Del Algoritmo al Hardware:
Aprendizaje Automático en Sistemas Embebidos

From Algorithm to Hardware: Machine Learning in Embedded Systems

1 al 11 de Abril, 2025. Universidad Nacional de Mar del Plata - Mar del Plata - Argentina.

Machine Learning:
From Theory to Practice

Romina Soledad Molina, Ph.D.
MLab-STI, ICTP

Mar del Plata, Argentina - 2025 -

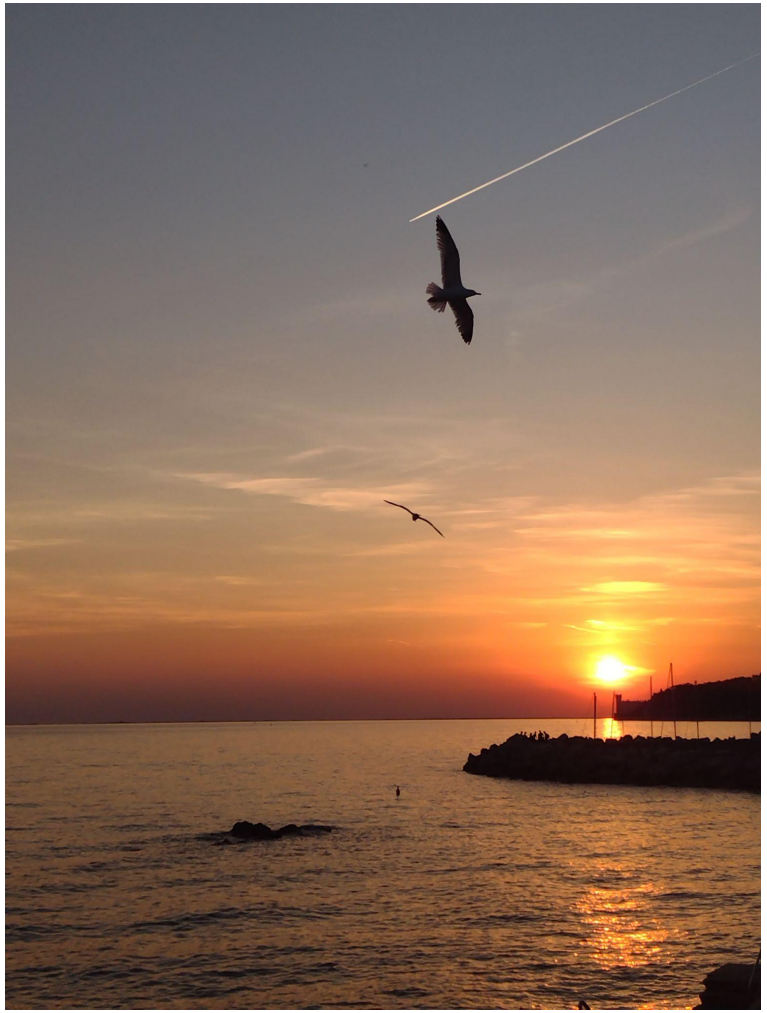
Universidad Nacional de Mar del Plata

Fundación Williams

The Abdus Salam International Centre for Theoretical Physics

Outline

- Machine learning.
- Machine learning: ANN, MLP, and CNN.
- Machine learning: training and inference.
- Basic ingredients.
- Metrics.
- General steps Keras+TensorFlow.
- Demo: MLP training for MNIST dataset



Machine learning

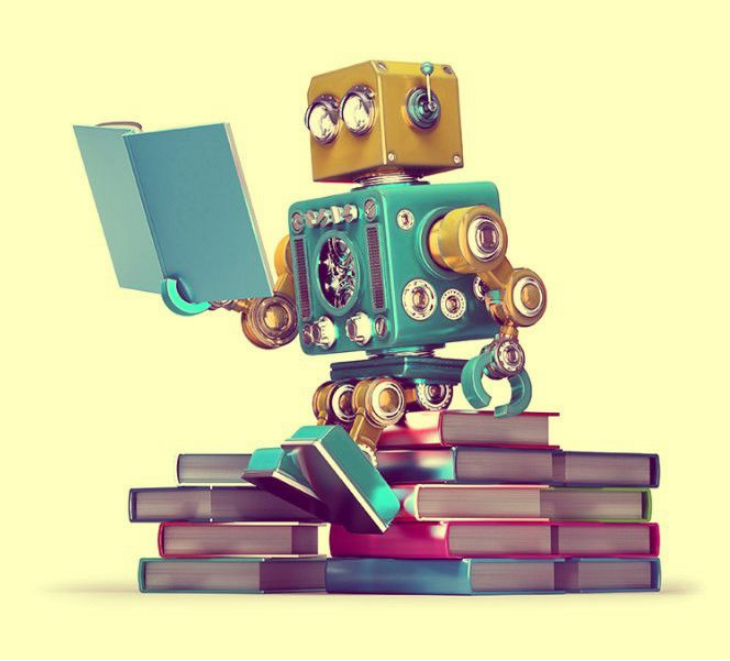
A simple game

Game - Developed by Code.org

https://studio.code.org/s/oceans/lessons/1/levels/2

Machine learning

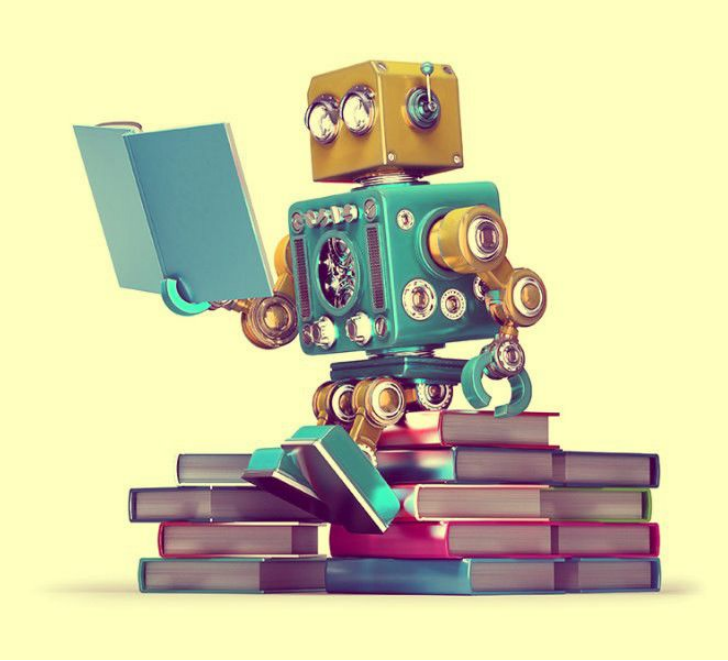
“Learning can be defined as the process of estimating the associations between input, outcomes, and system parameters using a limited number of observations.” – Vladimir Cherkassky et al.



Machine learning

“Learning can be defined as the process of estimating the associations between input, outcomes, and system parameters using a limited number of observations.” – Vladimir Cherkassky et al.

