. \tag{6}$$

In which $\alpha$ is the learning rate, a small positive number that determines the step size of the adjustment.

This formula ensures that the adjustment of the weights is proportional to the input value and the magnitude of the error, allowing a gradual convergence to the correct model.

### 3.2.1 Algorithm Steps

Training a simple perceptron involves a series of systematic steps designed to adjust the weights based on the input training data and corresponding outputs. The goal is to reduce the error between the predicted and actual classification values. Here is a description of each step:

1. **Initialization**: Start by initializing the weights and bias to small random values or zeroes. Set a learning rate $\alpha$, which controls how much the weights are updated during training.
2. **For each epoch**: An epoch consists of one full pass through the entire training dataset.
   a. **For each training example**
      i. Compute Output: Calculate the weighted sum of the inputs, including the bias (2) and procced them to acticavion function to get output (3) or (4).

ii. Update Weights and Bias: If the output $y$ does not match the target $t$, adjust the weights by (6) and for bias by (6), where $x_i = 1$.

3. **Repeat**: The process is repeated for a specified number of epochs, or until the weights stabilize and the error minimizes.
4. **Evaluation**: After training, evaluate the perceptron's performance on a separate validation or test set to check its accuracy and generalization capability.

### 3.2.2 Python Code Example

A common and somewhat more complex example used in perceptron training is the classification of data points into two categories based on their position relative to a straight line. This task can be visualized, making it more intuitive to understand the concept of linear separability that the perceptron handles. So let's use a dataset in which we classify points as being above or below a line in two-dimensional space, for example $x_1 > 2x_2 + 1$.

If we use the numpy library to work with arrays and the matplotlib library to display graphs, then the python code can be as follows.

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate random data and labels
np.random.seed(0)
X = np.random.randn(100, 2) * 10
y = (X[:, 1] > 2 * X[:, 0] + 1).astype(int)

# Train the perceptron
weights, bias = train_perceptron(X, y)

# Plotting
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='winter', edgecolor='k', s=50)
x_values = np.array([-20, 20])
y_values = 2 * x_values + 1
plt.plot(x_values, y_values, 'r--')
y_model = -(weights[0] * x_values + bias) / weights[1]
plt.plot(x_values, y_model, 'g-')
plt.show()
```

The function for training must be defined. In example it is the function ”train_perceptron” defined as follows.

```
# Training the perceptron
def train_perceptron(X, y, learning_rate=0.1, epochs=20):
    n_samples, n_features = X.shape
    weights = np.zeros(n_features)
    bias = 0

    for _ in range(epochs):
        for idx, x_i in enumerate(X):
            linear_output = np.dot(x_i, weights) + bias
            predicted = np.where(linear_output >= 0, 1, 0)
            update = learning_rate * (y[idx] - predicted)
            weights += update * x_i
            bias += update

