

MASTER'S ADMISSION

Oral examination / interview for admission to the Master's program

Smart Biotechnical Systems / Sisteme Biotehnice Inteligente (SBI)

(Field of Mechatronics and Robotics)

The oral examination results in a single grade, but it is structured in two parts:

- The first part is the interview in which the candidate introduces themselves, covering the following aspects: completed studies, thesis topic, work experience, areas of interest, motivation / determination to pursue this master's program.
- The second part of the oral examination consists of a discussion on the competition topic presented briefly but sufficiently for presentation at the exam.
-

Elements of Programmable Automation

PLC is an abbreviation for Programmable Logic Controller, which is a digital computer-based device used for industrial automation and control. It is used in a wide variety of applications, from controlling manufacturing processes to managing power grids. A PLC is made up of a central processor unit (CPU), memory, input/output (I/O) modules, and communication interfaces. PLCs are programmed with specialized software, often using ladder logic or other programming languages, to perform particular control tasks such as switching relays, controlling motors, and processing data.

PLCs have revolutionized the field of industrial automation by providing a dependable and flexible answer for controlling machines and processes. PLCs have dramatically reduced the complexity of controlling machines and processes by replacing traditional hard-wired relay systems. PLCs can handle sensor inputs, analyze data, and control outputs to different actuators to keep the desired process state.

PLCs are built to withstand harsh industrial settings, making them long-lasting and dependable. They can manage complex processes that require quick and accurate responses by performing real-time control tasks. PLCs also provide several advantages, such as freedom, modularity, and scalability. These advantages make it simple to adapt and modify the system as required, lowering downtime and increasing efficiency.

PLC programming languages, such as Ladder Logic, Function Block Diagram, and Structured Text, are intended to be intuitive and simple to learn, even for those with no programming experience. Because of this ease of use, programmers can easily build and modify control programs, reducing the time and expense of developing and deploying a system.

PLCs have allowed manufacturers to reach higher levels of productivity, dependability, and safety. PLCs have become an important component of modern industrial automation systems due to their ability to handle a wide range of input and output devices. PLCs are anticipated to become more

popular as automation technologies progress because they provide a reliable and flexible solution for controlling machines and processes.

PLC advantages over traditional control systems:

- PLCs are easily programmed and reprogrammable to adjust to changing process requirements or to add new control functions. This increases manufacturing and other industrial operations' flexibility.
- PLCs are more reliable than traditional control systems because they are designed to work in harsh and demanding environments. They can also be readily integrated with other automation systems like robotics and SCADA to increase overall system reliability.
- Scalability: PLCs are easily extended or updated to meet changing process requirements, making them a cost-effective industrial automation and control solution.
- PLCs can process data and execute program instructions rapidly, allowing them to control industrial processes in real-time.
- Safety: PLCs can be programmed to monitor and control safety functions such as emergency stops and interlocks to ensure worker safety and avoid equipment damage.
- Remote Access: PLCs can be accessed remotely, enabling operators to remotely watch and control processes. This can increase efficiency while decreasing the need for on-site employees.
- Data Logging: PLCs can record data on process factors like temperature and pressure, which can be used for process optimization and troubleshooting.

PLC disadvantages over traditional control systems:

- Programming and configuring a PLC can be time-consuming and complex, requiring specialized knowledge and abilities. This can reduce the accessibility of PLCs for small enterprises and those with limited technical resources.
- PLCs can be more costly than conventional control systems, particularly for small-scale applications. Hardware, software, and programming costs can rapidly add up, rendering PLCs less cost-effective for certain applications.
- Maintenance: PLCs require routine maintenance, including software updates and hardware replacements, to ensure dependability and prevent system malfunctions. This can be time-consuming and expensive, particularly with larger systems.
- Limited Memory: The memory capacity of PLCs is limited, which can be a hindrance when programming complex logic or managing large data collections.
- Security: PLCs are susceptible to cyberattacks, particularly if they are connected to a network or the internet. Additional measures and resources, such as firewalls and security protocols, are required to guarantee the safety of PLC systems.
- Compatibility: PLCs may not be compatible with legacy equipment or systems, necessitating additional hardware and software investments.