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed." – Arthur Samuel

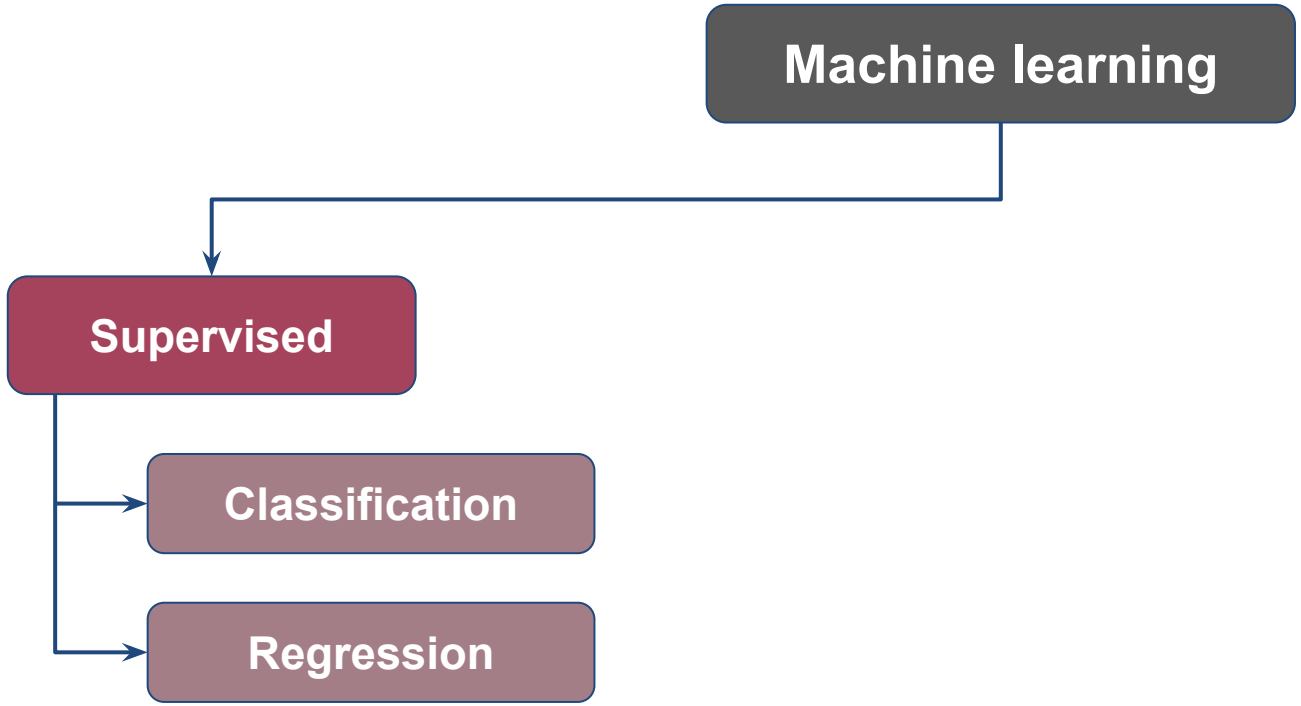


Machine learning

Classification

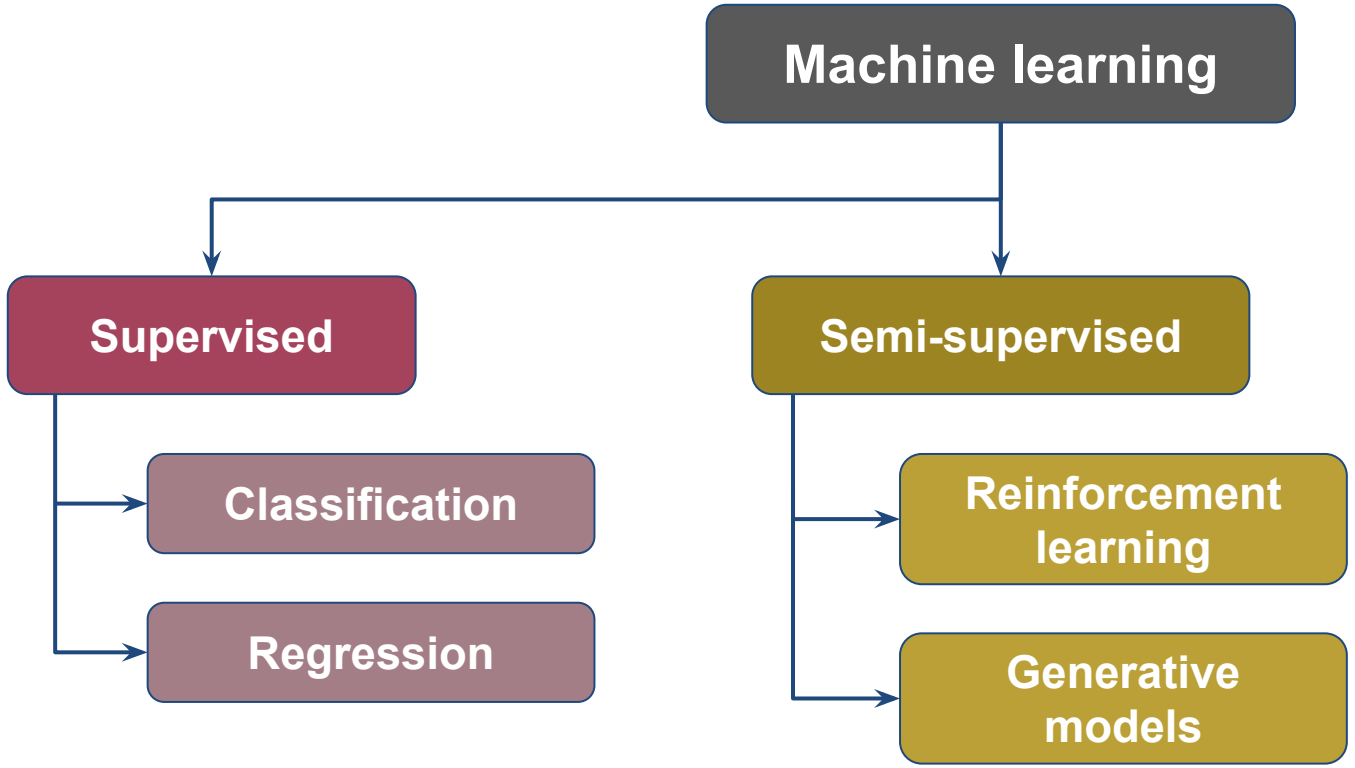
Machine learning

Classification



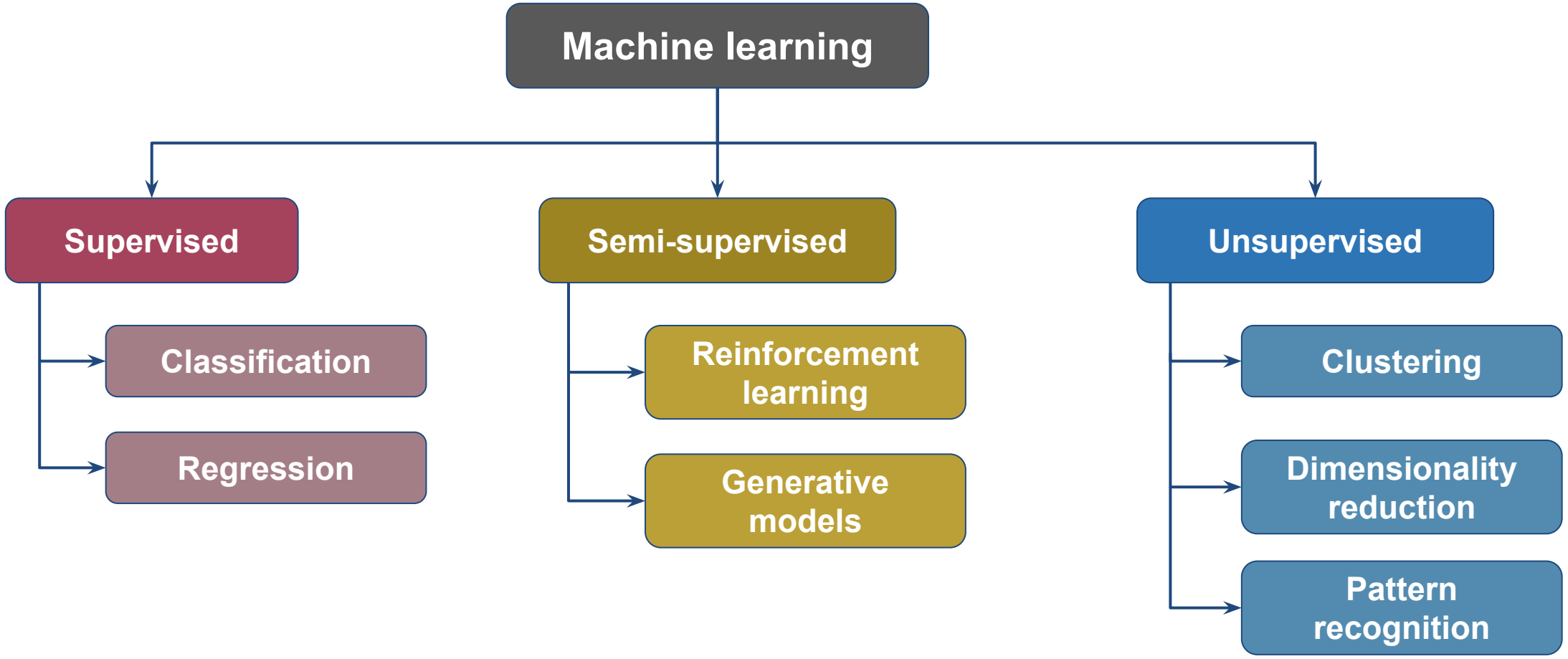
Machine learning

Classification



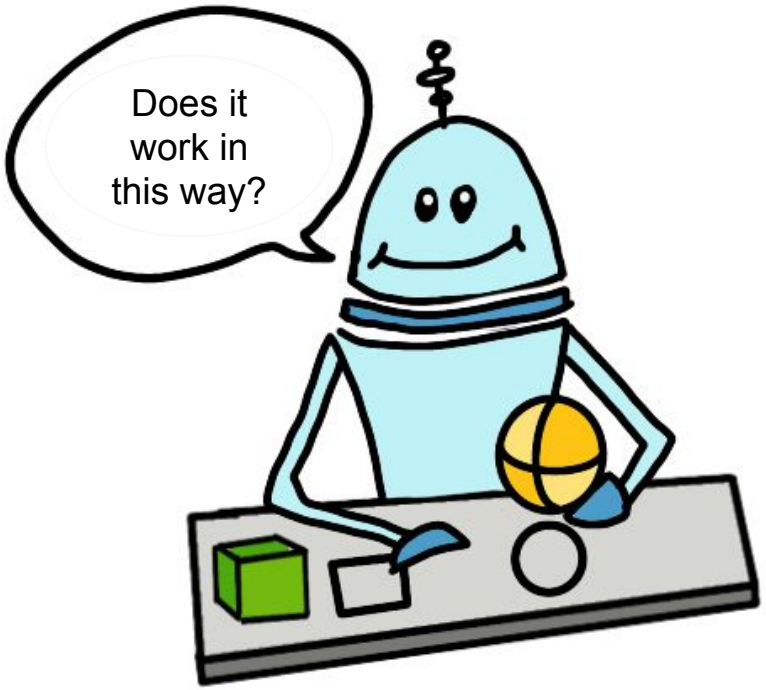
Machine learning

Classification



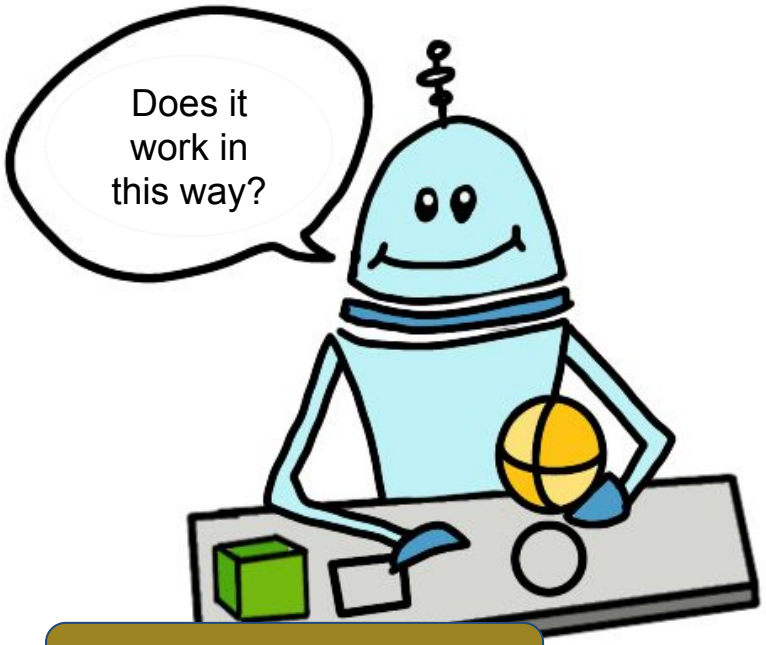
Machine learning

Classification



Machine learning

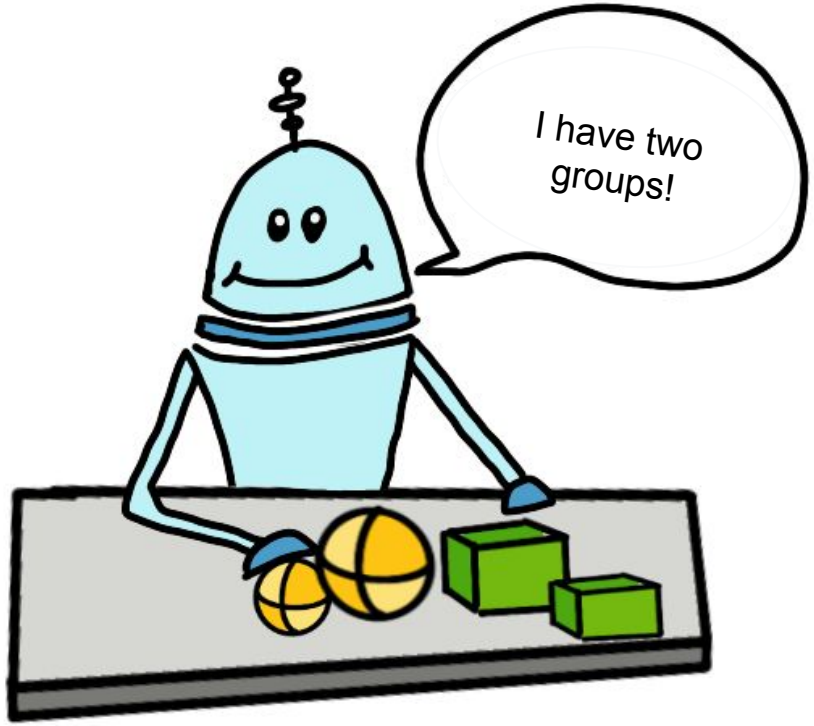
Classification



Semi-supervised

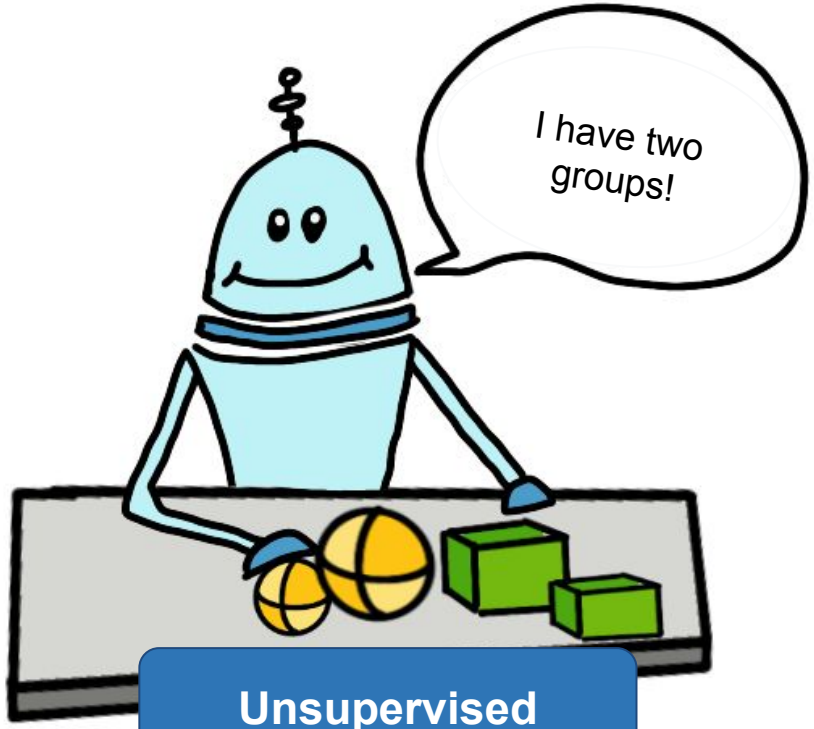
Machine learning

Classification



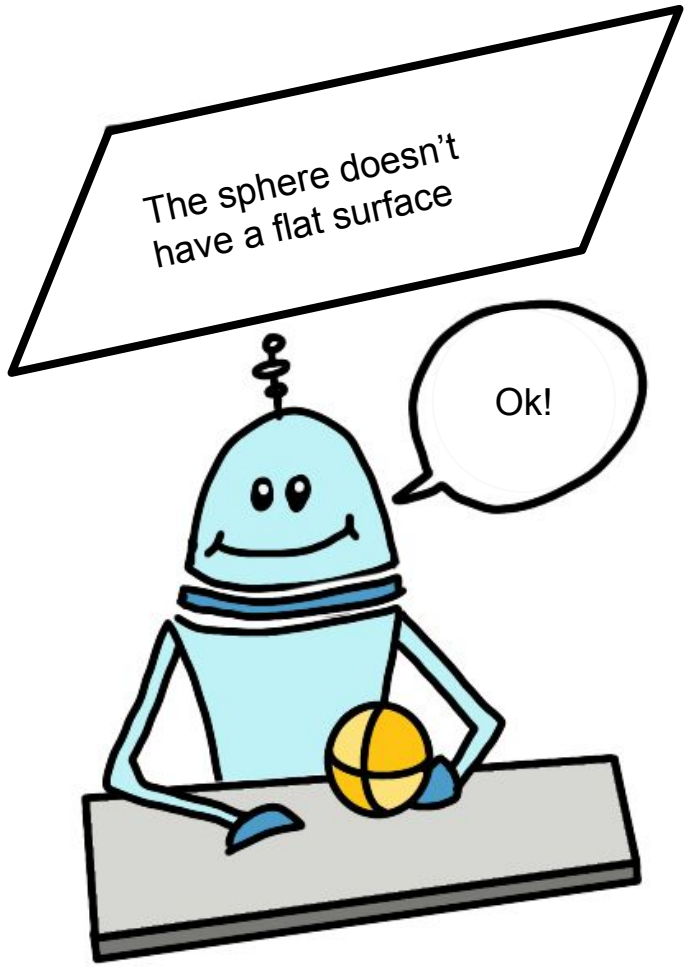
Machine learning

Classification



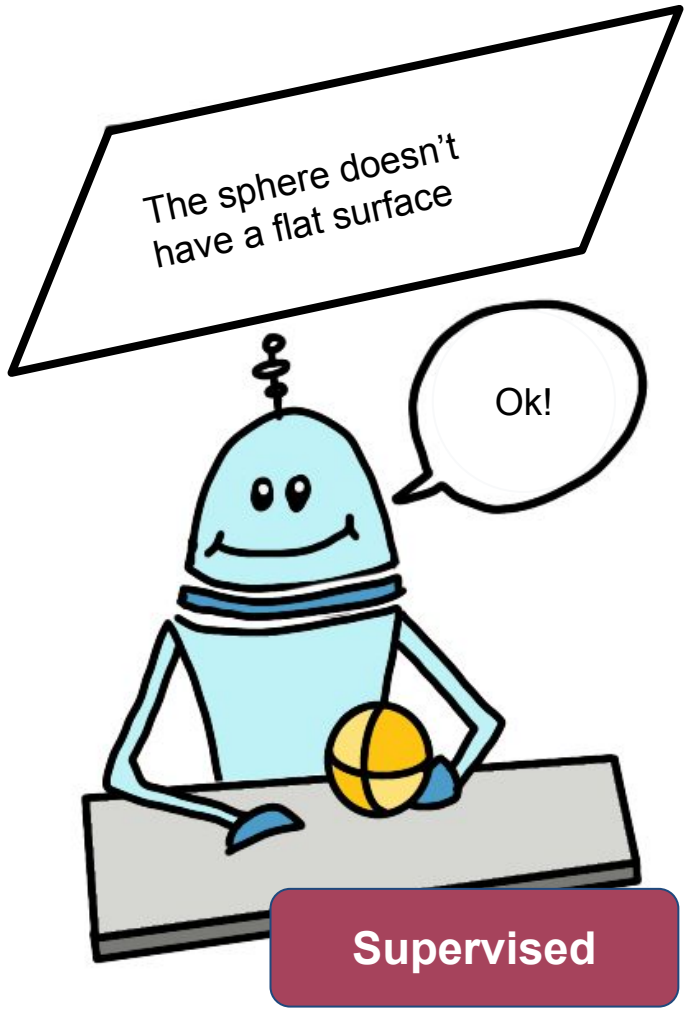
Machine learning

Classification



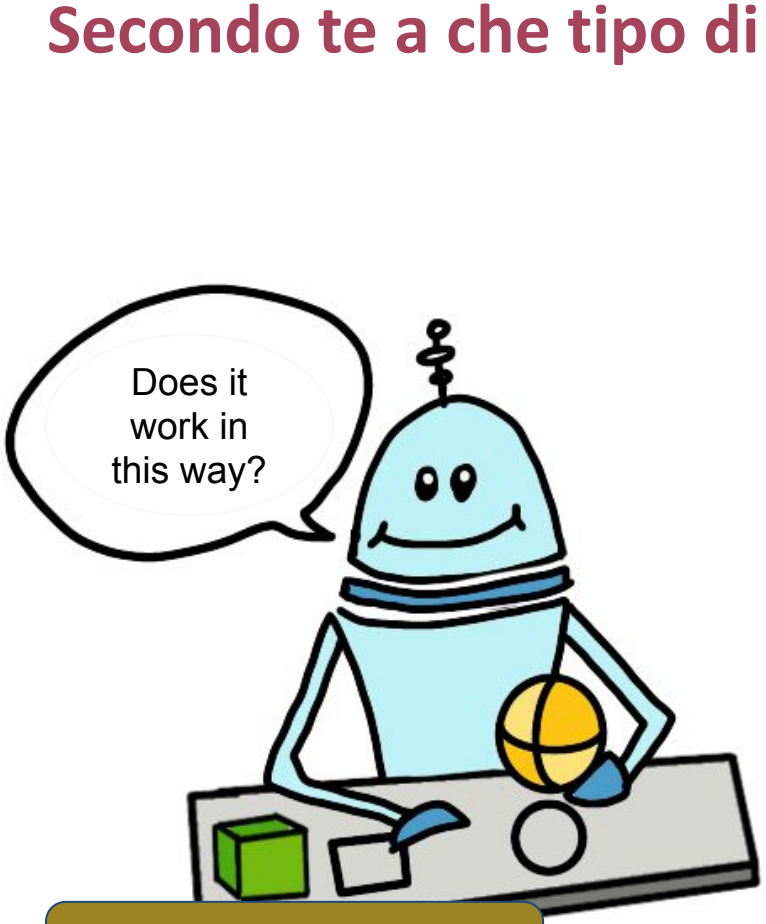
Machine learning

Classification

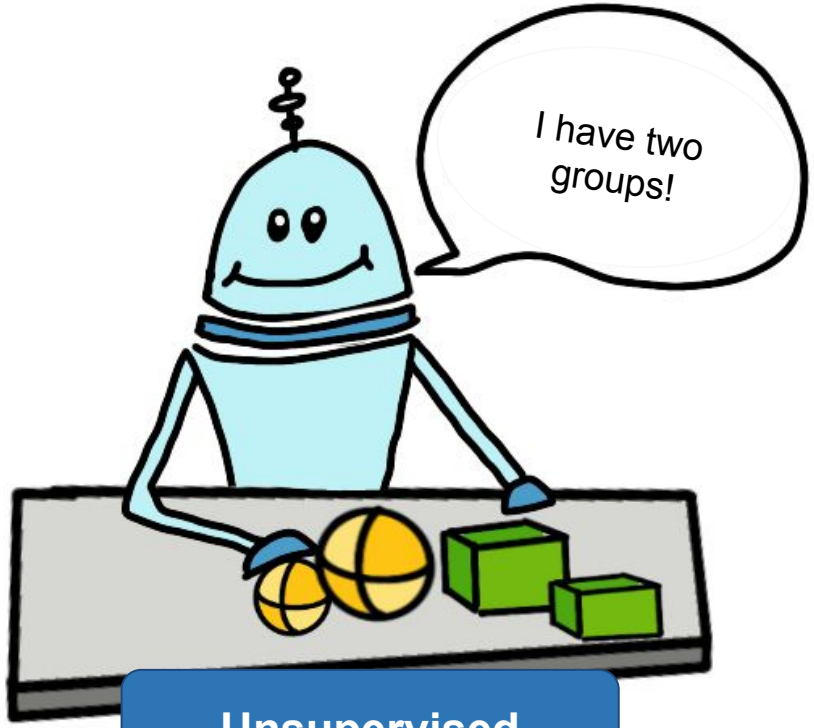


Machine learning

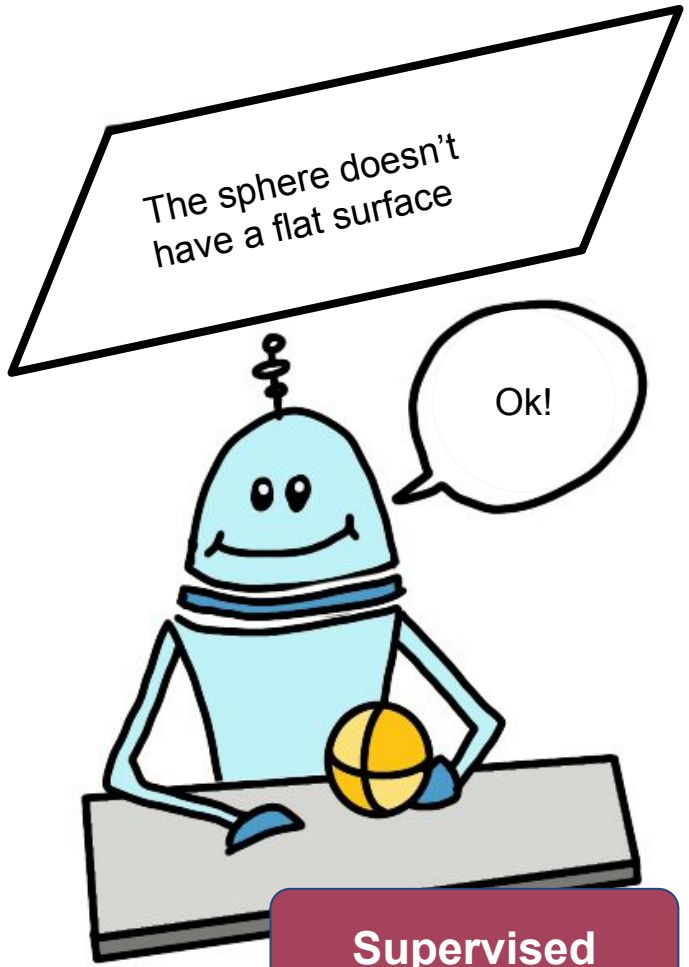
Secondo te a che tipo di apprendimento corrisponde?



Semi-supervised



Unsupervised



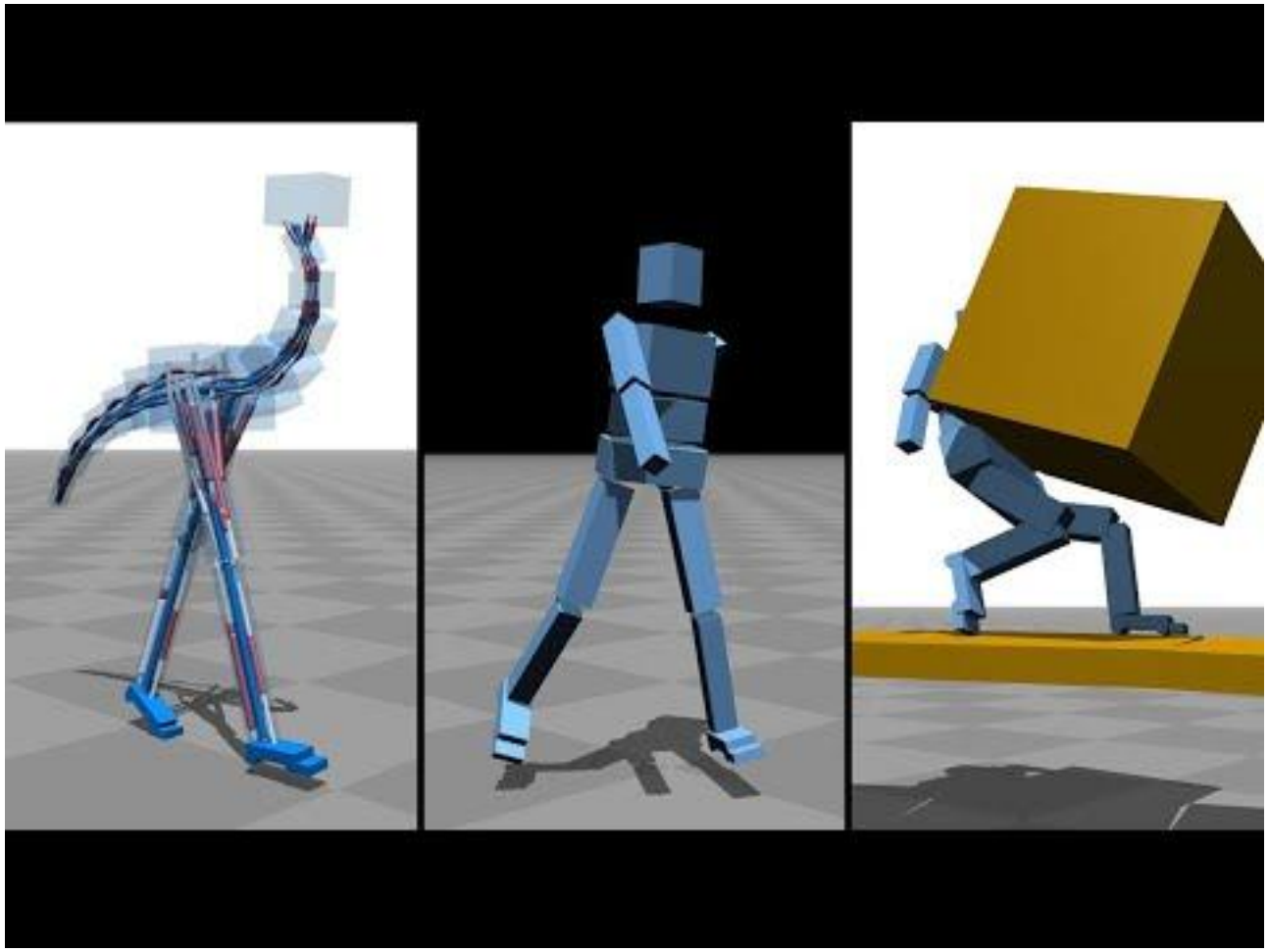
Supervised

A simple example

3blue1brown.com

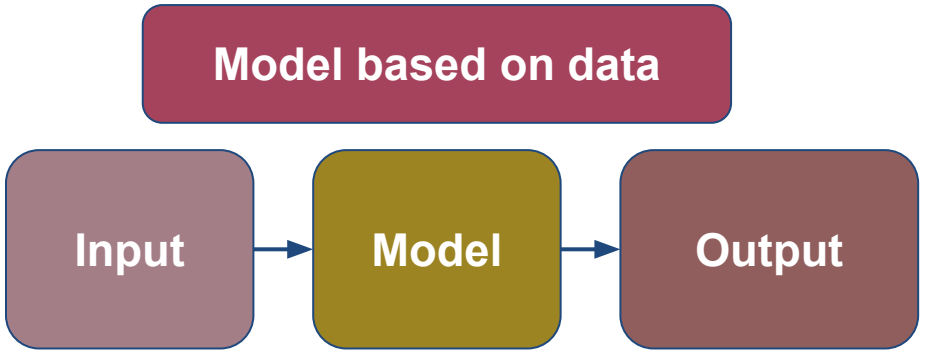
https://www.3blue1brown.com/lessons/neural-networks

Reinforcement learning

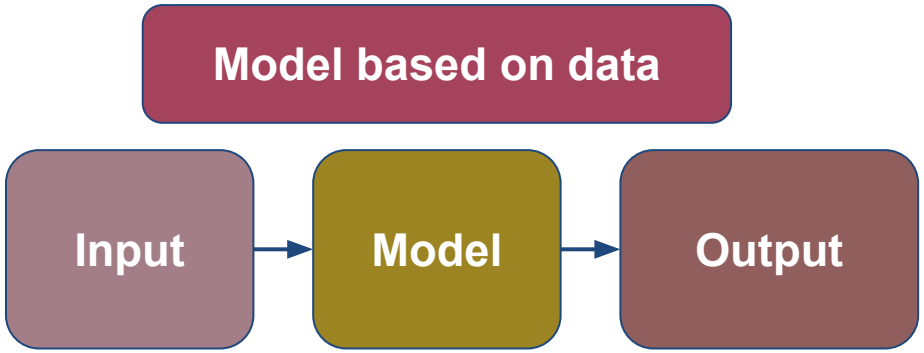


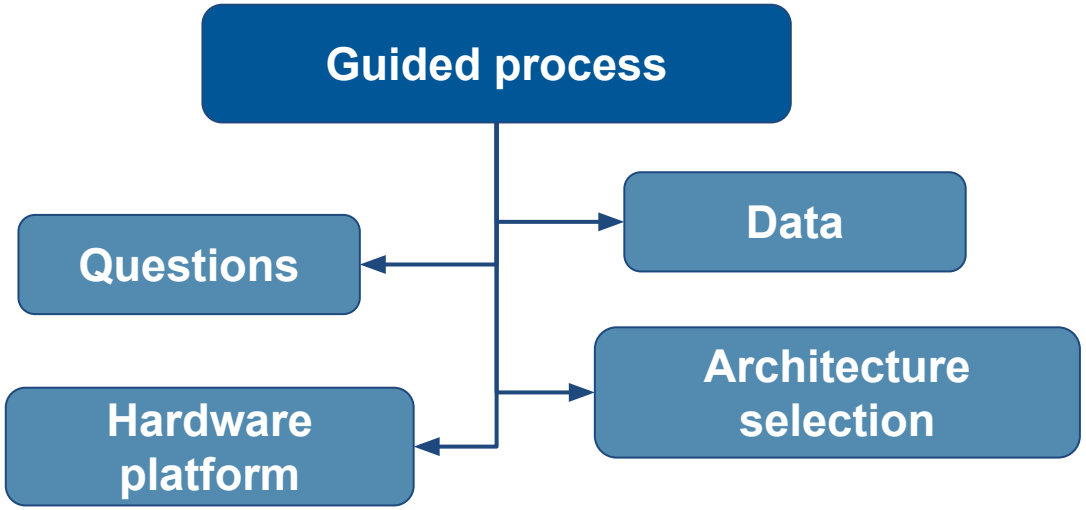
Video from Flexible Muscle-Based Locomotion for Bipedal Creatures https://www.youtube.com/watch?v=pgaEE27nsQw&t=21s

Machine learning

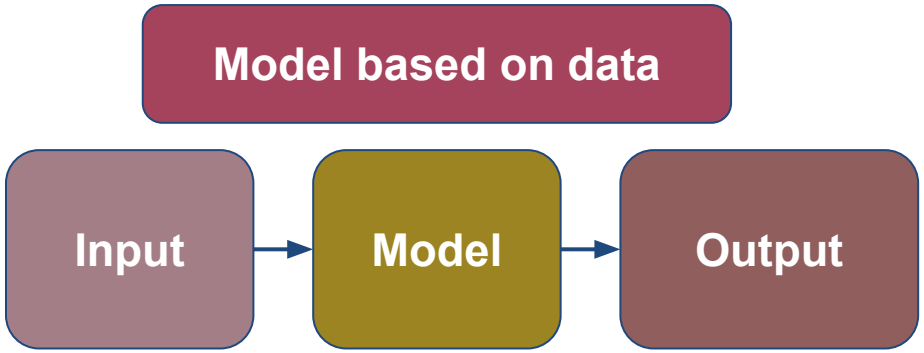


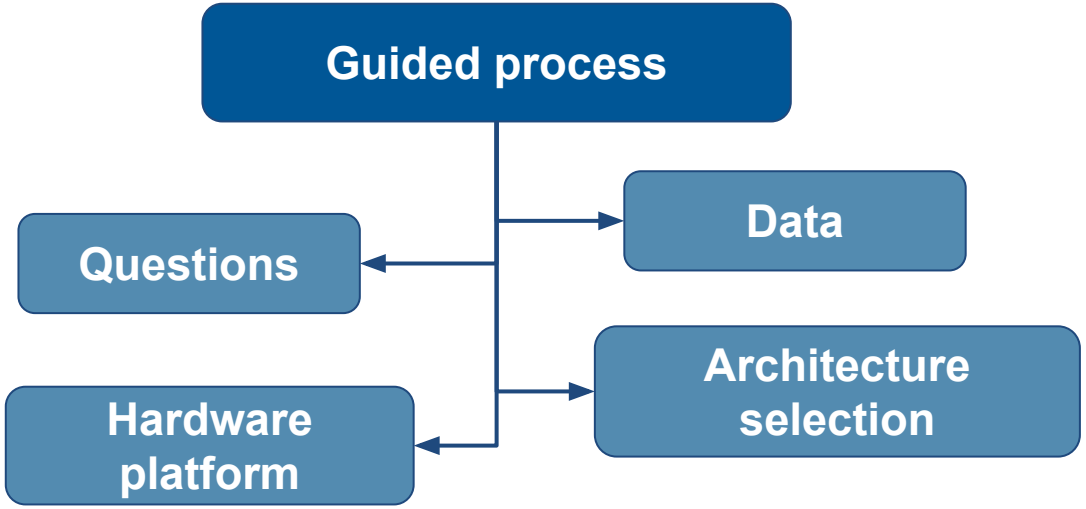
Machine learning

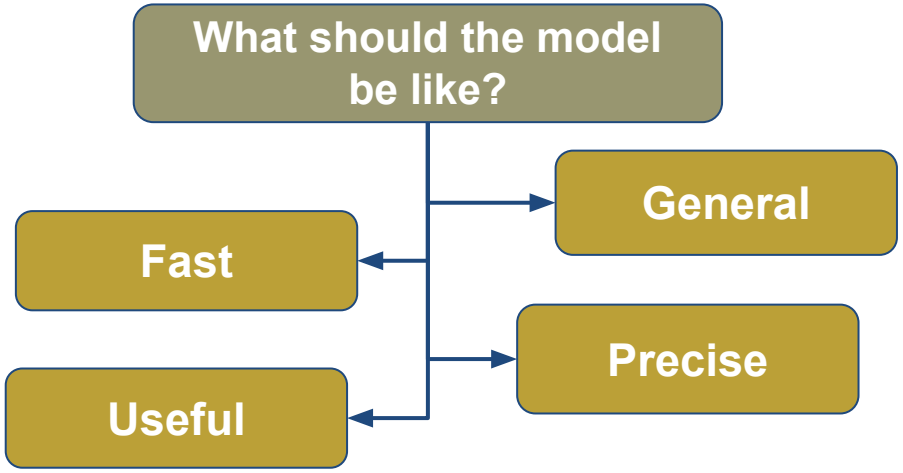




Machine learning







Machine learning

Generalization

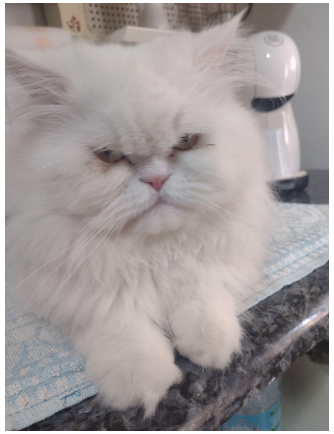
Image from
Togootogtokh, E., & Amartuvshin, A. (2018). Deep Learning Approach for Very Similar Objects Recognition Application on Chihuahua and Muffin Problem. *ArXiv, abs/1801.09573*.