    return weights, bias
```

After running the script, the output shows a graph for random data around the defined line, which can be separated into 2 categories - points below and points above the line. The output is shown at Figure 3.

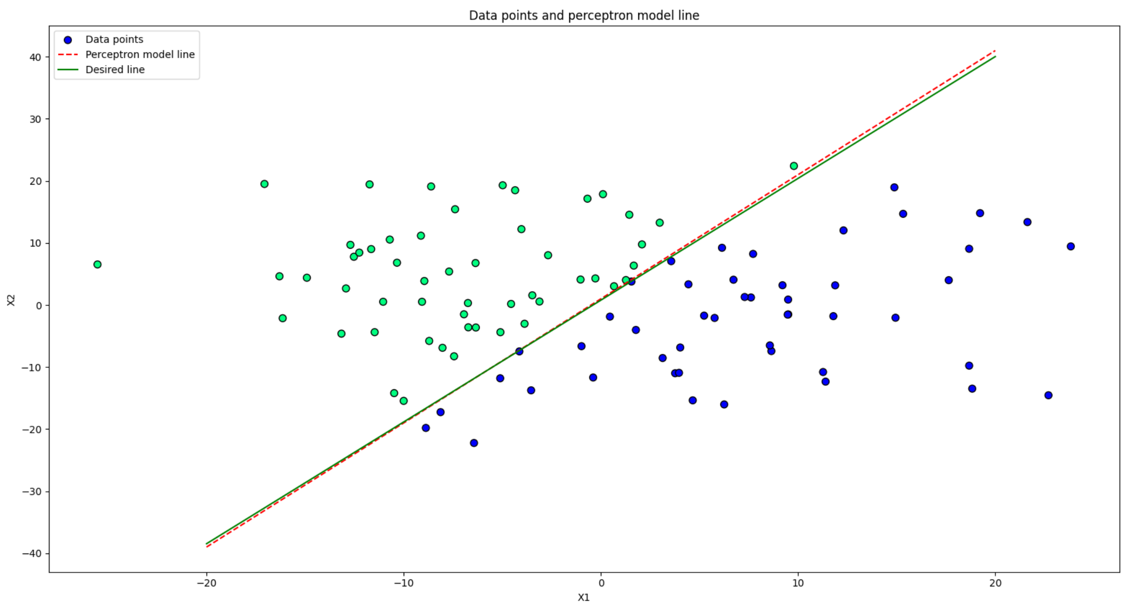


**Figure 3 – Code output example for binary classification with perceptron**

## 4 Conclusions

In this chapter we have discussed the simple perceptron, one of the earliest and most basic forms of neural networks. As a binary classifier, the perceptron serves as a fundamental building block for understanding more complex neural network architectures. Its simplicity, characterized by a single input and output layer with a straightforward learning algorithm, provides a clear illustration of how one can begin to make decisions with neural networks.

# 5 Literature

DOLEŽEL, Petr. Úvod do umělých neuronových sítí. Online. 1. Pardubice: Univerzita Pardubice, 2016. ISBN 978-80-7560-022-6.

HAYKIN, Simon. Neural networks: a comprehesive foundation. 2nd ed. Upper Saddle River: Prentice-Hall, 1999. ISBN 978-0132733502.

HEBB, Donald. The Organization of Behavior. Wiley and Sons, New York, 1949. ISBN 978-0805843002.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. Online. Psychological Review. 1958, issue 65, no. 6, p. 386-408. ISSN 1939-1471. At: https://doi.org/10.1037/h0042519.

## List of abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |

## Index

# Artificial intelligence in automation

## Topic 3: Multilayer Perceptron I – Definition, Back Propagation Algorithm.

**Study Objective**

The study objective is to equip students with a thorough understanding of multilayer perceptron as a cornerstone of modern artificial neural networks, focusing on their structure, the significance of their multiple layers, and the back-propagation algorithm which enables learning from complex data sets.

**Time Required for Study**

2 hours

**Keywords**

Multilayer Perceptron, Back Propagation, Neural Networks, Hidden Layers, Activation Functions, Gradient Descent

## 1 Introduction

In the following chapters we will only discuss the forward multilayer artificial neural network (ANN), as it is the most frequently used type. This network is usually represented by a name feed forward neural network (FFNN) because the signal is going only in one direction - forward. It has found applications as a universal approximator, predictor, decision maker and in pattern recognition. The basic element is the formal neuron, whose properties were described earlier.

### 1.1 Definition of a Multilayer Perceptron

The structure (topology) of a FFNN can be described by an arbitrary graph with vertices and oriented edges (Samarasinghe, 2006). The vertices of this graph are composed of neurons and the oriented edges represent the connections between these neurons. It is true that the output of one neuron is (after applying the appropriate connection weight) an input to several other neurons, just as the terminals of the axon of a biological neuron are connected via synaptic connections to the dendrites of other neurons.

If the edge (connection) between neuron *i* and neuron *j* is oriented in the direction *i* -> *j*, then neuron *i* is called a presynaptic (source) neuron, neuron *j* a postsynaptic (target) neuron.

The weight of the corresponding connection is denoted by $w_{ij}$. If we also denote the layer to which the connection is directed, the layer number will be given as a superscript - $w_{ij}^{k}$.

In addition to the vertex and oriented edges, the topology of a FFNN is also described by layers. The basic characteristic of a FFNN is that a neuron connection exists (and is complete) only between neurons of adjacent layers and is unidirectional. That is, there are no connections between neurons in their own layer and between other layers, and there are no feedback connections between neurons.

Signals from the environment enter the FFNN through source nodes (also input terminals), which can be seen as passive elements with unit transmission. The source nodes are used to distribute the input signals to the active neurons of the next layer. The output layer is used to transmit the signal from the neural network to the environment.

All intermediate layers are called hidden layers. Thus, a FFNN is made up of input nodes (terminals), one or several layers of hidden neurons and one layer of output neurons (Samarasinghe, 2006). A general schematic of the FFNN topology including labeling is shown in figure 1.

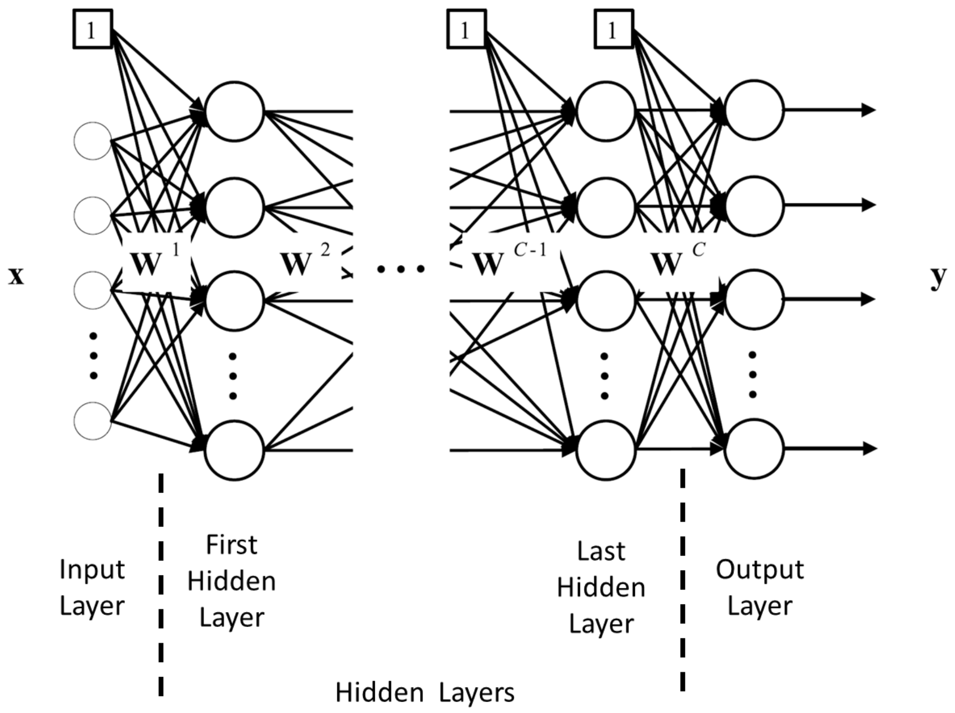


**Figure 1 - feed forward neural network (Doležel, 2016)**

## 1.2 FFNN Topology Design

There is no analytical procedure for designing the topology of a neural network. A number of different recommendations are given in the literature, but these are always specific to the problem to be solved and are thus highly subjective. They are mostly based on a series of simulation calculations or an experiment. The number of hidden layers is chosen according to the complexity of the problem to be solved. Experience shows that in a number of cases we achieve satisfactory behaviour in using a single hidden layer.

Although it is possible to find other recommendations in different sources, it should be noted that in the majority of cases it is more likely to proceed experimentally in basically two ways. First, a redundant neural network is chosen - a network with a high number of layers and neurons. This network is then gradually simplified (pruned) until the simplest possible network still gives satisfactory results. The second option is to initially choose a very simple network topology and gradually enrich it with additional neurons and layers until there is still an improvement in network performance.

The aggregation functions in the FFNN neurons are strictly defined as sums of (weighted) inputs, the activation functions are theoretically arbitrary. In practice, the problem to be solved, which often indicates the correct choice of activation functions, and the learning algorithm used are taken into account.

# 2 FFNN Response

The determination of the FFNN response is done by applying the input vector to the input terminals, saving the input potential and then the output of the neuron in the first hidden layer, then in the second hidden layer, and thus proceeds to the output layer. For neuron *j* in the *k*-th layer the relations

$$y_{aj}^{k} = \sum_i w_{ij}^{k} \cdot y_{i}^{k-1}, \tag{1}$$

$$y_j^k = \phi^k\left(y_{aj}^k\right). \tag{2}$$

Meaning that all outputs of all layers can be calculated by solving each particular equations between all layer each by each. Technically, this is the same calculation as for a simple perceptron, but extended to all participating neurons.

As with other topologies, it is advantageous to use a matrix approach to determine the response of the FFNN. The output vector of the whole layer *k* can be determined by the following pair of relations

$$y_a^k = (\mathbf{W}^k)^{\mathrm{T}} y^{k-1}\,, \tag{3}$$

$$y^k = \phi^k(y_a^k). \tag{4}$$

## 3 FFNN Training

Learning FFNN is done through supervised learning. Thus, we have a training (and possibly testing and validation) dataset, and the goal of learning is to minimize the difference between the actual and expected network response. The situation is shown schematically in Figure 2, where **x** is the vector of actual network inputs, **y** is the vector of network responses to the input vector **x**, and **t** is the vector of desired outputs obtained from the training set.

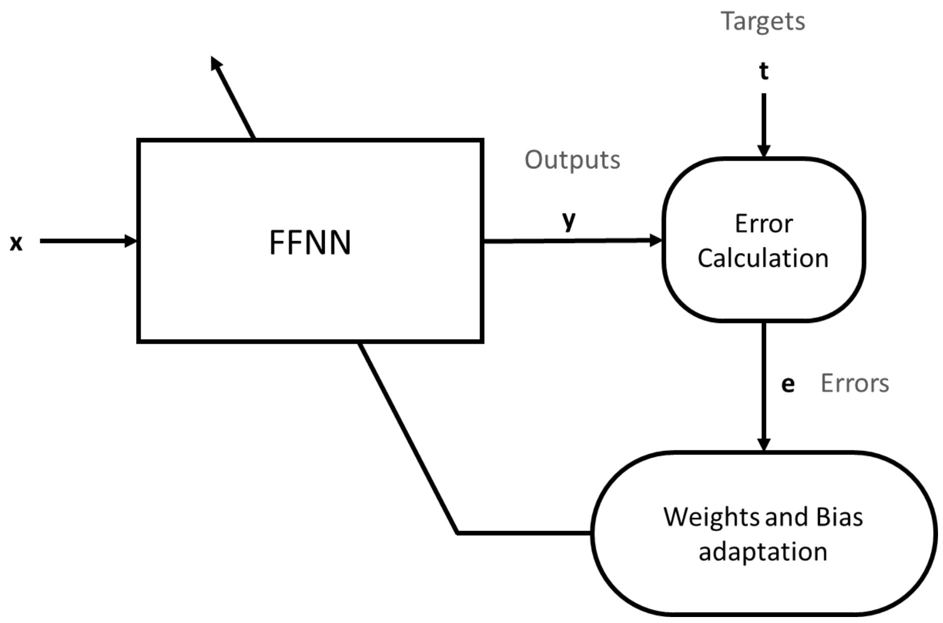


**Figure 2 - Principle FFNN learning**

Although FFNNs were known earlier, they were not widely used until the second half of the 1980s, when the back-propagation algorithm was "rediscovered" (Rumelhart, 1986).

In fact, the relatively well-known optimization method newly applied to FFNN, combined with the development of computational techniques, caused a huge increase in learning efficiency. Several learning algorithms are now available. However, many of them are based on the initial error back propagation algorithm and almost all of them use an iterative adjustment of the weights based on the actual error found in the output of the network. One iteration can be generally expressed by the following steps:

1. Provide the network with an input vector x that is currently in the training set.
2. Determine the response y to this input vector.
3. Compute the error vector **e = t - y**.
4. Compute the error vector **e** over all neurons in all layers.
5. Based on the error of each neuron, locally adjust the weights of these neurons.

The procedure is repeated for all samples of the training set, which constitutes one epoch of computation. Then the learning continues with further epochs until the termination condition is satisfied. According to the chosen algorithm tens to thousands of epochs are applied.

## 3.1 Error Back Propagation Algorithm

An error back propagation algorithm is derived for FFNNs with continuous differentiable activation functions, with sigmoid, hyperbolic-tangential activation functions and linear activation functions being the most used in practice. Although the goal of the algorithm is to minimize the error function, it is based on the idea that by changing the weights and threshold, the error of each sample defined by the relation (5)

$$E' = \sum_{j=1}^{Q}\left(t_j - y_j\right)^2 = \sum_{j=1}^{Q}\left(e_j\right)^2. \tag{5}$$

In which $Q$ is a number of outputs; $t_j$ are target values for *j*-th output; $y_j$ are real output values of *j*-th output; and $e_j$ are errors of *j*-th neuron output.

Thus, after calculating the response to the current training set sample, each weight is adapted by the increment. This increment is computed depending on the direction and magnitude of the gradient function, so the following dependence (6) can be considered

$$\Delta w_{ij}^{k} \sim \frac{\partial E'}{\partial w_{ij}^{k}}. \tag{6}$$

The determination of the derivative on the right-hand side of the dependence (6) is shown in detail in, for example, in (Hyakin, 1999) and will not be discussed here for clarity. The resulting relation for the increment of weights and bias is

$$\Delta w_{ij}^{k} = \alpha \delta_j^k y_i^{k-1}. \tag{7}$$

In which $\alpha$ is the learning rate; *i* is source neuron index; *j* is index of neuron for which the weight will be adopted; $\delta_j^k$ is the local gradient of neuron which weight is going to be adopted; and $y_i^{k-1}$ is the output of a source neuron.

From the principle, it is clear that to calculate the local gradients of the neuron in the *k*-th hidden layer, we already need to know the local gradients in the nearest layer of the network on the right (layer *k* + 1). Thus, the procedure for determining these values must start by determining the local gradients in the output layer, then in the last hidden layer, the second to last hidden layer, and so on up to the first hidden layer. Hence the name of the algorithm - the error in the output of the network is propagated to the hidden neurons in the form of local gradients against the direction of the signal when calculating the network response.

### 3.2 Initialization of the Weights and Bias of the Neuron

The initial values of the weights and the threshold define the point on the error function hyperplane from which the algorithm starts. For the error back propagation algorithm, its choice is crucial because the algorithm inherently finds the local minimum of the error function closest to the initial values in most cases. In the absence of a general valid procedures for choosing the correct initialization, the error propagation algorithm is always called repeatedly from different initial conditions to minimize the probability of inappropriate initialization.

### 3.3 Choice of the Learning Rate Coefficient

The learning rate coefficient defines the step size with which the algorithm moves along the hyperplane given by the error function. Its choice is also not unambiguous, because too large a step can lead to uncontrolled movement on the surface and consequently to not finding the minimum, while a small step, on the contrary, safely leads to the nearest minimum, but the convergence of the algorithm can be very slow.

Thus, the problem is again solved by repeatedly calling the error back propagation algorithm for different values of $\alpha$. If the training set contains values in the interval <-1; 1>, it is advisable to choose on the interval <0.01; 0.1>.

## 4 Conclusions

This chapter has provided a comprehensive introduction to the multilayer perceptron more known as feed forward neural networks (FFNN), a pivotal advancement in the field of neural networks that extends beyond the capabilities of the simple perceptron by incorporating multiple layers and non-linear activation functions. The depth and complexity of FFNN's allow them to model the non-linearity in real-world data effectively, making them a crucial tool in the machine learning toolkit.

## 5 Literature

DOLEŽEL, Petr. Úvod do umělých neuronových sítí. Online. 1. Pardubice: Univerzita Pardubice, 2016. ISBN 978-80-7560-022-6.

HAYKIN, Simon. Neural networks: a comprehesive foundation. 2nd ed. Upper Saddle River: Prentice-Hall, 1999. ISBN 978-0132733502.

SAMARASINGHE, S. Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition. Auerbach Publications, New York, 2006. ISBN 978-0849333750.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. Online. Psychological Review. 1958, issue 65, no. 6, p. 386-408. ISSN 1939-1471. At: https://doi.org/10.1037/h0042519.

RUMELHART, David E.; HINTON, Geoffrey E. a WILLIAMS, Ronald J. Learning representations by back-propagating errors. Online. Nature. 1986, issue 323, no. 6088, p. 533-536. ISSN 0028-0836. At: https://doi.org/10.1038/323533a0.

## List of abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| FFNN | Feed Forward Neural Network |
| MLP | Multilayer Perceptron |

## Index

# Artificial intelligence in automation

## Topic 4: Multilayer Perceptron II – Universal Approximation, Modeling of Dynamical Systems, Process Control.

**Study Objective**

The study objective is to elucidate the capabilities of feed forward neural networks (FFNN) as universal approximators, explore their role in modeling dynamical systems, and discuss their applications in process control, highlighting how these advanced functions contribute to solving complex real-world problems.

**Time Required for Study**

2 hours

**Keywords**

Multilayer Perceptron, Neural Networks, Time-Series Prediction, Universal Approximation

## 1 Introduction

This chapter will build on the basic concepts of feedforward neural networks (FFNNs) introduced earlier and focus on their advanced applications and theoretical foundations. FFNNs are not limited to simple pattern recognition tasks but can perform complex functions such as universal approximation, modeling of dynamical systems, and control of sophisticated processes. This chapter will discuss the versatility of FFNNs and show their effectiveness as universal approximators, their usefulness in the prediction and control of dynamical systems, and their strategic applications in process control. Through these discussions, the aim is to highlight how FFNNs can serve as a key tool in solving and addressing real-world problems in various scientific and engineering fields.

## 2 FFNN as a Universal Approximator

Universal approximation is a fundamental concept in the field of neural networks, which states that a neural network with one hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets, subject to certain assumptions about the activation functions. This theorem highlights the theoretical ability of the FFNN to serve as an incredibly powerful tool in a variety of approximation problems. The

Russian mathematician Kolmogorov proved that any continuous function of n variables can be expressed by a finite set of functions of one variable (Kolmogorov, 1957) The formal similarity of this theorem to feedforward neural networks was pointed out by the American computer scientist (Hecht-Nielsen, 1987). Finally, the Austrian statistician Hornik proved that any continuous function can be represented with the required accuracy using a feedforward multilayer artificial neural network with one hidden layer (Hornik, 1989). However, the above works do not provide an analytical procedure to find a particular network approximating a given function, and therefore the approach is usually taken experimentally. This problamatics will be described with a simple example below.

## 2.1 Example of Universal Approximation

Let's imagine that we want to create a converter of the temperature provided by the sensor to the real temperature. In the example, consider a temperature sensor measuring temperature in the range 0-500 °C. It is a digital sensor with a 10-bit analog-to-digital converter, so it returns a value of 0-1023 depending on the temperature. Using the calibration of the sensor, we need to design an optimal neural network that transforms the output of the sensor to the temperature of the measured environment. The situation is illustrated in figure 1.

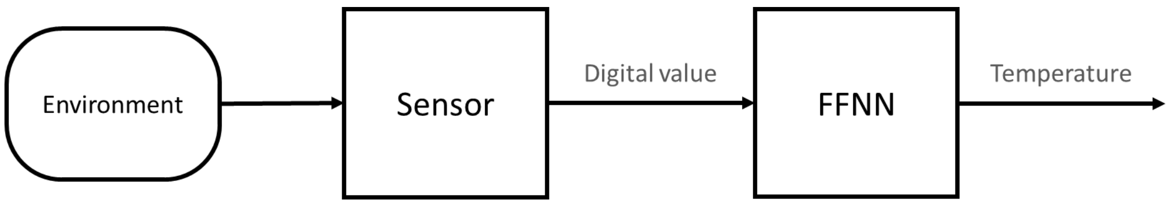


**Figure 1 – Approximation example.**

## 2.2 Training Data Obtaining

Data for training and validation of the network should be obtained by experimenting with a benchmark. The experiment setup is shown in Figure 2. The environment in which the sensor and the reference are placed should be set to different temperatures and the data measured by the standard and provided by the sensor under discussion should be recorded. This will give a table of corresponding values.

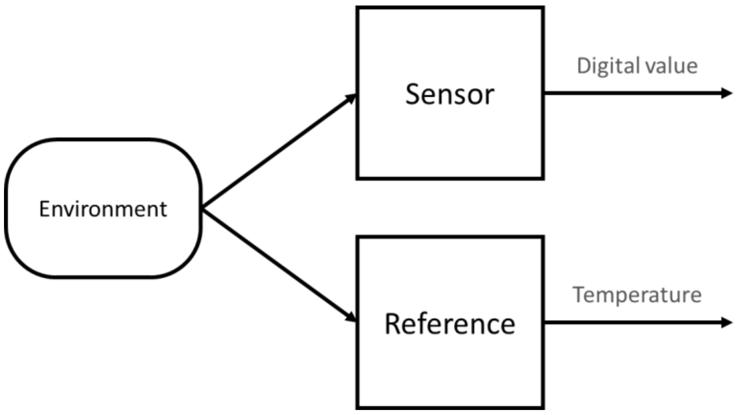


**Figure 2 – Experiment settings for data obtaining.**

Depending on the measurement, a table of digital values and corresponding measured temperatures provided by the reference is obtained. Usual practice is to create a set with enough data to cover the approximation problem. In our case it is optimal to measure let's say 100 input-output pairs of values. These values need to be suitably divided into training, validation, and test sets, where the inputs are digital values, and the expected output is the measured temperature. Typical split ratio between all sets is 70:15:15. The data itself should be preprocessed, which typically means normalizing each sample to the interval <-1; 1>.

## 2.3 Topology Design

Given the nature of the approximated dependence, a feedforward multilayer artificial neural network with one hidden layer seems to be appropriate. We choose a hyperbolic-tangential activation function in this layer and a linear activation function in the output layer. It is also important to note that the training set in our case contains relatively few values (70 pairs), and the number of optimized weights and thresholds should not exceed this number in any case, rather it should be an order of magnitude lower. Thus, the most complex topology that could theoretically be considered is a network with 23 neurons in the hidden layer. This is because it contains 23 weights between the input and the hidden layer, 23 weights between the hidden layer and the output, 23 thresholds of the neuron in the hidden layer and 1 threshold of the output neuron.

A suitable procedure for determining the topology is to perform a series of different experiments. Thus, we will use the back propagation error algorithm (learning rate coefficient $\alpha = 0.1$) to train different topologies and the criterion will be the value of the error function E(s) computed on the test set data. For each topology, we perform 100 training epochs to get statistically significant results. The final results of the E(s) values for the different topologies are shown in the form of box plots in Figure 3, where the middle "box" part of the plot is bounded above by the third quartile, below by the first quartile, and between them is a red line showing the median of the values. The lines extending from the middle part of the diagram perpendicularly up and down express the variability of the data below the first and above the third quartile. Outliers values are then plotted as individual crosses. The figure shows that the medians and minima of the final values of the error functions decrease approximately to a topology with 7 neurons in the hidden layer. Let us therefore state this topology as suitable.

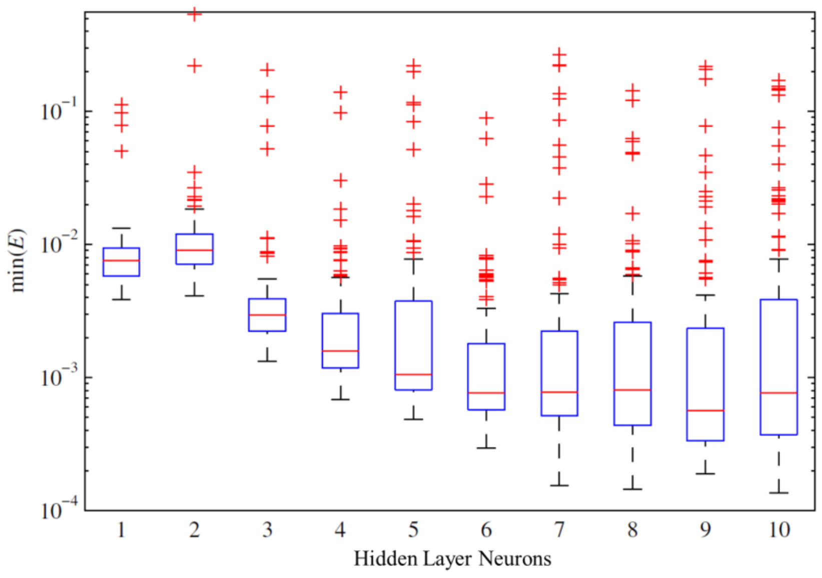


**Figure 3 - Finding a suitable network topology.**

## 2.4 Approximation model evaluation

The final step is to assess the quality of the resulting network, for which a test dataset is used. It is up to the user to decide if the accuracy achieved is sufficient. The data can of course be subjected to various statistical analyses to quantify the accuracy of the response of the resulting network. In case the network is not sufficient, the whole process of network building must be repeated. A further improvement can be the use of more efficient learning algorithms, which can achieve significantly better results.

# 3 FFNN for Modelling and Prediction

Mathematical modelling has long been the focus of attention of experts in many fields. Not only models of technical devices, but also of the most crucial economic, biological, and other systems are created. Models can then be used to better understand the system under investigation, to verify theoretical assumptions, to optimize a process, or to predict behavior.

Each system (the term process or system is also used) can be represented as a block, which is affected by input signals (inputs can be summarized in the vector **u**) and possibly disturbances, and whose behavior can be monitored based on output variables denoted by the vector **y**.

If a mathematical model describes the dependence between signals in a steady state form, it is called a static model and is an approximation problem. A dynamic model describes the behaviour of the system in the case (in a transient state). A static model is described by a system of algebraic equations, a dynamic model by a system of differential equations (continuous model) or a system of differential equations (discrete form model).

In principle, the methods of mathematical and physical analysis or experimental identification (or a combination thereof) may be used in the construction of the model. While mathematical-physical analysis decomposes the model into simple parts described by balance equations based on physical laws, experimental identification builds the model and its parameters based on signal measurements. The procedure is to compare the behavior of the process under investigation with the behavior of the chosen model according to the chosen criterion.

It is obvious that the resulting model may have a shape that does not correspond to the corresponding physical model, but on the other hand it is possible to create models of different systems by the same procedure without knowing their internal processes. Detailed information about modelling in both ways can be found, for example, in (Mikleš, 1999).

## 3.1 Static Model

The design of the static neural model is therefore proceeded in the way described in the previous subsection. The training, test and validation data sets are obtained by applying constant inputs (chosen sequentially from the whole range of inputs) to the modelled system and measuring the corresponding steady-state outputs (the measurement procedure corresponds to the determination of the static characteristics of the system). The measured data are then divided into individual sets by default.

## 3.2 Dynamic Model

The dynamic model of the system describes its behaviour in transient states. There are several ways to model a dynamic system using an artificial neural network. Probably the most natural approach consists in converting the model into a discrete form by choosing a suitable sampling interval and following a procedure analogous to experimental identification methods (Mikleš, 1999), only instead of the parameters of the differential equation, the optimized the values of the weights and thresholds in the artificial neural network.

This procedure requires the determination of the number of input and output of the artificial neural network and the use of a special shape of the training and testing set, but the type of topology itself neural network and the training algorithm remain identical to the topologies and algorithms used in the creation of the of the static model in the previous paragraphs.

For the sake of clarity, only the creation of a dynamic model of a single-input system will be outlined here and one output (SISO system), but the procedure for creating a multidimensional system model can be obtained by an obvious extension of the above procedure. A continuous system can be made by choosing a suitable sampling interval can be converted to discrete and then its behaviour can be described in general by the equation

$$y_s(k) = \vartheta[y_s(k-1), y_s(k-2), \dots, y_s(k-n), u(k-1), \dots, u(k-m)], m \leq n\ . \tag{1}$$

In which the *k* is a discrete time; *n* is the order of system; *u*(*k*) is system input in discrete time *k*; $y_s(k)$ is system output; and $\vartheta$ is general mathematical function.

Schematically, the differential equation can be represented as shown in Figure 4. The symbol $z^{-1}$ represents the shift of the respective values in the case, e.g. $z^{-1}u(k) = u(k - 1)$.

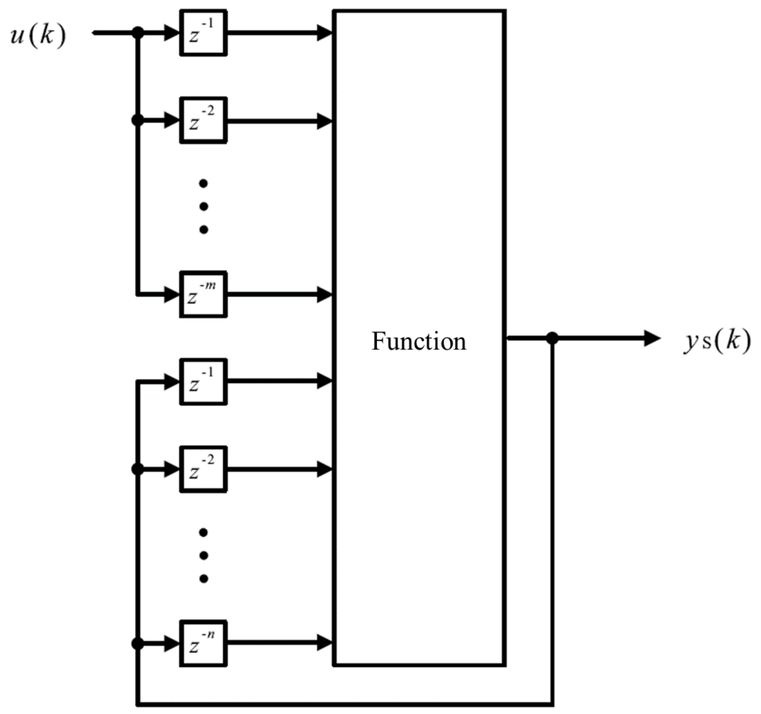


**Figure 4 – Example of a dynamic model.**

The aim of the design is to match the output *y* and $y_s$ as closely as possible with the same inputs. The network design procedure is again like the information presented in previous chapter. Thus, the first step is to determine the training, test, and validation data sets.

These data are obtained by experimenting with a modelled system, where a time-varying signal (linear, sinusoidal, rectangular, sawtooth, or some kind of random signal, or a combination of these) is applied to the input of the system and the response of the system is measured. It is highly desirable that the system be measured over the entire operating range, at all operating frequencies, and that steady states be present in the experiment if the system permits.

Thus, if appropriate values are measured, these values should be adjusted to suitably represent the training, testing and validation data set. At this point, however, the value of the system order *n* must already be known. Parameter *m* is usually chosen either *m* = *n* or *m* = 1. While in experimental identification linear systems, several procedures are known to determine the order of *n*, in the case of nonlinear systems it is not, so it is usually necessary to determine the order of the system experimentally, that is, to find the lowest order one by one, at which the properties of the model are still sufficient.

Thus, although the actual learning of an artificial neural network is not different from that of static modelling, the overall the process of building a dynamic neural model is more challenging and sensitive to a variety of circumstances (the choice of model order, choice of sampling interval, etc.). On the other hand, a dynamic model allows to predict the future behavior of the system, so it has several practical applications.

## 4 Conclusions

This chapter explores the advanced features and theoretical capabilities of feedforward neural networks (FFNNs) and highlights their versatility and effectiveness as tools in scientific and engineering applications. FFNNs are shown to be indispensable from the universal approximation, which highlights the theoretical potential of FFNNs to model any continuous function, to their practical applications in modeling dynamical systems and process control.

## 5 Literature

DOLEŽEL, Petr. Úvod do umělých neuronových sítí. Online. 1. Pardubice: Univerzita Pardubice, 2016. ISBN 978-80-7560-022-6.

MIKLEŠ, J., and FIKAR, M. Modelovanie, identifikácia a riadenie procesov I. STU, Bratislava, 1999. ISBN 80-2272-134-4

## List of abbreviations

FFNN - Feedforward Neural Network

SISO - Single Input Single Output

## Index

# Artificial intelligence in automation

## Topic 5: Introduction to Image Data Processing. Process of Digital Image Creation. Digital Image Representation, Basic Image Editing Operations.

**Study Objective**

The primary objective of this topic is to equip students with a comprehensive understanding of image data processing within the context of artificial intelligence and automation. As we delve into the fundamental principles and applications of image data processing, students will gain the necessary skills to analyze, interpret, and process digital images using different techniques.

**Time Required for Study**

2 hours

**Keywords**

Image Processing, Digital Images, Image Representation, Image Editing, Pixel, Color Models,

## 1 Introduction to Image Data Processing

Image data processing is a fundamental aspect of artificial intelligence that involves the interpretation and manipulation of image data. This area cuts across many disciplines, including computer vision, machine learning and robotics, and has important applications in automation. As industries increasingly focus on automation to increase efficiency and reduce human error, the role of image data processing is growing.

The process starts with image acquisition, which is usually captured by sensors or cameras. These images are then converted into digital form for further processing. This digital transformation is essential because it allows computers to perform various operations to extract valuable information from the images. (Gonzales, 2018)

Image processing technology relies heavily on algorithms and techniques developed in the field of machine learning. These algorithms can learn to identify patterns and features in images that are important for specific applications. For example, convolutional neural networks (CNNs) are particularly effective for image recognition tasks because they can automatically learn feature hierarchies that are most informative for classification.

Moreover, image data processing is not limited to image recognition or classification. It also includes operations such as image enhancement and restoration, object detection, segmentation, and image generation. Each of these operations plays a key role in enabling machines to interpret the visual world around them in a manner like humans, but with speed and accuracy that is unparalleled. (Szeliski, 2022)

More detailed information on more advanced applications will be discussed in later chapters, and the current study text will focus on the basic operations of image generation and processing.

# 2 Image Creation

The image is usually obtained by external stimulation, which means the interaction of radiation with matter. In specific cases, such as welding, the sensed object itself may emit radiation, which is used by the system to acquire the image. The illumination for external stimulation most often focuses on the visible spectrum (VIS) but can also include the near infrared (NIR) or ultraviolet (UV) region.

The main purpose of illumination in these systems is to maximize contrast in areas that are relevant to the application while minimizing contrast in other parts of the scene. This is important to ensure robustness and repeatability of measurements. Creating contrast can be achieved by differential absorption or emission of radiation by objects in the scene or by appropriate illumination that creates differences in brightness based on the illumination geometry.

Improving contrast and image robustness is also possible using optical filters. These filters can selectively block or pass light at specific wavelengths, thereby improving the quality of the acquired image for the desired application.

In practical applications, the lighting and filtering can be adjusted to highlight specific features of an object, such as material defects, bar codes, or other prominent features that are important for a particular inspection or recognition task. An example of the image formation is shown in the figure 2.

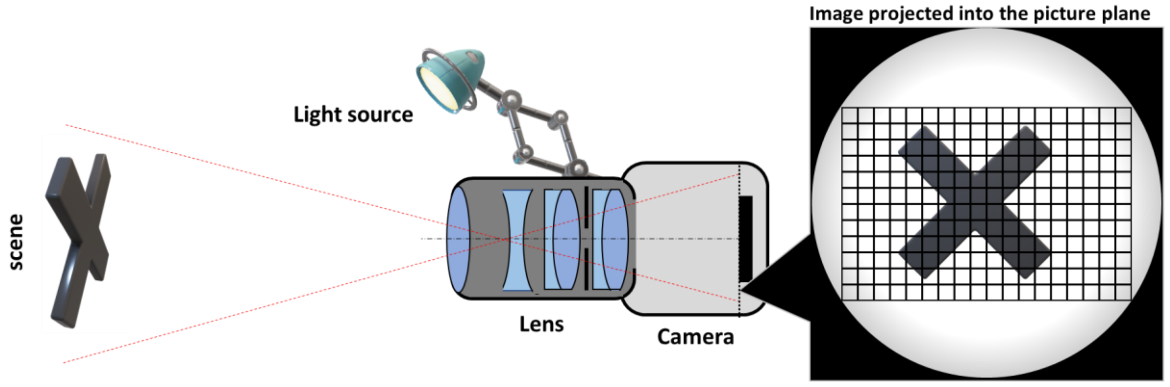