- **Limited Interoperability:** Different manufacturers' PLCs may not be interoperable, limiting the ability to incorporate systems from different vendors.

The **physical components** that comprise a Programmable Logic Controller (PLC) system are referred to as PLC hardware components. These components are intended to be used in conjunction to automate control procedures in industrial applications. Here are the important components:

- *Central Processing Unit (CPU)* is the “brain” of the PLC, responsible for data processing and instruction execution. It memorizes the PLC program and conducts computations depending on input and output data.
- *Input/Output (I/O) Modules* operate as a bridge between the PLC and external devices like sensors and actuators. I/O modules are classified into two types: input modules that accept signals from sensors and other input devices and output modules that deliver signals to actuators and other output devices.
- *Power Supply* supplies electricity to the PLC system. It converts the incoming alternating current or directs current voltage to the voltage required by the CPU and other components.
- *Programming Device* is used to create and edit the PLC program. Depending on the type of PLC system, it could be a personal computer or a handheld device.
- *Communication Module* interfaces the PLC with other devices, such as other PLCs, human-machine interfaces (HMIs), and supervisory control and data acquisition (SCADA) systems.
- *Expansion Modules* are extra I/O modules that can be added to a PLC system to expand its capacity or functionality. Analog I/O modules, specialist I/O modules, and communication modules are examples.
- *Rack* is the physical structure that connects the PLC components. It is intended to protect the components from harm while also making it simple to add or remove components as needed.

Specialized **programming languages** designed to generate logic circuits and control systems are used to program PLC systems. For PLC systems, various programming languages are utilized, each with its own set of advantages and disadvantages. Here's a rundown of the most popular PLC programming languages:

- *Ladder Logic (LAD)* is a graphical programming language that represents logical links between inputs and outputs using symbols and diagrams. It is the most extensively used programming language for PLC systems, and it is especially well-suited to discrete logic control applications.
- *Structured Text (ST)*, like Pascal or C, is a high-level programming language. It works in text format and is best suited for sophisticated calculations and data handling.
- *Function Block Diagram (FBD)* is a graphical programming language that employs blocks to represent functions and their relationships. It is best suited for complicated logic control applications.
- *Sequential Function Chart (SFC)* is a graphical programming language that represents control sequences with steps and transitions. It is best suited for sophisticated control applications requiring event sequences.

- *Instruction List (IL)* is a low-level programming language that represents machine code instructions in text form. It is best suited to applications that require fast processing.
- *Continuous Function Chart (CFC)* is a graphical programming language that represents control functions with continuous variables and equations. It is best suited to applications that require precise control of continuous variables.

PLC input devices transmit signals to a Programmable Logic Controller (PLC) system. Temperature, pressure, location, and speed are only a few of the physical factors represented by these signals. Sensors and switches are the two primary categories of PLC input devices. The following is a list of the most popular types of PLC input devices:

- **Sensors** are devices that take physical measurements and turn them into electrical signals. The following are the most frequent types of sensors found in PLC systems:
 - *Proximity* sensors use infrared, ultrasonic, or magnetic fields to detect the presence or absence of an object.
 - *Photoelectric* sensors use light beams to detect the presence or absence of an object.
 - *Temperature* sensors use thermocouples, resistance temperature detectors (RTDs), or thermistors to monitor the temperature of a system.
 - *Pressure* sensors use strain gauges or piezoelectric materials to monitor the pressure in a system.
 - *Position* sensors use potentiometers, encoders, or hall effect sensors to determine the location of a system.
- **Switches** are electronic devices that send a binary signal to the PLC system. The following are the most frequent types of switches found in PLC systems:
 - *Limit switches* use a mechanical lever to detect the presence or absence of an object.
 - *Push buttons* are used to deliver manual commands to the PLC system.
 - *Toggle switches* allow binary signals to be sent to the PLC system.
 - *Selector switches* are used to choose a specific control mode or parameter.