Machine learning for classification

A classifier as example

Machine learning for classification

A classifier as example

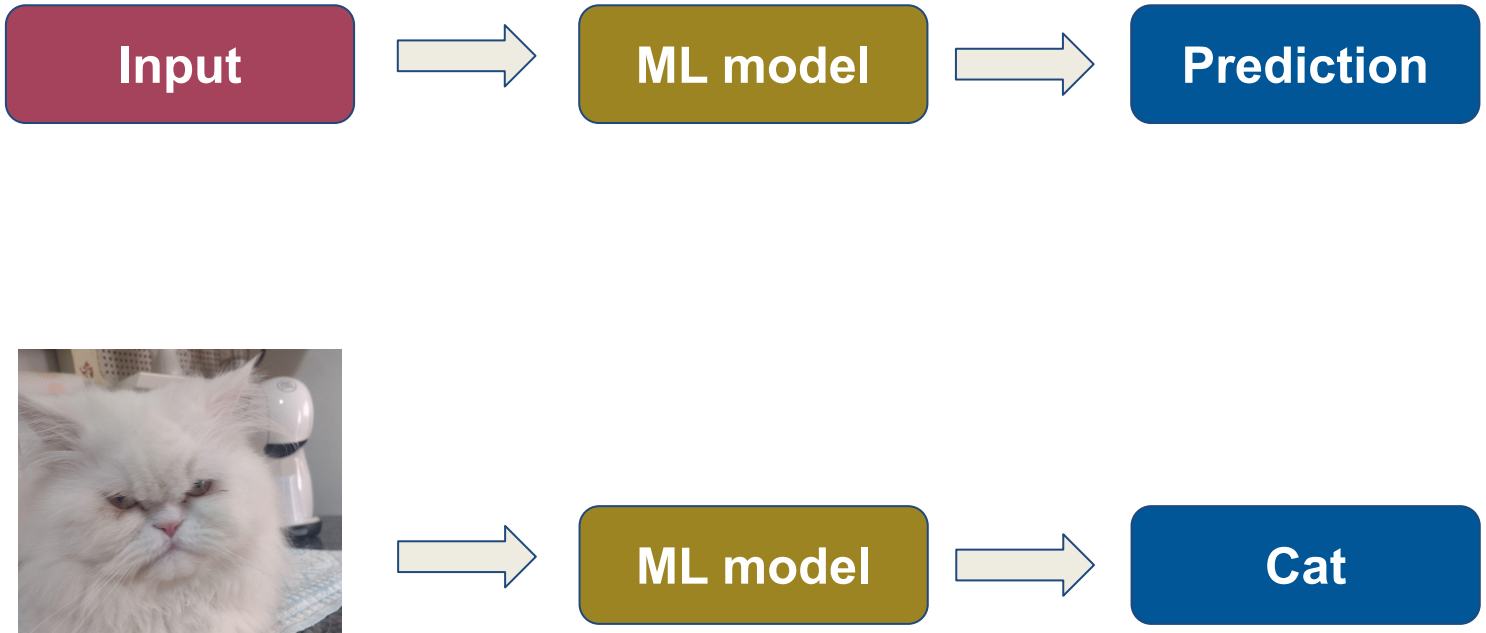


Machine learning for classification

A classifier as example

Machine learning for classification

A classifier as example

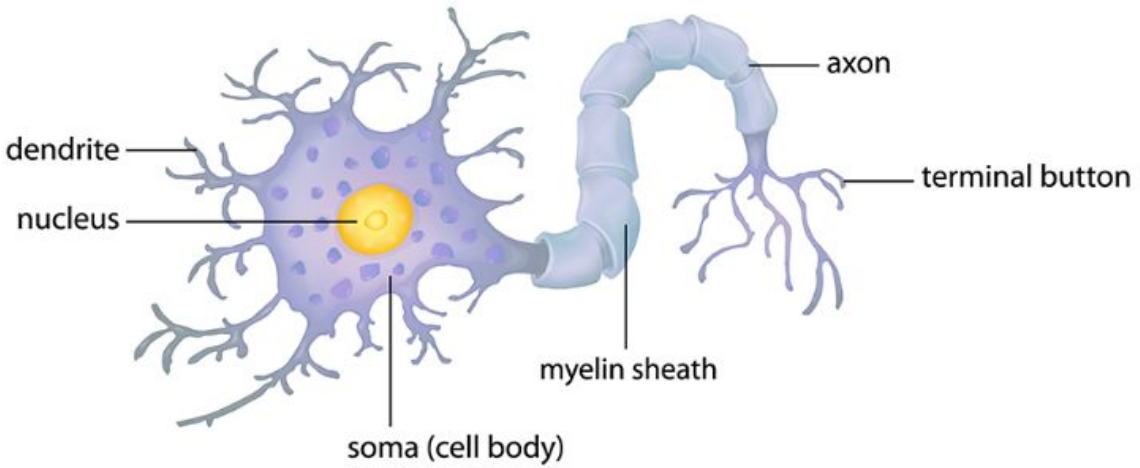


Machine learning: ANN, MLP, and CNN

Artificial neural networks

- **Inspired** by biological neurons [7].

- Each neuron processes electrochemical signals received from other neurons through its dendrites. If these signals are strong enough, the neuron becomes activated and transmits the signal through its axon, relaying it to the dendrites of other neurons, which may also be triggered. This is how message transmission takes place.

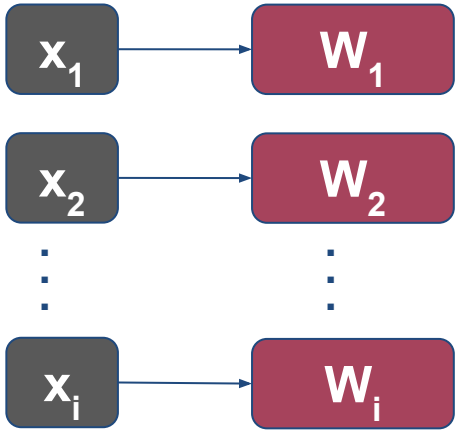


Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.

Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.

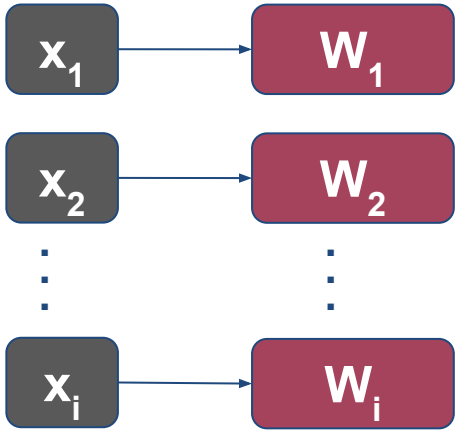


Weights are numerical values that represent the **importance of each feature or input in a model.**

They are **adjustable coefficients** that are fine-tuned during **training** to optimize performance and minimize prediction errors.

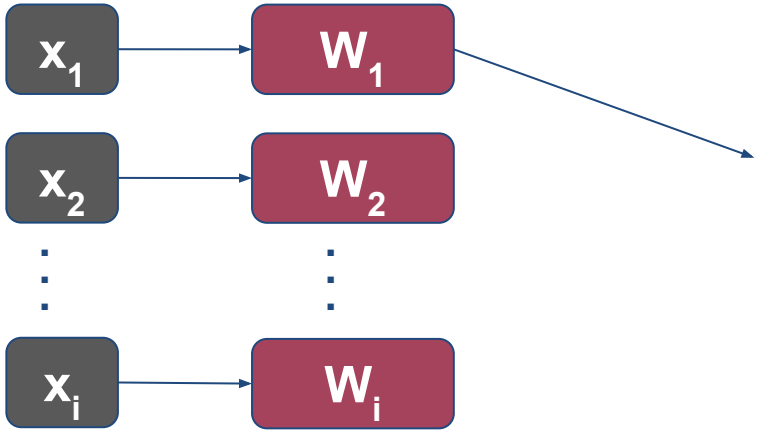
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



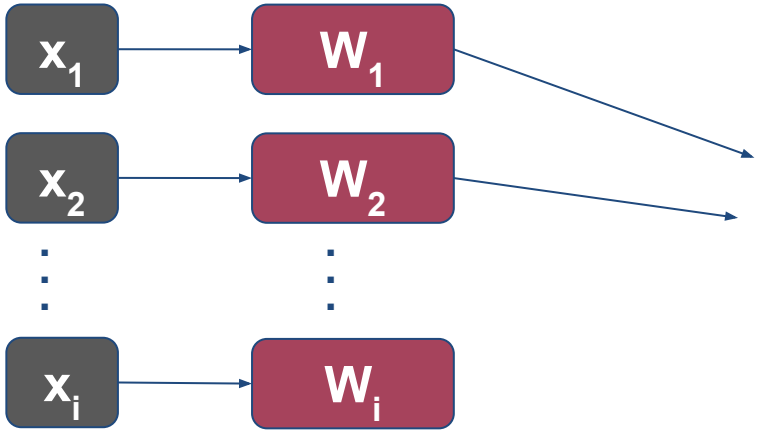
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



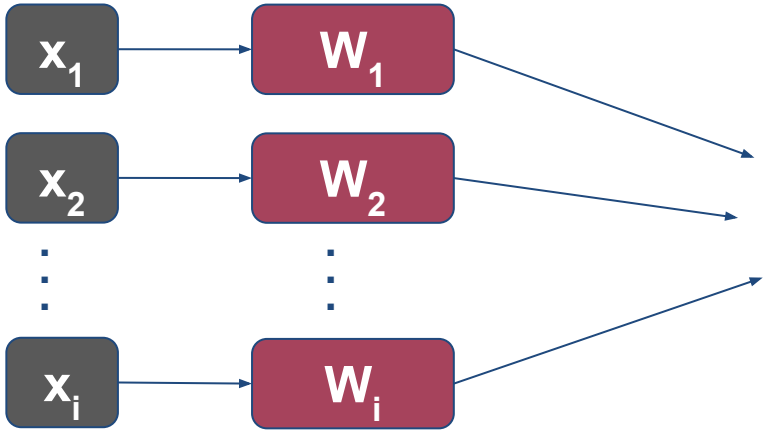
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



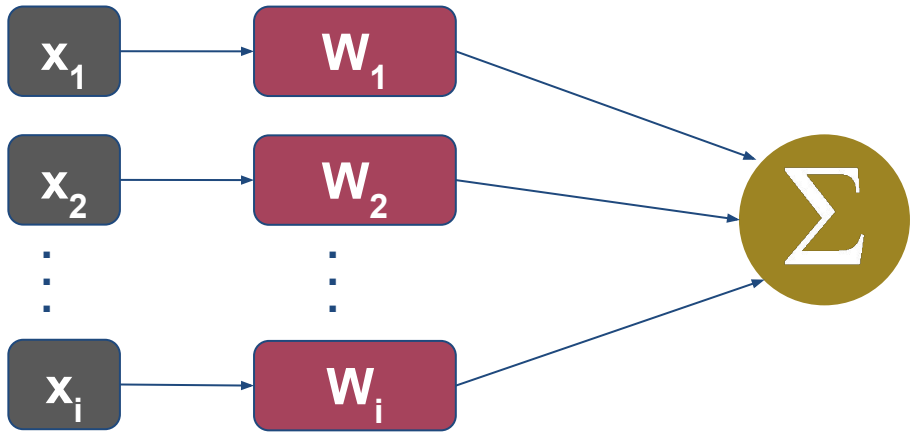
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



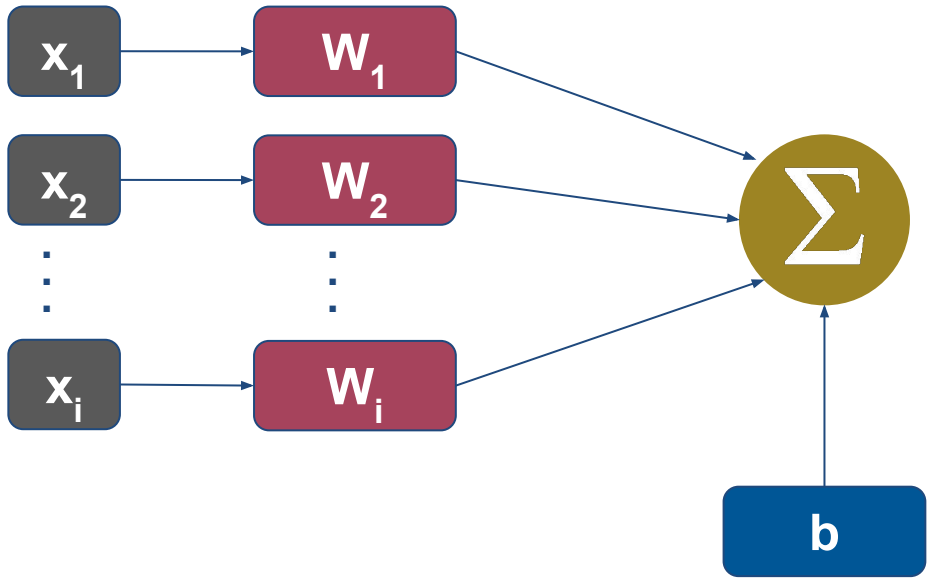
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



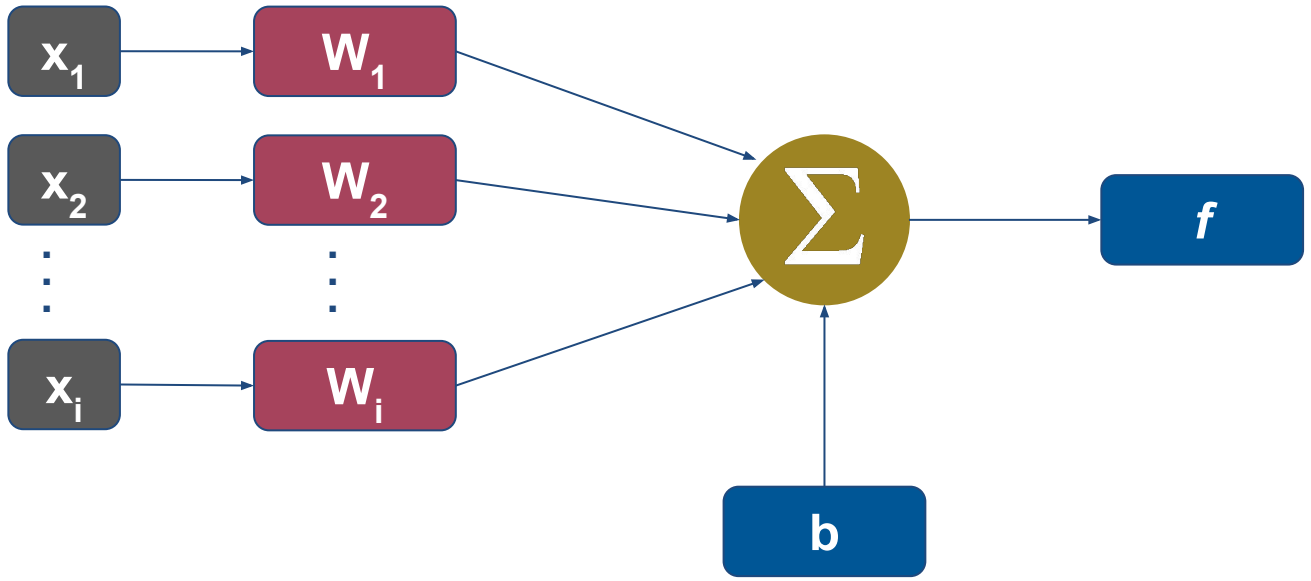
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



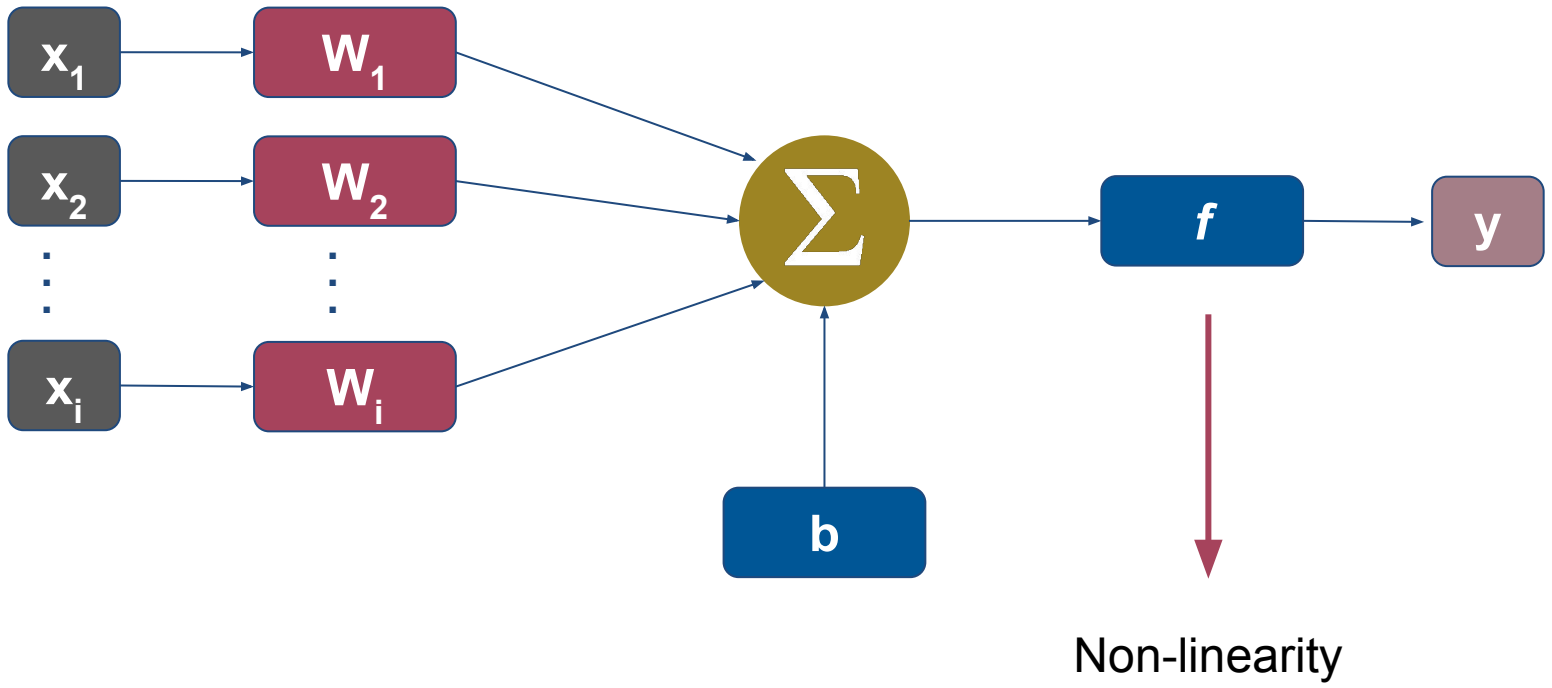
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



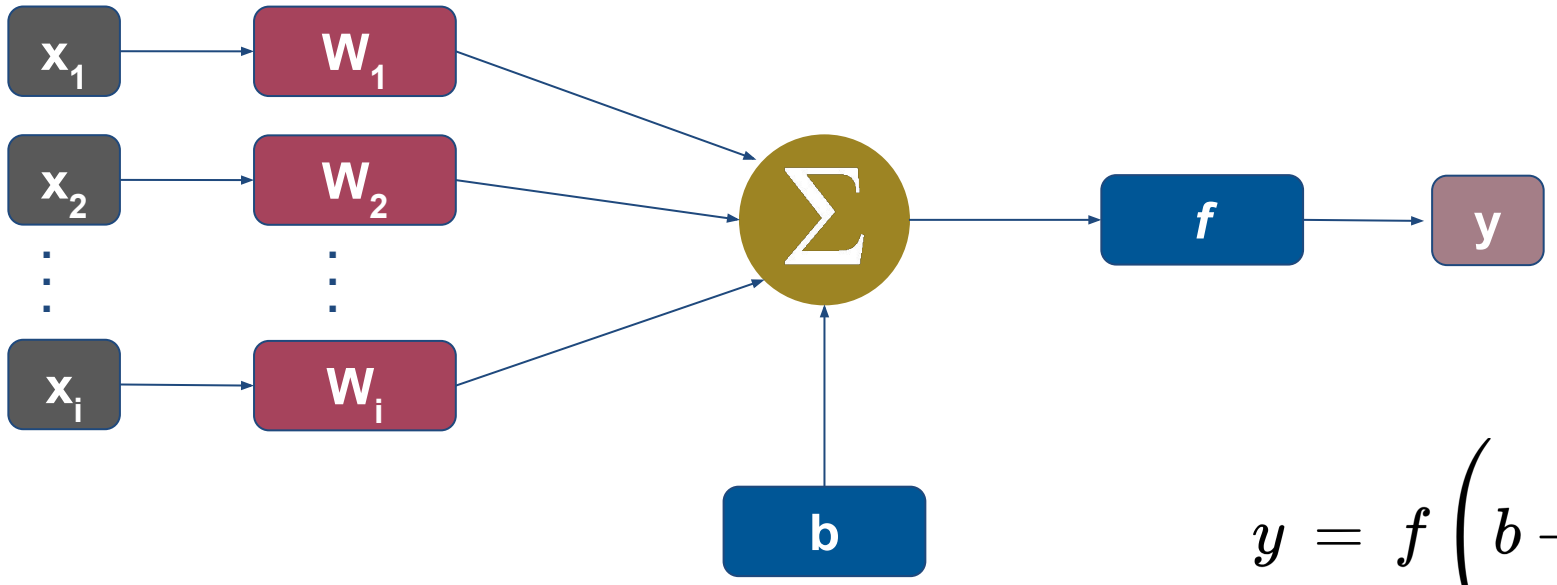
Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



Artificial neural networks

An artificial neural network (ANN) is composed of the interconnection of neurons (or nodes), distributed across different layers.



$$y = f\left(b + \sum_{m=1}^{i} x_m w_m\right)$$

Machine learning

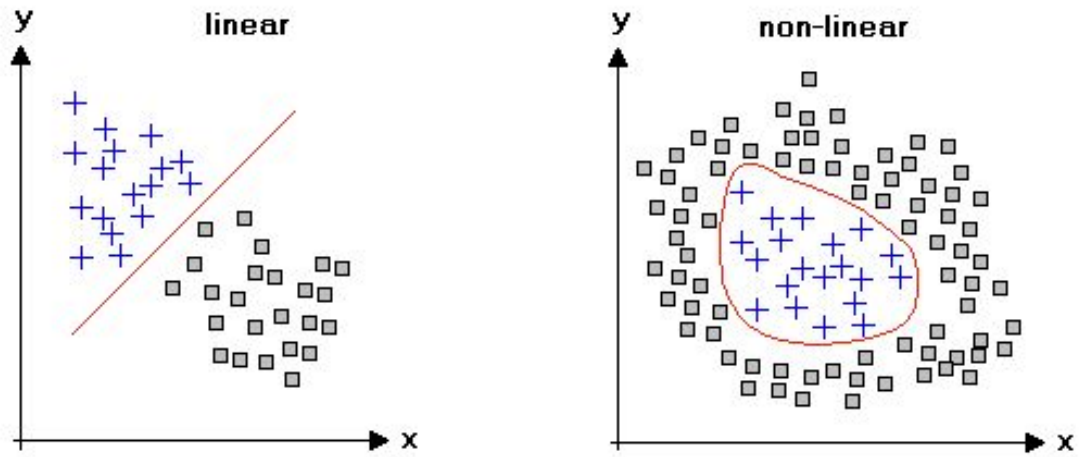
Activation functions

- *Question: Why activation functions are important?*

Machine learning

Activation functions

- *Question: Why activation functions are important?*
  - Activation functions introduce non-linearities into the network.