**Figure 1 – Image creation scheme.**

The reflection of the light from the object is therefore captured and steps are taken to digitize it. In a digital camera, the image is captured by an image sensor where each photodiode captures a portion of the projected image, and the original continuous image is divided into M×N samples arranged in a square grid. One of these points represents one image element of the captured image (pixel, px).

Subsequently, when reading data from the sensor, the analog signal is converted to digital (we perform quantization), where the quantization converts a continuous gray scale to a discrete gray scale. The number of luminance levels of the resulting scale determines the luminance resolution of the image, and quantization always results in irreversible loss of some information.

## 3 Color

The interaction of radiation with matter is a fundamental phenomenon that allows machine vision systems to detect and interpret objects. This phenomenon occurs when electromagnetic radiation strikes an object and excites valence electrons, i.e. electrons located in the outer orbits of atoms. The material of the object absorbs photon energy equal to the energy required to excite these electrons.

When the radiation is absorbed, it is transformed. The reflected radiation is then depleted of the absorbed photons and therefore a fraction of its wavelength. The energy of the photons is directly related to the wavelength - photons with different energies have different wavelengths. In practice, when white light containing all the wavelengths of the visible spectrum is incident on an object and some of them are absorbed, the reflected light is then perceived as a particular color. The color of the object that we perceive is a complementary color to those that have been absorbed. An example of complementary colors is shown in the following figure 2.

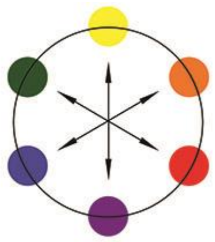


**Figure 2 – Complementary colors.**

If an object absorbs all wavelengths of light in the visible region, we perceive it as black. Conversely, an object that fully reflects all wavelengths is perceived as white. This phenomenon is the basis for color and hue discrimination in machine vision systems, which can be programmed to detect specific colors or patterns based on reflected light.

### 3.1.1 Color models

Color models are mathematical models that describe how colors can be represented in various devices such as digital cameras, monitors, printers, or machine vision systems. There are several basic models, each with its own specific applications and characteristics. Among those used in the creation of a digital image are the following:

- RGB Model
    - The RGB model is probably the best known and most used color model in video and digital imaging devices. The abbreviation RGB comes from the words Red, Green, Blue - red, green, blue. These three colors are considered additive primary colors because by combining them in different intensities, a wide range of other colors can be created. In machine vision systems.
- HSV model (Hue, Saturation, Value)
    - Trying to better match the human perception of color.
    - Hue – Specifies the type of color (red, green, blue, etc.). Hue is represented as an angle on the color wheel, where each color corresponds to an angle.
    - Saturation – Expresses the intensity or purity of a color. A saturation approaching 100% indicates a pure, strong color, while a lower saturation tends toward a whiter or grayer color.
    - Value – Also known as brightness, this component determines how light or dark a color is. A higher value indicates a lighter color, while a lower value represents a darker color.

# 4 Basic Image Operations

Image editing is a key part of digital image processing and involves techniques that modify an image to improve its aesthetics or extract important information. Basic image editing operations include cropping, resizing and filtering. These operations are not only fundamental in fields such as graphic design and digital art, but are also essential in technical fields such as medical imaging, satellite image analysis, and automated quality control systems.

Cropping is used to remove unwanted external areas from an image. This operation helps to focus the viewer's attention on the most important parts of the image and improves the overall composition.

Resizing changes the dimensions of an image, either to enlarge or reduce it. Resizing is done using various interpolation techniques such as nearest neighbor, bilinear and bicubic interpolation. Each technique has its own advantages and is selected based on the speed and image quality requirements of the application. For example, nearest neighbor interpolation is faster but may introduce jaggedness into the image, making it less suitable for high quality magnification.

Filtering involves changing pixel values in the image to achieve effects such as blurring, sharpening, edge detection, or noise reduction. Filters can be used to enhance image quality or to prepare an image for further analysis, for example in edge detection algorithms, which are crucial in object recognition and tracking systems. Mathematical insight:

- A common approach to filtering is to use a convolution matrix (or kernel) that is applied to each pixel and its neighbors in the image. For example, a simple blur might use a 3x3 matrix, where each value is 1/9, representing the average of a pixel and its eight neighbors.

In an automated production line, cameras continuously scan products for quality control. The image processing system can automatically crop the images to an area of interest, resize them to a standard format for analysis, and apply a filter to highlight features such as edges. This pre-processing can greatly improve the accuracy of subsequent automatic defect detection.

## 4.1 Edge Detection Filters

Edge detection filters are designed to highlight edges in images by detecting discontinuities in brightness or color. These filters work by enhancing the contrast at the boundaries between different areas of the image. The most commonly used edge detection filters include:

- Sobel filter – This operator uses two separate 3x3 kernels that are convolved with the original image to compute approximations of the derivatives - one for horizontal changes and one for vertical changes. This filter is particularly effective in detecting edges in high contrast areas. (Gonzales, 2018)
- Prewitt filter – Similar to the Sobel filter, the Prewitt filter uses two 3x3 kernels. Compared to the Sobel filter, it is simpler but less accurate in edge detection. The Prewitt filter places more emphasis on isotropic response, but is more sensitive to noise. (Gonzales, 2018)
- Canny detector – One of the most widely used edge detection tools, is an algorithm that aims to find the optimal edge detection by minimizing error, ensuring that the detected edges are as close as possible to the actual edges, and having a clear edge detection response. The process involves several steps including smoothing the image with a Gaussian filter, finding gradients, suppressing non-maximum values and hysteresis thresholding. (Szeliski, 2022)
- Laplacian of Gaussian (LoG) – This operator first smooths the image with a Gaussian filter to reduce its sensitivity to noise, and then uses a Laplacian filter to detect edges. The LoG filter helps to highlight areas of rapid intensity change and is particularly useful for detecting blobs in addition to edges. (Nixon, 2012)

The following figure 3 shows an example of edge extraction using the Sobel filter.

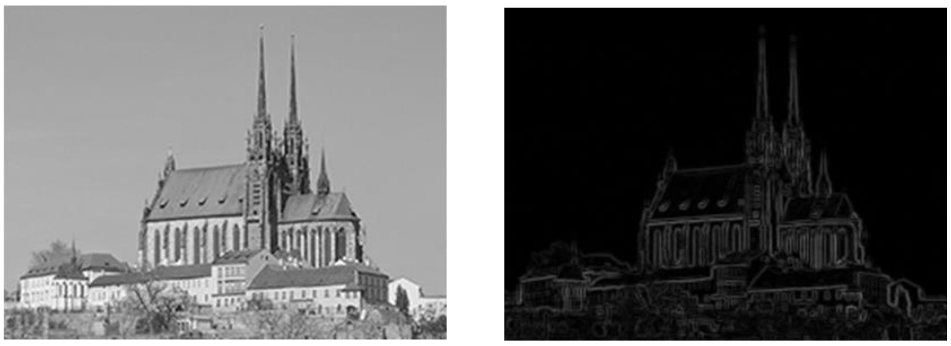


**Figure 3 - Example of edge extraction using the Sobel filter.**

## 5 Conclusions

In this topic, we have discussed the fundamental aspects of image data processing, a key field of artificial intelligence and automation that significantly enhances our ability to interpret and manipulate digital images. Starting with the process of creating digital images, we delved into the technical aspects of how images are captured and converted into digital formats that offer a foundation for further manipulation and analysis.

In discussing the representation of digital images, we revealed how images are encoded using different color models and pixel arrangements that are crucial for subsequent image editing and processing tasks. This led us to explore basic image editing operations such as cropping, resizing, and filtering, which are essential tools in engineering and scientific fields.

## 6 Literature

GONZALEZ, Rafael C. a WOODS, Richard E. Digital image processing. Fourth edition. New York: Pearson. 2018. ISBN 978-013-3356-724.

NIXON, Mark S. and AGUADO, Alberto S. Feature Extraction & Image Processing for Computer Vision. Online. Amsterdam: Elsevier, 2012. ISBN 9780123965493. At: https://doi.org/10.1016/C2011-0-06935-1.

SZELISKI, Richard. Computer Vision. Online. Texts in Computer Science. Cham: Springer International Publishing, 2022. ISBN 978-3-030-34371-2. At: https://doi.org/10.1007/978-3-030-34372-9.

## List of abbreviations

AI - Artificial Intelligence

CNN - Convolutional Neural Network

VIS - Visible Spectrum

NIR - Near Infrared

UV - Ultraviolet

RGB - Red, Green, Blue

HSV - Hue, Saturation, Value

## Index

# Artificial intelligence in automation

## Topic 6: Convolutional Neural Network I – Signal and Image Processing.

**Study Objective**

The aim of this topic is to provide a comprehensive introduction to convolutional neural networks (CNNs) with a particular focus on their use in signal and image processing. Students will learn about the CNN architecture, including the different layers that facilitate pattern recognition and feature extraction. The chapter will detail CNN mechanisms such as convolutional layers, pooling, and fully connected layers, and their role in efficient signal processing and two-dimensional images.

**Time Required for Study**

2 hours

**Keywords**

Convolutional Neural Networks, Feature Detection, Signal Processing, Image Classification

## 1 Introduction to Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of deep neural networks that are particularly powerful for tasks involving image and signal processing. CNNs are primarily designed to process data in the form of multiple arrays, such as a color image composed of three 2D arrays containing pixel intensities in an RGB color model, and automate the extraction of critical features, making them essential for computer vision tasks.

A CNN typically consists of an input layer, several hidden layers, and an output layer. The hidden layers typically include several convolutional layers that convolve their input with a set of learned filters, providing robust feature extraction capabilities. Convolutional layers are followed by pooling layers (also known as down-sampling layers), which reduce the spatial size of the representation, thereby reducing the number of parameters and computations required in the network, thus increasing efficiency. (Hyakin, 1999)

Each convolutional layer applies a series of filters to the input. Where each filter implements the detection of a feature that is present in the data. For example, a simple edge detection filter may highlight areas of intense color change, while others may detect specific shapes or textures. The result of this convolutional process is a feature map that highlights specific features in the input. The networks use nonlinear activation functions that introduce nonlinearities into the model and help the network learn complex patterns. (Horn, 2020)

## 2 History of Convolutional Neural Networks

Convolutional neural networks (CNNs) are one of the most powerful tools for image data analysis and pattern recognition today, but their roots go back to the mid-20th century. The origins of CNNs date back to 1959 with the pioneering work of David Hubel and Torsten Wiesel, who studied the human visual system and came up with the theory that certain types of cells in the visual cortex have special roles in pattern recognition. They identified two types of cells: Simple cells (S-cells), which respond to edges and stripes of a certain orientation in each perceptual field, and Complex cells (C-cells), which respond to similar stimuli but are tolerant of their shifts in the visual field. (Carandini, 2006)

These discoveries inspired Kunihiko Fukushima in 1980 to create the Neocognitron, a model of a self-organizing neural network for pattern recognition that is unaffected by their displacement in space. Neocognitron incorporated S-cells and C-cells implementing mathematical operations and introduced the concept of "simple to complex" in the context of machine learning. (Fukushima, 1982)

Another landmark was the development of Yann LeCun's LeNet in 1998, which applied gradient learning to document recognition and opened the door for modern CNN development (Lecun, 1998). In 2009, Fei-Fei Li and her team created ImageNet, a large dataset for computer vision research. This dataset contains over a million images divided into 1000 classes and spurred competition among research teams in image data classification, leading to the development of new topologies and architectures for convolutional neural networks.

In 2012, AlexNet was introduced by Alex Krizhevsky, which implemented ReLu activation function and used many filters in convolutional layers. This network also made use of parallel training architecture, data augmentation, and the introduction of a Dropout layer, a technique where the output of a certain percentage of neurons is set to zero to reduce neuron interdependencies and avoid overlearning the network. AlexNet achieved a breakthrough performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), validating the potential of CNNs in practice. (Krizhevsky, 2017)

These developments are critical to understanding how CNNs work today and how they have evolved into tools that have become the basis for many machine vision applications. With each of these advances, new opportunities for improvement, accuracy, and efficiency in image data recognition have opened, leading to a revolution in automation and enabling new applications such as autonomous driving, advanced diagnostics in healthcare, intelligent video analysis, as well as new image generation.

## 3 Problems to Solve Using CNNs

Convolutional neural networks (CNNs) are particularly powerful for tasks involving the analysis of visual images and have made significant inroads in processing other types of data fields. This subsection will detail some of the critical problems that CNNs are uniquely equipped to solve, demonstrating their versatility and effectiveness.

- Image Classification – One of the most common applications of CNNs is image classification, where the network must classify an input image into one of several predefined classes. CNNs excel in this area due to their ability to extract key features from images at different scales and through different filters.
- Object Detection – Unlike image classification, which labels the image, object detection identifies multiple objects in an image and classifies each one. This is crucial for applications such as surveillance, where objects (people, vehicles) need to be recognized in different scenes, or for autonomous driving, where detection of pedestrians, signs and vehicles is critical for safe navigation. Other applications can be detection for robotic manipulation.
- Semantic an Instance Segmentation – Semantic segmentation involves labeling each pixel in the image with the class of what is represented. This is particularly useful in medical imaging, for example, when segmenting different types of tissues, tumors, or organs in magnetic resonance imaging or computed tomography. Instance segmentation then also assigns its own class to each instance of the object.

Classification is concerned with assigning the input image to the appropriate class. For example, CNN can recognize whether an image depicts a cat or a dog. Localization goes one step further and not only determines the class of the object, but also finds its location in the input image. Detection is a combination of classification and localization where the system finds and classifies all objects in the image. Segmentation then divides the image into segments, with two main types: semantic segmentation, where each pixel is assigned to a class, and instance segmentation, which labels the pixels corresponding to each instance of the object. An example of mentioned problems to solve with CNN is shown in the figure 1.

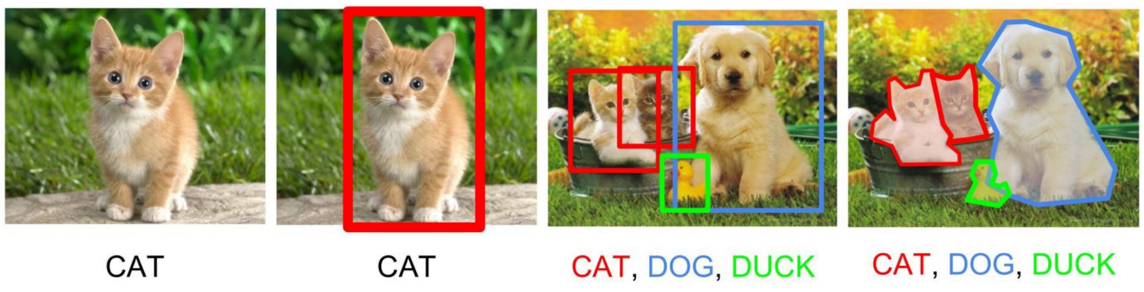


**Figure 1 – Examples of CNN usage.**

## 3.1 CNN for Classification

In convolutional neural networks, typically instead of manually selecting features such as edges, shapes, and colors, CNNs automatically detect features in the input data. These features, or patterns, can be of different sizes and locations and are captured by detectors that scan the data. These detectors are implemented by convolutional layers.

### 3.1.1 Convolutional Layer

A convolutional layer consists of a set of filters that perform a convolution operation on the input data. Each filter in the convolutional layer has a certain defined kernel size and its weights are obtained during the training process. For each pixel, the scalar product of the convolution filter with the input is applied, producing a matrix of flags. This matrix shows where the dominant features corresponding to the filter occur in the original image. The figure 2 shows a one step of procesing the convolution.

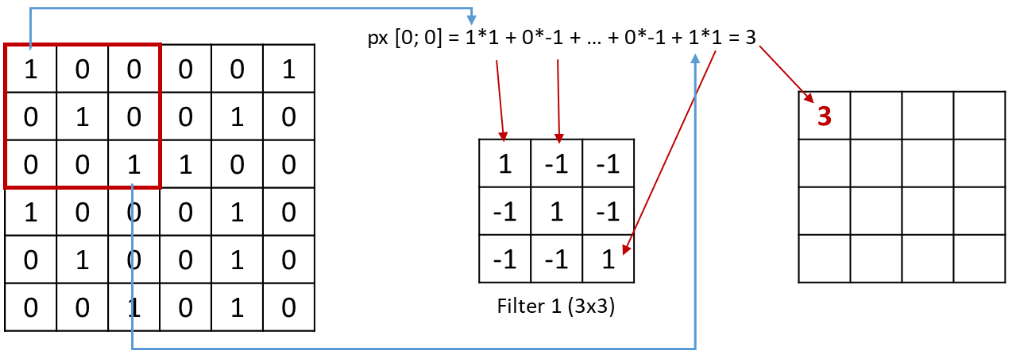


**Figure 2 – One step of calculating convolution.**

In the convolutional layer, in addition to the kernel size, we can also define the size of the shift, called stride, and then also the possible padding of the image to avoid reducing the size of the input image. An example of padding is shown in the following figure 3.

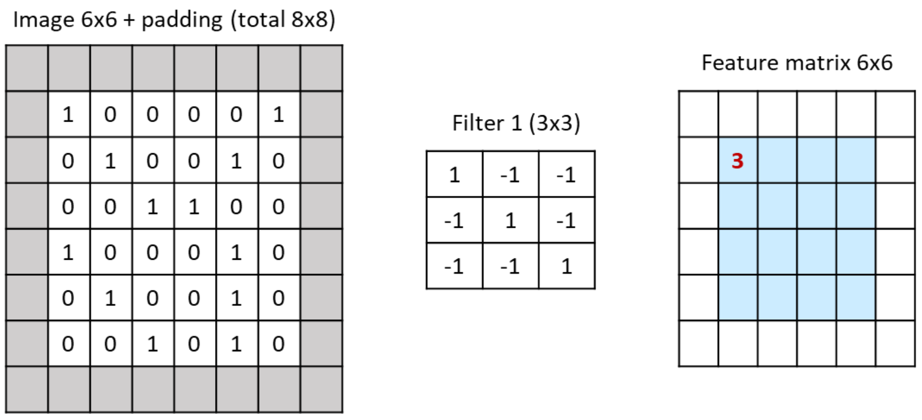


**Figure 3 - Example of padding.**

The next step after convolution is the selection of key features for further processing which is done by pooling layer.

### 3.1.2 Pooling Layer

Pooling layers then further process this map of flags by pooling groups of features into a single value, typically using a max pooling method that selects the maximum value from a given window, thus reducing the dimension of the input data. The principle is shown in the following figure 4.

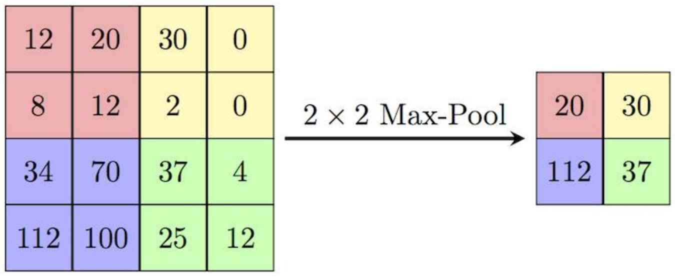


**Figure 4 – Max-Pooling example.**

### 3.1.3 Overall Structure

These layers (Convolutional and Pooling) are usually paired and arranged in sequence resulting in the extraction of increasingly complex features.

The overall structure then consists of a part that extracts features and a part that performs their classification. The extraction is performed by pairs of convolutional and pooling layers, and the classification is then performed by a simple feedforward neural network for classification, which has a softmax activation function at the output. The topology if CNN for classification is shown at figure 5.

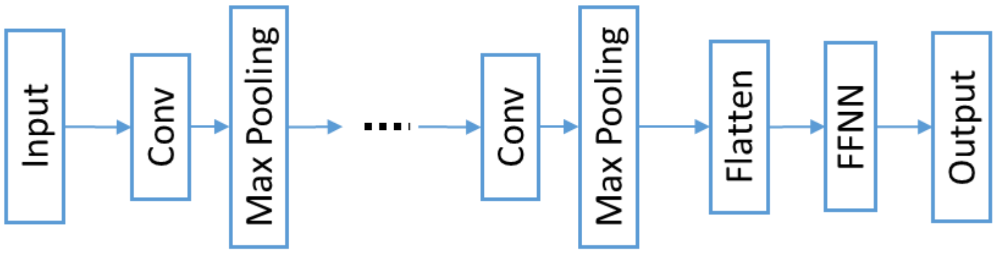


**Figure 5 – CNN for classification topology.**

## 3.2 CNN for Detection

Detection by CNN can be easily done using a method called the sliding window method. In this method, small windows are cut out from the original image and then each cut out is classified. This really requires a lot of classifications, and if we consider that the objects may be of unequal sizes, then it is almost, in a reasonable time, an unsolvable problem. For this problem, networks have started to be used that predict the occurrence of the seed region and then perform classification. The first representative is R-CNN, which improves the performance of object detection by combining high-capacity convolutional neural networks with region design techniques (Girshick, 2014). Here we show how it works in each step:

- Region Suggestion – R-CNN initially uses a selective search algorithm to generate approximately 2000 region suggestions from the image. These region suggestions are candidate bounding boxes that are likely to contain objects.
- Feature extraction – Each proposed region is then deformed to a fixed size and inserted into a pre-trained CNN (e.g., AlexNet) to extract a fixed-length feature vector. The CNN acts as a feature extractor and the output are a 4096-dimensional vector (if using AlexNet).
- Classification – The extracted features are then used by a set of class-specific linear SVMs (Support Vector Machines) to classify the object within each region. Unlike the use of softmax in typical classification tasks, R-CNN uses SVMs for more accurate classification based on the features extracted by the CNN.
- Bounding box regression - To improve the accuracy of the bounding box around the detected objects, R-CNN also uses a linear regressor for each region design to output narrower bounding boxes.

## 3.3 CNN for Segmentation

Image data segmentation is an advanced method in computer vision that assigns a class (label) to individual pixels in an image. This method is crucial in applications where accurate recognition and localization of objects and their boundaries is important, such as medical imaging, autonomous vehicles, or remote sensing.

Segmentation is divided into semantic segmentation, where each pixel is assigned to a class that represents a particular category of objects without regard to individual instances, and instance segmentation, which distinguishes between the individual instances of objects in a class, which is useful for detailed analysis and visual recognition in complex scenes.

Models based on encoder-decoder or auto-encoder architectures are often used for image data processing and segmentation. These models use convolutional layers to "encode" the input information (image) into a more compact form, while the decoder is responsible for "decoding" this information back into a format that displays the segmented image.

The encoder progressively reduces the dimensions of the input image, focusing on extracting key features, while the decoder re-enlarges this information and reconstructs the original image dimensions, with emphasis on locating and identifying individual segments. During this process, several up-sampling techniques, such as replication, averaging or un-pooling, can be used to recover the details in the image. An example of segmentation using an encoder-decoder scheme is the SegNet architecture shown in Figure 6.

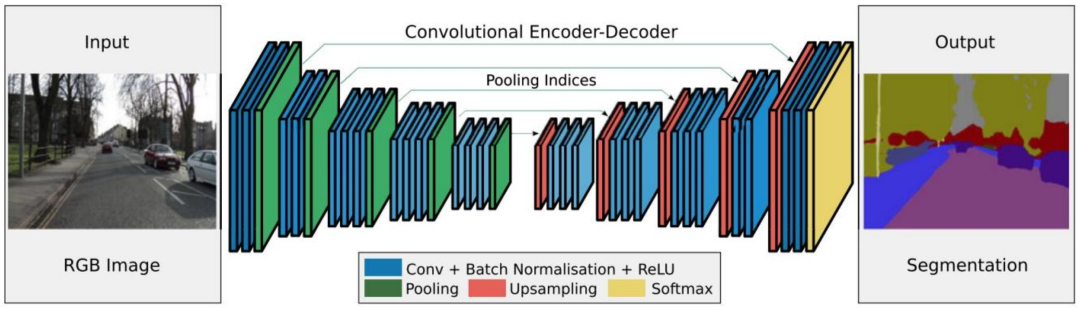


**Figure 6 – SegNet topology. (Badrinarayanan, 2017)**

## 4 Conclusions

In this chapter, we have discussed the rich history, basic principles, and various applications of convolutional neural networks (CNNs) in image and signal data processing. From their roots in the study of the human visual system to their evolution into sophisticated architectures such as LeNet, AlexNet, and others, CNNs have fundamentally changed the landscape of computer vision and machine learning. We delved into the versatility of CNNs and discussed how they solve complex problems such as image classification, object detection, and semantic segmentation.

## 5 Literature

BADRINARAYANAN, Vijay; KENDALL, Alex a CIPOLLA, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. Online. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017, issue. 39, no. 12, p. 2481-2495. ISSN 0162-8828. At: https://doi.org/10.1109/TPAMI.2016.2644615.

CARANDINI, Matteo. What simple and complex cells compute. Online. The Journal of Physiology. 2006, issue 577, no. 2, p. 463-466. ISSN 0022-3751. At: https://doi.org/10.1113/jphysiol.2006.118976.

FUKUSHIMA, Kunihiko and MIYAKE, Sei. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. Online. In: AMARI, Shun-ichi and ARBIB, Michael A. (ed.). Competition and Cooperation in Neural Nets. Lecture Notes in Biomathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, p. 267-285. ISBN 978-3-540-11574-8. At: https://doi.org/10.1007/978-3-642-46466-9_18.

GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor a MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014, p. 580-587. ISBN 978-1-4799-5118-5. At: https://doi.org/10.1109/CVPR.2014.81.

HAYKIN, Simon. Neural networks: a comprehesive foundation. 2nd ed. Upper Saddle River: Prentice-Hall, 1999. ISBN 978-0132733502

HORN Berthold. 6.801: Machine Vision. Online. 2020. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA. At: https://ocw.mit.edu/courses/6-801-machine-vision-fall-2020/

KRIZHEVSKY, Alex; SUTSKEVER, Ilya a HINTON, Geoffrey E. ImageNet classification with deep convolutional neural networks. Online. Communications of the ACM. 2017, issue 60, no. 6, p. 84-90. ISSN 0001-0782. At: https://doi.org/10.1145/3065386.

LECUN, Y.; BOTTOU, L.; BENGIO, Y. a HAFFNER, P. Gradient-based learning applied to document recognition. Online. Proceedings of the IEEE. Issue 86, no. 11, p. 2278-2324. ISSN: 00189219. At: https://doi.org/10.1109/5.726791.

SZELISKI, Richard. Computer Vision. Online. Texts in Computer Science. Cham: Springer International Publishing, 2022. ISBN 978-3-030-34371-2. At: https://doi.org/10.1007/978-3-030-34372-9.

## List of abbreviations

CNN - Convolutional Neural Network

SVM - Support Vector Machine

ReLU - Rectified Linear Unit

R-CNN - Regions with Convolutional Neural Networks

ILSVRC - ImageNet Large Scale Visual Recognition Challenge

## Index

# Artificial intelligence in automation

## Topic 7: Convolutional Neural Network II – Industrial Applications.

**Study Objective**

The aim of this chapter is to elucidate the practical applications of convolutional neural networks in the industrial automation sector, demonstrating how these advanced tools can optimize operations, enhance precision, and streamline processes.

**Time Required for Study**

2 hours

**Keywords**

Convolutional Neural Networks, Feature Detection, Signal Processing, Image Classification

## 1 CNNs in Industrial Automation

In the field of industrial automation, convolutional neural networks (CNNs) have become a foundational technology that is driving significant advances in a variety of industries. With their exceptional ability to process and interpret visual data, CNNs have enabled automation systems to perform tasks requiring complex visual recognition that often surpass human accuracy and speed.

The scientific foundations of CNNs can be traced back to the Neocognitron, which was introduced by Kunihiko Fukushima in the 1980s (Fukushima, 1980), and later refined by Yann LeCun and others who developed practical applications using back-propagation algorithms to efficiently train deep neural networks (Lecun, 1998). These developments laid the groundwork for the use of CNNs in complex image recognition tasks, which are crucial in various industrial applications today.

The transformation of industries through visual recognition is evolving in some key areas:

- Manufacturing – CNNs are widely used in manufacturing for automated visual inspection systems. These systems use CNNs to detect minor defects in high-resolution images of manufactured products, outperforming traditional methods in both speed and accuracy. The ability of CNNs to learn from the vast amount of labeled data allows them to identify patterns and defects that are imperceptible to human inspectors.

- Automotive – In the automotive industry, CNNs facilitate the development of advanced driver assistance systems (ADAS) by processing and interpreting image data in real time. CNNs enable vehicles to recognize objects, assess the environment, and make decisions in a fraction of a second, which is crucial for autonomous driving technologies.
- Pharmaceuticals – for example, the pharmaceutical sector is currently using CNNs to ensure compliance with quality standards through automated inspection of packaging and labels. These applications are essential to maintain patient safety and product reliability.

Research suggests that CNNs can reduce inspection time by up to 50% while improving defect detection rates by approximately 30% compared to manual inspection methods. Automation through CNNs significantly reduces operational costs and minimizes waste, offering a return on investment that is compelling for industry. Once trained, the CNN model can be deployed across multiple facilities without the need for significant retraining, making it a cost-effective solution for large-scale industrial applications.

## 2 Predictive Maintenance

In industrial maintenance, predictive strategies using convolutional neural networks (CNNs) are revolutionizing the way companies approach machine maintenance and operations. Renowned for their ability to process and interpret image data, CNNs are also exceptionally capable of analysing time series data from a variety of sensors. This capability makes them invaluable for predictive maintenance, which aims to predict machine failure before it occurs, thereby minimizing downtime and reducing costs.

A compelling example of the use of CNNs in predictive maintenance can be seen in the wind energy industry. Wind turbines are critical assets that require constant monitoring due to their mechanical complexity and exposure to variable environmental conditions. Traditionally, maintenance procedures have been either reactive or scheduled at regular intervals regardless of the current condition of the equipment. However, through the use of CNNs, power companies can now analyse vibration data obtained from sensors mounted on turbines. These sensors capture a myriad of data points that reflect the operational status of the turbines in real time.

The process starts with collecting a huge amount of vibration data that tells the state of the turbine. This data, characterized by patterns that can indicate normal or abnormal conditions, is fed into the CNN. Trained on historical data marked as normal or indicative of specific faults (such as gearbox misalignment, bearing wear, or blade damage), the network learns to recognize subtle patterns that might elude human analysts or traditional rules-based monitoring systems.

# 3 Visual Inspection Systems

In the manufacturing industry, especially in electronics assembly, ensuring the quality of printed circuit boards is crucial. Defects such as soldering errors, missing components or poor alignment can lead to product failure. Traditional inspection methods include manual checks or simple automated systems that may not catch all defects or adapt to different types of circuit boards. The challenge is to create a system that can accurately and consistently identify defects without slowing down the production line.

Convolutional Neural Network (CNN) is an excellent tool for this application due to its ability to learn and identify complex patterns in image data. CNNs can be trained on a set of labeled images to recognize different types of defects on printed circuit boards. Once trained, the CNN can analyze images of circuit boards taken during manufacturing and classify them as defective or defect-free.

Advantages of using CNN for visual inspection systems:

- High Accuracy – CNNs can achieve higher accuracy in image classification tasks compared to traditional image processing techniques due to their deep learning capabilities.
- Automation and speed – Once trained, CNNs can process images quickly, enabling real-time inspection without delaying production.
- Adaptability – CNNs can be retrained or fine-tuned with new data to adapt to different or evolving products without extensive rework of the inspection system.

## 3.1 Custom CNN Implementation for Visual Inspection

Overall, the preparation of the actual network for this task can be divided into the following. CNN with 2 or 3 pairs of convolutional and max pooling vrsrtvy is suitable for the task. The following figure shows a possible architecture.

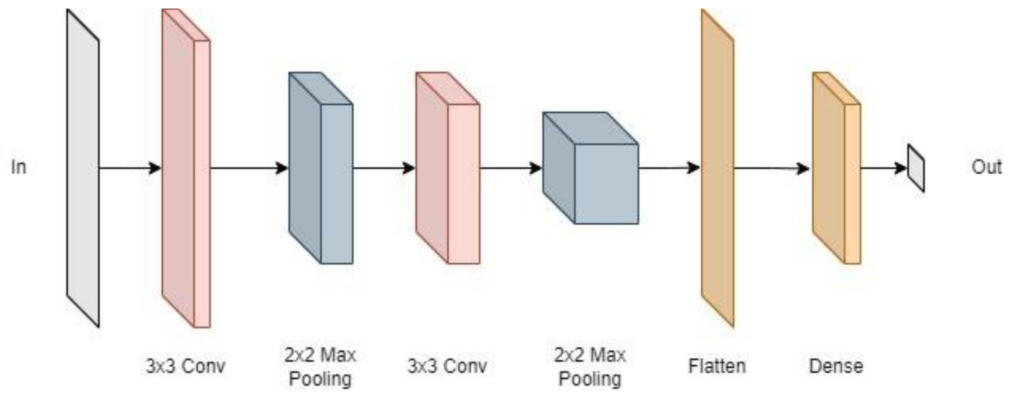


**Figure 1 - Possible architecture for binary visual inspection.**

### 3.1.1 Setting up the model

With the use of plug-in libraries such as Keras and Tesorflow, it is easy to prepare custom CNN architectures suitable for classifying images of manufacturing processes such as PCB inspections to "OK" or "NOK". The Python code using the mentioned libraries is as follows to solve the given problem.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

def build_model():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(128, 128, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))  # Binary output for
defective or not

    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
    return model
```