PLC output devices receive signals from a Programmable Logic Controller (PLC) system and act on them. These output devices are commonly referred to as “actuators” and are divided into two categories: motors and other devices. The following is a list of the most popular types of PLC output devices:

- **Motors** are used to control the motion of machines and equipment by converting electrical energy into mechanical energy. The following are the most frequent types of motors found in PLC systems:
 - *Electric motors (AC)* are used to control the speed and direction of alternating current (AC)-powered machines.

- *DC motors* are used to regulate the speed and direction of DC-powered machinery.
- *Servo motors* are used to precisely regulate the location and speed of machinery.
- *Stepper motors* are used to precisely position and move machines.
- **Other devices** include lights, heaters, and solenoids, are employed to control physical processes. PLC output devices that are commonly used include:
 - *Lights* are used to identify a machine's or system's status.
 - *Heaters* regulate the temperature of a system.
 - *Solenoids* are electronic devices that control the movement of valves and other mechanical components.
 - *Relays* are used to switch high-voltage or high-current loads.

Artificial Neural Networks - Glossary of basic terms

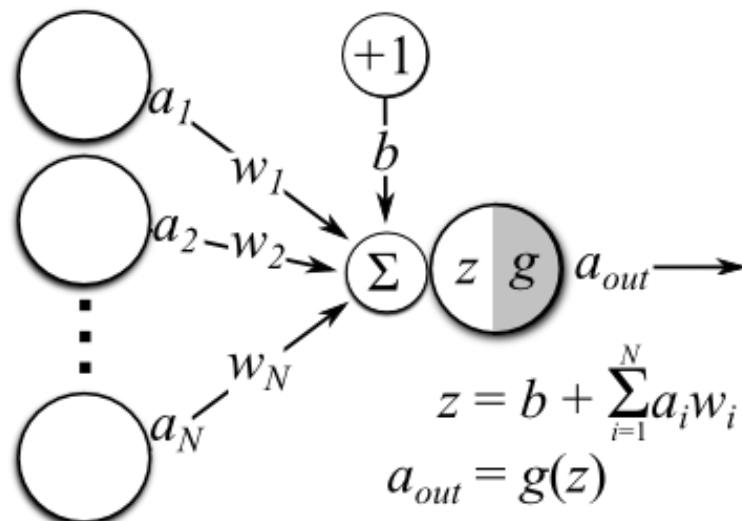
Neuron — the basic unit of a neural network. It receives a certain number of inputs and a bias (a 'bias' value, a scaling factor). When a signal (value) arrives, it is multiplied by a weight value. If a neuron has 4 inputs, it will have 4 weight values, which will be adjusted during training. The general formula is:

$$Y = \sum (\text{weight} * \text{input}) + \text{bias}$$

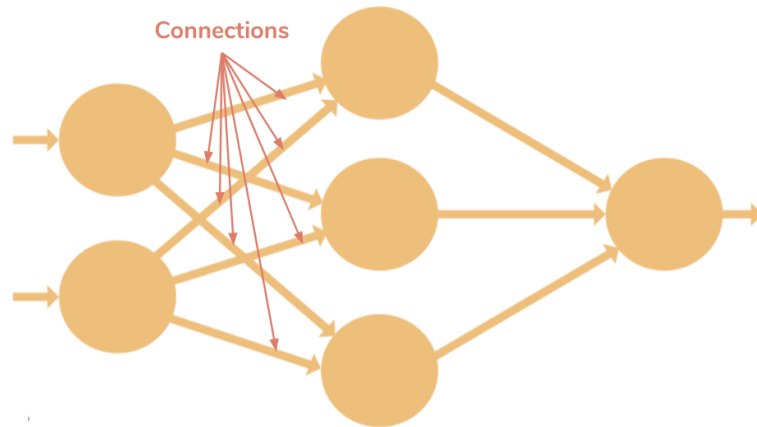
$$z = x_1 * w_1 + x_2 * w_2 + \dots + x_n * w_n + b * 1$$