Machine learning

Activation functions

- Mathematical functions applied to the output of a neuron in a neural network. They **determine whether a neuron should be activated**, introducing non-linearity to the model, which helps it learn complex patterns.

Machine learning

Activation functions

- Mathematical functions applied to the output of a neuron in a neural network. They **determine whether a neuron should be activated**, introducing non-linearity to the model, which helps it learn complex patterns.

- **Types of activation functions**
  - Linear
  - Non-Linear: Sigmoid, ReLu, Leaky ReLu, Softmax.

Machine learning

Activation functions

- Mathematical functions applied to the output of a neuron in a neural network. They **determine whether a neuron should be activated**, introducing non-linearity to the model, which helps it learn complex patterns.

- **Types of activation functions**
  - Linear
  - Non-Linear: Sigmoid, ReLu, Leaky ReLu, Softmax.

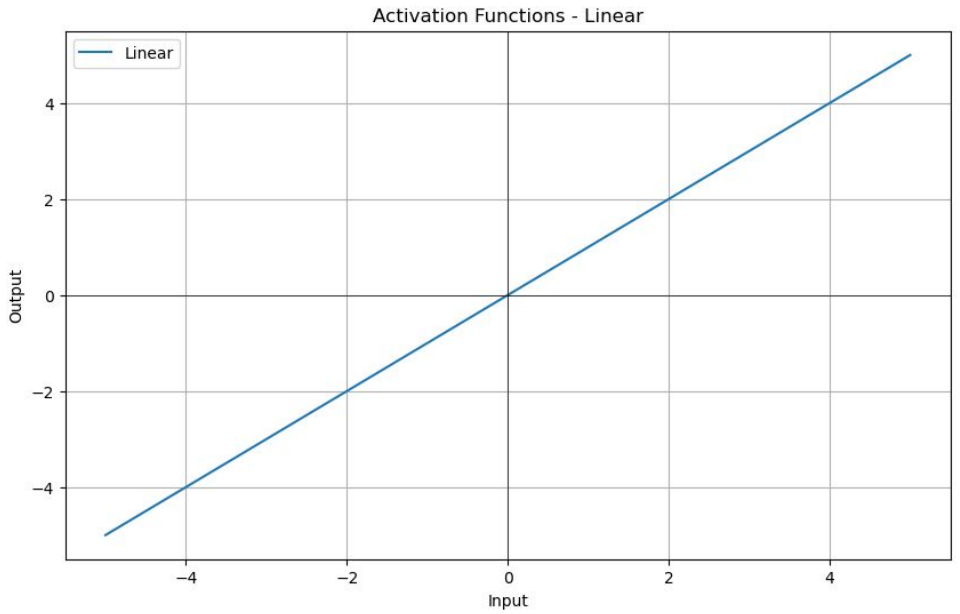
- **Tasks:**
  - Classification: Sigmoid (binary), Softmax (multi-class).
  - Hidden layers: ReLU, Leaky ReLU.
  - Regression: Linear activation or ReLU.

Machine learning

Activation functions

Linear

$$f(x) = ax$$



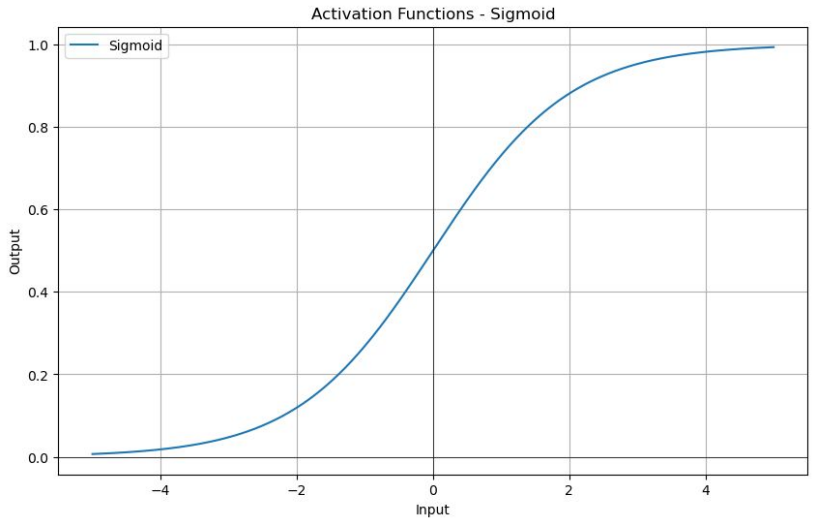
- Simple
- Useful for regression
- Cannot model non-linearity

Machine learning

Activation functions

Sigmoid

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



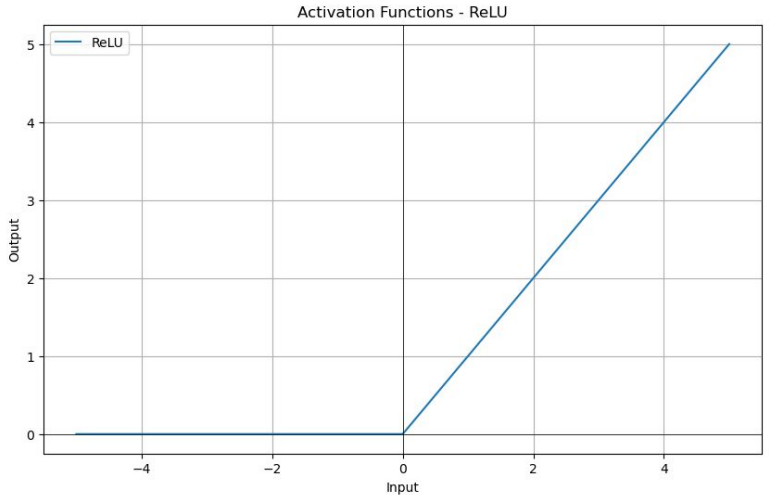
- Binary classification
- Slow training
- Good for probabilities

Machine learning

Activation functions

ReLu

$$f(x) = \max(0, x)$$



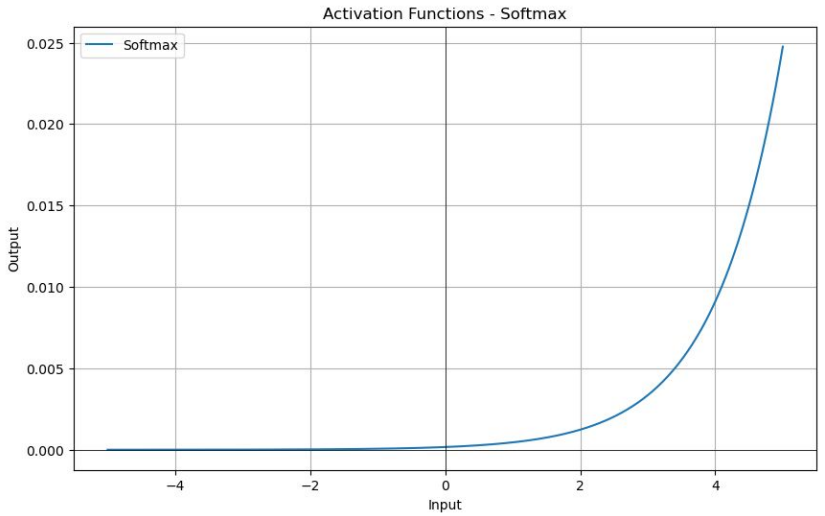
- Deep networks
- Neurons can stuck at zero
- Fast

Machine learning

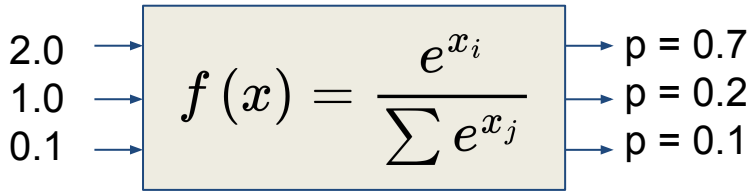
Activation functions

Softmax

$$f(x) = \frac{e^{x_i}}{\sum e^{x_j}}$$



- Multi-class classification
- Computationally expensive
- Converts outputs into probabilities



Machine learning

Visualization: activation functions
https://developers.google.com/machine-learning/crash-course/neural-networks/interactive-exercises?hl=es-419

Machine learning

Deep Neural Networks

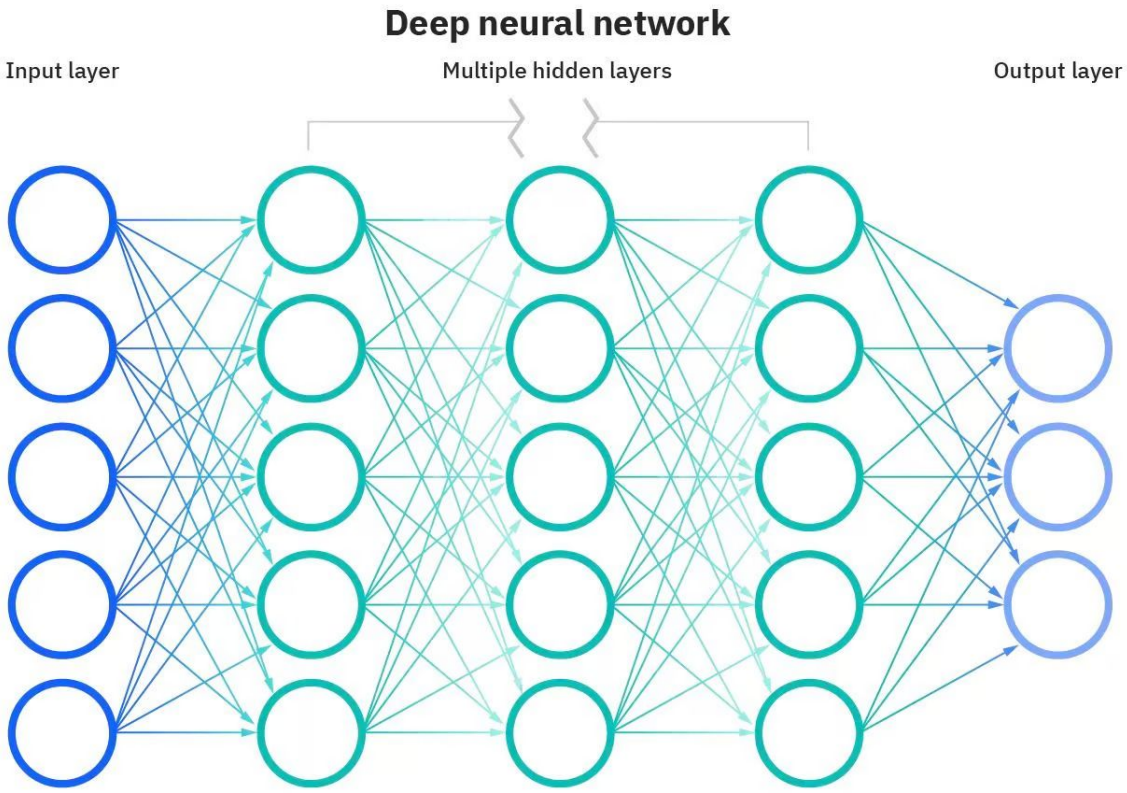


Image from https://www.ibm.com/think/topics/neural-networks

Artificial neural networks

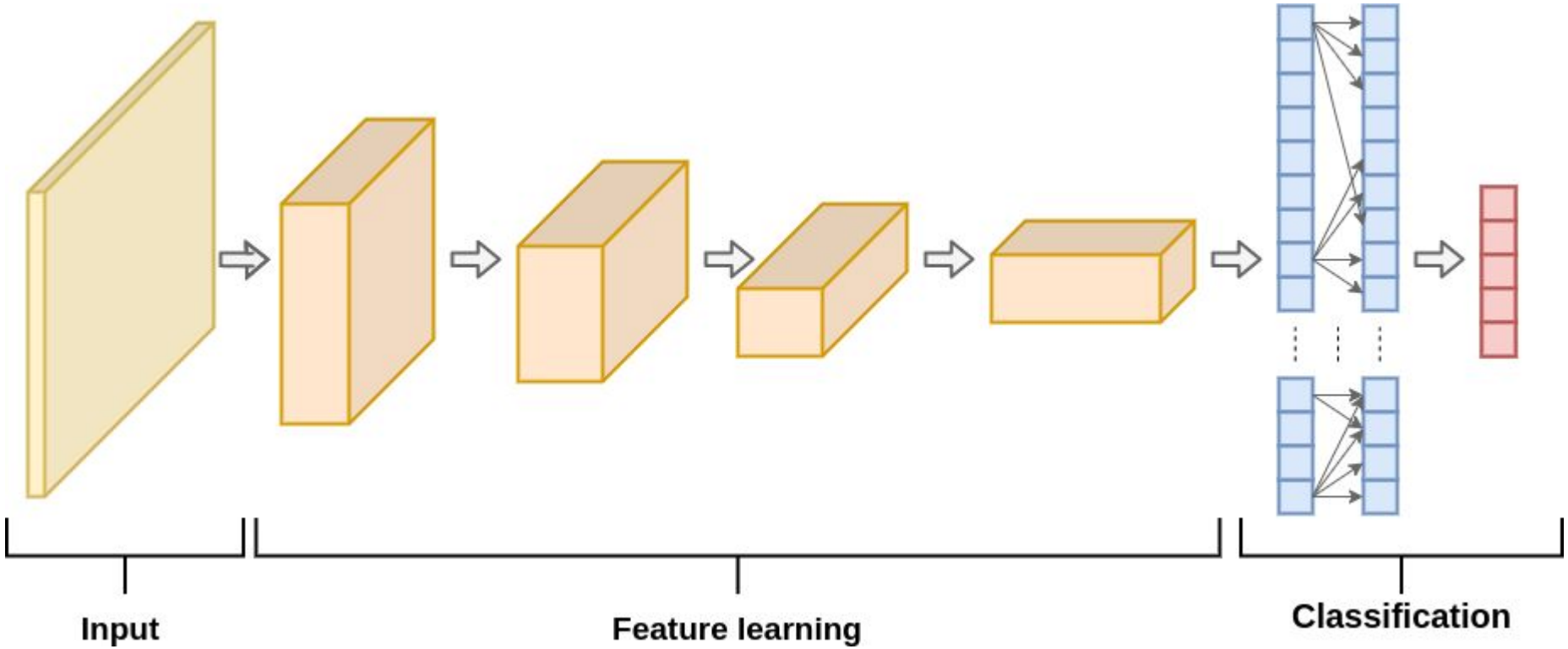
Considering a given layer l, the output $\mathbf{a}^{(l)}$ can be defined as:

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$$
$$\mathbf{a}^{(l)} = \boldsymbol{f}(\boldsymbol{z}^{(l)})$$

$\boldsymbol{W}^{(l)}$: weights matrix of the layer l.
$\boldsymbol{a}^{(l-1)}$: output of the previous layer (l-1)
$\boldsymbol{b}^{(l)}$: bias vector of layer l.
$\boldsymbol{f}(.)$: activation function

Machine learning

Convolutional Neural Network (CNN)



Machine learning

Basic layers that make up the CNN architecture

- The first layer: captures basic features.
  - For example, horizontal and vertical edges.
- The output moves to the next layer, which identifies more complex features.
  - For instance, corners or combinations of edges.
- As the network deepens, it becomes capable of recognizing even more intricate features, such as faces, objects, and more.

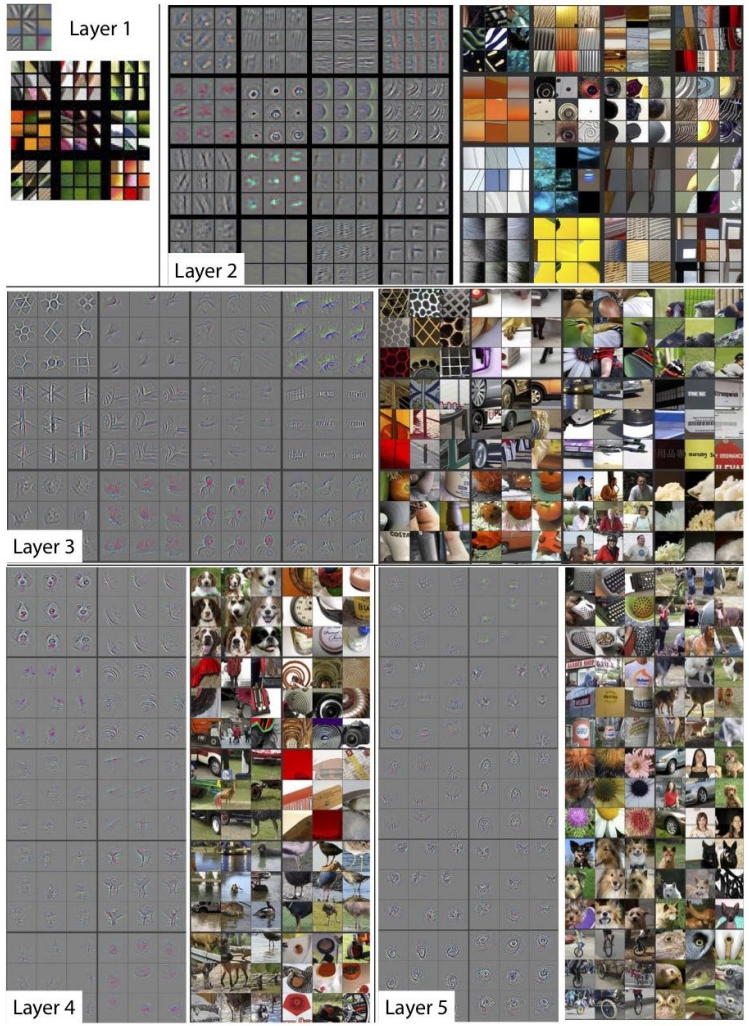
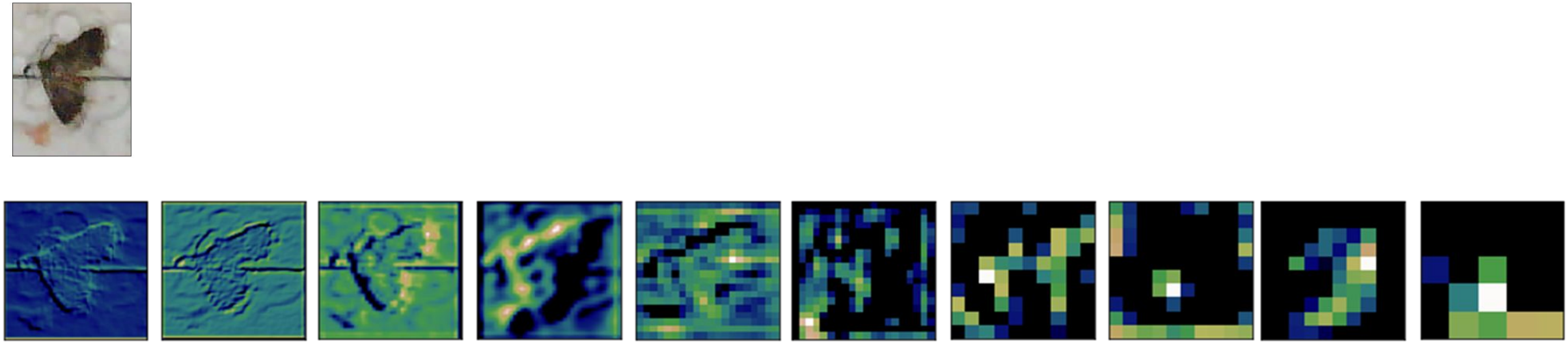


Image from Zeiler, Matthew D. and Rob Fergus. “Visualizing and Understanding Convolutional Networks.” ECCV (2014) [6]