### 3.1.2 Data preparation

In a real scenario, a dataset of PCB images marked as "defective" (OK) or "not defective" (NOK) is needed. For the sake of demonstration, let us assume that we have such a dataset, created by steps:

- Data collection – Images of PCBs taken during production.
- Labeling – Images must be manually labeled by experts to provide training data for the model. In the case of binary classification, this is usually a breakdown into relevant components.
- Pre-processing – typically, resizing is done to a relatively small size that does not neglect details and normalization of values to the interval <0, 1> is done to allow standard methods to work on the images.

### 3.1.3 Training the Model

The model is trained using batches of images, which helps to efficiently optimize the weights of the neural network without requiring excessive memory resources. The training process involves feeding the network with batches of training images and their labels, adjusting the weights to minimize the loss (error) between the predicted and actual labels. An example for training our previously mentioned model and additionally using data augmentation using Keras and Tensorflow is as follows.

```
from keras.preprocessing.image import ImageDataGenerator

# Define paths to your training and validation image folders
train_data_dir = 'path_to_train_data'
validation_data_dir = 'path_to_validation_data'

# Initialize data generators with data augmentation parameters
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

val_datagen = ImageDataGenerator(rescale=1./255)

# Prepare data flow from directories and match the image input dimensions
of the model
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary')

validation_generator = val_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary')

# Fit the model
model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples //
validation_generator.batch_size)
```

Data augmentation techniques such as rotating, zooming and panning images are used to improve the ability of the model to generalize to new, previously unseen images. This simulates different scenarios that may occur in a real environment.

### 3.1.4 Model Deployment

Once the model has been trained and validated, it can be integrated into the production line. This involves setting up a system in which real-time images from cameras on the production line are fed into the model. If trained optimally, the trained CNN operates in real time and analyses images of the PCBs that pass through the production line. The system must be capable of high-speed image processing to keep up with the pace of the production line and provide immediate feedback and alerts when defects are detected.

## 4 Conclusions

The integration of convolutional neural networks (CNNs) into industrial automation has demonstrated significant efficiency gains in industries as diverse as manufacturing, automotive and pharmaceuticals. CNNs excel in complex visual recognition tasks, outperforming traditional methods in speed and accuracy, and thus play a key role in the development of automated systems. These neural networks not only optimize manufacturing processes but also improve predictive maintenance strategies, thereby reducing downtime and operating costs. As industries continue to evolve, the adaptation and integration of CNN technologies is a critical component in achieving greater productivity and operational excellence in the industry.

## 5 Literature

FUKUSHIMA, Kunihiko and MIYAKE, Sei. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. Online. In: AMARI, Shun-ichi and ARBIB, Michael A. (ed.). Competition and Cooperation in Neural Nets. Lecture Notes in Biomathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, p. 267-285. ISBN 978-3-540-11574-8. At: https://doi.org/10.1007/978-3-642-46466-9_18.

GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor a MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014, p. 580-587. ISBN 978-1-4799-5118-5. At: https://doi.org/10.1109/CVPR.2014.81.

LECUN, Y.; BOTTOU, L.; BENGIO, Y. a HAFFNER, P. Gradient-based learning applied to document recognition. Online. Proceedings of the IEEE. Issue 86, no. 11, p. 2278-2324. ISSN: 00189219. At: https://doi.org/10.1109/5.726791.

## List of abbreviations

| | |
|---|---|
| CNN | Convolutional Neural Network |
| ADAS | Advanced Driver Assistance Systems |
| PCB | Printed Circuit Board |
| NOK | Not OK – Indicating defects |
| OK | OK – Indicating no defects |
| AI | Artificial Intelligence |

## Index

# Artificial intelligence in automation

## Topic 8: Fuzzy Sets Theory. Fuzzy Relations and Operations with Fuzzy Relations. Linguistic Variable, Fuzzy Logic.

**Study Objective**

The main objective of this topic is to explore the basic concepts of fuzzy set theory and to provide students with a comprehensive understanding of fuzzy relations, operations on fuzzy relations and the concept of linguistic variables. This knowledge will enable students to effectively apply the principles of fuzzy logic in various automation scenarios. By addressing both theoretical and practical aspects, the course aims to prepare students to understand fuzzy logic problems and enhance their ability to develop solutions in the field of automation and control systems.

**Time Required for Study**

2 hours

**Keywords**

Fuzzy sets, Fuzzy relations, Linguistic variables, Fuzzy logic

## 1 Introduction to Fuzzy Sets Theory

Fuzzy sets were introduced by Lotfi A. Zadeh in 1965 as an extension of classical set theory. Traditional set theory classifies objects into different categories, represented by binary values (0 or 1) that indicate membership or non-membership to the set. In contrast, fuzzy sets allow objects to have degrees of membership, represented by values between 0 and 1. This approach reflects the way humans process information, making it a valuable tool for processing imprecise or vague information. (Zadeh, 1965)

The importance of introducing non-binary set memberships is not obvious to people at first glance, so it is best to give a suitable example. Imagine that you oversee a hundred sheep that you must corral for the evening and that the sheep must be sorted into white and black. The situation is complicated by the fact that the sheep are not necessarily all white or all black as shown in Figure 1. (Škrabánek, 2014)

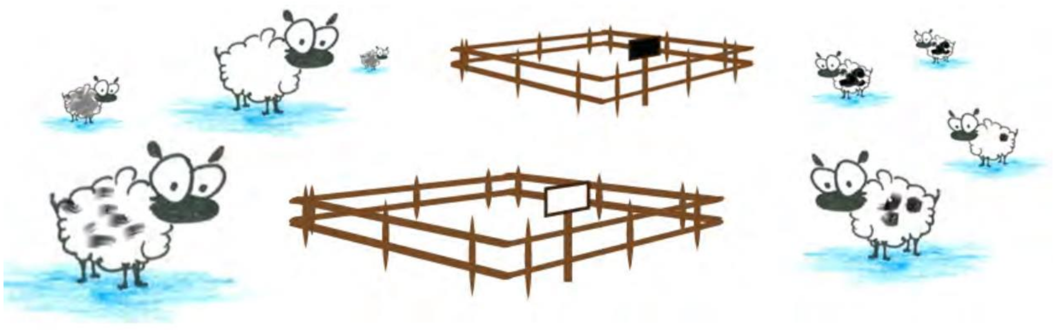


**Figure 1 - Uncertainty in the real world: the problem of black and white sheep. (Škrabánek, 2014)**

This task probably did not take much time to solve (except for a few undecided individuals). Each of you will eventually come to your own sense of sorting the sheep. Everyone will eventually herd the sheep into a "fairer" corral for them, even though the corrals do not contain only pure white or black sheep. Further, it is not expected that anyone would be greatly disturbed if the sheep were sorted by another person in a different way. It is this variability that is the benefit of fuzzy sets. (Škrabánek, 2014)

## 1.1 Basic Principles

Fuzzy sets represent a generalization of classical sets and are a mathematical means of working with imprecise concepts. In contrast to sharp sets, which assign only binary membership values to their elements (either they belong to the set or they do not), fuzzy sets allow to assign to each element x of the universe X a degree of membership to the fuzzy set. This degree of membership, which can take values between 0 and 1, expresses the degree of membership of an element to the set. Fuzzy sets are usually denoted by capital letters from the beginning of the alphabet, most commonly A, B and C.