$$\hat{y} = a_{out} = \text{sigmoid}(z)$$

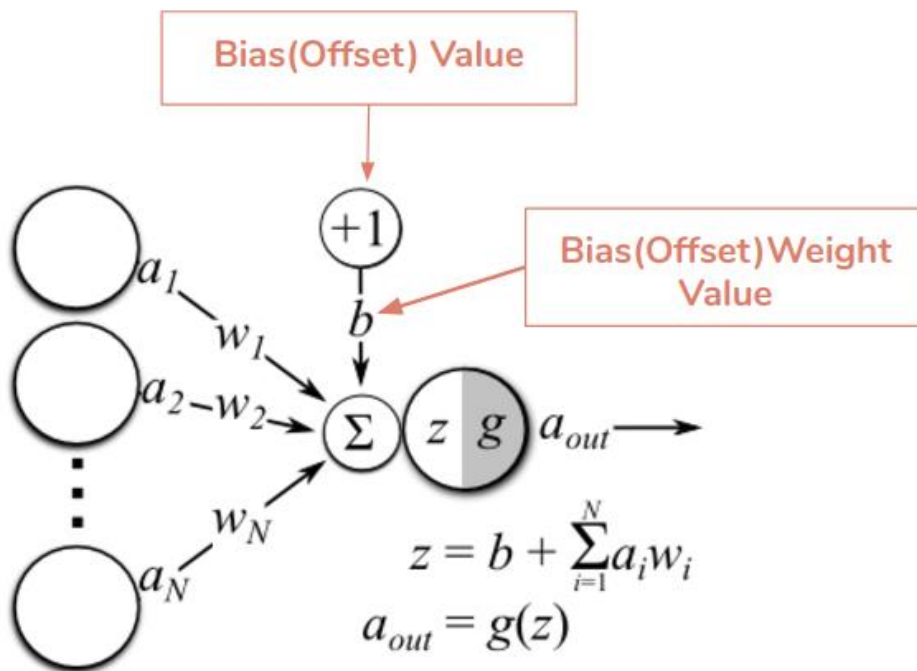
$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$



The operations that occur at a neuron level in a neural network.

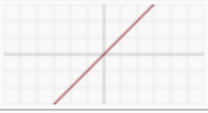

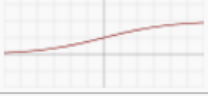
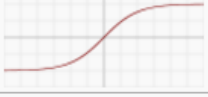
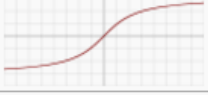






Connections — connects a neuron from one layer to another neuron from a different layer or the same layer. A connection always has an associated weight. The purpose of training is to update this weight value to reduce the error (loss).