Machine learning - CNN

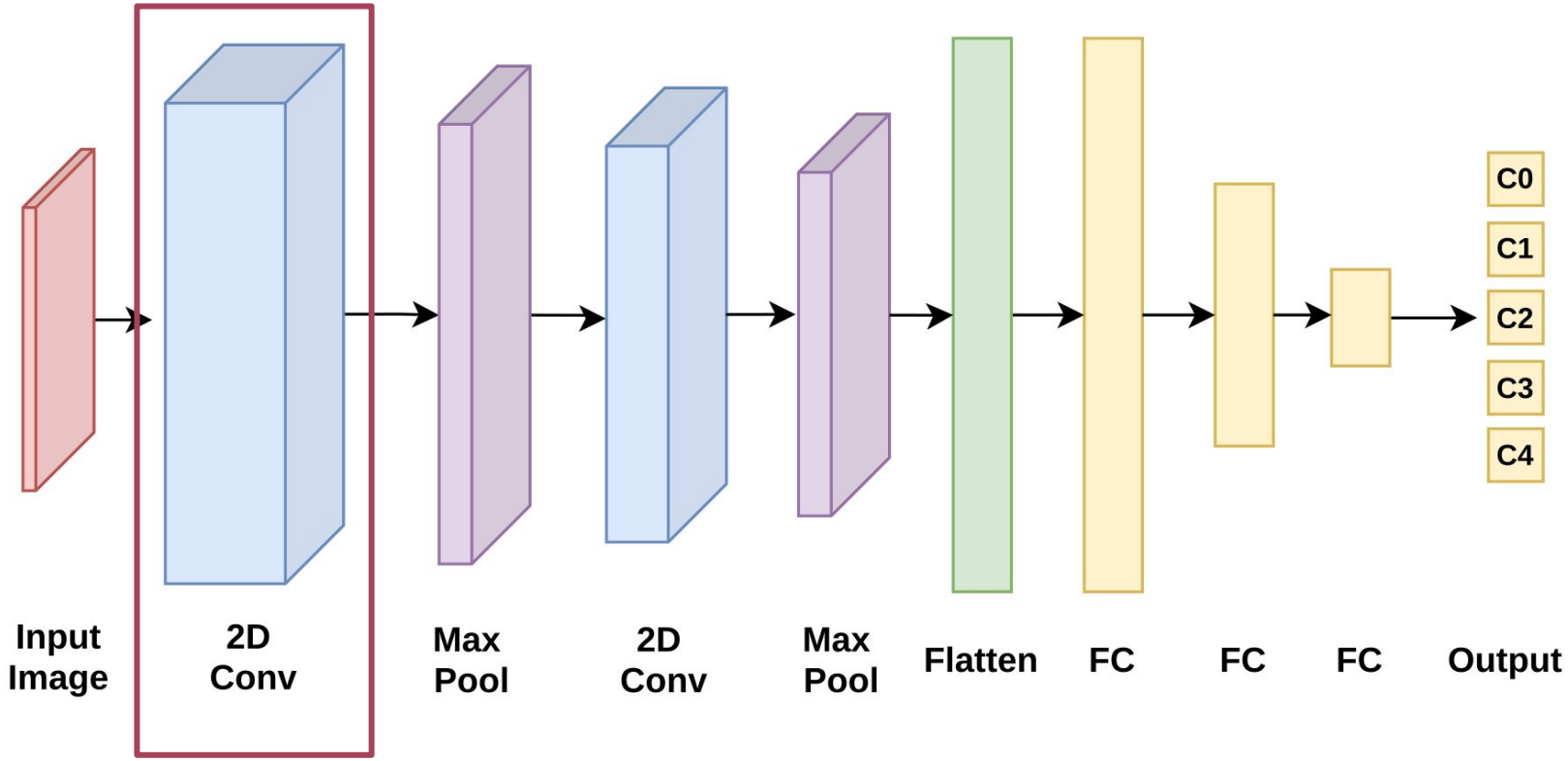
Feature extraction



Machine learning

Fundamental layers in CNN architecture

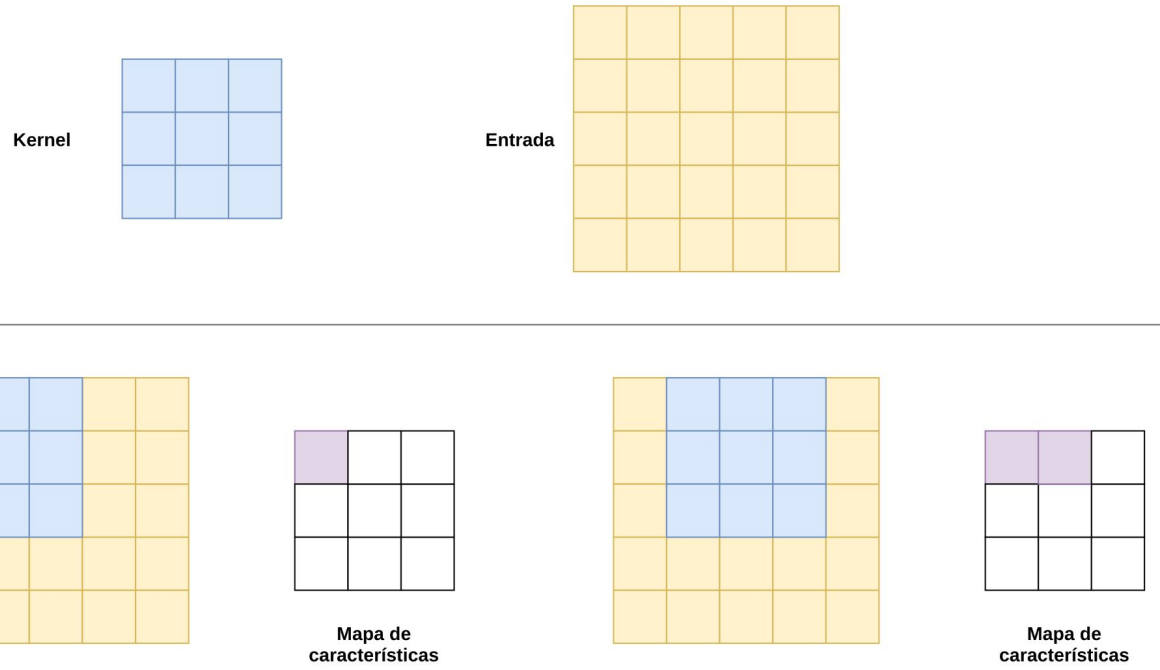
- **Convolution (1D - 2D)**
- Pooling
- Flatten
- Fully connected
- Dropout
- Batch Normalization
- Reshape

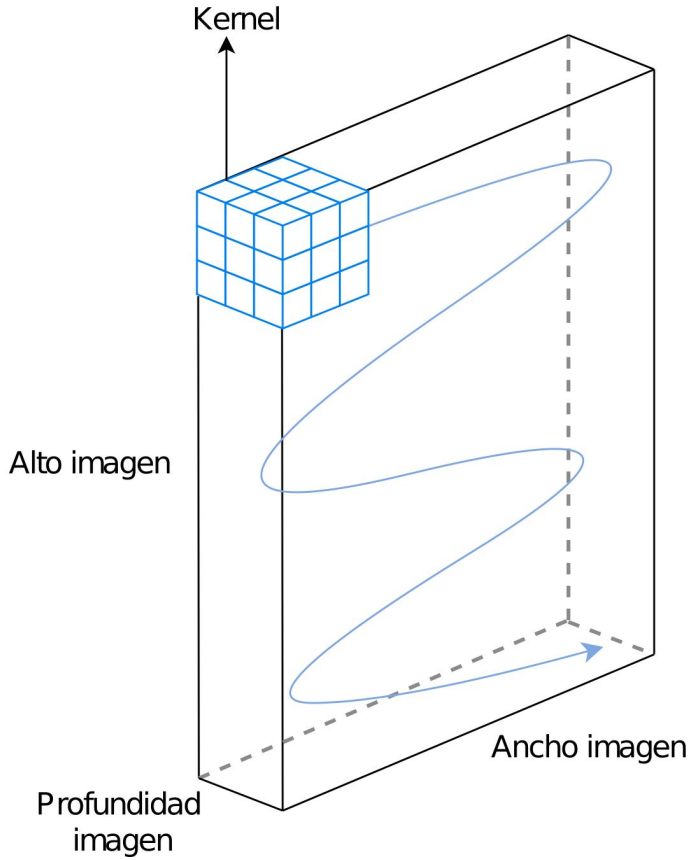


Machine learning

Fundamental layers in CNN architecture

- Convolution (1D - 2D)

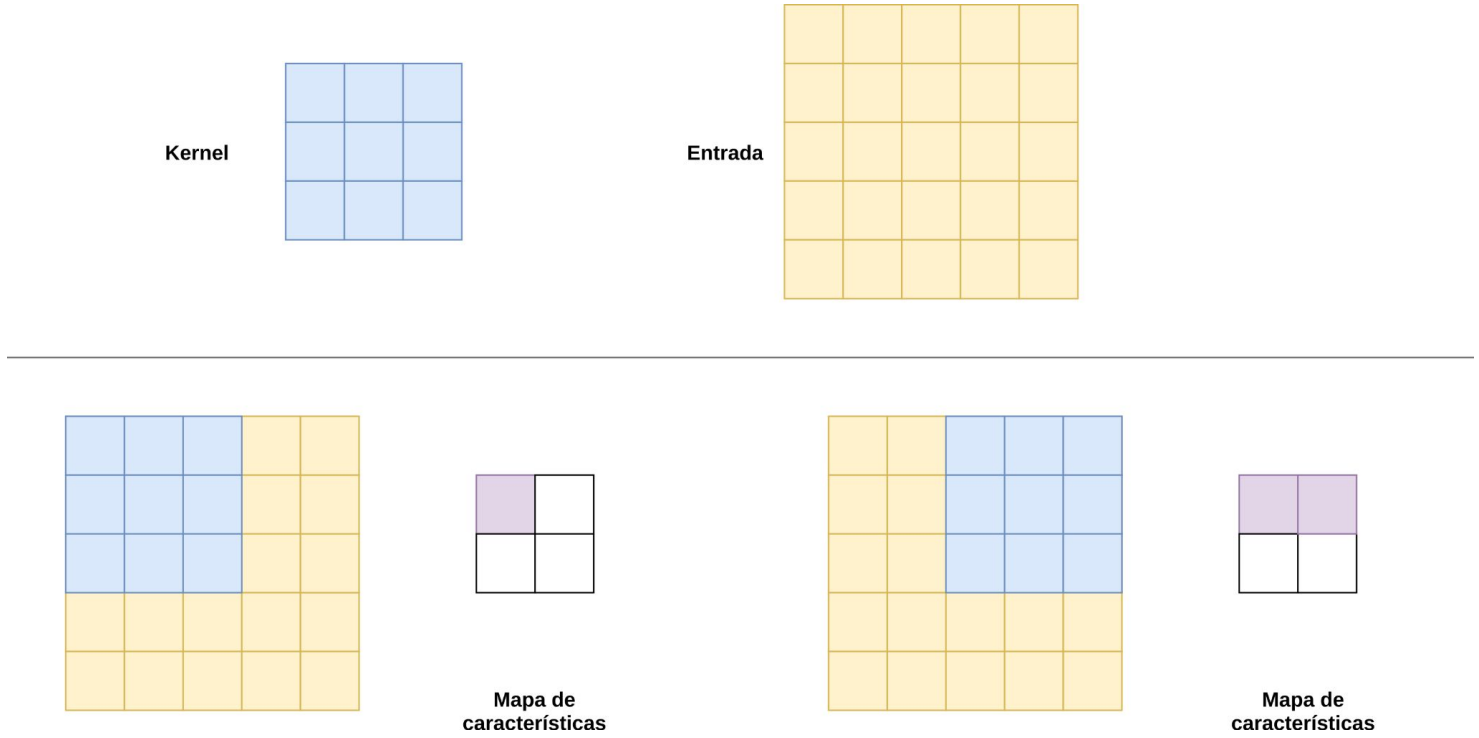




Machine learning

Fundamental layers in CNN architecture

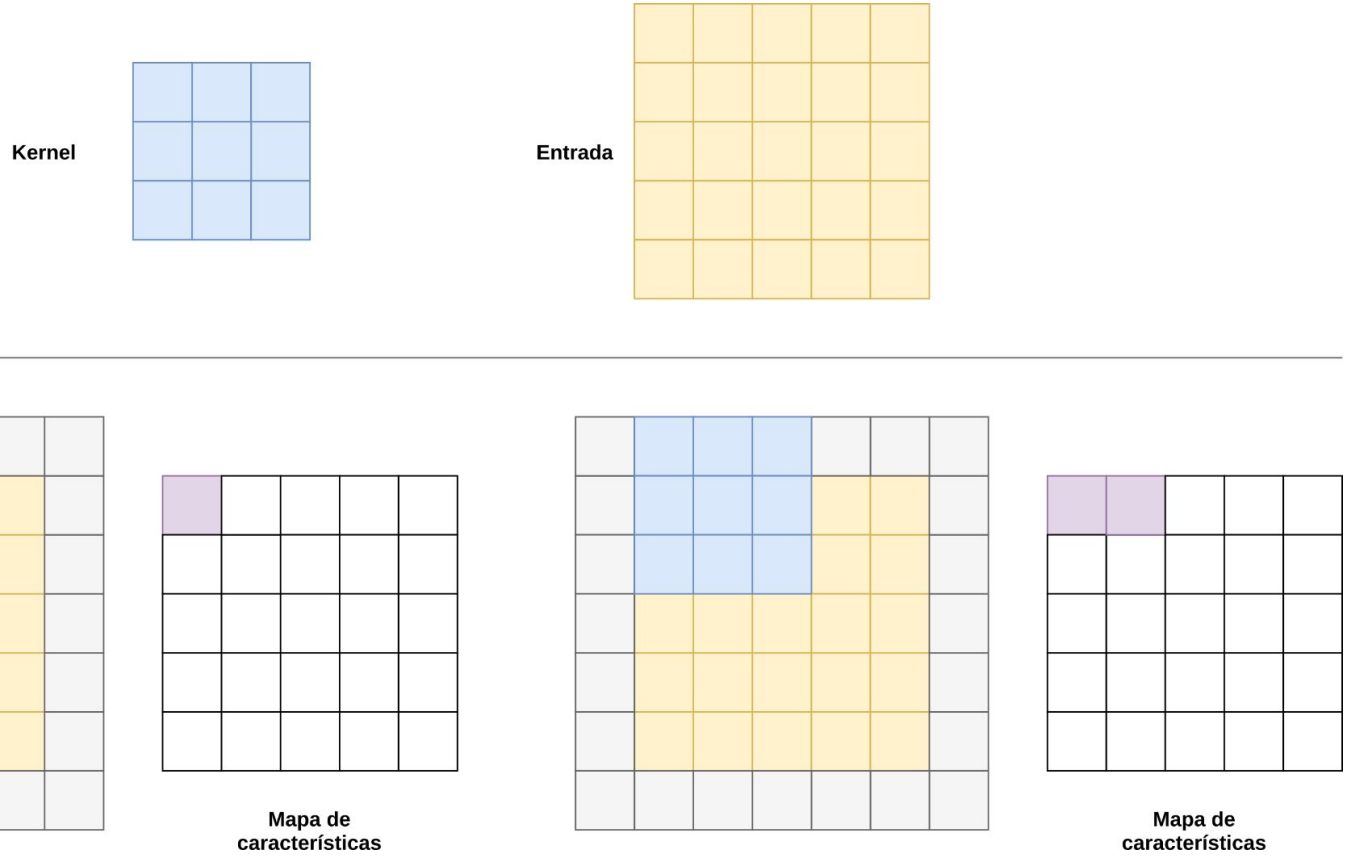
- Convolution (1D - 2D)
  - Padding none



Machine learning

Fundamental layers in CNN architecture

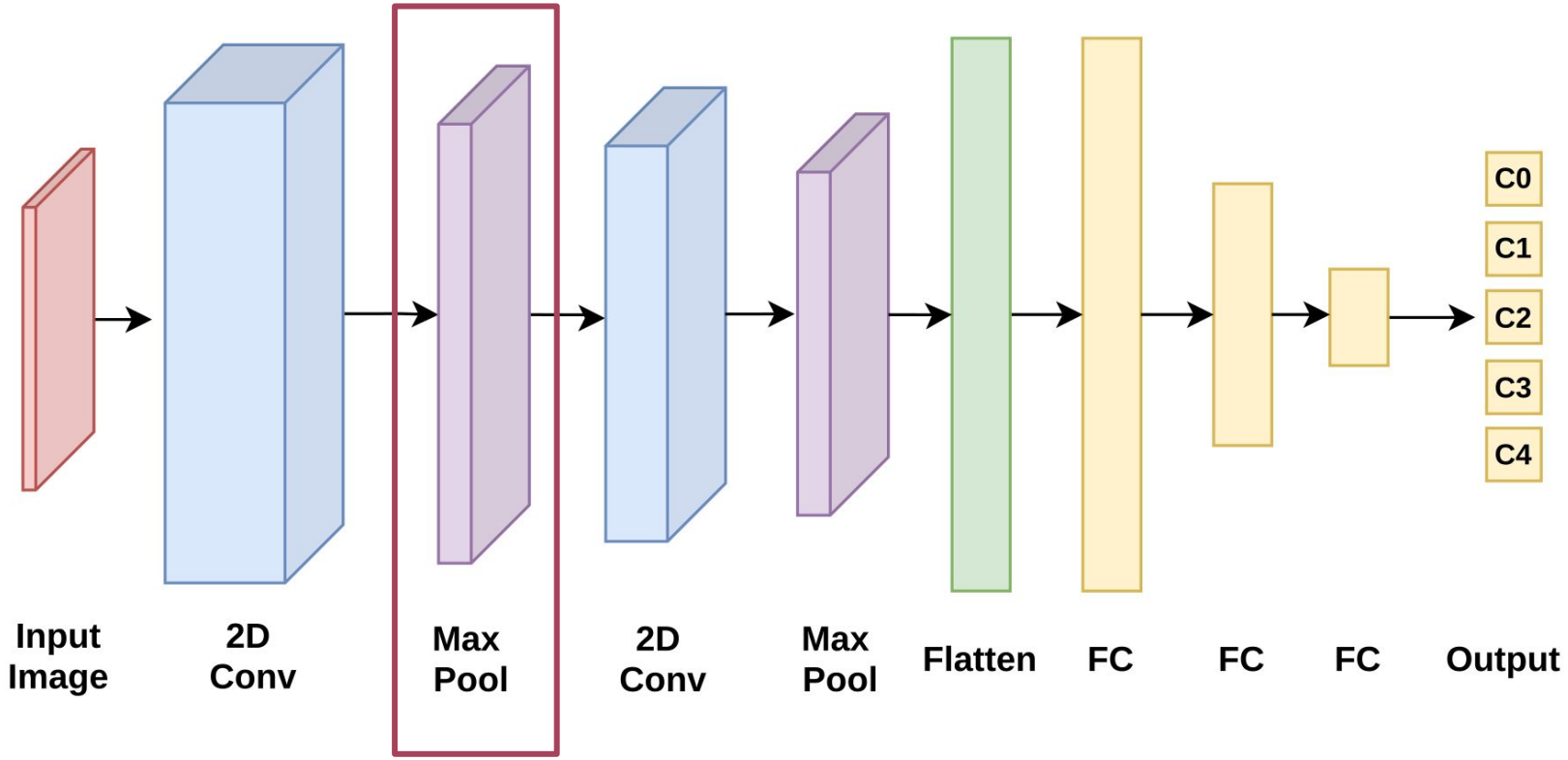
- Convolution (1D - 2D)
  - Padding same



Machine learning

Fundamental layers in CNN architecture

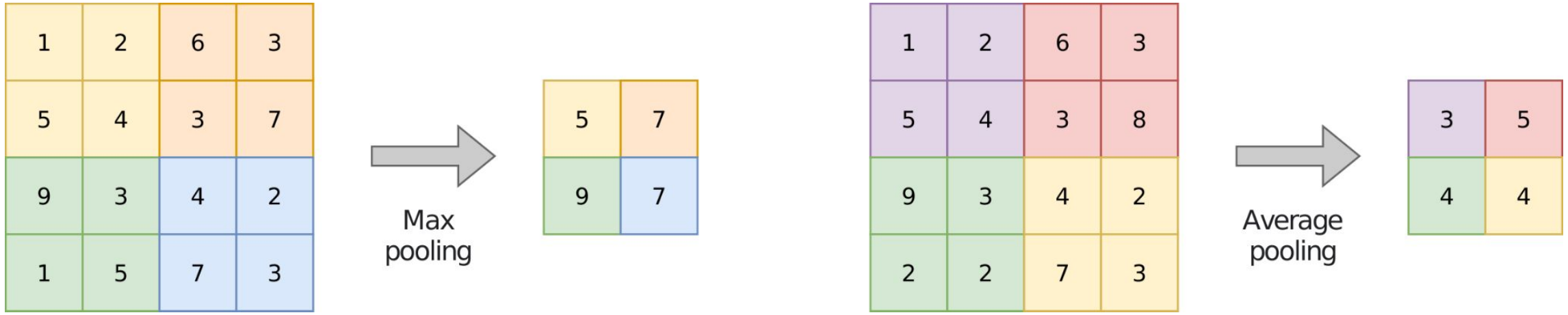
- Convolution (1D - 2D)
- **Pooling**
- Flatten
- Fully connected
- Dropout
- Batch Normalization
- Reshape