A sharp set is a special case of a fuzzy set where the degree of membership can only take on values of 0 or 1, which corresponds to the classical definition of sets in binary logic.

Fuzzy sets can be written in different ways. One is an enumeration of elements where each element is accompanied by a degree of membership, for example A = {(x, 0.7), (y, 0.5), (z, 0.9)}, which means that element x belongs to the set A with probability 70%, y with 50% and z with 90%. Another way is to identify elements based on some property or criterion. A set can also be expressed in the form of "summation" or "integration", where the degrees of membership of all elements of the set are added together.

Basic concepts include an empty fuzzy set that is defined on the universe X but contains no elements with positive degrees of membership. A fuzzy subset is defined such that all elements of the subset have a degree of membership in the fuzzy set less than or equal to the degree of membership in the parent set. Equality of two fuzzy sets A and B on a universe X occurs if for each element x of the universe, the degree of membership of A is the same as the degree of membership of B.

A fuzzy set is characterized by a membership function that assigns each element a degree of membership from zero to one. This function is the cornerstone of fuzzy logic and allows to handle uncertainty and vagueness in everyday reasoning. The concept of a fuzzy set is key to understanding how fuzzy systems model real-world situations where answers do not necessarily fall into the categories of "yes" or "no“ but can be anywhere in between. (Zadeh, 1965)

## 1.2 Membership Function

The membership function is a key concept in fuzzy logic that gives each element x of the universe X assigns a degree of membership to a given fuzzy set. This degree of membership is usually expressed as a number between 0 and 1, where 0 means that the element does not belong to the set at all, and 1 means that the element is fully a member of the set.

The membership function can be formally described by an equation or a set of equations. For finite universes, it may also be practical to use a tabular notation that directly shows the degree of membership for each element. In practice, there may be multiple fuzzy sets on a single universe, which means that there are also multiple membership functions that may partially overlap.

For continuous sharp sets, which are defined in conventional mathematical terms, the membership functions are rectangular, meaning that the degree of membership of each element is either 0 or 1, with no intermediate degrees. In contrast, for fuzzy sets, there are a variety of different membership functions that reflect different distributional possibilities and characteristics of the membership of the elements. Examples of membership functions are shown in Figure 2.

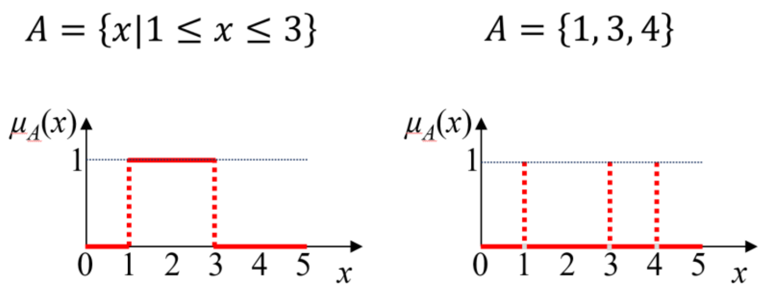
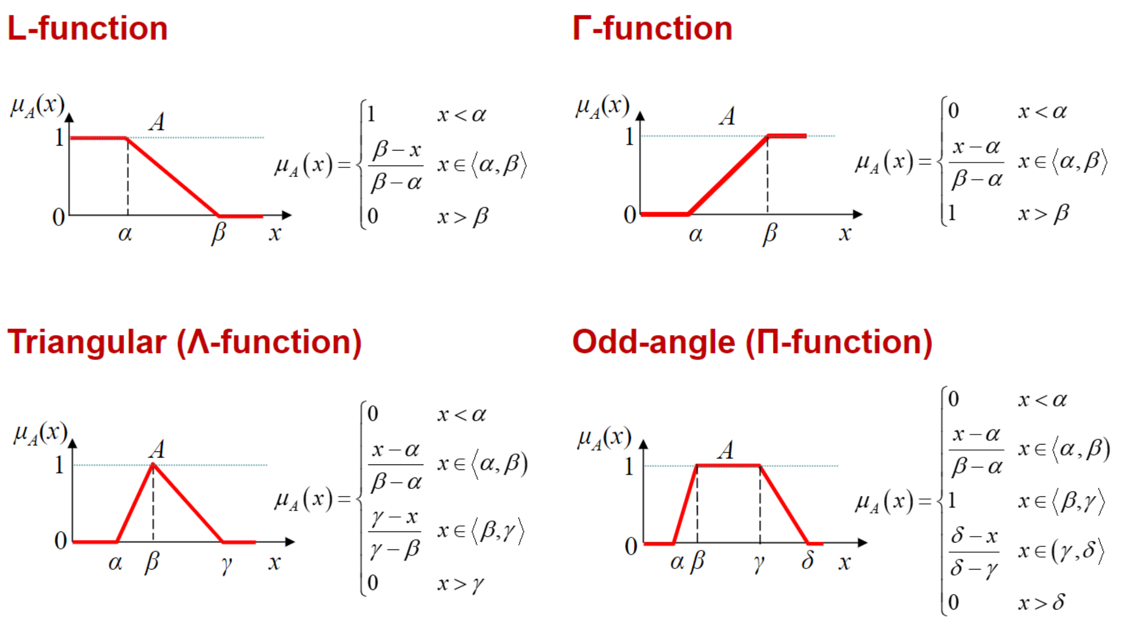


**Figure 2 – Examples of membership function**

### 1.2.1 Commonly Used Membership Functions

The most used forms of membership functions in fuzzy sets allow models to be adapted to real-world situations where different levels of uncertainty and membership need to be captured. There are specific shapes that are used for different purposes (figure 3):

- L-function - This function is shaped like the letter "L" and is used to model situations where an element with increasing value quickly goes from full membership to no membership. It is typically used for situations where a sharp threshold is needed above which the element no longer belongs to the set.
- Γ-function - The opposite form to the L-function, it resembles the letter "Γ". This function is suitable for situations where an element quickly gains full membership in the set with increasing value and then retains it. The Γ-function is useful in cases where it is important to define a minimum threshold beyond which an element becomes fully accepted as a member of the set.
- Triangular function (Λ-function) - As name speaks, it is triangular, where the vertex symbolizes the maximum degree of membership. This shape is useful for representing elements that have the greatest degree of membership at one point and gradually lose their membership as they move away from that point.
- Odd-angle function (Π-function) - Provides more flexibility than the triangular function, as it allows for the existence of an interval in which an element maintains its maximum degree of membership before it begins to decrease. This form is ideal for modeling situations where there is a well-defined range of values within which the element is fully accepted.



$$\mu_A(x)=\begin{cases}1 & x<\alpha\\ \dfrac{\beta-x}{\beta-\alpha} & x\in\langle\alpha,\beta\rangle\\ 0 & x>\beta\end{cases}$$

$$\mu_A(x)=\begin{cases}0 & x<\alpha\\ \dfrac{x-\alpha}{\beta-\alpha} & x\in\langle\alpha,\beta\rangle\\ 1 & x>\beta\end{cases}$$

$$\mu_A(x)=\begin{cases}0 & x<\alpha\\ \dfrac{x-\alpha}{\beta-\alpha} & x\in\langle\alpha,\beta)\\ \dfrac{\gamma-x}{\gamma-\beta} & x\in\langle\beta,\gamma\rangle\\ 0 & x>\gamma\end{cases}$$

$$\mu_A(x)=\begin{cases}0 & x<\alpha\\ \dfrac{x-\alpha}{\beta-\alpha} & x\in\langle\alpha,\beta)\\ 1 & x\in\langle\beta,\gamma\rangle\\ \dfrac{\delta-x}{\delta-\gamma} & x\in(\gamma,\delta\rangle\\ 0 & x>\delta\end{cases}$$

**Figure 3 - Commonly used membership functions**

## 1.3 Basic Operations

Other key concepts of fuzzy sets that are introduced in this slide include α-cut, carrier, kernel, fuzzy singleton, and height of the fuzzy set (Figure 4). These concepts are the building blocks for deeper analysis and manipulation of fuzzy sets in various applications:

- α-cut – An α-cut is a technique used to define a subset of a universe X based on some threshold α. Specifically, the α-cut of a fuzzy set A is the set of all elements x of X whose degree of membership in A is greater than or equal to α. This concept is useful for simplifying fuzzy sets and for analyzing their properties at different membership levels.
- Carrier of a fuzzy set (support) – The carrier of a fuzzy set A is the set of all elements x of the universe X that have a non-zero degree of membership in A. In other words, it is all elements that "somehow" belong to the set A.
- Core of a fuzzy set (core) – The core is the set of elements x from the universe X that have a maximum degree of membership of 1 in the fuzzy set A. This set represents the "most uncertain" elements that are fully accepted as members of set A.
- Fuzzy singleton – A fuzzy singleton is a special type of fuzzy set where only one element of the universe has a non-zero degree of membership, while all other elements have a degree of membership of 0. This element usually has a degree of membership of 1. Fuzzy singletons are useful for representing very specific data or results.
- Height of a fuzzy set (height) – The height of a fuzzy set is the highest degree of membership that any element in the set achieves. Height provides an overview of the degree to which elements are included in the set, and can be used as a measure of the "intensity" or "strength" of the fuzzy set.

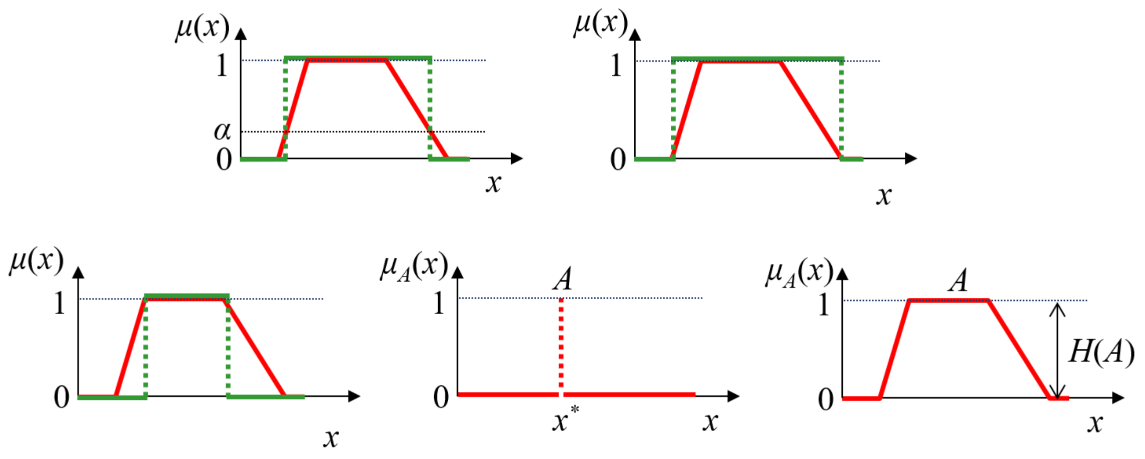


**Figure 4 – α-cut (top left), carrier (top right), kernel (bottom left), fuzzy singleton (bottom middle), height of the fuzzy set (bottom right).**

## 2 Linguistic Variable

The linguistic variable is a key tool in fuzzy logic, which allows to link linguistic expressions with numerical values, thus bridging the gap between human language and mathematical modelling. This variable is defined as an ordered five that includes the following components:

- Name of the language variable – This is an identifier or name of the variable that allows one to uniquely identify what the variable refers to. Examples might include "age", "temperature", "satisfaction", etc.
- Set of all terms of the language variable – This is the set of all possible terms or expressions that can be used for the language variable. For example, for the variable "temperature" there may be terms such as "low", "medium", "high".
- Reference variable universe – This is the set of all possible values that a language variable can take on. For "temperature" this could include a numeric range from -30 to 50 degrees Celsius.
- Syntactic Rule – These rules define how terms can be formed or combined to be grammatically correct. These rules may include grammar for forming terms and specify how terms may be combined or modified.
- Semantic rule – Semantic rules assign each term its meaning or definition in the form of a fuzzy set. These rules specify what mathematical model or membership function each term represents, allowing ambiguous or subjective concepts to be quantified.

## 2.1 Linguistic Operators

In fuzzy logic, linguistic operators play a crucial role in how the meanings of basic terms can be modified, and thus their corresponding membership functions. These operators allow different degrees of modification to be applied to basic fuzzy sets, thereby achieving the desired modification in their semantic interpretation. Modifications include adjusting the width, height or contrast of the membership function without changing the position of the carrier of the original fuzzy set.

### 2.1.1 Concentration Operator

This operator is used to intensify the meaning of terms, resulting in reduced vagueness. For example, the term "very warm" would have a sharper and narrower membership function than the plain term "warm". Thus, the concentration operator narrows the range of the membership function, increasing the clarity and specificity of the term.

### 2.1.2 The Dilation Operator

On the contrary, it widens the membership function, which increases the vagueness and produces a broader interpretation of the term. It is useful for expressions such as "roughly", "more less" that imply a less specific or less certain value. Using this operator leads to a wider range of values where the term can be considered acceptable.

### 2.1.3 Contrast Enhancement Operator

This operator is aimed at highlighting differences in membership, which is useful for terms such as "definitely", "unambiguously". The use of this operator makes the distinction between values with higher and lower degrees of membership more pronounced, which helps to better define and emphasize the criteria that determine the term.

### 2.1.4 Summary

These operators not only allow fine-tuning the meaning of terms within fuzzy logic, but also offer robust tools for handling and interpreting uncertain or fuzzy data in many applications such as expert systems, decision systems, and automation. These operators are key to effectively communicating and interpreting complex and fuzzy information in the real world, allowing systems to better respond to human language and natural language expressions.

# 3 Fuzzy Logic

Fuzzy logic is an advanced mathematical concept that extends classical binary logic to express truth in different degrees between absolute truth and falsity. This approach is essential for modeling and processing uncertain or fuzzy information that is typical in the real world, and offers tools for effective decision making in complex situations.

The truth value can take any value in the interval <0,1>, which allows to express different degrees of truth. Logical connectives are defined using truth tables and equivalent operations from fuzzy set theory:

- t-norm (triangular norm) – Used for the "AND" operation in fuzzy logic. Examples of t-norms include a minimum that selects the smaller value of two elements, or an algebraic product that multiplies the values of the elements.
- s-norm (supremum norm) – Used for the "OR" operation in fuzzy logic. Common s-norms include a maximum, which is the higher value of two elements, or an algebraic sum with a restriction to a maximum value of 1.
- Fuzzy negation – Negation is applied as a function that converts the truth value of x to 1-x, which is the opposite of the given value.

## 3.1 Specifics of Fuzzy Logic

Fuzzy logic does not contain direct ways to express operations such as implication, which is common in classical logic. Fuzzy implication is usually modeled using different approaches based on a combination of t-norms and s-norms, or special functions adapted for specific applications that consider context-dependent truth logic. Due to these features, fuzzy logic has become popular in various fields, including control systems, artificial intelligence, and decision making, where traditional binary approaches fail in their ability to effectively handle uncertainty and subjective interpretations of human judgments.

## 4 Literature

ŠKRABÁNEK, Pavel. Teorie fuzzy množin a její aplikace. Online. 1. Pardubice: Univerzita Pardubice, 2014. ISBN 978-80-7395-875-6.

ZADEH, L.A. Fuzzy sets. Online. Information and Control. 1965, issue 8, no. 3, p. 338-353. ISSN 00199958. At: https://doi.org/10.1016/S0019-9958(65)90241-X.

## List of abbreviations

| | |
|---|---|
| CNN | Convolutional Neural Network |
| L-function | Left function |
| Γ-function | Gamma function |
| Λ-function | Lambda function (Triangular function) |
| Π-function | Pi function (Odd-angle function) |

## Index

# Artificial intelligence in automation

## Topic 9: Fuzzy Logic Systems and Their Use for Process Control.

**Study Objective**

The main objective is to showcase the adaptability of fuzzy logic in improving decision-making, operational efficiency, and system robustness, while highlighting both the advantages and challenges faced in real-world implementations. Through this exploration, the chapter will provide valuable insights into designing and deploying effective fuzzy control systems.

**Time Required for Study**

2 hours

**Keywords**

Fuzzy Logic, Fuzzy Controllers, System Design, Decision Support

## 1 Fuzzy Logic Systems

Fuzzy logic does not contain direct ways to express operations such as implication, which is common in classical logic. Fuzzy implication is usually modeled using different approaches based on a combination of t-norms and s-norms, or special functions adapted for specific applications that consider context-dependent truth logic. Due to these features, fuzzy logic has become popular in various fields, including control systems, artificial intelligence, and decision making, where traditional binary approaches fail in their ability to effectively handle uncertainty and subjective interpretations of human judgments.

Fuzzy logic systems use the concept of fuzzy implication to model and control processes where uncertainty or indeterminacy is a key factor. Fuzzy implication extends the classical logical concept of implication (if p, then q) to a domain where truth values can take on arbitrary levels between 0 and 1, which is characteristic of fuzzy logic. (Šrabánek, 2014)

### 1.1 Fuzzy Implication

Fuzzy implication can be expressed in several different ways that vary depending on the requirements of the specific application and the properties they are intended to satisfy:

- Zadeh's Implication - This type of implication is based on the maximum and minimum of t-norms. If the truth value of the premise x is less than or equal to the truth value of the consequent y, then the truth value of the implication is 1. Otherwise, the truth value of the implication is equal to the truth value of y. (Zadeh, 1965)
- Kleene-Dienes Implication - In this form of implication, the logical OR operation associated with the negation of the first argument is used. Thus, the implication is expressed as a maximum of 1-x and y, which allows flexible treatment of truth values. (Kleene, 2009)
- Łukasiewicz Implication – This method defines the implication as the minimum of 1 and 1-x+y. This approach provides a continuous transition between truth values and is very useful in systems where smoothness of the change of truth levels is important. (Łukasiewicz, 1970)

## 1.2 Implications in Engineering Applications

In fuzzy logic and especially in engineering applications, special forms of fuzzy implications are used that are designed to better meet the needs of practical applications such as process control or decision making under uncertain conditions. Among the most commonly used are Mamdani and Larsen implications, which are known for their effectiveness in preserving the "cause → effect" principle. (Šrabánek, 2014)

### 1.2.1 Mamdani's Implication

Mamdani implication, named after Ebrahim Mamdani, is widely used in control systems because of its intuitive interpretation and ease of implementation. This method uses the t-norm minimum for applying the implication. In practice, this means that the output fuzzy set is bounded by the degree of truth of the input condition. Thus, if the rule says "If the temperature is high, then the fan speed is high," and the temperature is "high" with a truth value of 0.8, then the maximum truth value for "high fan speed" will also be 0.8.

### 1.2.2 Larsen's Implication

Larsen's implication, named after Ronald R. Larsen, uses production rules to apply the implication. Unlike Mamdani's method, which uses a minimum, Larsen's method uses multiplication. That is, the truth values of the output set are the result of multiplying the truth value of the input condition. Thus, if the input condition has a truth value of 0.8, the truth values of the output set are scaled by a factor of 0.8.

### 1.2.3 Practical Use

These two expressions of implication are very popular in practice due to their direct application to fuzzy control systems and models. They allow systems to respond effectively to changing environmental or process conditions, and provide a clear and easy-to-understand path from inputs to outputs. These implications are key for fuzzy control systems used in automotive, robotics, home automation and many other applications where uncertainty is common.

# 2 Structure of Fuzzy Logic Systems

Fuzzy logic systems (FLS) are based on the principle of fuzzy logic and enable efficient transfer of human knowledge and expert judgments into a form that is easily processed by a computer. These systems are designed to mimic the way human decision making and judgment is used in situations where uncertainty and vagueness dominate. (Šrabánek, 2014)

Fuzzy logic systems transform knowledge into the following components:

- Language variables – These variables allow the representation of fuzzy or qualitative concepts such as "temperature", "speed", "satisfaction", etc., in a form that can be easily manipulated within the system.
- Logical Conjunctions – Including operations such as AND, OR and NOT, which are defined using t-norms, s-norms and fuzzy negation, allow individual statements to be combined to form more complex logical structures.
- Inference mechanism – This mechanism uses a set of rules to draw conclusions from given data and situations. The rules are often formulated in the form "If ... Then ...", allowing the system to respond to inputs according to predefined logical structures.
- Rules – They are the key to creating a system that can effectively simulate expert judgments and decision making. Rules define how outputs are generated from inputs based on fuzzy logic.