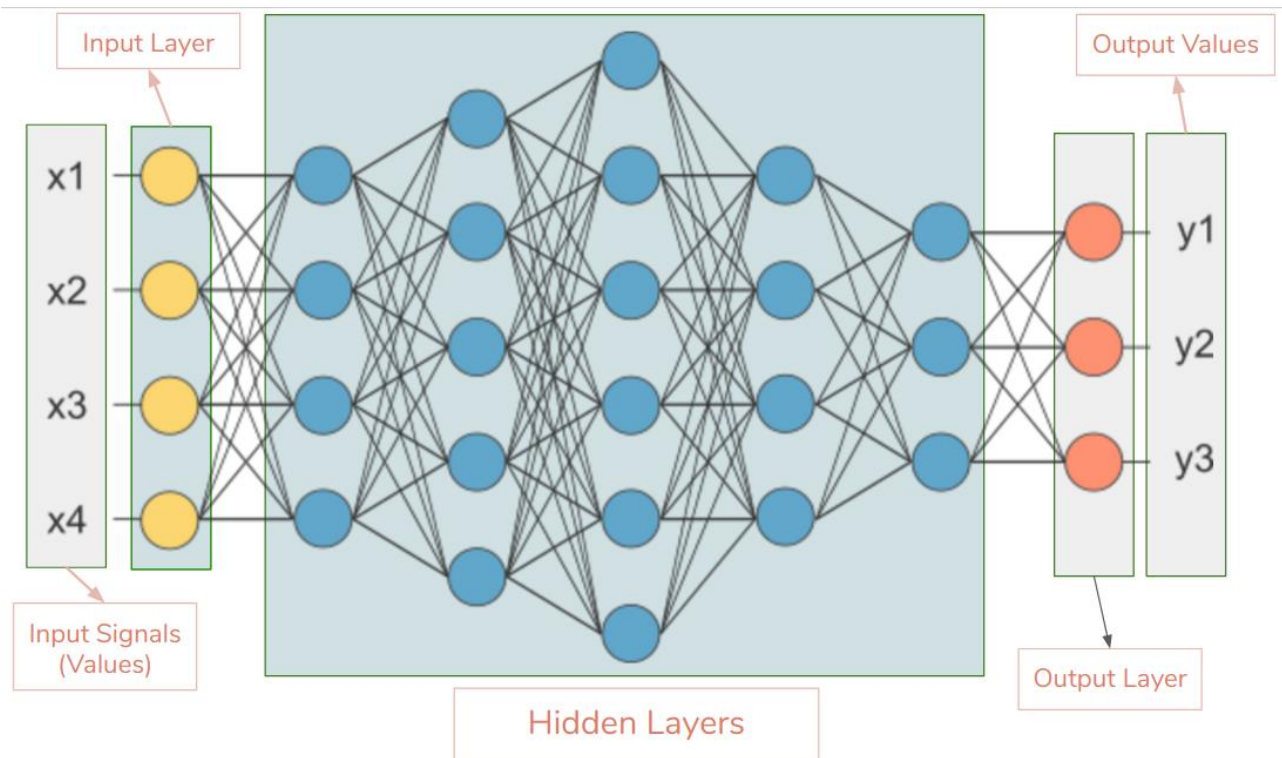


Bias (Offset) — an additional input for neurons, which always has the value of 1 and has its own connection weight. This ensures that even when all inputs are zero, a non-zero signal will still enter the neuron.

Activation Function, Transfer Function — activation functions are used to introduce nonlinearity into neural networks. The role of these functions is to confine values within a smaller range (for example, a sigmoid activation function reduces all values to the range [0, 1]). There are many activation functions used in deep learning. ReLU, SeLU, and TanH are often preferred over the sigmoid function.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Input Layer — the first layer in the neural network. It receives input signals (values) and transmits them to the next layer. It does not perform any operations on the input signals (values) (it does not have an activation function), nor does it have weights or biases. In the network shown in the figure, there are 4 input signals: x_1 , x_2 , x_3 , x_4 .



The basic structure of a neural network

Hidden Layers — hidden layers have neurons (nodes) that apply different transformations to the input data. A hidden layer is a collection of neurons arranged vertically (in our representation). In the image above, there are 5 hidden layers. The first hidden layer has 4 neurons (nodes), the second has 5 neurons, the third has 6 neurons, the fourth has 4 neurons, and the fifth has 3 neurons. The last hidden layer transmits the values to the output layer. All neurons in a hidden layer are connected to each neuron in the following layer, therefore we have fully connected hidden layers.