Machine learning

Fundamental layers in CNN architecture

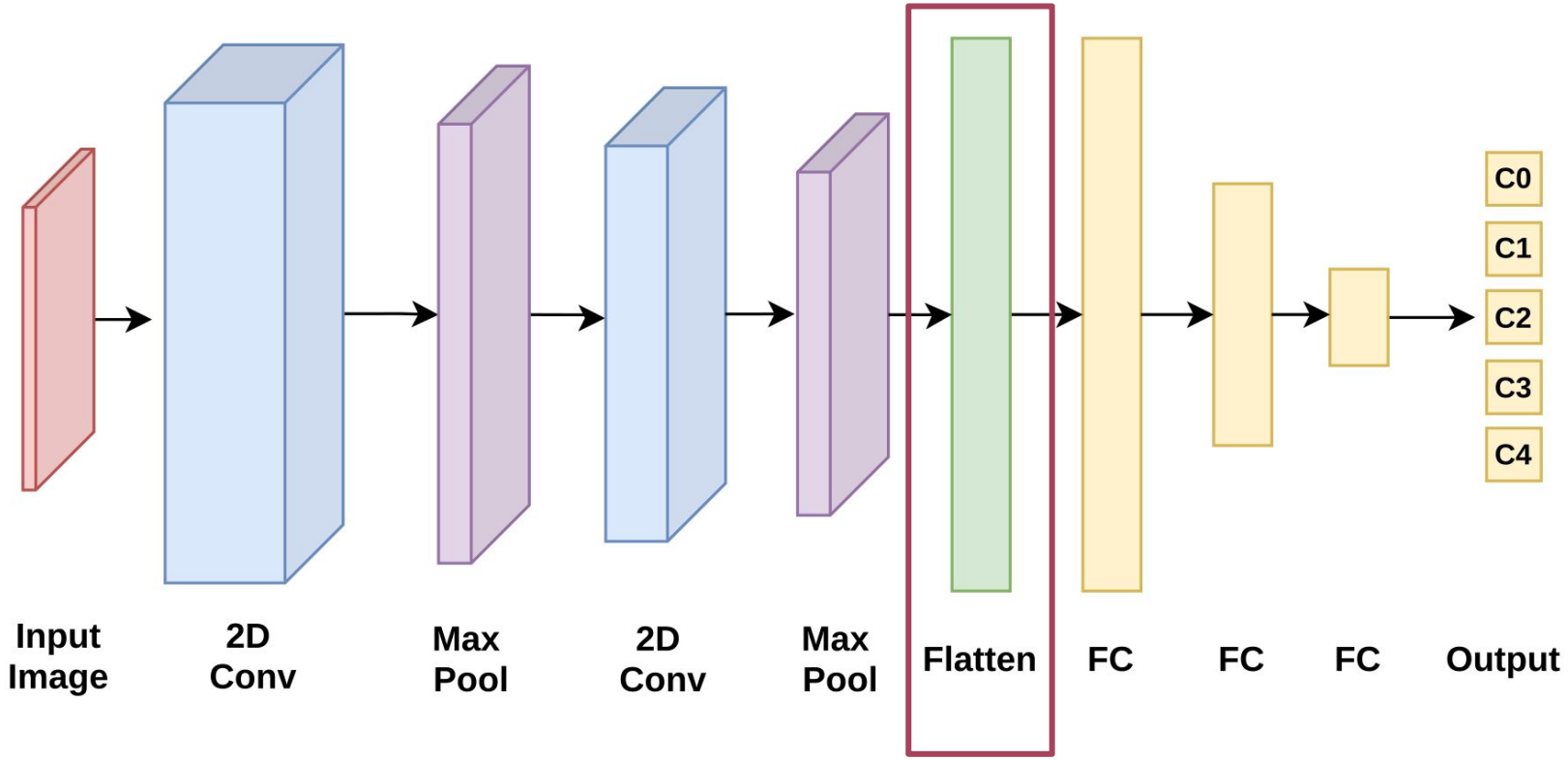
- Pooling (Average and Max)



Machine learning

Fundamental layers in CNN architecture

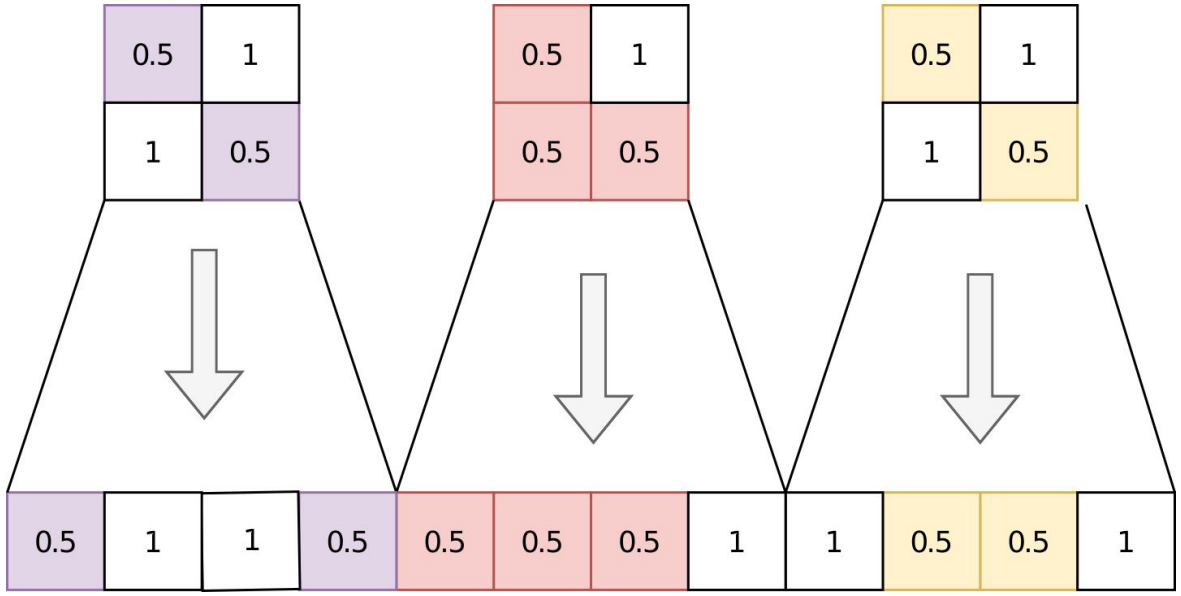
- Convolution (1D - 2D)
- Pooling
- **Flatten**
- Fully connected
- Dropout
- Batch Normalization
- Reshape



Machine learning

Fundamental layers in CNN architecture

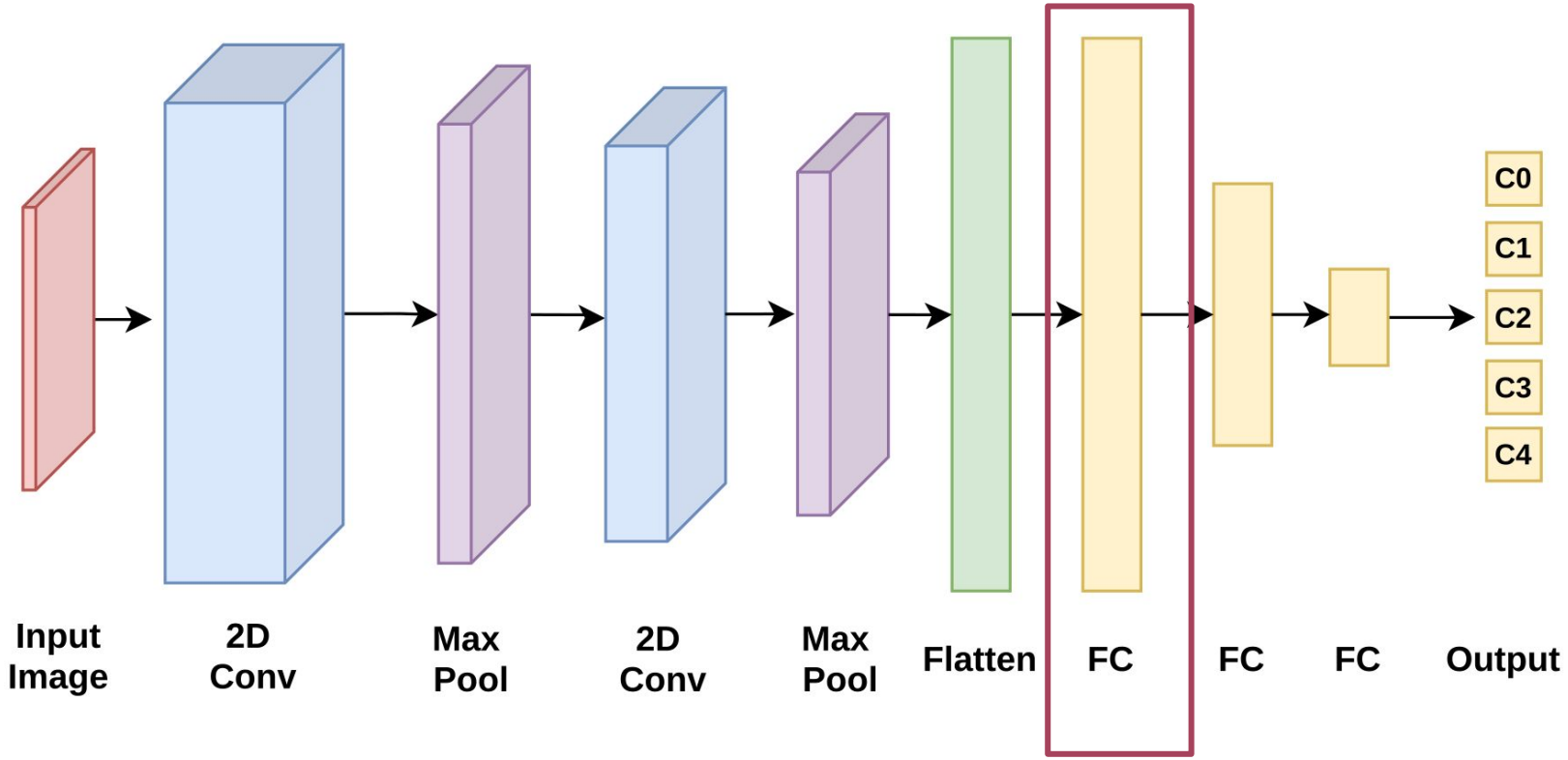
- Flatten



Machine learning

Fundamental layers in CNN architecture

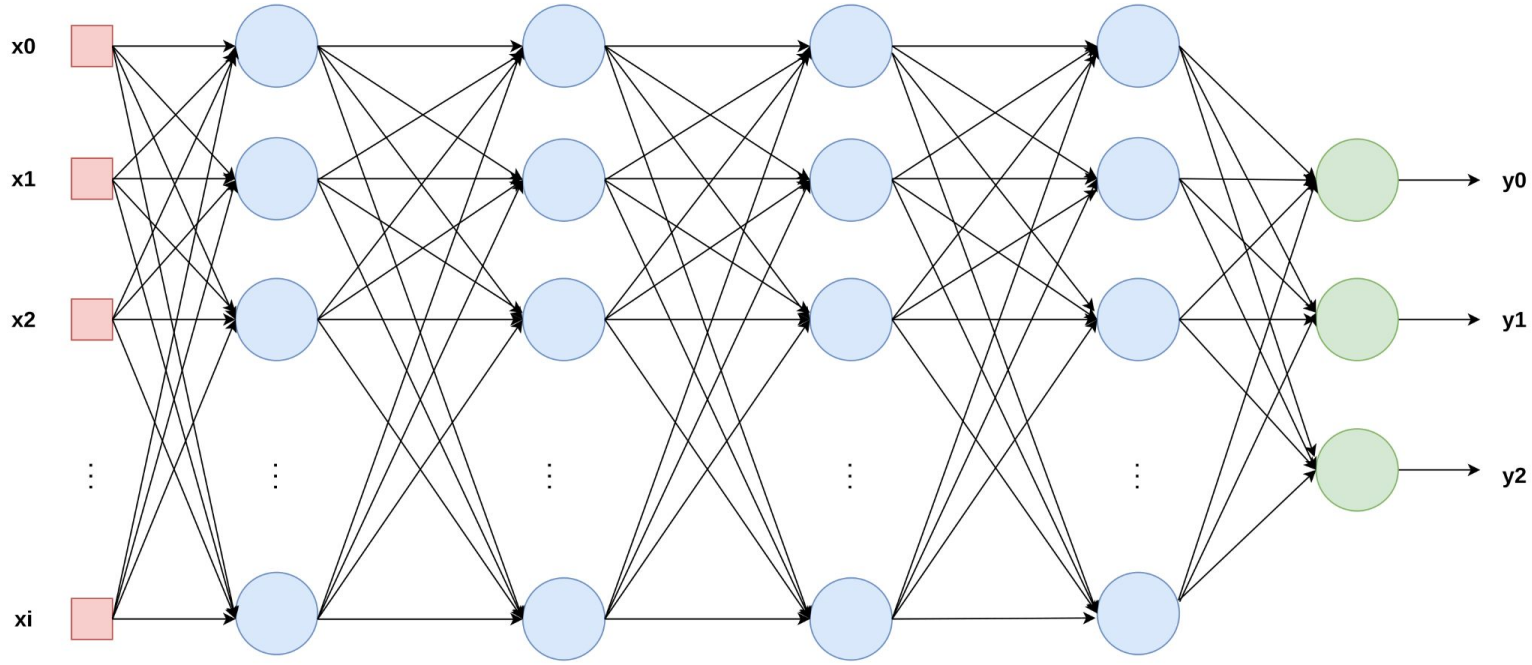
- Convolution (1D - 2D)
- Pooling
- Flatten
- **Fully connected**
- Dropout
- Batch Normalization
- Reshape



Machine learning

Fundamental layers in CNN architecture

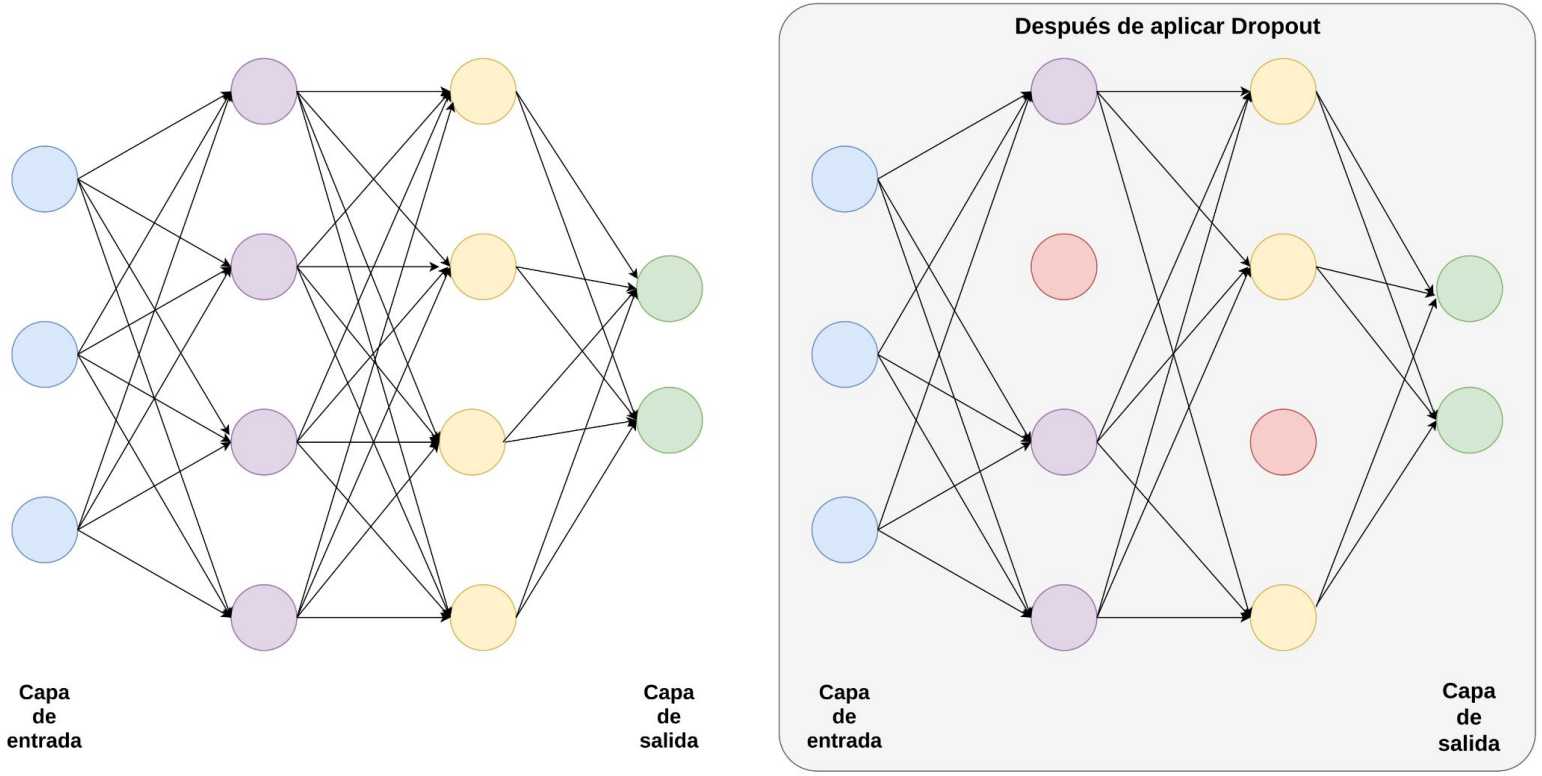
- Fully connected



Machine learning

Fundamental layers in CNN architecture

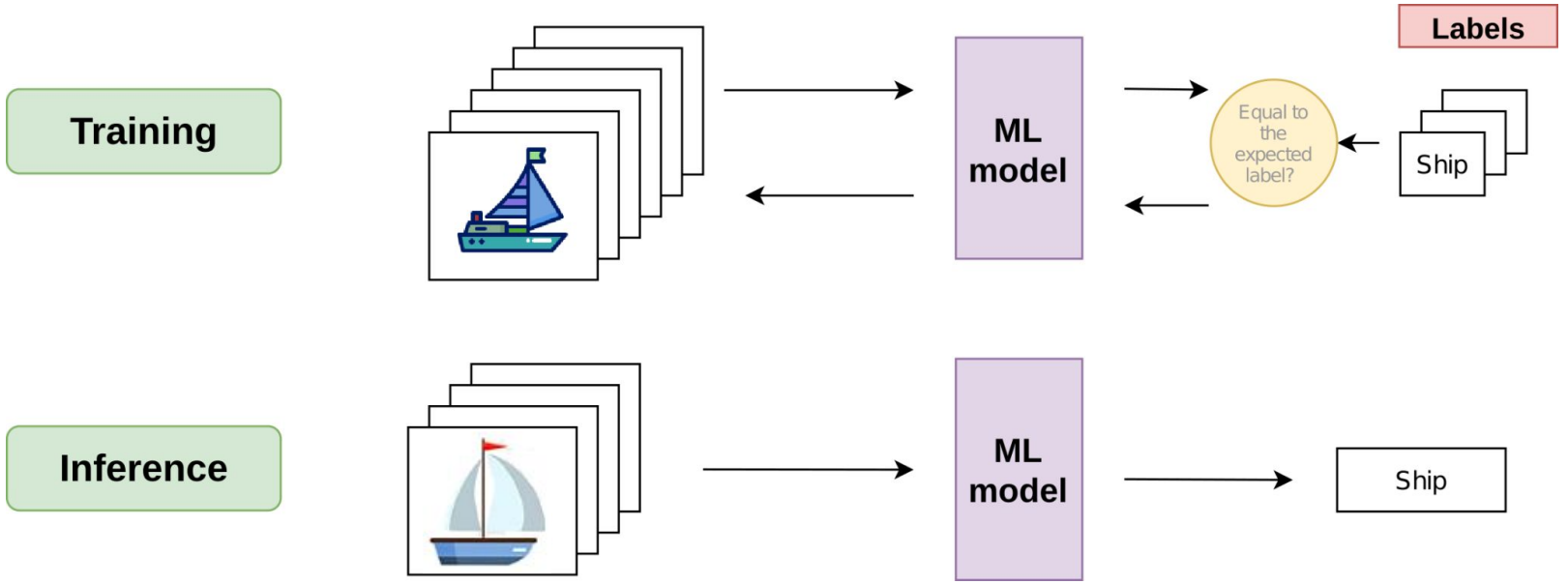
- Dropout



Machine learning: Training and inference

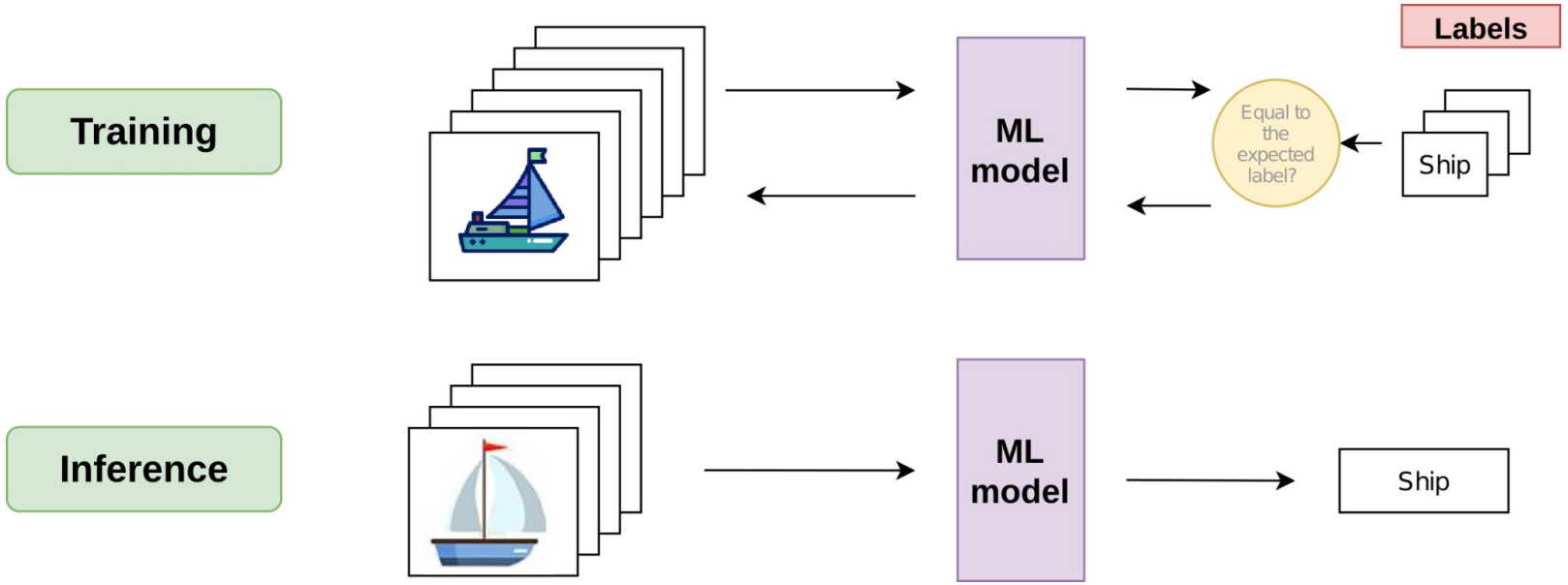
Machine learning

Training and inference



Machine learning

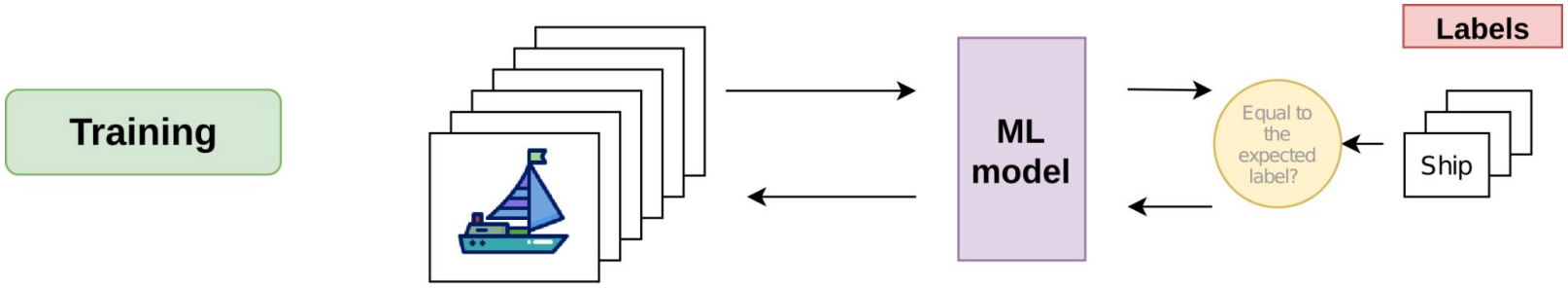
Training and inference



- In a classifier, **the input is assigned to a specific class**.
- **Supervised training:** The network compares the predicted output with the expected output. The difference between them is minimized through backpropagation.