There are two main types of fuzzy logic systems, which differ in the way they define the output:

- Mamdani type FLS – This type is most commonly used because of its intuitiveness and ease of implementation. The outputs are defined using fuzzy sets, and the inference process is based on minimum and maximum operators. Mamdani-type systems are suitable for applications where the output can be expressed in the form of linguistic terms.
- Sugeno-type FLS – These systems, sometimes known as Takagi-Sugeno-Kang systems, are useful when the outputs are best in the form of specific numerical values, which is often advantageous for control applications. Compared to the Mamdani system,

the Sugeno system is often more computationally efficient because the output functions are linear or constant, making calculations easier.

Fuzzy logic systems are an important tool in areas such as robotics, automotive, medical diagnosis, financial analysis, and many others where there is a need to manage the uncertainty and complexity of the real world. With the ability to model complex processes and simulate human decision making, fuzzy systems make a valuable contribution to the development of intelligent technologies.

Fuzzy logic systems (FLS) use several processes that allow conversion between sharp (exact) and fuzzy (indeterminate) values. These processes, including fuzzification and defuzzification, are key to the operation of FLS and allow the system to communicate effectively with the real world. The general structure of the fuzzy logic system is shown at figure 1.

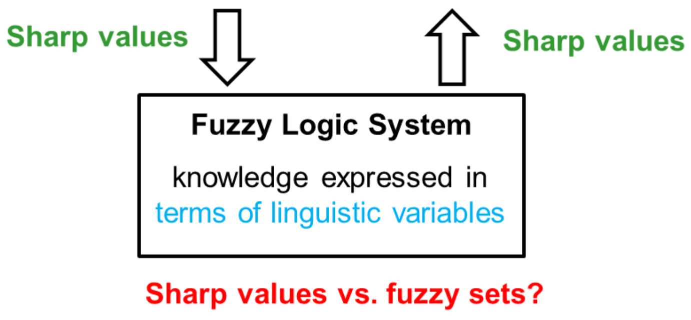


**Figure 1 – General structure of the fuzzy logic system.**

## 2.1 Fuzzification

Fuzzification is a process where sharp (numeric) input values are converted into fuzzy sets. This step is necessary because it allows the system to handle the fuzzy and imprecise information typical of fuzzy logic. During fuzzification, the input value is compared with the membership functions of the different fuzzy sets, and based on this comparison, the degree to which the value belongs to each fuzzy set is determined.

## 2.2 Defuzzification

Defuzzification is the process of converting a fuzzy set back to a sharp value, which is often required at the output of a system where a specific action or value is needed. There are several methods of defuzzification, such as the center of gravity method, where an average value is computed based on the degrees of membership in the fuzzy set to determine the "most representative" value for the fuzzy set.

# 3 FLS type Mamdani for Process Control

Mamdani's fuzzy logic system model is particularly valued for its accessibility and ease of implementation, making it a popular choice for engineers and developers who want to quickly translate expert knowledge into effective control systems. Because of its ability to integrate qualitative and fuzzy information, Mamdani's system enables effective problem solving where traditional approaches fail.

The Mamdani FLS was originally designed for process control purposes. Its basic idea is to use human knowledge, which is usually expressed through rules like "If this situation occurs, then do this". This approach allows effective control of even very complex systems without the need for deep technical or theoretical understanding. (Mamdani, 1975)

## 3.1 Structure of Mamdani FLS

Mamdani's FLS includes several key components that together define its functionality:

- Fuzzifiers – each input is transformed into fuzzy values through predefined membership functions, allowing the system to handle fuzzy data.
- Rule block – It forms the core of the system where all the rules are defined and stored. These rules are based on human knowledge and intuition and provide the basis for inference.
- Inference Engine – It applies fuzzy rules to the actual situation processed by the fuzzifiers, thus generating output fuzzy sets.
- Defuzzifier – In Mamdani's system, the output fuzzy sets are converted into one specific value that can be used for control or decision. This step is essential for converting fuzzy logic into practical, quantifiable actions.

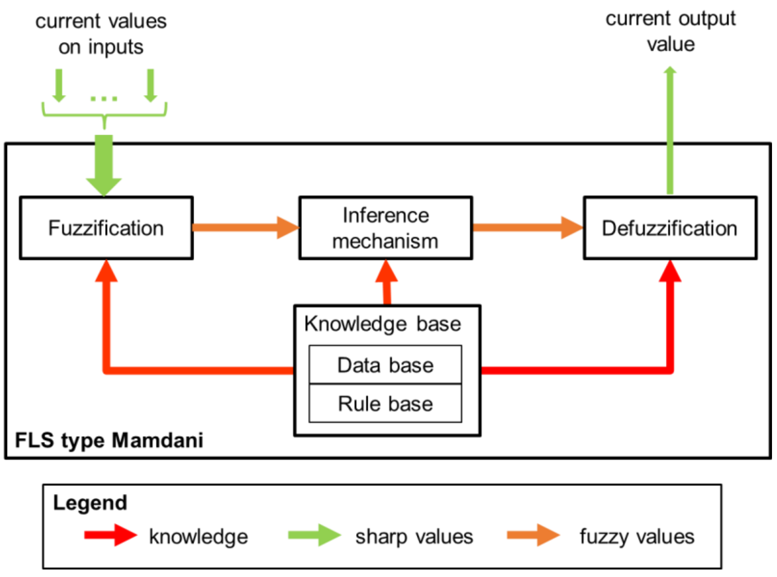


**Figure 2 – Structure of Mamdani's FLS**

### 3.2 Creation of FLS Type Mamdani

Creating a Mamdani-type FLS involves several key steps:

- Data Base:
    - Definition of the language variables for the inputs and output of the system. These variables allow the system to work with fuzzy values and translate them into specific actions or outputs.
    - Setting up the system, including the selection of logical connectors, inference mechanisms, and other system parameters that define how the system will respond to different situations.
- Rule base:
    - The definition of the rules that determine the behavior of the system. The rules are formulated in the format "If ... Then ...", which allows the system to respond to different input conditions with appropriate actions.

### 3.3 Applications

Mamdani's FLS is used in a wide range of industrial and commercial applications:

- Industrial automation – control of temperature, pressure, speed, and other factors in chemical and manufacturing processes.
- Automotive – Vehicle driving assistance systems such as adaptive cruise control and lane keeping systems.
- Smart Home Systems – Home automation such as temperature control, lighting and security systems that adapt to user preference and changing conditions.

Thus, Mamdani's FLS enables efficient and intuitive problem solving in situations where there is a limited amount of accurate data available and where decision making involves a high degree of uncertainty and variability.

## 4 Conclusion

Fuzzy logic systems (FLS) have demonstrated remarkable adaptability and efficiency in a variety of domains, bridging the gap between traditional binary logic systems and the complex, uncertain realities of practical applications. From engineering and manufacturing to healthcare and automation, FLSs leverage the concept of fuzzy implication and the flexibility of fuzzy rules to model environments where precision is obscured by ambiguity. By incorporating a continuum between absolute truth values, these systems mimic human decision-making processes and enhance the ability of machines to operate under varying degrees of uncertainty. Among other things, Mamdani's implications highlight the system's ability to preserve cause-effect relationships in fuzzy terms, making it an indispensable tool in industries where responsive and adaptive control is paramount.

## 5 Literature

KLEENE, Stephen Cole a BEESON, Michael. *Introduction to metamathematics*. New York: Ishi Press, 2009. ISBN 9780923891572.

ŁUKASIEWICZ, Jan & BORKOWSKI, L. Selected Works. Amsterdam: North-Holland, 1970, ISBN 978-0720422528.

MAMDANI, E. H.; ASSILIAN, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man-Machine Studies, issue 7, no. 1, 1975: p. 1-13, ISSN 0020-7373.

ŠKRABÁNEK, Pavel. Teorie fuzzy množin a její aplikace. Online. 1. Pardubice: Univerzita Pardubice, 2014. ISBN 978-80-7395-875-6.

ZADEH, L.A. Fuzzy sets. Online. Information and Control. 1965, issue 8, no. 3, p. 338-353. ISSN 00199958. At: https://doi.org/10.1016/S0019-9958(65)90241-X.

## List of abbreviations

| | |
|---|---|
| FLS | Fuzzy Logic Systems |
| S-norm | Standard Norm |
| T-norm | Triangular Norm |

## Index

# Artificial intelligence in automation

## Topic 10: Creation of digital twins of technological processes

**Study Objective**

The objective of this topic is to understand the fundamental concepts, methodologies, and applications of creating digital twins for technological processes. Students will learn to define digital twins, recognize their critical role in enhancing automation and process optimization, and understand the technical steps involved in their creation. By the end of this topic, students should be able to conceptualize and evaluate the application of digital twins.

**Time Required for Study**

2 hours

**Keywords**

Digital Twin, Automation, Technological Process, Simulation, Real-time Data

## 1 Introduction to Digital Twins

### 1.1 Definition and Overview

Digital twins represent the convergence of the virtual and physical worlds, where every industrial product can have a dynamic digital representation. A digital twin is essentially a virtual model designed to faithfully mimic and synchronize with its physical counterpart. This innovative technology integrates various areas including the Internet of Things (IoT), artificial intelligence (AI), machine learning and big data analytics, allowing it to perform complex simulations, predict future conditions and drive decision-making processes. Scheme of digital twins components is shown at figure 1.

By digitally simulating physical objects, enterprises can visualize products in operation and under different conditions and predict future performance or failures before they occur. This capability is key for industries that depend on precision operations and proactive maintenance, such as aerospace, automotive and healthcare. Digital twins also play a key role in improving product development, increasing operational efficiency and reducing time and cost to market. (Grieves, 2014)

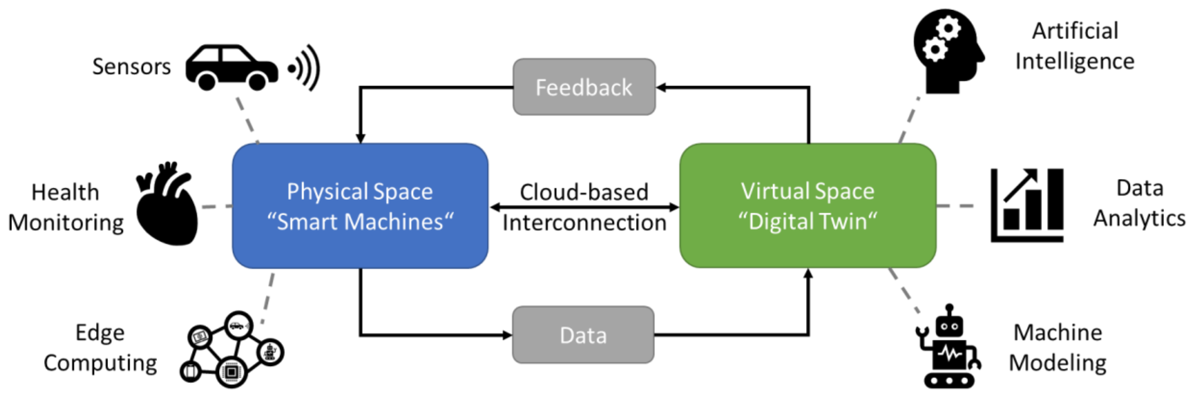


**Figure 1 - Ingredients and features of the digital twin ecosystem**

### 1.2 Historical Background

The concept of digital twins originated in aerospace, where NASA was the first to use them during space missions. The need for an exact replica of a distant spacecraft to simulate real-time conditions led to the development of the first forms of digital twins. This concept has evolved significantly over the years, spurred by advances in computing power and the advent of the Internet of Things. Today, digital twins are developed using high-fidelity 3D models that are continuously updated using data collected from sensors embedded in their physical counterparts. This evolution from simple simulations to complex predictive models reflects a broader trend in digital innovation, where technology increasingly serves as a bridge between the digital and physical realms. The growth in digital twin capabilities is accompanied by advances in data analytics and connectivity, making them more realistic and an integral part of operational strategies across industries. (Glaessgen, 2012)

## 2 Key Components of Digital Twins

### 2.1 Physical Assets

Digital twins rely extensively on accurate representations of the physical assets that are the basic elements they replicate. These assets can range from small mechanical components to large-scale infrastructures such as buildings or even entire cities. The accuracy of the digital replication of these assets is critical because it directly affects the reliability and utility of the digital twin. High-fidelity models ensure that simulations and analyses can accurately predict real-world behavior under a variety of scenarios, thereby aiding decision-making and strategic planning. This replication process involves detailed mapping of the physical properties and operational characteristics of the asset, which are continuously updated to reflect real-time changes and maintenance updates. (Tao, 2019)

## 2.2 Data Layers

At the core of the digital twin is its data architecture, which consists of several layers of data including operational data, environmental conditions and historical maintenance records. This data is collected through sensors and IoT devices and is critical to the twin's operation as a faithful replica. Integrating this data into the digital twin allows for dynamic updates, meaning the twin evolves in sync with its physical counterpart. The data layers must be structured and managed efficiently to provide accurate information, support predictive maintenance and increase operational efficiency. Advanced data analytics and machine learning algorithms are often applied to these data layers to extract patterns, predict outcomes and optimize performance. (Tao, 2019)

## 2.3 Connectivity

Connectivity is a key component that links physical assets to their digital twins. Using IoT and other networking technologies, digital twins can receive real-time data from their physical counterparts, ensuring that the virtual model is always an accurate mirror, both physically and operationally. This interconnection needs to be robust and secure to handle the huge amount of data being transferred, often over global networks. Examples include wired and wireless systems, cloud technologies, and edge computing, each of which play a role in increasing the efficiency and speed of data flows, which is essential for real-time digital twin functionality. (Rosen, 2015)

## 2.4 Role of Sensors and IoT

Sensors are the eyes and ears of the digital twins, collecting key data such as temperature, pressure, motion and other operational parameters directly from physical assets. This sensor data is transmitted through various platforms, such as IoT, to support a seamless flow of information between the physical and digital realms. The integration of these technologies enables continuous monitoring and analysis, facilitating immediate responses to any changes or anomalies detected in physical assets. This setup not only helps with preventive maintenance, but also improves overall system reliability and performance.  (Rosen, 2015)

# 3 Process of Creating a Digital Twin

## 3.1 Data Collection and Processing

The first step in creating a digital twin is the careful collection and processing of data from the physical asset. This involves deploying sensors and other data collection equipment to gather a wide range of operational, environmental and maintenance data. The quality of the digital twin is largely dependent on the accuracy and completeness of this collected data. Once collected, the data undergoes thorough processing to ensure that it is clean, reliable and efficiently structured for integration into the digital twin. Advanced data processing techniques, including data normalization and error correction, are used to prepare the data for further use in simulations and predictive analysis. (Qi, 2018)

### 3.2 Building the Virtual Model

The creation of the virtual model is a critical stage where the processed data is transformed into a digital replica of the physical asset.  Engineers and developers strive to ensure that models not only replicate the physical dimensions and properties of the asset, but also mimic its behavior in different operational scenarios. An important part of this process is validation, which involves thoroughly testing the digital twin against real data and conditions to ensure its accuracy and reliability. (Qi, 2018)

### 3.3 Integration with Real-Time Data

It is essential for the accuracy and usefulness of the digital twin that it is updated with real-time data. This integration is typically achieved through the continuous flow of data enabled by IoT technologies. As new data is received from the physical asset, it is immediately reflected in the digital twin, enabling real-time monitoring and analysis. This timely synchronization helps identify potential issues before they become critical, enabling proactive asset management and maintenance. (Tao, 2017)

### 3.4 Use of AI and Machine Learning for Predictive Analytics

Artificial intelligence and machine learning are playing a transformative role in improving the predictive capabilities of digital twins. Using these technologies, digital twins can not only simulate different scenarios, but also predict future conditions and design optimal responses. For example, machine learning algorithms can analyze historical and real-time data to predict equipment failures or optimize operational parameters for energy efficiency. Such predictive analysis brings significant value in decision making, reducing downtime and optimizing performance. (Tao, 2017)

## 4 Applications and Case Studies

In today's world, the prevalence of digital twins is extensive, but it is also important to remember that this is a fundamental modelling task. Industrial applications involve systems of varying complexity and it is this complexity that relates to the need for an appropriate modelling tool; for simple processes, approximation models or models based on knowledge of physical dependencies may be appropriate. For complex models, multilayer feedforward neural networks are a suitable tool. An example of a digital twin is shown in Figure 2.

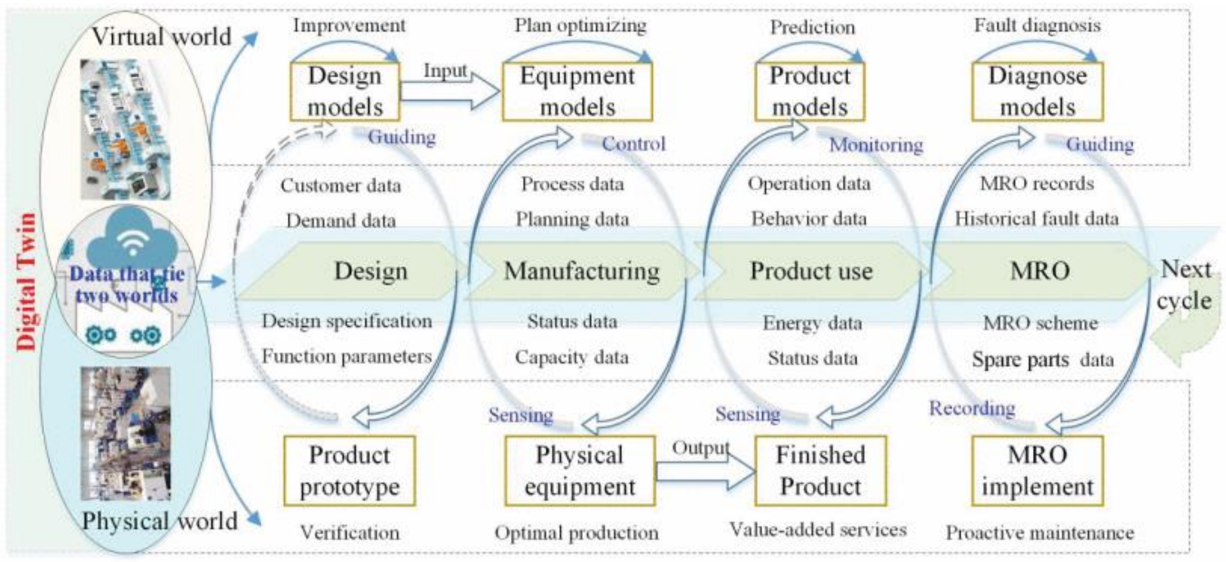


**Figure 2 – Example of digital twin in manufacturing. (Qi, 2018)**

## 4.1 Manufacturing Automation

In manufacturing, digital twins are the key to automating complex production lines. They enable real-time monitoring and control, allowing adjustments to be made on the fly to optimize production efficiency. A prominent example is a global automotive manufacturer that has implemented digital twins to fully automate its vehicle assembly lines. Digital twins simulate assembly processes to identify bottlenecks and predict equipment failures before they occur, enabling proactive maintenance and minimal downtime. This automation not only increases production speed, but also significantly improves product quality and worker safety. (Kritzinger, 2018)

## 4.2 Energy Sector Automation

Digital twins in the energy sector are changing the way traditional and renewable energy installations operate. They play a key role in automating the control systems of power plants, especially in integrating and controlling different energy sources such as wind, solar and hydro power plants within smart grids. For example, one European utility company uses digital twins to simulate and optimize the operation of its smart grid, increasing the efficiency of energy distribution and automating maintenance schedules for its infrastructure. This leads to improved energy production and reduced operating costs. (Kritzinger, 2018)

## 4.3 Automated Logistics and Supply Chain Management

Automation through digital twins in logistics and supply chain management is revolutionizing the way goods are stored, tracked and distributed. Digital twins enable logistics companies to create highly efficient warehouse operations and optimize everything from inventory allocation to delivery schedules. A case study from a leading logistics service provider shows how digital twins are being used to automate warehouse operations, enabling real-time adjustments based on inventory levels, demand forecasts and delivery schedules. This application ensures optimal inventory levels, reduces waste and improves delivery efficiency, significantly increasing customer satisfaction. (Kritzinger, 2018)

## 5 Conclusions

In exploring digital twins, we examined their basic concepts, the complex process of their creation, and their diverse applications in various industries, with a focus on industrial automation. Digital twins represent the peak of modern technology and embody the convergence of physical assets and digital sophistication. Through real-time data integration, advanced analytics and machine learning, digital twins offer unprecedented capabilities to predict outcomes, optimize processes and improve decision-making.