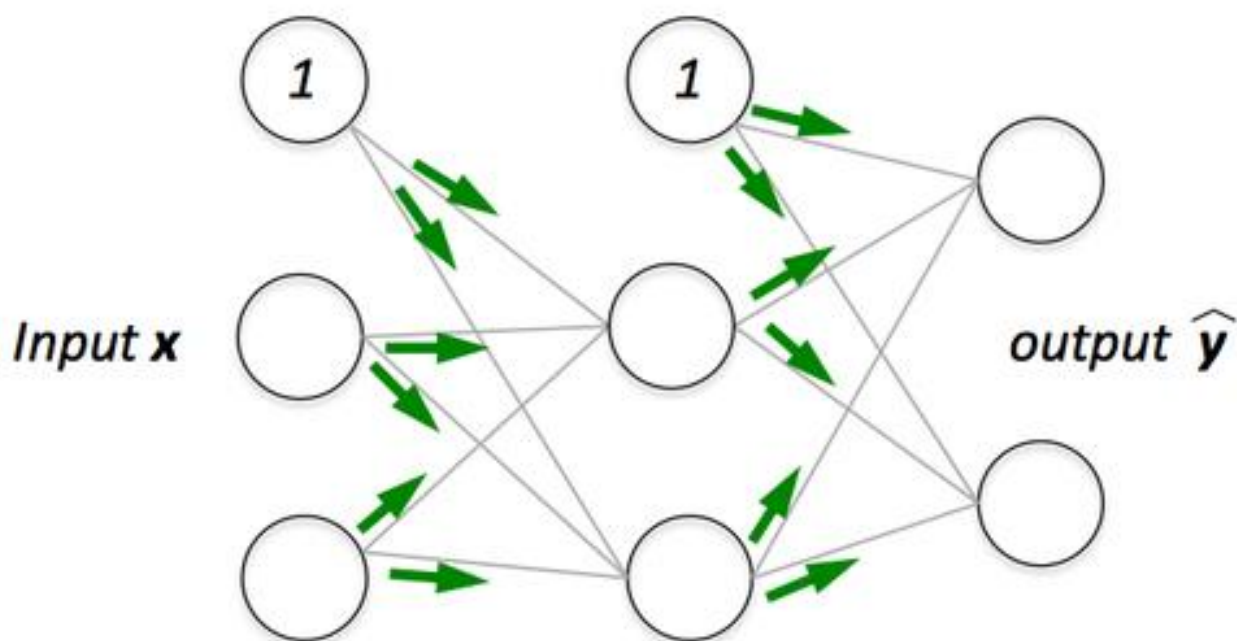
Output Layer — the last layer in the network and receives input from the last hidden layer. With this layer, we can obtain the desired number of values and within a desired range. In this network, there are 3 neurons in the output layer: y_1 , y_2 , y_3 . In the case of a network for recognizing handwritten digits from 0 to 9, there would be 10 neurons in the output layer (one for each digit).

Input Shape — the size of the input matrix that we pass to the input layer. The input layer of our network has 4 neurons and expects 4 values for a data sample. The input size for our network is $(1, 4, 1)$, if we feed it with one sample at a time. If we feed 100 samples, the input shape would be $(100, 4, 1)$. The input size depends on how the data is structured/coded to solve the problem.

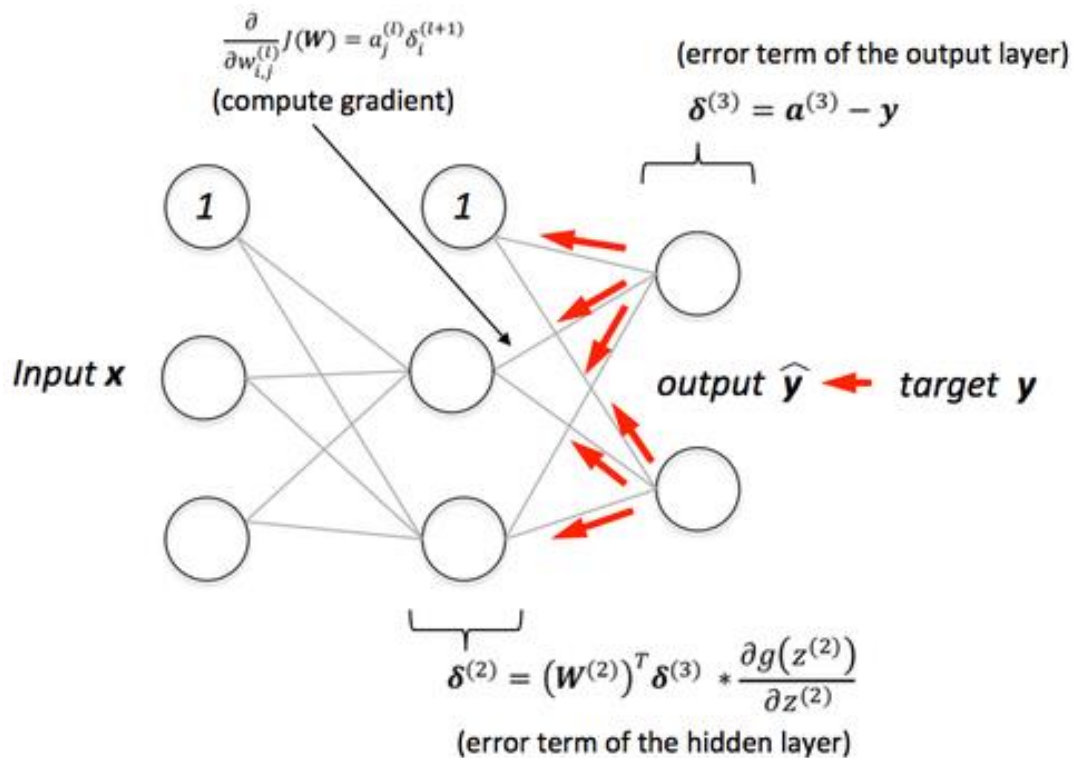
Weights — represent the importance of the connection between neurons (nodes). If the weight from node 1 to node 2 has a higher value, it means that neuron 1 has a greater influence on neuron 2.

A smaller weight means a lower importance of the value entering the neuron. Weights close to zero imply that a change in these inputs will hardly alter the output value. Negative weights mean that an increase in these inputs will decrease the output signal. The value of a weight shows how much influence an input to a neuron will have on the output from that neuron.

Forward Propagation — the process of feeding input values into the neural network in order to obtain an output that we call a predicted value. Sometimes we refer to forward propagation as inference. When we supply the input values to the first layer of the neural network, the propagation occurs without any operation. The second layer takes the values from the first layer and applies operations of multiplication, addition, and activation, then passes this value to the next layer. The same process repeats for subsequent layers. Finally, we obtain an output value from the last layer.

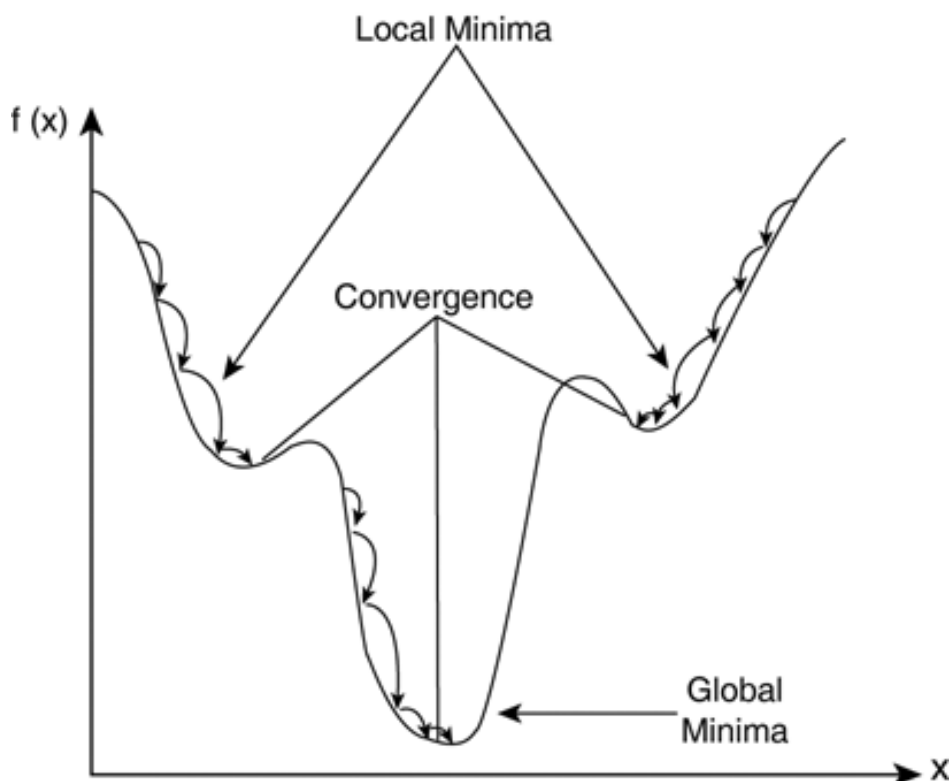


Back-Propagation — after forward propagation, we obtain an output value which is the predicted value. To calculate the error, we compare the estimated value with the actual output value (the one labeled by us for each training data set). For this purpose, we use a 'loss' function to calculate the error value, after which we calculate the derivative of the error value in relation to each weight in the neural network. Back-Propagation uses the chain rule from differential calculus. In the derivation chain, we first calculate the derivatives of the error value in relation to the weights of the last layer. These derivatives are called gradients. We use these gradient values to compute the gradients of the last layer. We repeat this process towards the input layer until we obtain gradients for every weight in the neural network. Then we subtract this gradient value from the weight value to reduce the error value. In this way, we approach a minimum value.



Backward Propagation

Learning rate — when training neural networks, we typically use the Gradient Descent method to optimize the weights. In each iteration, we use back propagation to calculate the derivative of the error calculation function (loss function) in relation to each weight. We subtract this value from the respective weight. The learning rate determines how quickly or slowly we want to update the weight values. The learning rate should be high enough that the network does not take too long to converge and low enough to find local minima and ultimately the global minimum.



Convergence — convergence occurs when, as iterations continue, the output increasingly approaches a specific value.

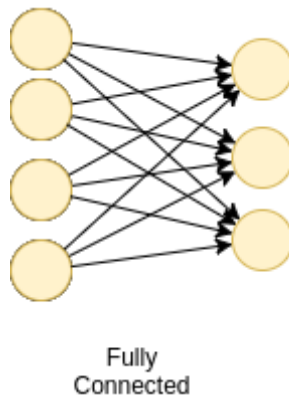
Regularization — is used to avoid the problem of overfitting. Through regularization, we penalize the error by adding an L1 norm (LASSO) or L2 norm (Ridge) to the weight vector w .

$$L(\text{loss function}) + \lambda N(w)$$

where λ is the regularization term and $N(w)$ is either the L1 or L2 norm.

Normalization — the process of rescaling one or more attributes (usually to the range from 0 to 1). Normalization is a useful technique when the data distribution is unknown or is known not to be Gaussian.

Fully Connected Layers — network architecture where the outputs of all nodes from one layer reach every node in the next layer. When all nodes in the N -th layer are connected to all nodes in the $(N+1)$ -th layer, we call these layers fully connected layers.



Loss/Cost Function — calculates the error for an input from the training dataset. The cost function is the average of the loss functions for the entire training set. Commonly used functions include: mse - mean squared error; binary_crossentropy - for binary logarithmic error (logloss); categorical_crossentropy - for multi-class logarithmic losses (logloss).

Model Optimizers — a search technique used to update weights in the model more efficiently. Examples include: SGD - Stochastic Gradient Descent; RMSprop - an adaptive learning rate optimization method (Geoff Hinton); Adam - Adaptive Moment Estimation, which uses adaptive learning rates.

Performance Metrics — used to measure the performance of the neural network. Metrics such as accuracy, loss, validation accuracy, validation loss, mean absolute error, precision, sensitivity, and the f1 score are some of the performance indicators.

Batch Size — the number of training examples used in one forward/backward pass (one forward + one backward). The larger the batch size, the more memory space is required.

Training Epochs — the number that shows how many times the model is trained with the entire training dataset. One epoch equals one forward pass and one backward pass with all the examples from the training dataset.