Machine learning

Training

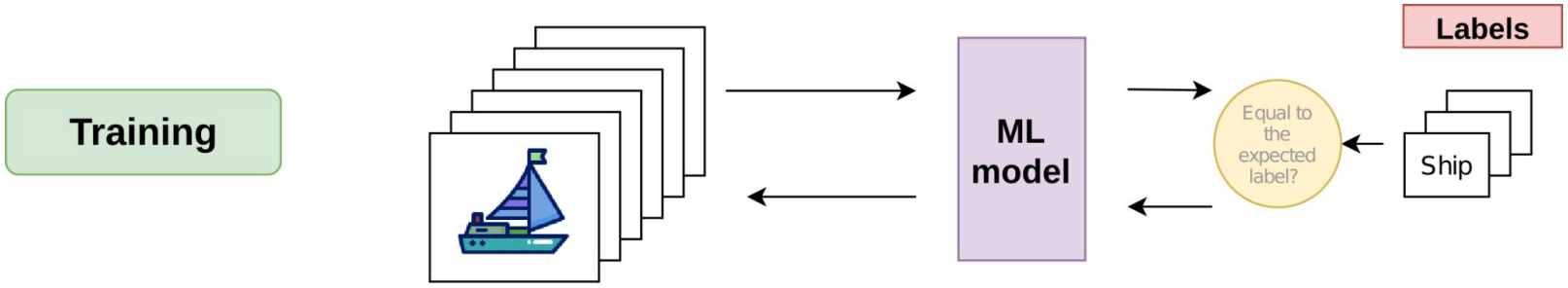


The **training phase** involves adjusting the weights and connections between nodes, allowing the neural network to learn.

One widely used method is the **backpropagation algorithm**, which is based on gradient computation.

Machine learning

Training

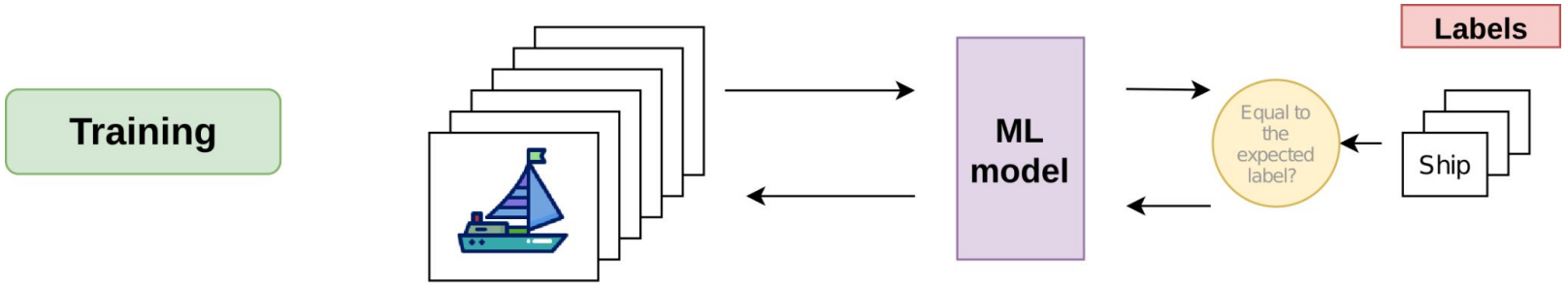


The **backpropagation algorithm** consists of two phases:

- **Forward pass:** Inputs flow through the network to produce classification outputs.

Machine learning

Training



The **backpropagation algorithm** consists of two phases:

- **Forward pass:** Inputs flow through the network to produce classification outputs.
- **Backward pass:** The gradient of the loss function is computed and iteratively applied to adjust the network's weights.

Machine learning

Visualization: neural network training
https://mlu-explain.github.io/neural-networks/

Machine learning

K-fold cross validation

- A method for evaluating a machine learning model by dividing the dataset into multiple subsets, or **folds**.
- This approach helps improve the model's ability to generalize to new data by minimizing the effects of data variability.

Machine learning

K-fold cross validation

- How it works?

The dataset is divided into K equal-sized folds.

Machine learning

K-fold cross validation

- How it works?

The dataset is divided into K equal-sized folds.

The model is trained K times, each time using K-1 folds for training and 1-fold for validation.

Machine learning

K-fold cross validation

- How it works?

The dataset is divided into K equal-sized folds.

The model is trained K times, each time using K-1 folds for training and 1-fold for validation.

Each fold serves as the validation set exactly once, while the remaining folds are used for training.

Machine learning

K-fold cross validation

- **How it works?**

The dataset is divided into K equal-sized folds.

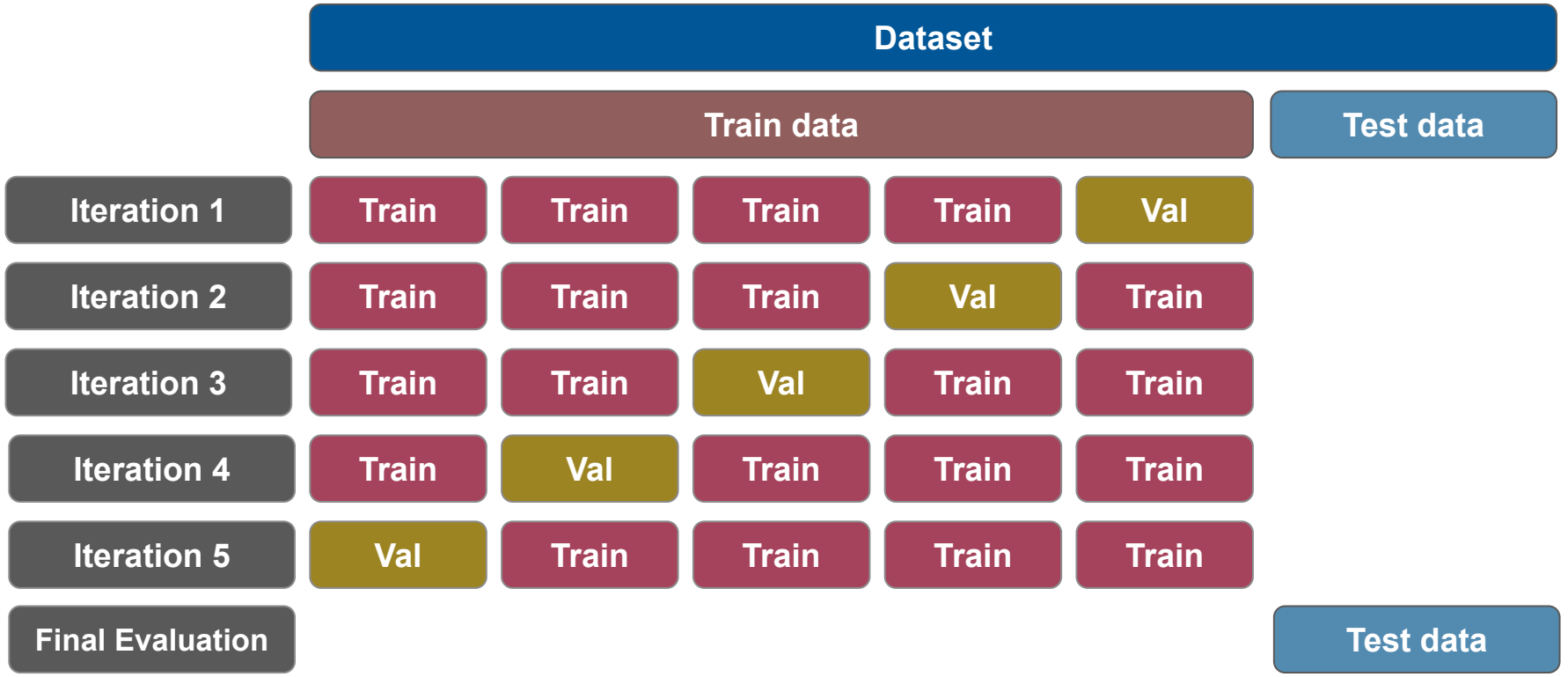
The model is trained K times, each time using K-1 folds for training and 1-fold for validation.

Each fold serves as the validation set exactly once, while the remaining folds are used for training.

The final performance of the model is computed by averaging the results across all K iterations.

Machine learning

K-fold cross validation - Graphical representation



Machine learning

Underfitting, Overfitting, and Optimal

Image from
Togootogtokh, E., & Amartuvshin, A. (2018). Deep Learning Approach for Very Similar Objects Recognition Application on Chihuahua and Muffin Problem. *ArXiv, abs/1801.09573*.

Machine learning

Underfitting, Overfitting, and Optimal

Underfitting: This typically happens when the model is not complex enough or doesn't learn enough from the training data.

Machine learning

Underfitting, Overfitting, and Optimal

Underfitting: This typically happens when the model is not complex enough or doesn't learn enough from the training data.

- The model performs poorly on both the training set and the test set.

Machine learning

Underfitting, Overfitting, and Optimal

Underfitting: This typically happens when the model is not complex enough or doesn't learn enough from the training data.

- The model performs poorly on both the training set and the test set.

Overfitting: The model essentially memorizes the training data, which reduces its ability to generalize to new, unseen data.

Machine learning

Underfitting, Overfitting, and Optimal

Underfitting: This typically happens when the model is not complex enough or doesn't learn enough from the training data.

- The model performs poorly on both the training set and the test set.

Overfitting: The model essentially memorizes the training data, which reduces its ability to generalize to new, unseen data.

- The model performs well on the training set but poorly on the test set.

Machine learning

Underfitting, Overfitting, and Optimal

Underfitting: This typically happens when the model is not complex enough or doesn't learn enough from the training data.

- The model performs poorly on both the training set and the test set.

Overfitting: The model essentially memorizes the training data, which reduces its ability to generalize to new, unseen data.

- The model performs well on the training set but poorly on the test set.

Optimal: The model finds the right balance between underfitting and overfitting. It captures the underlying patterns in the data without memorizing noise or irrelevant details.

Machine learning

Underfitting, Overfitting, and Optimal

Underfitting: This typically happens when the model is not complex enough or doesn't learn enough from the training data.

- The model performs poorly on both the training set and the test set.

Overfitting: The model essentially memorizes the training data, which reduces its ability to generalize to new, unseen data.

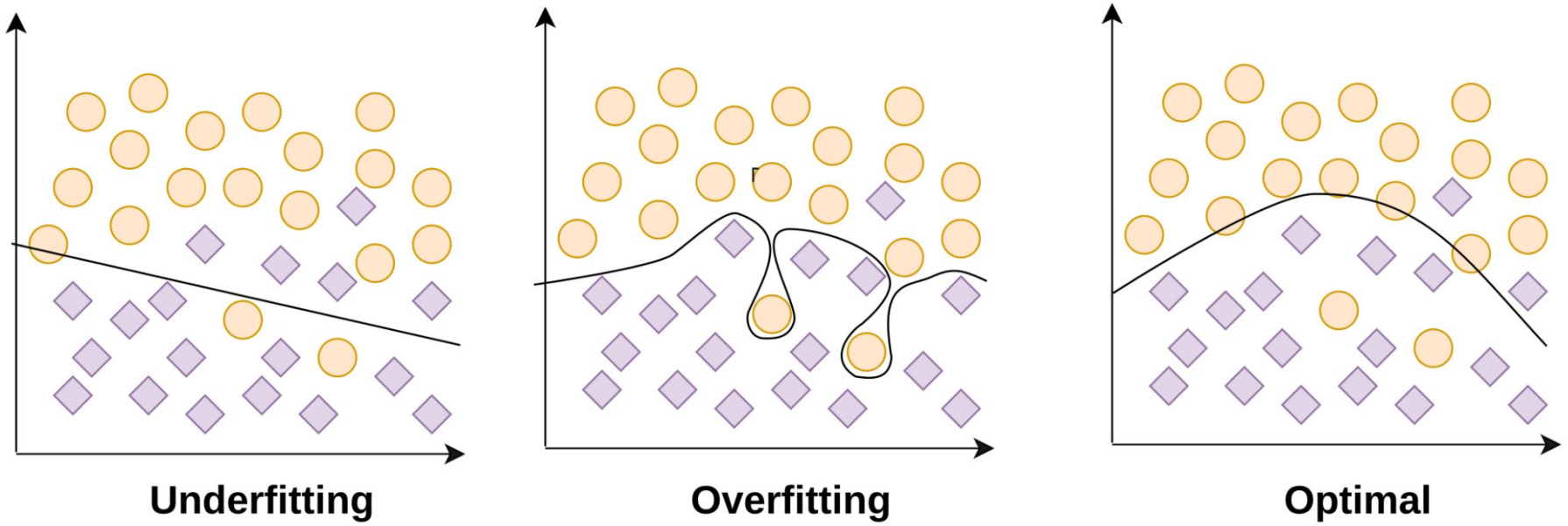
- The model performs well on the training set but poorly on the test set.

Optimal: The model finds the right balance between underfitting and overfitting. It captures the underlying patterns in the data without memorizing noise or irrelevant details.

- It performs well on both the training set and the test set.

Machine learning

Underfitting, Overfitting, and Optimal



Machine learning

Inference

Once the learning phase has been completed, the network can be used to perform the task it was trained for, a process known as **inference**.

Basic ingredients

Machine learning

Basic ingredients

Four basic ingredients

- **Dataset**
- **Loss function**
- **Model**
- **Optimizer**

Machine learning

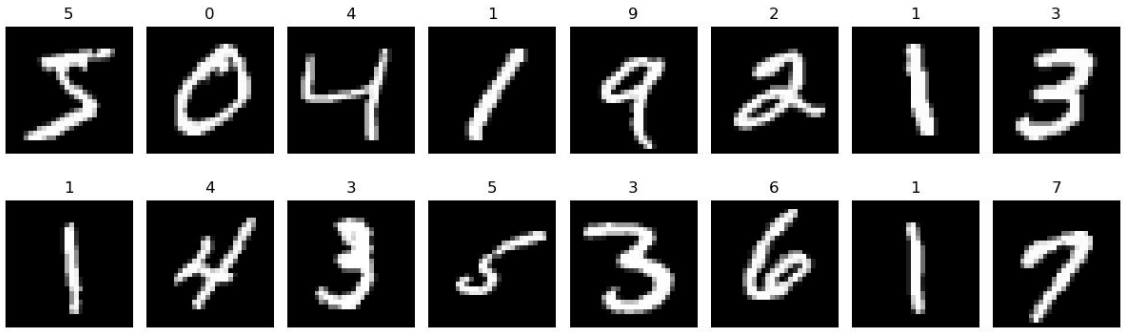
Basic ingredients

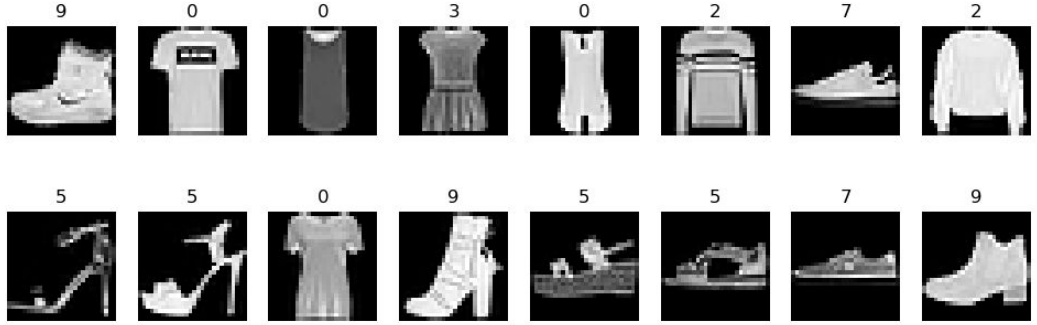
Four basic ingredients

- **Dataset**
- **Loss function**
- **Model**
- **Optimizer**

Machine learning

Dataset





Machine learning

Basic ingredients

Four basic ingredients

- **Dataset**
- **Loss function**
- **Model**
- **Optimizer**

Machine learning

Loss-Function

- A loss function quantifies the difference between a model's predictions and the actual target values.
- In machine learning, it serves as a key metric to assess performance.
- **The goal of training is to minimize this loss, enhancing the model's accuracy.**

Machine learning

Loss-Function

- A loss function quantifies the difference between a model's predictions and the actual target values.
- In machine learning, it serves as a key metric to assess performance.
- **The goal of training is to minimize this loss, enhancing the model's accuracy.**

For classification tasks:

- Number of classes > 2: **categorical cross-entropy**
- Number of classes = 2: **binary cross-entropy**

Machine learning

Basic ingredients

Four basic ingredients

- **Dataset**
- **Loss function**
- **Model**
- **Optimizer**

Machine learning

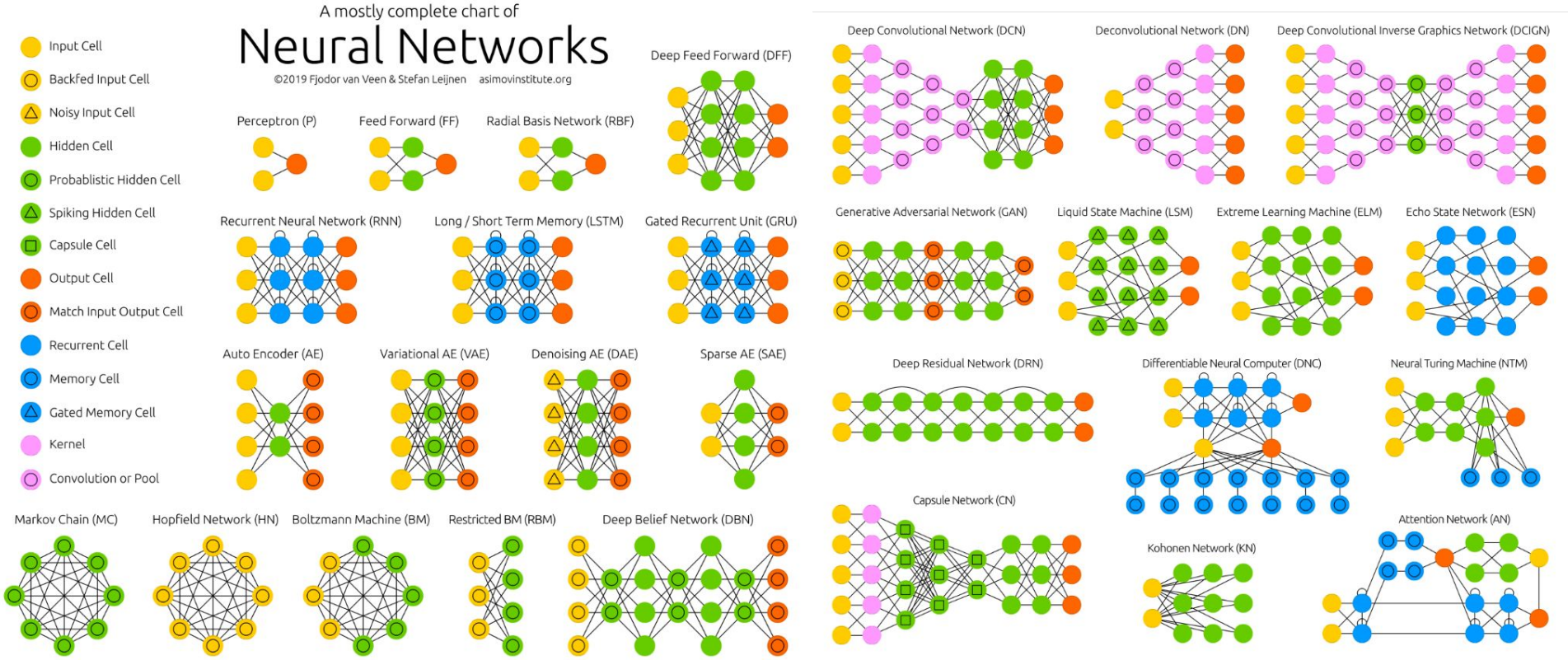
Topology and Model

- **Topology** structure or architecture of the model, which determines how the components (e.g., neurons in a neural network) are connected.

- The **model** is the final output of the learning process after training.
  - Algorithms or equations that map input data to output predictions.

Machine learning

Topology



Fjodor van Veen from Asimov institute compiled a cheatsheet on Neural Networks topologies. | https://www.asimovinstitute.org/author/fjodorvanveen/

Machine learning

Basic ingredients

Four basic ingredients

- **Dataset**
- **Loss function**
- **Model**
- **Optimizer**

Machine learning

Optimizers

- In machine learning, optimizers are algorithms designed to **minimize the loss function**.

Machine learning

Analogy

Objective: Find the fastest and most efficient route to the lowest point.

Machine learning

Optimizers

- In machine learning, optimizers are algorithms designed to **minimize the loss function**.
  - *Why? What do you think?*

Machine learning

Optimizers

- In machine learning, optimizers are algorithms designed to **minimize the loss function**.
  - *Why? What do you think?*

- They achieve this by adjusting the model's parameters (weights and biases) during training.

Machine learning

Optimizers

- In machine learning, optimizers are algorithms designed to **minimize the loss function**.
  - *Why? What do you think?*
- They achieve this by adjusting the model's parameters (weights and biases) during training.
- The primary goal of an optimizer is to enhance the model's performance. **This is done by reducing the error between the predicted output and the actual target values.**

Machine learning

Optimizers

- **SGD**
  - It uses a fixed learning rate for all parameters.
  - SGD does not estimate any moments of the gradient.

Machine learning

Optimizers

- **SGD**
  - It uses a fixed learning rate for all parameters.
  - SGD does not estimate any moments of the gradient.

- **Adam**
  - It adapts the learning rate to each parameter individually.
  - Adam uses first- and second-order moments to dynamically adjust the learning rate at each iteration.

Machine learning

Optimizers

- **SGD**
  - It uses a fixed learning rate for all parameters.
  - SGD does not estimate any moments of the gradient.

- **Adam**
  - It adapts the learning rate to each parameter individually.
  - Adam uses first- and second-order moments to dynamically adjust the learning rate at each iteration.

- The **gradient is a vector that indicates the direction and magnitude of the fastest change of a function**.

- In Machine Learning and optimization, the gradient of a loss function L with respect to the model parameters tells us how to adjust those parameters to minimize the loss.