Digital twins represent not just a technological advance, but a paradigm shift in how we interact with and manage our technological environment. They enable a deeper understanding of our systems and processes, allowing us to make more informed decisions and implement more effective solutions. As this technology continues to evolve, it will undoubtedly play a key role in shaping the future of the industry and beyond.

## 6 Literature

GLAESSGEN, Edward a STARGEL, David. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. Online. In: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference: 20th AIAA/ASME/AHS Adaptive Structures, Conference: 14th AIAA. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2012, -. ISBN 978-1-60086-937-2. At: https://doi.org/10.2514/6.2012-1818.

GRIEVES, M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. [Whitepaper], 2014.

KRITZINGER, Werner; KARNER, Matthias; TRAAR, Georg; HENJES, Jan a SIHN, Wilfried. Digital Twin in manufacturing: A categorical literature review and classification. Online. IFAC-PapersOnLine. 2018, issue: 51, no. 11, p. 1016-1022. ISSN 24058963. At: https://doi.org/10.1016/j.ifacol.2018.08.474.

QI, Qinglin a TAO, Fei. Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison. Online. IEEE Access. 2018, issue 6, p. 3585-3593. ISSN 2169-3536. At: https://doi.org/10.1109/ACCESS.2018.2793265.

ROSEN, Roland; VON WICHERT, Georg; LO, George a BETTENHAUSEN, Kurt D. About The Importance of Autonomy and Digital Twins for the Future of Manufacturing. Online. IFAC-PapersOnLine. 2015, issue: 48, no. 3, pages 567-572. ISSN 24058963. At: https://doi.org/10.1016/j.ifacol.2015.06.141.

TAO, Fei a ZHANG, Meng. Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing. Online. IEEE Access. 2017, issue 5, p. 20418-20427. ISSN 2169-3536. At: https://doi.org/10.1109/ACCESS.2017.2756069.

TAO, Fei; ZHANG, Meng a NEE, A.Y.C. Digital Twin Driven Smart Manufacturing. Academic Press, 2019. ISBN 978-0-12-817630-6.

## List of abbreviations

| | |
|---|---|
| IoT | Internet of Things |
| AI | Artificial Intelligence |
| 3D | Three Dimensional |

## Index

# Artificial intelligence in automation

## Topic 11: Introduction to reinforcement learning. Fundamentals of grasping and manipulation.

**Study Objective**

The main objective of this topic is to provide students with a basic understanding of reinforcement learning (RL) and its specific applications in robotics, with a particular focus on the area of grasping and manipulation.

**Time Required for Study**

2 hours

**Keywords**

Reinforcement Learning, Robotics, Grasping, Manipulation, Deep Learning, Q-Learning

## 1 Introduction to RL Concepts

Reinforcement learning is a type of machine learning where an agent learns to make decisions by performing actions and receiving feedback in the form of rewards or penalties. Unlike supervised learning, where the training data includes the correct answers, RL involves learning the best actions to take in a given situation through trial and error. This method is particularly powerful in environments where the right answers are not known beforehand, making it ideal for dynamic and complex decision-making scenarios such as robotic manipulation. The scheme of main RL principle is shown at figure 1.

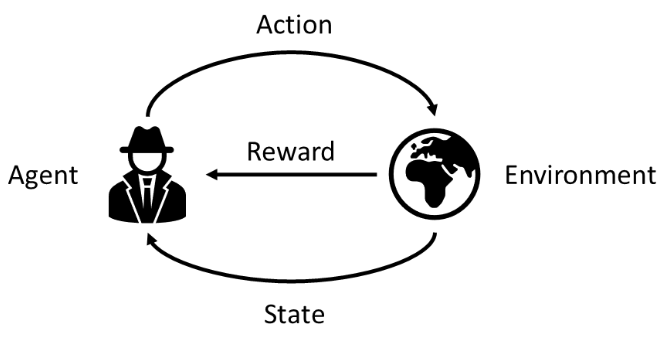


**Figure 1 - Scheme of RL principle.**

### 1.1 Key RL Algorithms

Among the many RL algorithms, there are several that are considered essential due to their efficiency and wide use. These include Q-learning, Deep Q Networks (DQN), Policy Gradient methods or Actor-Critic algorithms. Each of these algorithms has its own unique approach and area of application, but they share basic principles that underline the reinforcement learning paradigm.

Despite the differences in approach and implementation, the key RL algorithms have several key similarities. First, all of these algorithms involve a trade-off between exploration and exploitation, where the agent must balance between exploring new actions to discover potentially better long-term rewards and exploiting known actions that yield the highest rewards. This tradeoff is critical to ensure that the agent can adequately learn the environment without getting stuck in suboptimal solutions.

Another common aspect is the use of rewards as signals for policy learning. In any RL algorithm, the actions performed by the agent are evaluated based on the rewards obtained from the environment. These rewards help the algorithms to adjust their parameters, whether it is updating the Q-table in Q-learning or adjusting the weights of the neural network in DQN.

In addition, most RL algorithms work on the principle of iterative improvement. After each episode or step, they incrementally improve their estimates.

Finally, all of these algorithms are set in the context of sequential decision making, where each decision affects not only the immediate reward but also subsequent decisions. This decision process is modeled as a Markov Decision Process (MDP) in almost all RL scenarios, which provides a mathematical framework for the analysis and design of RL algorithms.

In conclusion, although methods may vary, the underlying goals and principles of reinforcement learning remain consistent and drive the development of algorithms capable of learning complex behavior in dynamic environments.

#### 1.1.1 Q-learning

Q-learning is a foundational algorithm in the field of reinforcement learning, providing a robust framework for agents to learn optimal policies in a model-free environment. It is particularly powerful in scenarios where the model of the environment is unknown and the agent must learn to make decisions based solely on the rewards received. (Bertsekas, 2019)

It uses Bellman equation to update the Q-values (1), which are estimations of the expected rewards for an action taken in a given state

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]. \qquad (1)$$

The $Q(s,a)$ is the Q-value for a given state *s* and action *a*. It represents the expected utility of taking action *a* in state *s*. Coefficient $\alpha$ is a learning rate, which determines how much new information affects the existing Q-value. A higher $\alpha$ allows the agent to learn faster, adjusting more significantly with new data. Coefficient *r* is reward received after taking action *a* in state *s*. This reward influences the Q-value update, encouraging actions that lead to higher rewards. Coefficient $\gamma$ is the discount factor, which represents the difference in importance between future rewards and immediate rewards. The function (2)

$$\max_{a'} Q(s', a') \tag{2}$$

represents the maximum predicted reward obtainable from the next state *s′*, under any action *a′*. This term allows the agent to update its Q-value based on the best possible future outcome, aligning with a strategy of maximizing future rewards.

This algorithm helps an agent to evaluate the expected utility of taking a given action in a given state, updating its policy based on a balance of immediate rewards and future returns. It’s particularly useful in discrete action spaces.

### 1.1.2 SARSA (State-Action-Reward-State-Action)

SARSA standing as a prime example of an on-policy learning algorithm, directly incorporates the action taken by the current policy into its updates, unlike Q-learning which uses the maximum reward action. It’s ruled by assign (3)

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s', a') - Q(s,a)]. \tag{3}$$

SARSA is essential for understanding how actions taken according to a specific policy lead to learning about the value of state-action pairs directly under that policy. This characteristic makes it suitable for learning policies that need to consider the risks of exploratory actions, such as in robotic pathfinding where safety is a priority.

### 1.1.3 Deep Q-Networks

DQN extends Q-learning by using deep neural networks to approximate the Q-value function, allowing it to handle complex environments with high-dimensional state spaces. The use of a neural network helps to generalize across states, making it more efficient in environments where state representations are large or continuous.

## 2 Reinforcement Learning in Robotics

Reinforcement learning has become a powerful tool to enable robot autonomy, especially in environments that require adaptability and real-time decision making. Robots equipped with RL algorithms can learn optimal strategies by trial and error and autonomously improve their task performance without explicit programming for all possible scenarios. (Kober, 2013) Some examples of RL robots are shown at figure 2.

**Figure 2 – Robots with behaviors that were reinforcement learned (From left: OBELIX, autonomous helicopter, Sarcos humanoid DB) (Kober, 2013)**

## 2.1 Specifics of Applications on Grasping and Manipulation

Robotic grasping and manipulation involve tasks in which the robot must physically interact with objects in its environment, such as lifting, moving or assembling objects. Reinforcement learning is particularly suited for these tasks because it allows robots to learn from a continuous stream of feedback and adjust their strategies based on the results of their actions. For example, a robotic arm trained with RL can adapt its grasping technique to handle objects of different shapes, sizes, and fragility. This adaptability is crucial in industries such as warehousing and manufacturing, where robots are expected to efficiently handle a variety of objects. (Levine, 2018)

## 2.2 Technical Challenges and Solutions

Implementing RL in robotics faces several practical challenges, primarily the discrepancy between simulated training environments and the real world. This issue, known as the "reality gap," can hinder the transfer of learned behaviors from virtual to real settings. Techniques such as domain randomization, where the training simulation varies widely in its parameters, appear to help improve the robustness and transferability of learned behavior. In addition, combining simulation data with limited real-world training data - known as simulation-to-reality transfer - has shown promise in bridging this gap. (Rusu, 2017)

# 3 Techniques and Technologies

## 3.1 Sensors and Hardware

Robotic systems designed for grasping and manipulation tasks are equipped with various sensors that enable precise interaction with the physical world. Tactile sensors allow robots to recognize the texture and contours of objects and adjust their grip based on the properties of the material they encounter. Force and moment sensors provide important feedback about the forces and moments acting on the robotic arm and gripper, which is important for tasks requiring fine manipulation or precise application of force. Vision systems, including advanced cameras and sometimes 3D scanners, help robots identify, locate and orient themselves relative to objects. (Siciliano, 2008)

The hardware supporting these sensors must allow precise control and fast feedback. Robotic arms are usually equipped with sophisticated articulated actuators for smooth and precise movements. Central computing units, often augmented by graphics processing units (GPUs), are key to managing the significant flow of sensor data and processing this information in real time to respond quickly and efficiently to dynamic environments.

### 3.2 Software and Computational Methods

Robotic systems rely on advanced computational frameworks to manage the complex data and algorithms involved in reinforcement learning. Robotic Operating System (ROS) is widely used as a structured environment that provides hardware abstraction, device control, and a communication platform between the software components of the robot. Deep learning frameworks, such as TensorFlow and PyTorch, are essential for implementing neural networks that underlie deep reinforcement learning, and facilitate the training and deployment of models that predict optimal actions based on sensory inputs. (Quigley, 2009)

The challenges associated with real-time data processing are significant and require the system to analyze and respond to sensor data in fractions of a second to be effective. Computational efficiency is a primary concern, with ongoing efforts focused on optimizing the performance of algorithms to ensure fast decision making. Another critical issue is scalability, as systems must handle increasingly complex tasks and larger volumes of data as robots evolve.

### 3.3 Conclusions

The integration of sophisticated sensors and robust computational frameworks is fundamental to the success of reinforcement learning in robotic grasping and manipulation. These technologies provide the necessary data and computational power required for efficient operation and define the possibilities and potential of artificial intelligence in robotics.

# 4 Simple Reinforcement Learning Example

Let’s do the grid system example. Imagine a small 4x4 grid where the robot starts at the top left corner and needs to reach the bottom right corner as efficiently as possible. The grid cells are either open or have obstacles, and the robot can move up, down, left, or right. For simplicity, assume all cells are open, and the goal is simply to learn the shortest path to the goal with using Q-Learning algorithm. This scheme is shown at figure 3.

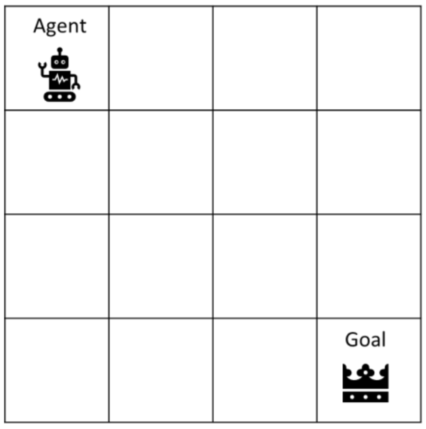


**Figure 3 – RL example scenario**

At first there must be an environmental setup, which includes definitions as follows:

- States: Each cell in the 4x4 grid represents a state (total 16 states).
- Actions: Four possible actions at each state – move up, down, left, or right.
- Rewards/Punishments:
    - -1 for each move to incentivize the shortest path
    - 100 for reaching the goal
    - -10 for bumping into walls (attempting to move outside the grid).

At second there must be a Q-learning setup, which includes definitions as follows:

- Initialize the Q-table: 16x4 matrix of zeros (16 states and 4 actions per state).
- Learning rate (α): 0.1
- Discount factor (γ): 0.9

Now as the algorithm and environment are set, then the Q-learning algorithm process can start. Algorithm goes as follows:

1. Start at the initial state (0,0).
2. Select an action:
    - Use an ε-greedy policy: Choose random actions with probability ε (say 0.1) for exploration, and the best action according to the Q-table otherwise for exploitation.
3. Perform the action:
    - If the action is to move right from (0,0), the new state becomes (0,1).
4. Receive the reward:
    - Since (0,1) is not the goal and it's a valid move, the reward is -1.
5. Update the Q-table using the Q-learning formula (1):
    - For moving right from (0,0) to (0,1):
    - $Q((0,0),"right") \leftarrow 0 + 0.1[-1 + 0.9 \cdot 0 - 0] = -0.1$
6. Repeat till Q-values converge.

## 4.1 Expected Outcome after Training

After sufficient training, the Q-table will guide the robot to always take the shortest path to the goal with no unnecessary moves. The optimal policy will direct the robot to move right three times and down three times from the start to the goal under ideal learning conditions.

# 5 Literature

BERTSEKAS, Dimitri. Reinforcement Learning and Optimal Control. 1. Belmont, Massachusetts: Athena Scientific, 2019. ISBN 978-1-886529-39-7.

KOBER, Jens a PETERS, Jan. Reinforcement Learning in Robotics: A Survey. Online. In: WIERING, Marco a VAN OTTERLO, Martijn (ed.). Reinforcement Learning. Adaptation, Learning, and Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 579-610. ISBN 978-3-642-27644-6. At: https://doi.org/10.1007/978-3-642-27645-3_18.

LEVINE, Sergey; PASTOR, Peter; KRIZHEVSKY, Alex; IBARZ, Julian a QUILLEN, Deirdre. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Online. The International Journal of Robotics Research. 2018, no. 4-5, p. 421-436.

QUIGLEY, Morgan, CONLEY, Ken, GERKEY, Brian, et al. (2009). ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software. 3. ISSN 0278-3649. At: https://doi.org/10.1177/0278364917710318.

RUSU, A.A., VECERIK, M., et al. (2016). Sim-to-Real Robot Learning from Pixels with Progressive Nets. ArXiv, abs/1610.04286. At: https://doi.org/10.48550/arXiv.1610.04286.

SICILIANO, Bruno a KHATIB, Oussama (ed.). Springer Handbook of Robotics. Online. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN 978-3-540-23957-4. At: https://doi.org/10.1007/978-3-540-30301-5.

# List of abbreviations

| | |
|---|---|
| RL | Reinforcement Learning |
| DQN | Deep Q Networks |
| MDP | Markov Decision Process |
| ROS | Robotic Operating System |
| GPU | Graphics Processing Unit |
| SARSA | State-Action-Reward-State-Action |

## Index

# Artificial intelligence in automation

## Topic 12: Learning-based grasping and manipulation.

**Study Objective**

The aim of this topic is to explore in detail learning-based approaches to robotic grasping and to highlight the integration of deep learning technologies in accurate and efficient object detection and manipulation. The course will also cover methodologies for training convolutional neural network models along with challenges in practical implementation of the system in real-world applications.

**Time Required for Study**

2 hours

**Keywords**

Robotic Grasping, Object Manipulation, Camera Calibration, Convolutional Neural Networks

## 1 Object Detection and Recognition

### 1.1 Introduction to Object Detection in Robotics

Object detection is an important component in the field of robotic manipulation, enabling robots to efficiently perceive the necessary environment. This process involves using cameras and sensors to capture visual information, which is then analyzed using advanced algorithms to identify and locate objects. Convolutional neural networks (CNNs), a class of deep learning models, are particularly adept at processing and interpreting image data. These models are trained to recognize objects by learning from thousands of images labeled with information about what they contain.

Among the various architectures developed for CNNs, models such as AlexNet, VGG and ResNet show significant success in object classification tasks due to their ability to extract and learn from complex patterns in visual data. These networks have been discussed extensively in previous topics of this course, and networks suitable for object detection are built on their core. Therefore, these networks are often referred to as backbone networks in object detection networks. (Canziani, 2016)

Unlike image classification models, which output a single prediction for the entire image, object detection models produce multiple predictions, each associated with specific objects in the image. Key architectures that have contributed to the development of object detection include:

- Region-Based Convolutional Neural Networks (R-CNNs) and their faster variants (Fast R-CNN and Faster R-CNN): These models apply CNNs to propose regions of interest and then classify each region into object categories while refining their bounding boxes. (Girshick, 2014)
- You Only Look Once (YOLO): This model frames object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. YOLO is renowned for its speed and efficiency, making it particularly suitable for real-time applications. (Redmon, 2016)
- Single Shot MultiBox Detector (SSD): This model uses a single deep neural network to detect multiple objects within the image in one go, which makes it faster than R-CNNs and effective for real-time processing. (Liu, 2016)

### 1.2 Challenges in Object Detection for Robotic Grasping

Robotic grasping presents a unique challenge that differs from typical object detection tasks due to the additional requirement to not only identify but also manipulate objects. The main challenges of machine vision include:

- Variability in object shapes and sizes – Robotic arms often need to grasp a wide variety of objects that can vary greatly in shape, size, and texture, requiring robust detection algorithms capable of handling this diversity.
- Occlusions and overlapping objects – In many real-world situations, objects may be partially or completely obscured by other objects, making detection and sometimes accurate grasping difficult.
- Real-time processing needs – In many industrial applications, decisions need to be made quickly, requiring very high processing speeds without sacrificing accuracy.

## 2 Robotic Manipulation Techniques

This section explores strategies and technologies that enable robots to manipulate objects more efficiently and optimally, with an emphasis on the transition from traditional, hard-coded robotic systems to adaptive, learning-based approaches.

### 2.1 Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm in which agents learn to make decisions by interacting with the environment and receiving feedback in the form of rewards or punishments. In the field of robotic manipulation, RL can be used for autonomous learning of complex tasks such as object grasping and interaction. Algorithms such as Deep Q-

Networks (DQN) and Proximal Policy Optimization (PPO) are key to the use of RL in robotics. Compared to Q learning, DQN implements a neural network to help process the high-dimensional sensory inputs that are integral to robotic systems, while PPO offers an easier implementation. (Sutton, 2018)

RL training includes simulated environments that reflect real-world conditions but allow for accelerated and safer experimentation. Setting up a reward system is key; for example, the robot learns to grasp by receiving small rewards for moving toward an object and larger rewards for successfully picking it up. However, RL faces challenges such as efficiency, controllability due to sampling, and safety of the learning process, as RL requires many episodes (scenarios of passage from a location to a target) to converge and may involve dangerous actions. (Sutton, 2018)

### 2.2 Supervised Learning

Supervised Learning in robotics involves training models on datasets for which the manipulation processes are known, and associating sensory inputs with the corresponding outputs. This method is crucial for tasks that require precise control, such as component assembly or complex object manipulation. Convolutional neural networks (CNNs) are often used to process visual inputs to identify objects and their details, while recurrent neural networks (RNNs) and long short-term memory (LSTM) networks can process the sequence of actions that are necessary for time-dependent tasks in single-step manipulation and trajectory planning. (Caldera, 2018)

Creating an efficient dataset is probably the biggest hurdle in supervised learning for robotics. It involves collecting a diverse set of examples involving different objects, angles, and lighting conditions and then accurately annotating the correct actions. Challenges in supervised learning include the model's ability to generalize, which means it will be able to identify previously unseen scenarios. (Caldera, 2018)

## 3 Building a Robotic Grasping System

This section introduces the components and goals of a robotic grasping system and explains the integration of hardware and software to achieve accurate and efficient object manipulation. It will focus on setting up the camera and lighting, calibrating the system for accurate object detection, and planning the robot's movements to perform the grasping task.