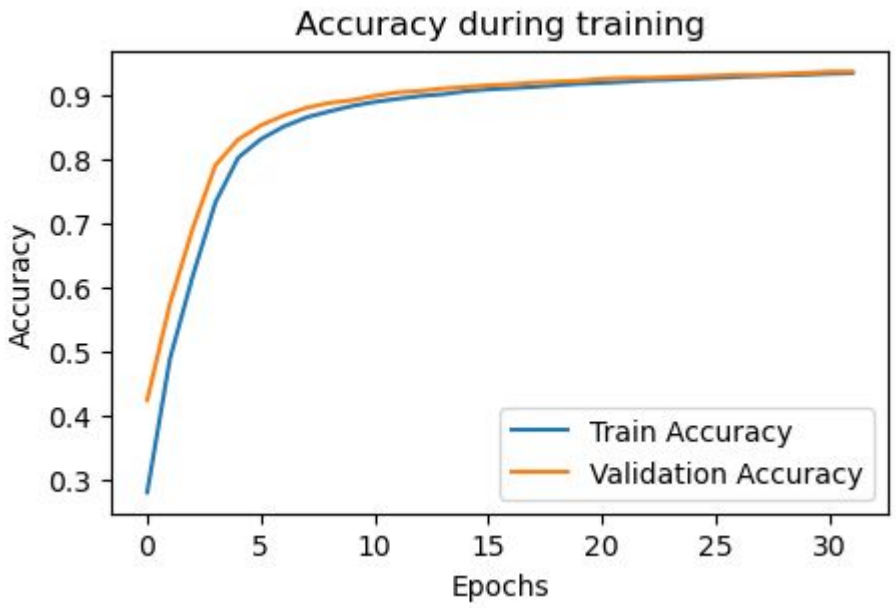
Metrics

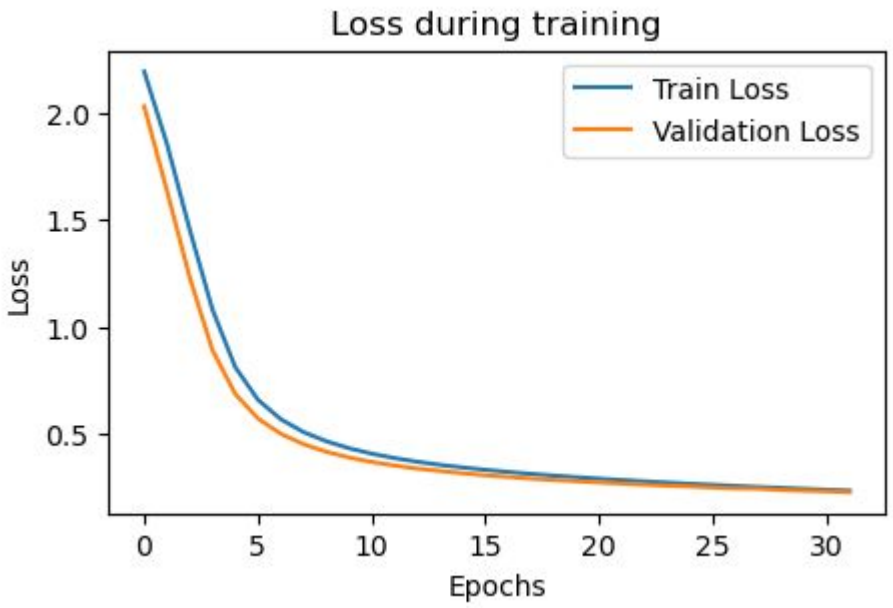
Metrics

- *Why metrics are important?*

Metrics

Accuracy and loss during training





Metrics

Confusion matrix

- **True Positives (TP)**
  - The number of instances where the model correctly predicted the positive class.
- **True Negatives (TN)**
  - The number of instances where the model correctly predicted the negative class.
- **False Positives (FP)**
  - The number of instances where the model incorrectly predicted the positive class.
- **False Negatives (FN)**
  - The number of instances where the model incorrectly predicted the negative class.

Metrics

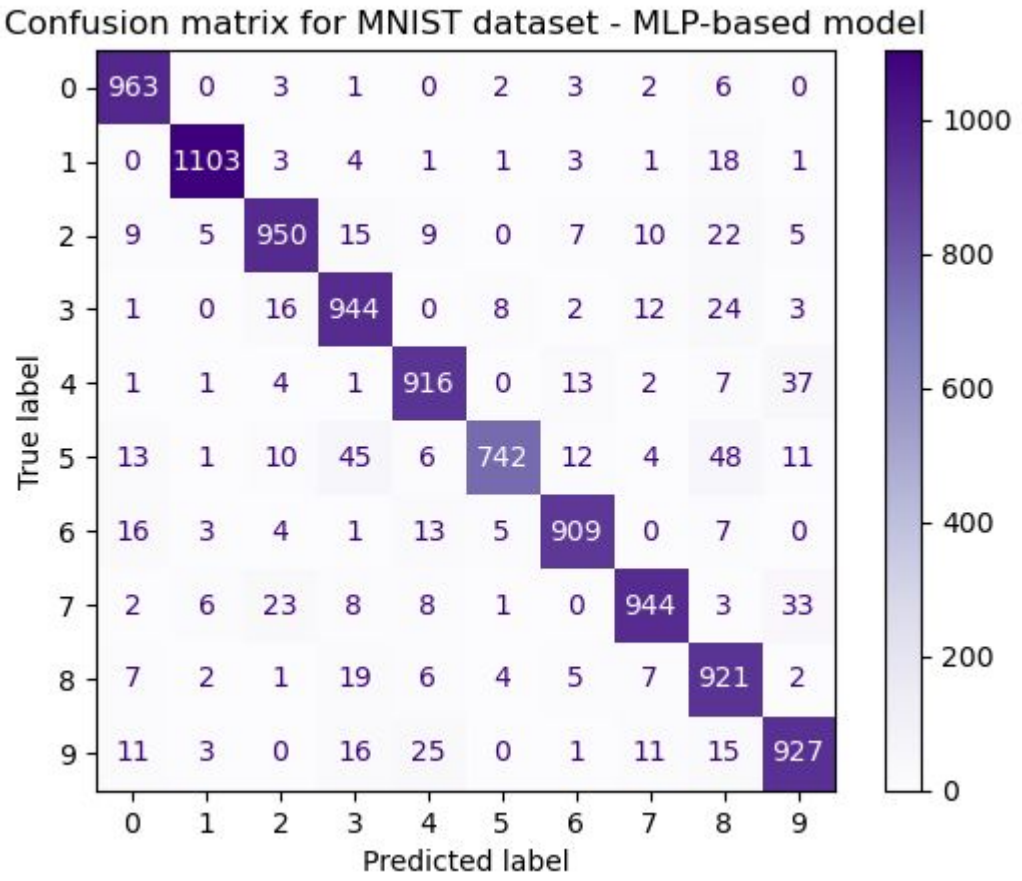
Confusion matrix

	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Metrics

Confusion matrix

	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative



Metrics

Receiver operating characteristic (ROC) curve

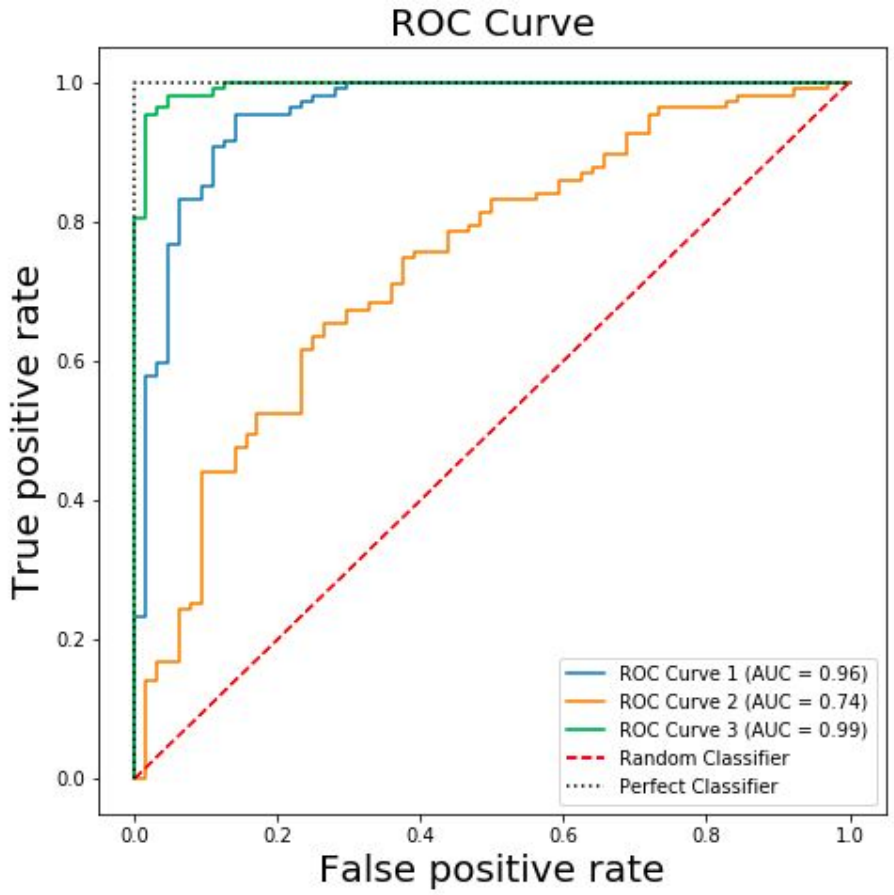
- It illustrates the trade-off between the **true positive rate (TPR)** and the **false positive rate (FPR)** at various threshold settings.

  TPR = TP/(TP+FN) | **FPR** = FP/(FP+TN)

- The ROC curve helps evaluate how well a model distinguishes between classes.

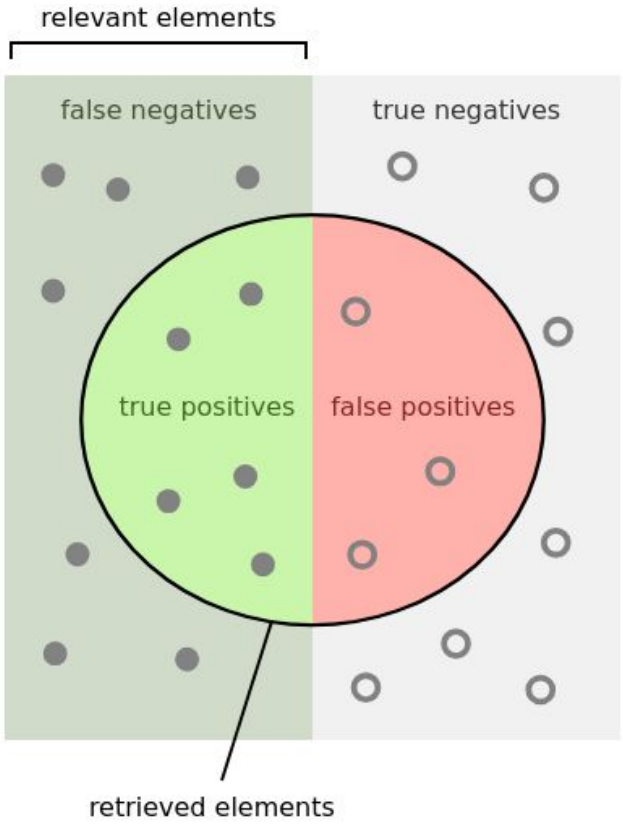
- The ROC curve is always plotted between 0 and 1 on both axes because both TPR and FPR are proportions (ratios) and therefore can only range from 0 to 1.

- **Area Under the Curve (AUC)** measures the overall ability of the model to distinguish between the classes. An AUC score of 1 means perfect classification, while an AUC score of 0.5 means the model performs no better than random guessing.



Metrics

Precision and Recall



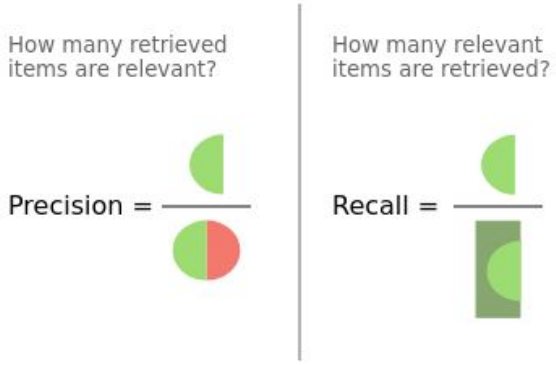
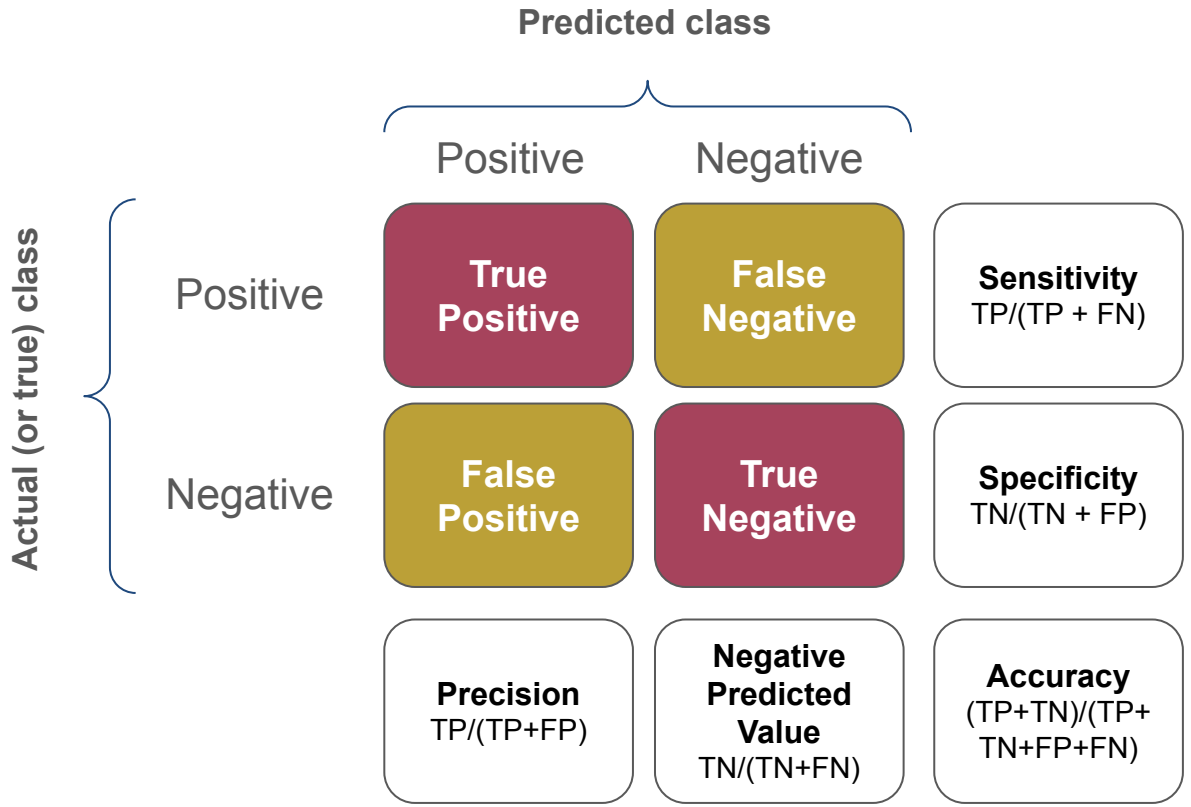


Image from https://en.wikipedia.org/wiki/Precision_and_recall

Metrics

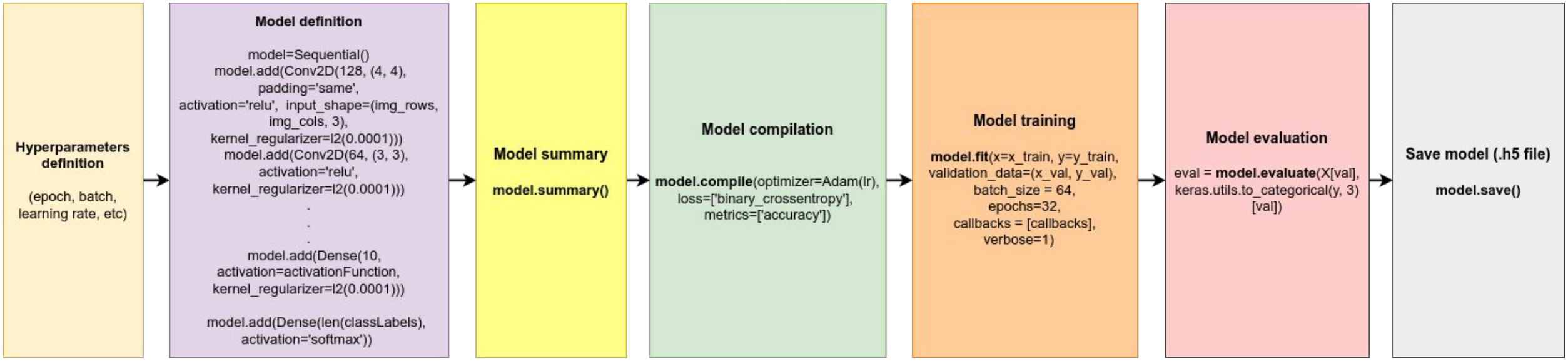
Accuracy and loss during training



General steps Keras+TensorFlow

Machine learning

General steps Keras+TensorFlow



Machine learning

General steps Keras+TensorFlow

General overview

- The first two steps focus on **defining the hyperparameters and configuring the machine learning architecture**. Afterward, a model summary provides an overview of how the model was constructed.

- Once the model is created, parameters such as the optimizer, loss function, and metrics are configured using the **model.compile()** function.

- Finally, training is performed with the **model.fit()** function, where the dataset, batch size, number of epochs, and callbacks, among other settings, are specified.

Machine learning

General steps Keras+TensorFlow

```
model= Sequential([

    Flatten(input_shape=(w, h)),
    Dense(256, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(n_classes, activation='softmax')

])
```

Model definition

```
model.summary()
```

Model summary

Machine learning

General steps Keras+TensorFlow

```
learningRate = 0.001
optimizer = Adam(learningRate)
Epochs = 32
Batch = 16
```

Defining some of the hyperparameters

Machine learning

General steps Keras+TensorFlow

```
model.compile(loss='sparse_categorical_crossentropy', optimizer=op, metrics=['accuracy'])
```

Model compile

Loss: A metric that measures how far the model's predictions are from the actual values.

Optimizer: An algorithm that adjusts the weights of the neural network to minimize the loss function.

Learning Rate: A hyperparameter that controls the size of the adjustments the optimizer makes to the model's weights during each iteration.

Metrics: Additional values monitored during training to evaluate the model's performance. For example, accuracy (used in classification).

Machine learning

General steps Keras+TensorFlow

```
history = model.fit(x_train_norm, y_train, epochs= 32, batch_size = 50, validation_split=0.2)
```

→ Model fit

x_train_norm: normalized dataset obtained by applying a transformation to x_train.

y_train: labels (or expected values) corresponding to the training data.

batch: number of samples processed before updating the model's weights.

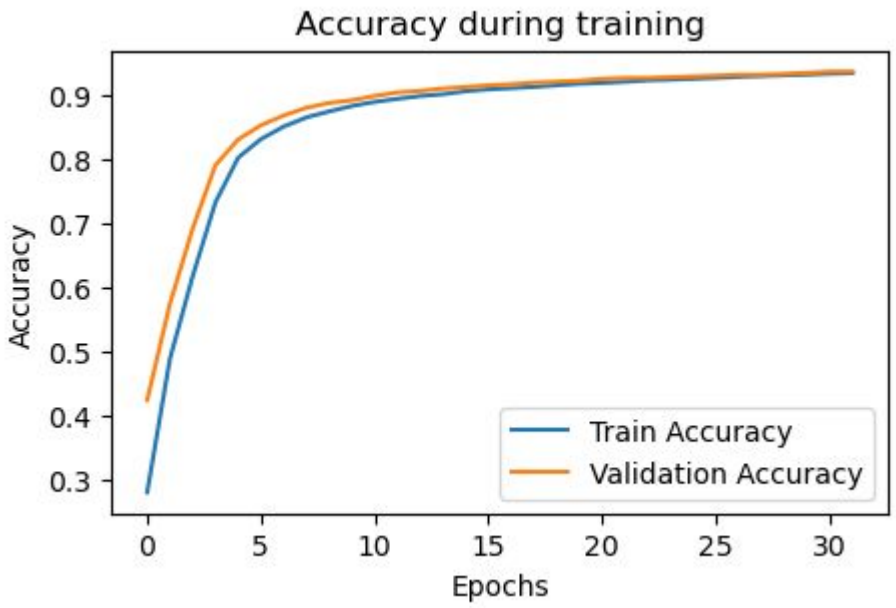
epochs: number of times the model will go through the entire training dataset.

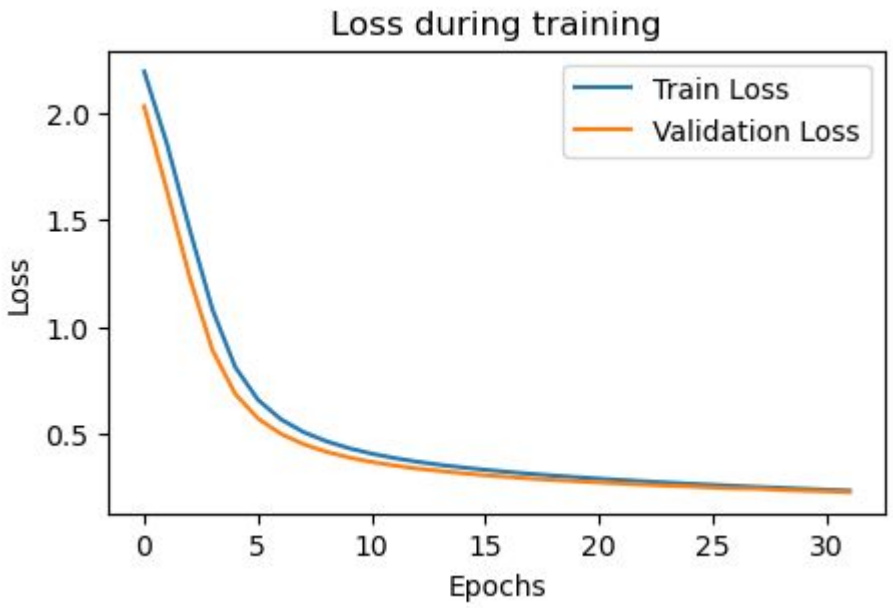
validation_split: percentage of the training dataset (x_train, y_train) reserved for validation.

Machine learning

General steps Keras+TensorFlow

Plot the Accuracy and Loss from the **history** variable during training





Demo:
MLP training for MNIST dataset

Thank you!

Romina Soledad Molina, Ph.D.
MLab-STI, ICTP

Mar del Plata, Argentina - 2025 -