### 3.1 Hardware Setup

The first step is to choose the right camera and lens, which is essential for taking clear and detailed images of the objects to be handled. Factors to consider include resolution, frame rate and field of view. The focal length and lens aperture must match the operating distance of the system and the lighting conditions to ensure a sharp image. It is the proper choice of lens that will occupy the area of interest.

The next step is to consider the lighting conditions and, if necessary, to provide them. Proper lighting is essential for consistent image quality. Diffused lighting will help avoid shadows and reflections that could confuse the object detection system. The location and type of lights should be configured to illuminate objects evenly and not create glare.

An example of a robotic grasping system is shown in Figure 1.

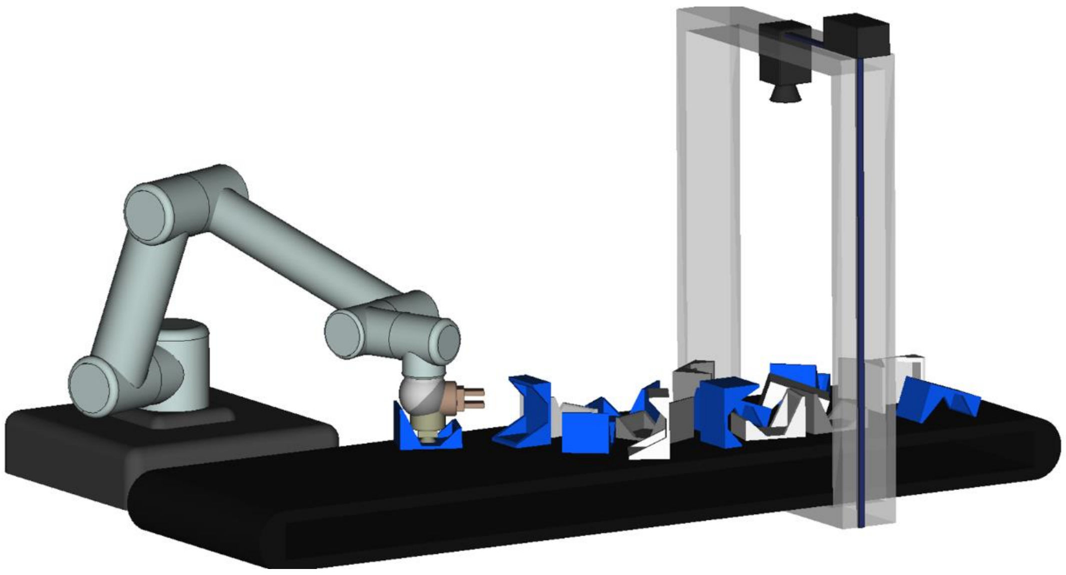


**Figure 1 – Example of robotic grasping system.**

## 3.2 System Calibration

To ensure correct operation, it is typical to perform a calibration in which we try to adjust the world visible from the camera to the world in which the robot operates. Thus, the effort is to find a system of translation and rotation matrices that maps the different coordination systems to each other.

Camera calibration is required to convert the 2D image into real world coordinates. This process involves determining the internal (focal length, skew, distortion) and external (position and orientation relative to the robot) parameters of the camera. Tools such as OpenCV provide functions to perform these calibrations using standard formulas in which the location of significant points in both the image and the real environment can be defined. Ideal for this task are checkerboard shapes where black and white alternate in adjacent squares of defined size. The coordinates captured by the camera must be converted to robot coordinates, a process that involves mapping the camera pixels to real dimensions. This conversion ensures that the robot can accurately locate objects within its operational space.

### 3.3 Software for Object Detection and Manipulation

The moment we can accurately map an object in the real world (in the robot's coordinates), we can use the detection algorithm to detect objects, or object grasping points.

Using deep learning, in particular CNN, the system can identify and categorize objects in the camera's field of view. In order for the model to learn to recognize the different objects that the robot will manipulate, training data must be collected and accurately labeled. Data collection can also take place before the actual placement and physical configuration of the system, but it should be noted that in this case the laboratory environment in which the data collection takes place should match as closely as possible the environment in which detection will then take place.

Once the detection algorithm is working with sufficient accuracy, we can move on to the next steps. Once the object is detected, the robot must next plan a path to grasp it. This involves calculating the safest and most efficient trajectory from the robot's current position to the object. In such cases, so-called safe zones are defined in which the robot can move. In the case of more advanced applications, other possible collisions with other objects or obstacles in space that may be detected by another detection algorithm must also be avoided.

The last step is to perform the gripping movement. The robot uses the planned trajectory to reach and grasp the object. Feedback mechanisms, such as force sensors in the gripper, ensure that the robot exerts enough pressure to grasp the object safely and not damage it.

The actual movement along the trajectory is usually handled by the robotic systems themselves in its control system. However, if the robotic arm is composed independently of actuators and sensors, the actual control of the actuators and sensors needs to be done in a different way. Typically, the control is performed in a feedback manner with control of the rotation angles of the individual robot joints that form the kinematic chain. In addition, control can be implemented in other ways, for example based on reinforcement learning.

## 4 Conclusions

The use of sophisticated object detection technologies, in particular convolutional neural networks (CNNs), represents a major advance in robotic manipulation. These deep learning models, including AlexNet, VGG and ResNet, have greatly improved the ability of robots to visually perceive their surroundings.

Robotic grasping faces challenges such as variability in object shapes and sizes, occlusions, and the need for real-time processing. Techniques such as region-based CNNs and fast models such as YOLO and SSD address these issues by providing both speed and accuracy, which is essential for real-time applications. The transition from traditional, hard-coded systems to adaptive, learning-based approaches is facilitated by Reinforcement Learning (RL) and Supervised Learning, which improve decision making and accuracy in manipulation tasks. RL allows robots to learn autonomously through interaction with the environment, while Supervised Learning uses pre-labeled data to learn accurate control tasks.

Building a robotic grasping system involves careful hardware setup, system calibration, and sophisticated object detection software. Precise camera calibration and efficient lighting are key to accurately matching digital and real coordinates. Software algorithms using deep learning convert visual data into actionable insights for task execution. These integrated technologies enable robots to perform complex manipulations and adapt in real time to ensure both efficiency and safety.

## 5 Literature

CALDERA, Shehan; RASSAU, Alexander a CHAI, Douglas. Review of Deep Learning Methods in Robotic Grasp Detection. Online. Multimodal Technologies and Interaction. 2018, issue 2, no. 3. ISSN 2414-4088. At: https://doi.org/10.3390/mti2030057.

CANZIANI, Alfredo; PASZKE, Adam and CULURCIELLO, Eugenio. An Analysis of Deep Neural Network Models for Practical Applications. Online. 2016. At: https://doi.org/10.48550/arXiv.1605.07678.

GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor and MALIK, Jitendra. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014, p. 580-587. ISBN 978-1-4799-5118-5. At: https://doi.org/10.1109/CVPR.2014.81.

LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott et al. SSD: Single Shot MultiBox Detector. Online. In: LEIBE, Bastian; MATAS, Jiri; SEBE, Nicu and WELLING, Max (ed.). Computer Vision – ECCV 2016. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, p. 21-37. ISBN 978-3-319-46447-3. At: https://doi.org/10.1007/978-3-319-46448-0_2.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross and FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. Online. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016, p. 779-788. ISBN 978-1-4673-8851-1. At: https://doi.org/10.1109/CVPR.2016.91.

SUTTON, Richard S. a BARTO, Andrew G. Reinforcement learning: an introduction. Second edition. Cambridge, Massachusetts: The MIT Press. 2018. ISBN 978-026-2039-246.

## List of abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| CNN | Convolutional Neural Network |

| | |
|---|---|
| DQN | Deep Q-Network |
| LSTM | Long Short-Term Memory |
| PPO | Proximal Policy Optimization |
| R-CNN | Region-Based Convolutional Neural Network |
| RL | Reinforcement Learning |
| RNN | Recurrent Neural Network |
| SSD | Single Shot MultiBox Detector |
| YOLO | You Only Look Once |

## Index

# Artificial intelligence in automation

## Topic 13: Case studies of industrial applications.

**Study Objective**

The aim of this section is to explore practical applications of AI in industry and highlight innovative uses and case studies that demonstrate the impact of AI. Through the analysis of real-world examples, students will gain insight into the challenges and opportunities that AI presents in automation. The goal is to equip students with knowledge of how AI technologies are implemented to optimize processes, increase efficiency, and support innovation in various industries.

**Time Required for Study**

2 hours

**Keywords**

Artificial Intelligence, Automation, Industrial Applications, Case Studies

## 1 Introduction

This introductory section will explain what case studies are and why they are important in scientific research and education, especially in the context of AI applications in industry.

### 1.1 What Are Case Studies?

Case studies are in-depth analyses of specific cases where technology is used to solve real-world problems. In the context of AI-solvable problems, case studies provide valuable insights into the theoretical and practical aspects of AI deployment in automation. They offer an in-depth look at the different types of challenges that need to be faced during implementation. As the culmination of this course, these case studies serve a dual purpose: they reinforce previously learned concepts by putting them into concrete scenarios, and they demonstrate the versatility and impact of AI technologies in different industries. By mentioning these examples, the aim is to better understand the application of AI theories in real industrial settings, thus bridging the gap between academic teaching and practical application.

## 2 Predictive Maintenance Using Neural Networks in Manufacturing

In the rapidly evolving environment of industrial automation, predictive maintenance is one of the most important applications of artificial intelligence. The deployment of neural networks in this area has revolutionized the way manufacturers approach maintenance. Previously unexplored operational effects have begun to be extensively investigated thanks to the capabilities of artificial intelligence. Along with the aforementioned shift, there has been a move away from traditional reactive or planned approaches to a more effective data-driven (machine learning) strategy. (Lei, 2013)

Predictive maintenance uses advanced neural network algorithms to analyze the vast amount of data collected from sensors embedded in manufacturing equipment. These sensors monitor various parameters such as vibration, temperature, pressure, and sound to provide real-time insight into the performance and condition of the equipment. By processing this data, neural networks can identify patterns and anomalies that prevent equipment failures. This capability allows maintenance professionals to intervene before a failure occurs, thereby preventing downtime and/or extending the life of the machinery. (Lei, 2013)

A compelling example of this technology in action can be seen in the automotive industry, where manufacturers have integrated AI-driven systems to monitor critical equipment. In one case, a major car manufacturer implemented neural networks to monitor the performance of its engine assembly lines. The AI system was learned from historical sensor data that included instances of equipment failures and instances of normal operation. Through learning and ample data, the system learned to distinguish between normal fluctuations and actual signs of impending failure. (Lei, 2013)

Following the information in (Lei, 2013), which was also used for the previous paragraphs, the results were significant. The introduction of predictive maintenance resulted in a 30% reduction in unplanned downtime and a 25% reduction in maintenance costs. In addition, production quality improved as the likelihood of faults was reduced due to better maintained equipment. These improvements not only increased operational efficiency, but also increased the overall profitability of the manufacturing operations.

## 3 AI in Logistics: From Warehousing to Delivery

The logistics industry, which is crucial to global trade, is undergoing significant changes thanks to advances in artificial intelligence. From warehouse automation to sophisticated optimisation of delivery routes, AI applications in logistics are fundamentally changing the way goods are stored, managed and transported. This integration of AI not only streamlines operations, but also reduces costs and improves service delivery, increasing overall supply chain efficiency.

### 3.1 Automated Warehousing with Robotic AI

In warehousing, AI-powered robotic systems can be seen as a revolutionary step forward. These AI-equipped systems perform many tasks traditionally performed by human workers, such as staging, packing, sorting and storing goods. A prominent example is the use of robotic systems in Amazon's fulfillment centers. These robots navigate large warehouses using complex algorithms that allow them to avoid obstacles, manage inventory and pick goods at remarkable speed. (Wurman, 2008)

The impact of deploying AI robotics in warehousing is profound. At Amazon, the introduction of Kiva robots, now known as Amazon Robotics, has led to a 50% reduction in operating costs. The robots have also reduced the time it takes to pick items from 1.5 hours to just 15 minutes, dramatically increasing the speed and accuracy of order processing. (Wurman, 2008)

## 4 AI in Energy: Smart Grid Management

The energy sector is undergoing a major transformation with the introduction of smart grid technologies. Artificial intelligence plays a key role in this transformation by improving network management through real-time data analysis, predictive assessment and efficient load balancing. These advances not only increase the resilience and efficiency of power systems, but also support the broader goals of sustainability and reducing environmental impact.

### 4.1 AI-Enhanced Grid Operation and Renewable Integration

One of the most critical applications of artificial intelligence in the energy sector is the management of power grids. Smart grids equipped with AI can dynamically adapt to changes in energy supply and demand, manage outages and integrate a higher share of renewable energy sources. An important example is their use by Enel, one of Europe's leading energy companies, to optimise grid operations and manage the variability of renewable energy sources such as wind and solar power. (Rolnick, 2023)

Enel's AI systems analyse data from thousands of collection points (sensor systems) across the grid to predict demand patterns, detect potential faults and recommend maintenance activities. In addition, these systems optimize the flow of renewable energy and adapt to the unpredictability of weather conditions to maximize efficiency and minimize waste. The result is a more resilient and flexible grid that can handle a higher percentage of renewable energy without compromising reliability. (Rolnick, 2023)

# 5 AI in Agriculture: Precision Farming

Agriculture is essential to sustaining the world's population, but faces many challenges, including climate change, land resource constraints and ensuring sustainability. Artificial intelligence offers innovative solutions to these challenges through precision agriculture, which uses AI to optimise the amount and timing of resources (such as water, fertiliser and pesticides) used to grow crops.

## 5.1 Implementing AI for Crop Health Monitoring and Yield Prediction

One of the transformative applications of AI in agriculture is monitoring crop health and predicting crop growth. Using image recognition and AI-based data analysis, farmers can detect plant diseases, pests or nutrient deficiencies. The deployment of AI technology by John Deere, a leading agricultural machinery manufacturer, is a case study, as it is incorporating AI and machine learning into its tractors and combines. (Maimaitijiang, 2020)

John Deere's technology uses cameras and sensors to analyse crop conditions in real time as the machines move through the fields. The data collected is processed using machine learning algorithms that compare it to historical data patterns to predict future crop health and potential yields. This system allows farmers to make more informed decisions about when to irrigate, fertilise or apply pesticides, leading to increased crop yields and reduced wastage. (Maimaitijiang, 2020)

# 6 AI in Healthcare: Automation in Diagnosis and Treatment

The healthcare industry is using AI to address some of its most pressing challenges, such as improving diagnostic accuracy, reducing treatment errors and managing patient data efficiently. AI technologies are particularly important in the field of diagnosis and treatment, where they offer the potential for fast and accurate analysis of complex medical data.

## 6.1 Enhancing Diagnostic Accuracy in Radiology

Another key area where AI is making significant advances is radiology. AI algorithms are increasingly being used to interpret medical images such as X-rays, CT scans and MRIs with greater speed and accuracy than ever before. A case in point is the use of AI systems in the diagnosis of lung disease, where AI tools analyse lung images to detect anomalies that may indicate diseases such as tuberculosis or lung cancer. (Hosny, 2018)

One such system has been developed and integrated into clinical workflows at several large hospitals, where it helps radiologists by highlighting problem areas on images and suggesting possible diagnoses. This AI assistance helps achieve earlier and more accurate detection of lung disease, which is essential for effective treatment. (Hosny, 2018)

# 7 AI in Automotive: Autonomous Driving and Quality Control

The automotive industry is at the forefront of the introduction of artificial intelligence, which is revolutionising the design, manufacture and operation of vehicles. The impact of AI is particularly evident in the development of autonomous driving technologies and the improvement of quality control processes within production lines.

## 7.1 AI in Autonomous Vehicle Technology

One of the most interesting applications of artificial intelligence in the automotive industry is the development of autonomous vehicles. This technology relies heavily on AI systems that interpret sensor data, make real-time decisions and learn from the vast amount of driving data to improve safety and efficiency. A prominent example is the use of AI by companies such as Tesla and Waymo, which are pioneering the development of self-driving cars. (Montemerlo, 2009)

These vehicles use a combination of cameras, radar and lidar, processed by sophisticated AI algorithms, to navigate and understand their surroundings. AI systems are trained on data collected from millions of miles driven to recognize objects, predict behavior, and make driving decisions that mimic human judgment but with better reaction time and accuracy.

(Montemerlo, 2009)

# 8 Conclusions

This chapter comprehensively discusses practical applications of AI in various industries and shows its potential not only in industrial automation. From predictive maintenance in manufacturing to advances in logistics, energy management, agriculture, and healthcare, the role of AI in improving efficiency, reducing costs, and promoting sustainability is profound.

Through case studies, the chapter highlights how neural networks and AI algorithms can help predict equipment failures, optimize logistics operations, manage the load or stress on energy networks, monitor crop health, or assist in medical diagnosis. Each case study not only illustrates AI's ability to solve complex real-world problems, but also demonstrates the technology's ability to adapt to different industry needs.

The insights gained from these applications underscore the critical importance of integrating AI into industrial processes, not only to increase operational efficiency, but also to foster innovation and maintain a competitive advantage in a rapidly evolving global environment. As industries continue to embrace AI, the insights from these case studies will serve as valuable guidance for future implementations.

# 9 Literature

HOSNY, Ahmed; PARMAR, Chintan; QUACKENBUSH, John; SCHWARTZ, Lawrence H. a AERTS, Hugo J. W. L. Artificial intelligence in radiology. Online. Nature Reviews Cancer. 2018, issue 18, no. 8, p. 500-510. ISSN 1474-175X. At: https://doi.org/10.1038/s41568-018-0016-5.

LEI, Yaguo; LIN, Jing; HE, Zhengjia a ZUO, Ming J. A review on empirical mode decomposition in fault diagnosis of rotating machinery. Mechanical Systems and Signal Processing. 2013, issue 35, no. 1-2, p. 108-126. ISSN 08883270. At: doi.org/10.1016/j.ymssp.2012.09.015.

MAIMAITIJIANG, Maitiniyazi; SAGAN, Vasit; SIDIKE, Paheding; HARTLING, Sean; ESPOSITO, Flavio et al. Soybean yield prediction from UAV using multimodal data fusion and deep learning. Online. Remote Sensing of Environment. 2020, Issue 237. ISSN 00344257.At: https://doi.org/10.1016/j.rse.2019.111599.

MONTEMERLO, Michael; BECKER, Jan; BHAT, Suhrid; DAHLKAMP, Hendrik; DOLGOV, Dmitri et al. Junior: The Stanford Entry in the Urban Challenge. Online. In: BUEHLER, Martin; IAGNEMMA, Karl a SINGH, Sanjiv (ed.). The DARPA Urban Challenge. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, p. 91-123. ISBN 978-3-642-03990-4. At: https://doi.org/10.1007/978-3-642-03991-1_3.

ROLNICK, David; DONTI, Priya L.; KAACK, Lynn H.; KOCHANSKI, Kelly; LACOSTE, Alexandre et al. Tackling Climate Change with Machine Learning. Online. ACM Computing Surveys. 2023, issue 55, no. 2, p. 1-96. ISSN 0360-0300. At: https://doi.org/10.1145/3485128.

WURMAN, Peter; D'ANDREA, Raffaello a MOUNTZ, Mick. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. Online. AI MAGAZINE. 2008. At: https://doi.org/10.1609/aimag.v29i1.2082.

## List of abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| CT | Computed Tomography |
| MRI | Magnetic Resonance Imaging |

## Index

UNIVERZITA
PARDUBICE
FAKULTA
ELEKTROTECHNIKY
A INFORMATIKY

Vytvořeno v rámci projektu **Studijní program Automatizace (SPAUT)**
**na Univerzitě Pardubice**, reg. č. NPO_UPCE_MSMT-16591/2022.

Financováno Evropskou unií NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY