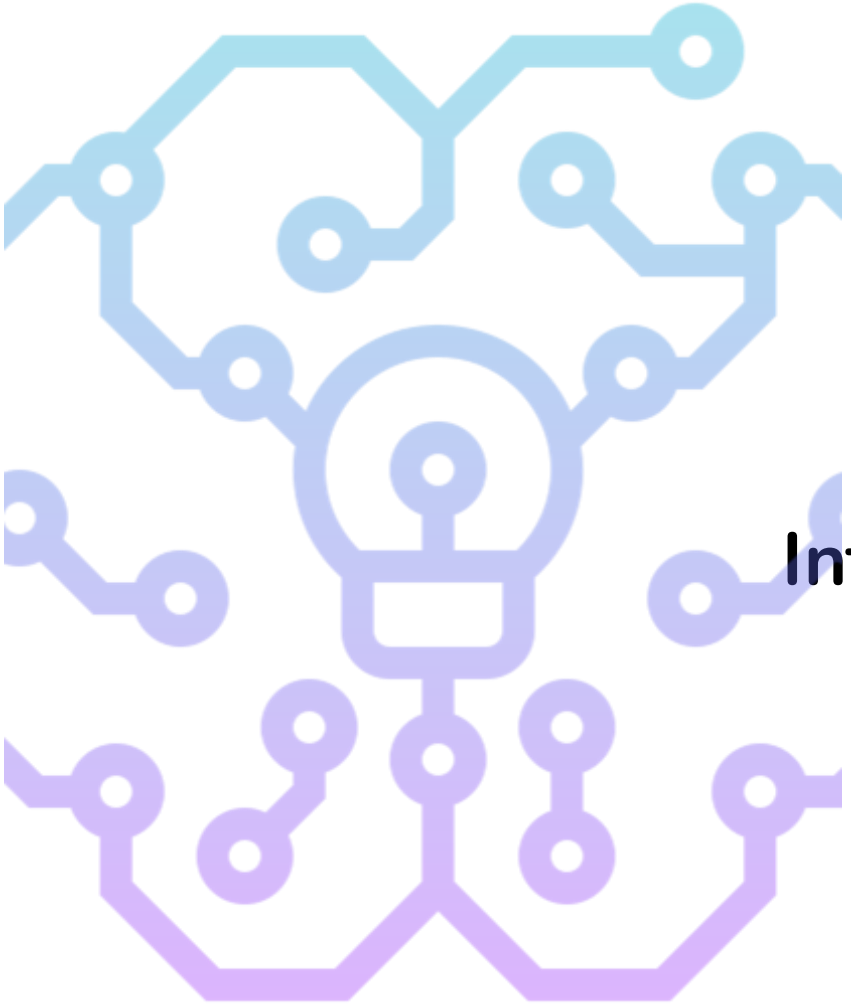


UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

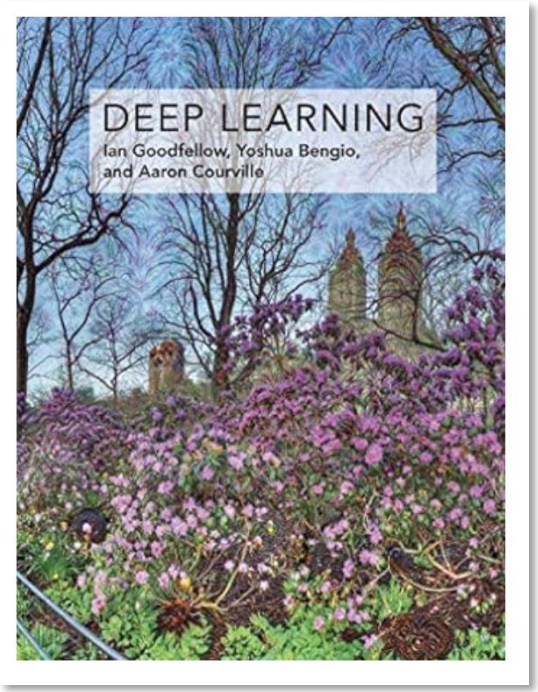
CCTVal
CENTRO CIENTÍFICO
TECNOLÓGICO
DE VALPARAÍSO

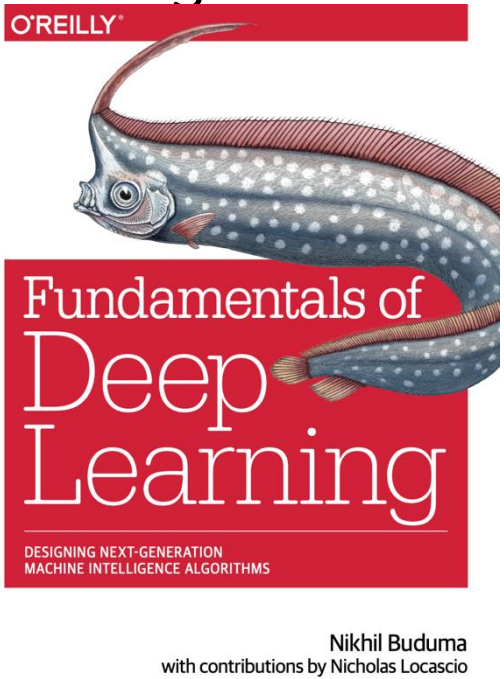
Machine learning basics:
Introduction to Deep Learning

Raquel Pezoa Rivera
raquel.pezoa@usm.cl
Departamento de Informática
Centro Científico Tecnológico de Valparaíso
Universidad Técnica Federico Santa María

Deep Learning

Interesting Books to learn Deep Learning



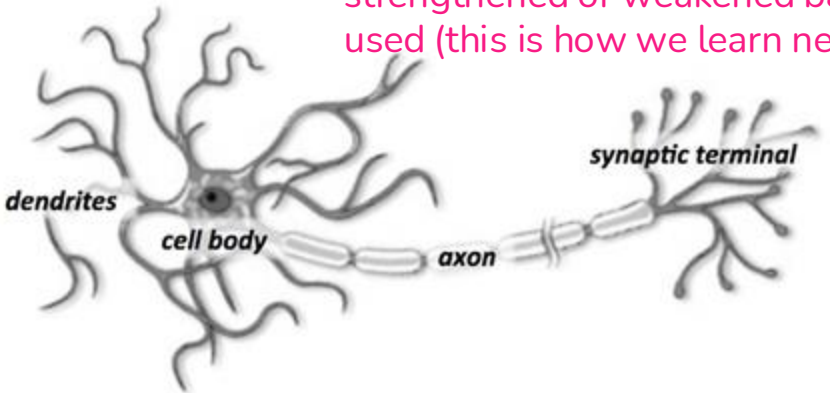


Online versión:
https://www.deeplearningbook.org

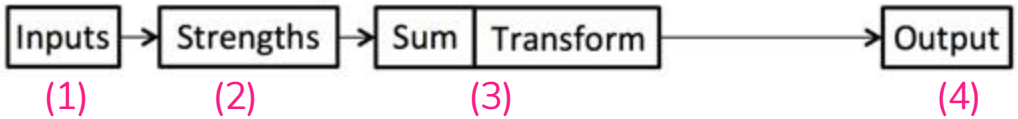
Introduction – Biological Neuron

(2) Each of the input signals are dynamically strengthened or weakened based on how often it is used (this is how we learn new concepts!)

(1) Dendrites receive the input signals



(3) After being weighted by the strength of their respective connections, the inputs are summed together in the cell body.

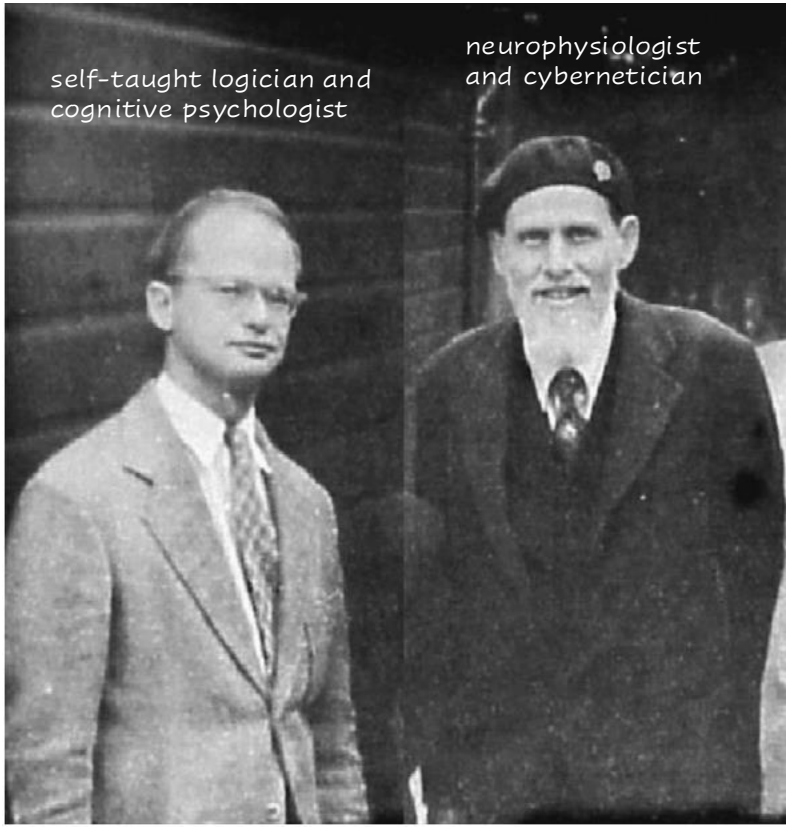


(4) This sum is then transformed into a new signal that's propagated along the cell's axon and sent off to other neurons.

Buduma, N., & Locascio, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.".

Artificial Neuron

- The **artificial neuron** emerges from the behavior of the biological neuron (in a very simplified form).
- The first model of an artificial neuron was proposed by **McCulloch & Pitts** in 1943.

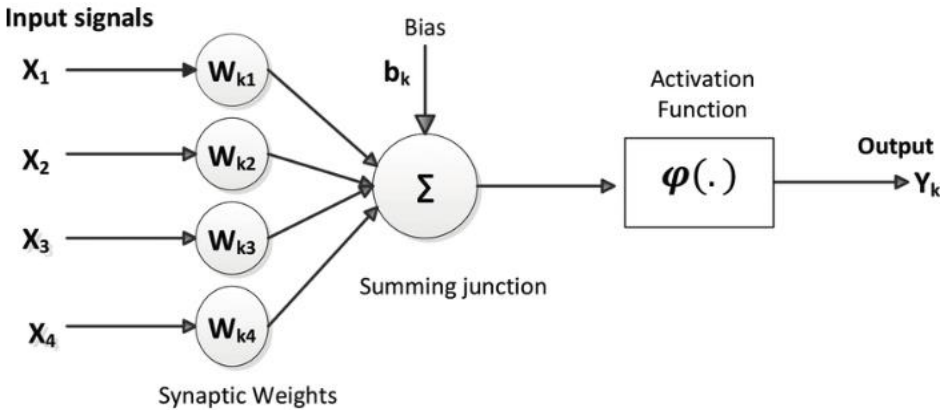


Warren Sturgis McCulloch (right) and Walter Harry Pitts (left) in 1949.

https://www.historyofinformation.com/detail.php?entryid=782

Artificial Neuron

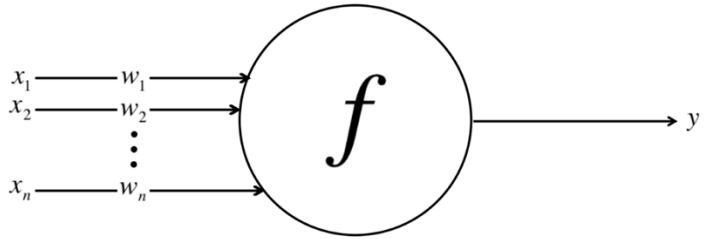
- The artificial neuron receives inputs, and each input is multiplied by the weights.



Artificial Neural

- This weighted sum produces the **logit** of the neuron:

$$z = \sum_{i=0}^{n} w_i \, x_i$$



- The logit may include a **bias**, which corresponds to a constant value (not included in this figure).

- The **logit** is the input to the function to produce the output, and this output can be transmitted to other neurons.

Artificial Neural Networks

- Let's reformulate the input as a vector $x = [x_1, x_2, \dots, x_n]$ and the weights of the neuron as $w = [w_1, w_2, \dots, w_n]$.

- Thus, the output of the neuron is :

$$y = f(x \cdot w + b)$$

where b is the "bias" term.

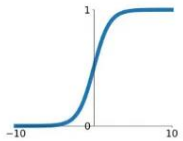
- Notice that f is a function that we called activation function.

Activation Functions

- The main goal is **to add non-linearity**, allowing the network to learn complex patterns.
- Common activation functions:

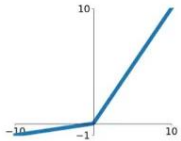
Sigmoid

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



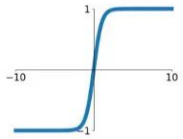
Leaky ReLU

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



tanh

$$\tanh(x)$$

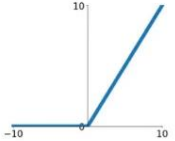


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

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

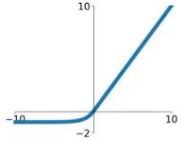
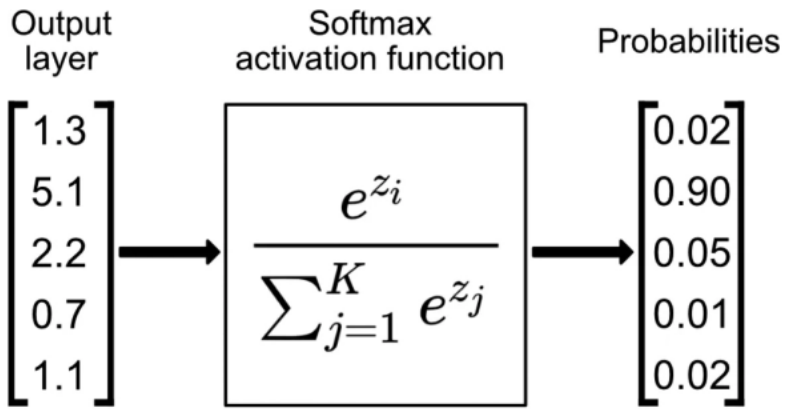


Image source: https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

Activation Functions : Softmax

- It used in the last layer of an ANN for classification tasks



- It converts raw output scores — also known as logits — into “probabilities” by taking the exponential of each output and normalizing these values by dividing by the sum of all the exponentials.
- This process ensures the output values are in the range (0,1) and sum up to 1, making them interpretable as probabilities.

Image source: https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

Activation Functions for imbalanced classes

For instance:

- Class-balance loss: it weighs the loss function by the inverse of the number of samples in each class.

$$\text{Loss} = -\frac{1}{N}\left(\frac{1}{N_i}\sum_{k\in i} y_k \log(\hat{y}_k) + \frac{1}{N_j}\sum_{k\in j}(1-y_k)\log(1-\hat{y}_k)\right)$$

- Focal loss:

$$\text{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

the model's estimated probability for the true class

Image source: https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

Artificial Neural Network (ANN)

- Thus, an ANN is a non-linear function with many parameters.

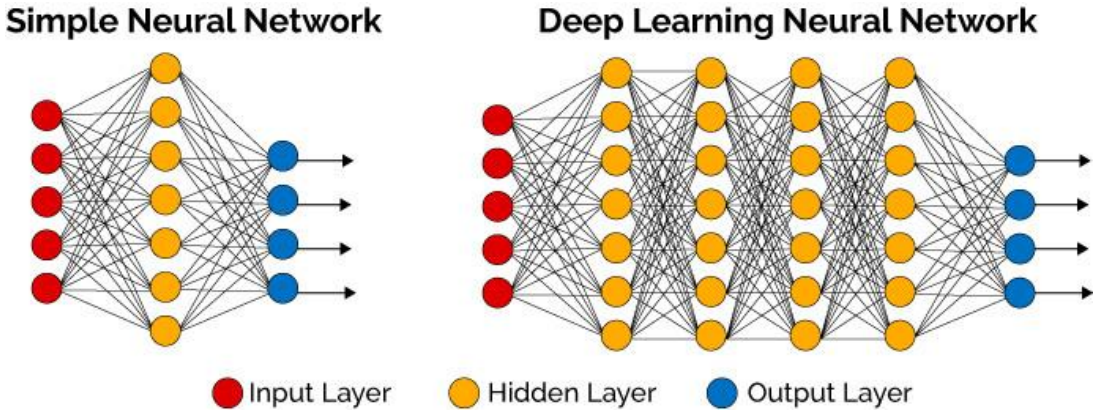


Image source: https://thedatascientist.com/what-deep-learning-is-and-isnt/

What is deep learning?

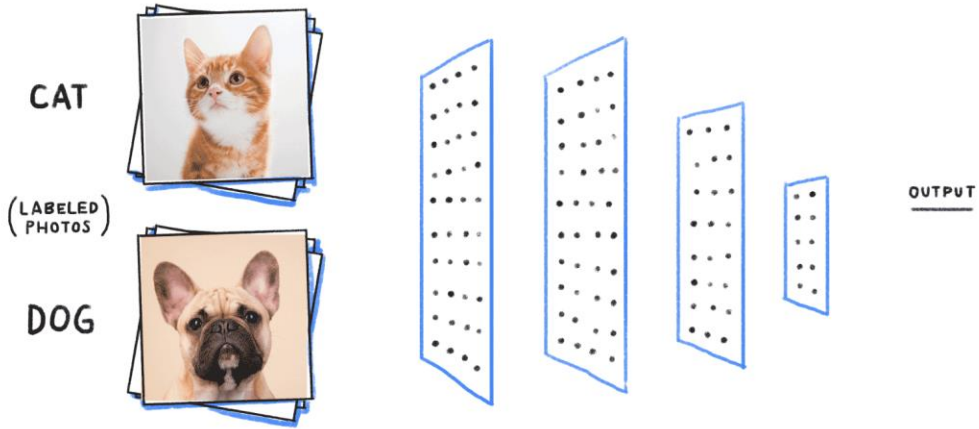
An artificial neural network with more than one hidden layer.

Applications of Artificial Neural Networks

Classification:
Predicting categories (e.g., spam detection, image labeling).

Regression:
Predicting continuous values (e.g., house prices, stock trends).

Clustering:
Grouping data into meaningful clusters without labels.



Dimensionality Reduction:
Reducing feature space while preserving information (e.g., PCA, autoencoders).

Sequence Prediction:
Time series forecasting (e.g., weather prediction, sales trends).

Generative Tasks:
Creating new data (e.g., GANs for image generation, text generation).

Image source: https://raihanrnj.medium.com/deep-learning-simple-image-classification-using-convolutional-neural-network-dog-and-cat-8c99aef29e8

Feed-Forward Neural Network

- An FFNN is the simplest type of ANN where information flows in one direction: from input to hidden layers to output.

- How to train an ANN? → Minimizing the error between the actual and predicted value → as usual!

- How to measure the error? → Loss function

Loss functions

- In simple words is a function that compares the true and the predicted values, measuring how well the ANN models the training data.

- The loss function is minimized during training using optimization algorithms like the gradient descent.

- The choice of loss function depends on the type of problem and output format.
  - **Regression**: Use MSE or MAE (Mean Absolute Error).
  - **Binary Classification**: Use Binary Cross-Entropy.
  - **Multi-Class Classification**: Use Categorical Cross-Entropy.

Loss functions

- Regression: Mean Square Error

Mean Error Squared

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

https://suboptimal.wiki/explanation/mse/

- Binary Classification: Binary cross-entropy

$\hat{Y}_i$ represents the "probability" of belonging to class 1 (a score in the range [0,1].

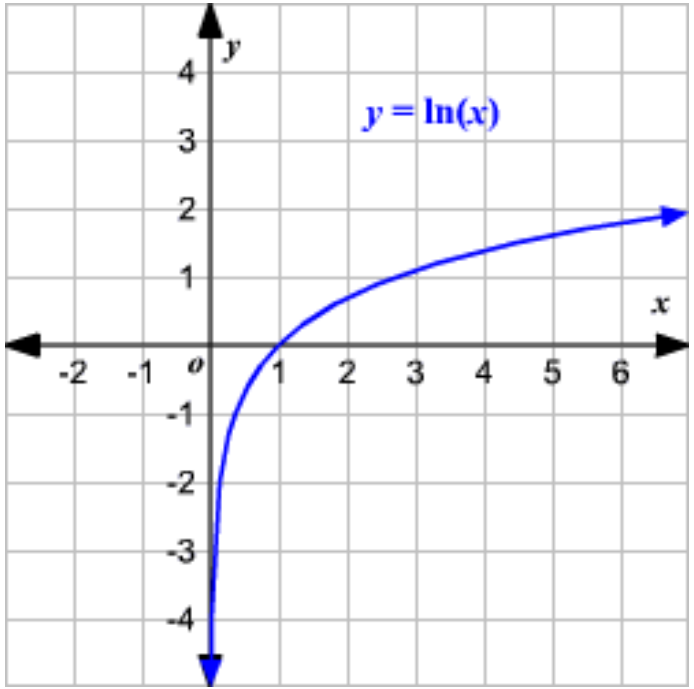
$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Loss functions

- Binary Classification: Binary cross-entropy

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

You can build your own loss function!



Training

- We want to minimize the loss function
- The main steps are:

1. Define the loss function (e.g., Mean Squared Error, Cross-Entropy).
2. Compute gradients of the loss with respect to weights and biases using **backpropagation**.
3. Adjust parameters iteratively to minimize the loss.

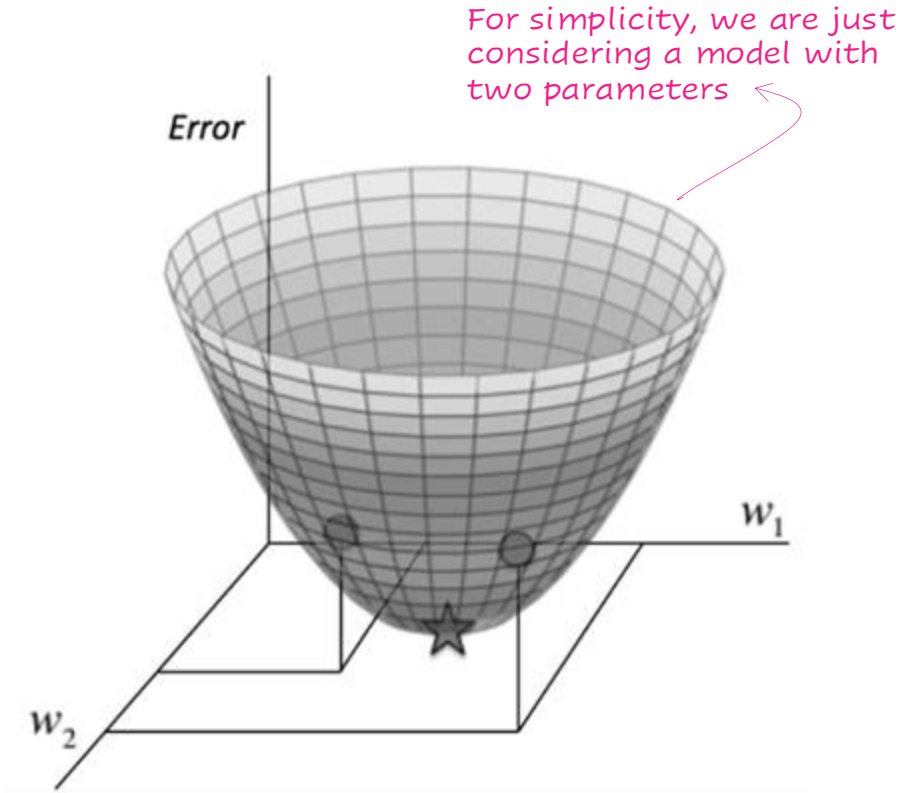
Training

- To have an idea of the minimization process – the training – let's consider the following loss function(*):

$$E = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2,$$

A quadratic function

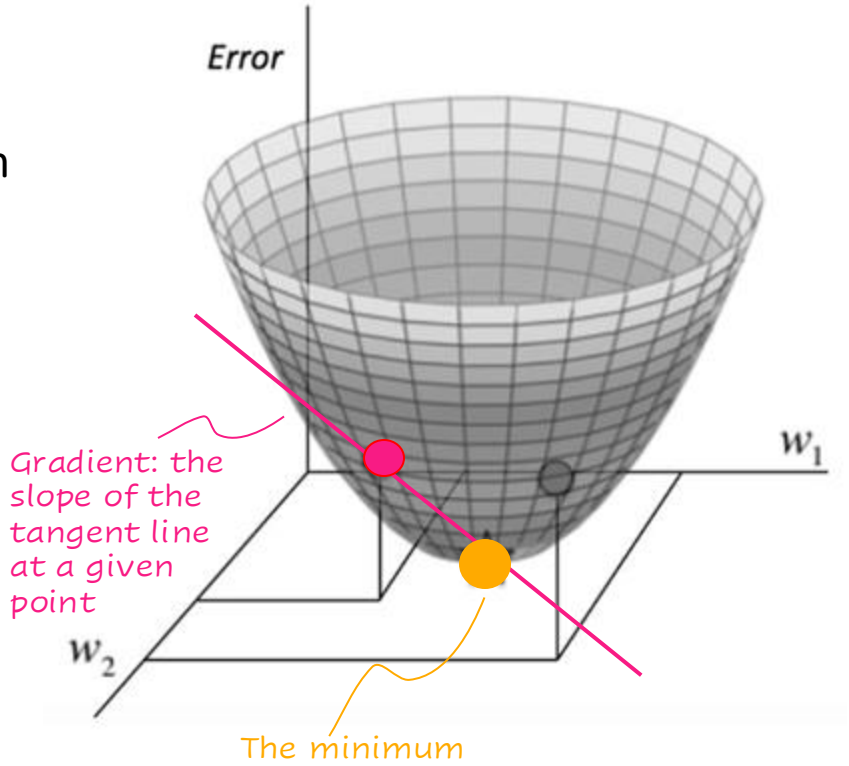
where y_i is the actual value and $\hat{y}_i$ is the predicted value.



(*) Using notation and explanation from: Buduma, N., & Locascio, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.".

Training

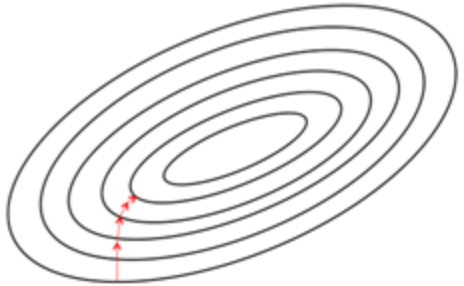
- **Minimize error** → optimization problem that seeks the parameter values that result in the lowest error.
- The gradient (**partial derivatives**) evaluated at a given point indicates the direction of the "steepest slope".
- The idea is to "move" along the surface to find the minimum.

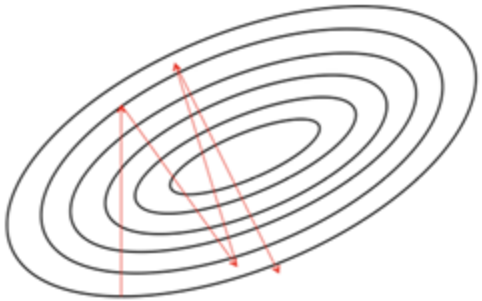


(*) Using notation and explanation from: Buduma, N., & Locascio, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.".

Training

- This strategy of "moving" (changing the values of w_i) based on the directions indicated by the gradient is the **gradient descent algorithm**.
- In practice, each step involves moving perpendicular to the contour of the surface.
- How far should we move? Multiply the gradient by a value called the **learning rate**.





(*) Using notation and explanation from: Buduma, N., & Locascio, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.".

Learning Rate

- The learning rate η is the size of each step taken during optimization.
- It determines the speed and precision of the algorithm's convergence to the optimal solution.
- Impact:
  - A smaller learning rate leads to slower but more precise convergence.
  - A larger learning rate may speed up convergence but risks overshooting or diverging.

Learning Rate Types

- Fixed Learning Rate: The learning rate remains constant throughout the training process.

- Adaptive Learning Rates: The learning rate changes dynamically based on the training progress, often shrinking as training continues. Examples:
  - **Learning Rate Decay**: Gradually reduces η over time to ensure finer adjustments as the model approaches the minimum.
  - **Optimization Algorithms**: Advanced techniques like Adam, RMSprop, and Adagrad adapt the learning rate for each parameter individually, optimizing training efficiency and stability.

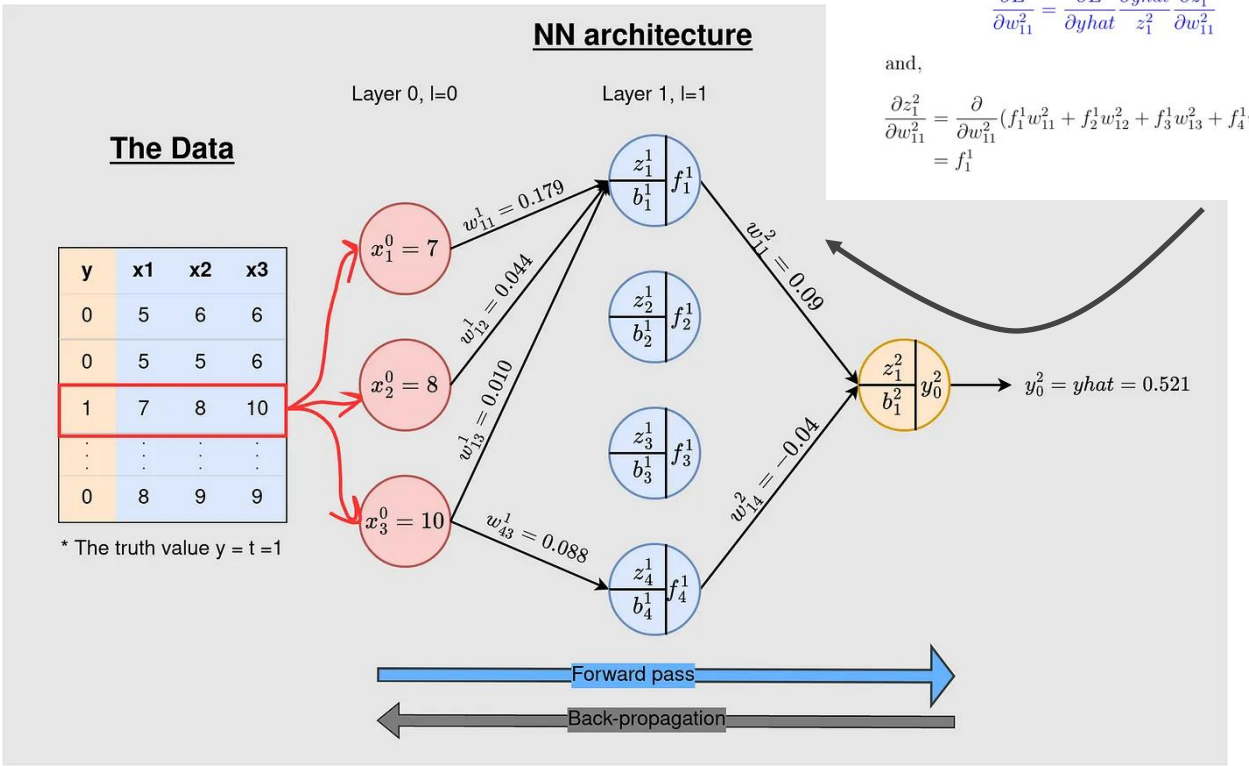
Backpropagation

- This algorithm was pioneered by David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams in 1986.

- Backpropagation is used to compute the gradients of the loss function with respect to the model's parameters (weights and biases).

- Idea:
  - Initialize the weights (for instance, randomly)
  - Forward pass o the data through the layer
  - Back-propagation

Forward and backward pass



$$\frac{\partial E}{\partial w_{11}^2} = \frac{\partial E}{\partial yhat}\frac{\partial yhat}{z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2}$$

and,

$$\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial}{\partial w_{11}^2}(f_1^1 w_{11}^2 + f_2^1 w_{12}^2 + f_3^1 w_{13}^2 + f_4^1 w_{14}^2 + b_1^2) = f_1^1$$

Image source: https://towardsdatascience.com/how-does-back-propagation-work-in-neural-networks-with-worked-example-bc59dfb97f48

ANN using Python

```
from keras.layers import Dense, Dropout
from keras.models import Sequential
import tensorflow  as tf



model = Sequential()
model.add(Dense(300, input_shape=(X_train.shape[1],), activation='relu'))
model.add(Dense(150, activation = "relu"))
model.add(Dense(100, activation = "relu"))
model.add(Dense(50, activation = "relu"))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

https://keras.io/

https://pytorch.org/

FFNN Example

- GitHub repository:

https://github.com/rpezoa/ML-HEP-School/blob/main/notebooks/Deep_Learning_FFNN.ipynb

Convolutional Neural Network (CNN)

- A neural network architecture is mainly used for visual tasks such as classification, object detection, segmentation, and more.
- FFNN cannot effectively capture the dependencies between pixels in an image because the image must be “flattened” into a one-dimensional array.
- CNNs take advantage of the multidimensional structure of images capturing the spatial relationships between pixels.

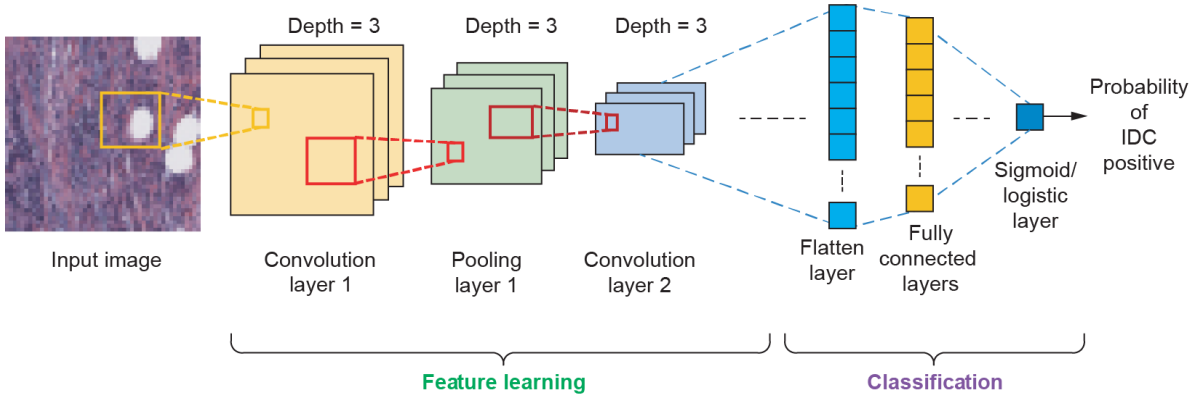
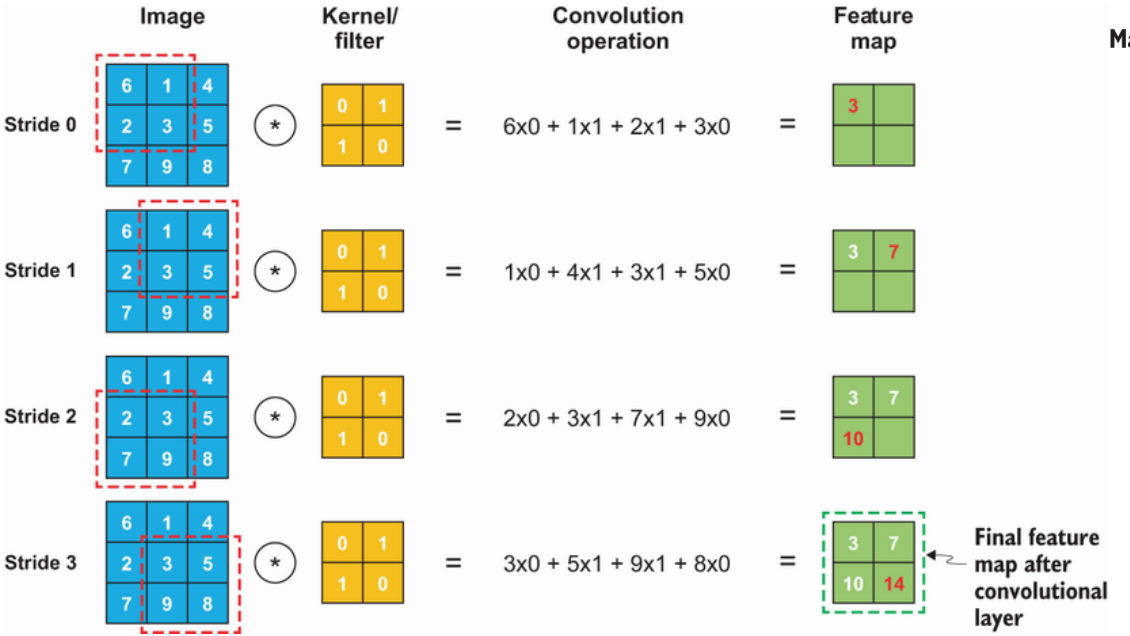
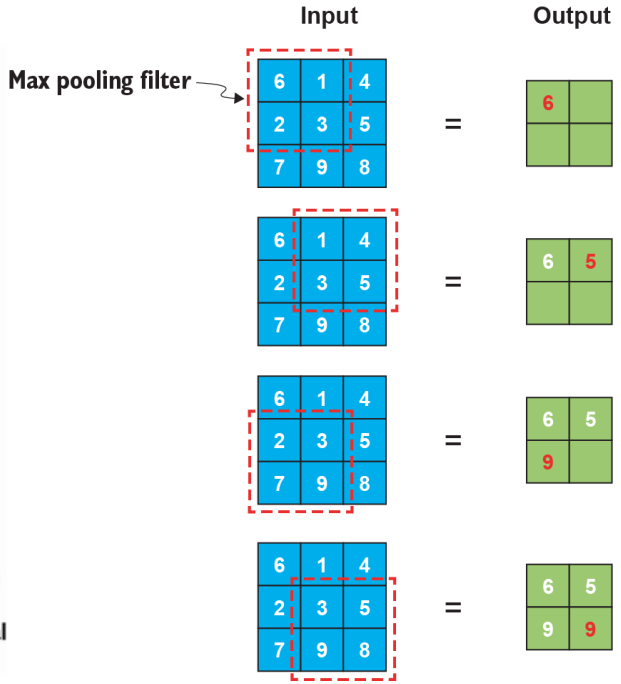


Image source: https://livebook.manning.com/book/interpretable-ai/chapter-5/25

Main CNN's layers



Convolutional layer



Max Pooling layers

Imagen obtenida de: https://livebook.manning.com/book/interpretable-ai/chapter-5/25

Convolutional layer

- Uses a **filter** or **kernel** whose elements are part of the parameters that define the CNN.

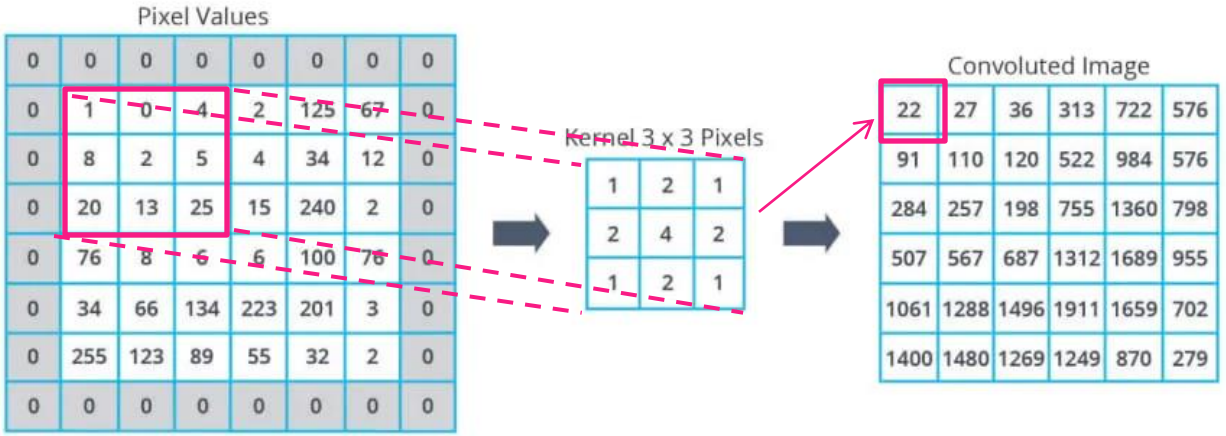
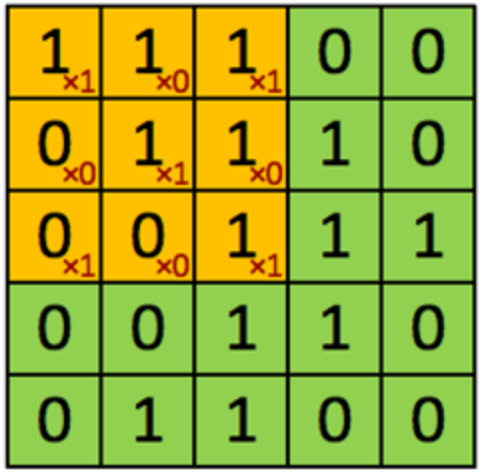
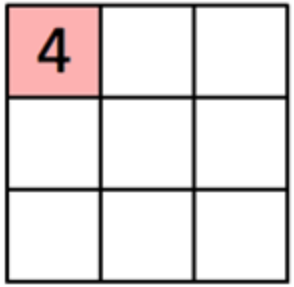


Image source: https://medium.com/@abhishekjainindore24/all-about-convolutions-kernels-features-in-cnn-c656616390a1

- **Filter size** (kernel size): The dimensions of the convolutional filter.
- **Number of output channels**: Specifies the depth of the **feature map** produced by the layer.
- **Stride**: The step size for sliding the kernel over the input.
- **Padding**: The method used to handle edge elements when the kernel does not fit perfectly.

Convolutional layer

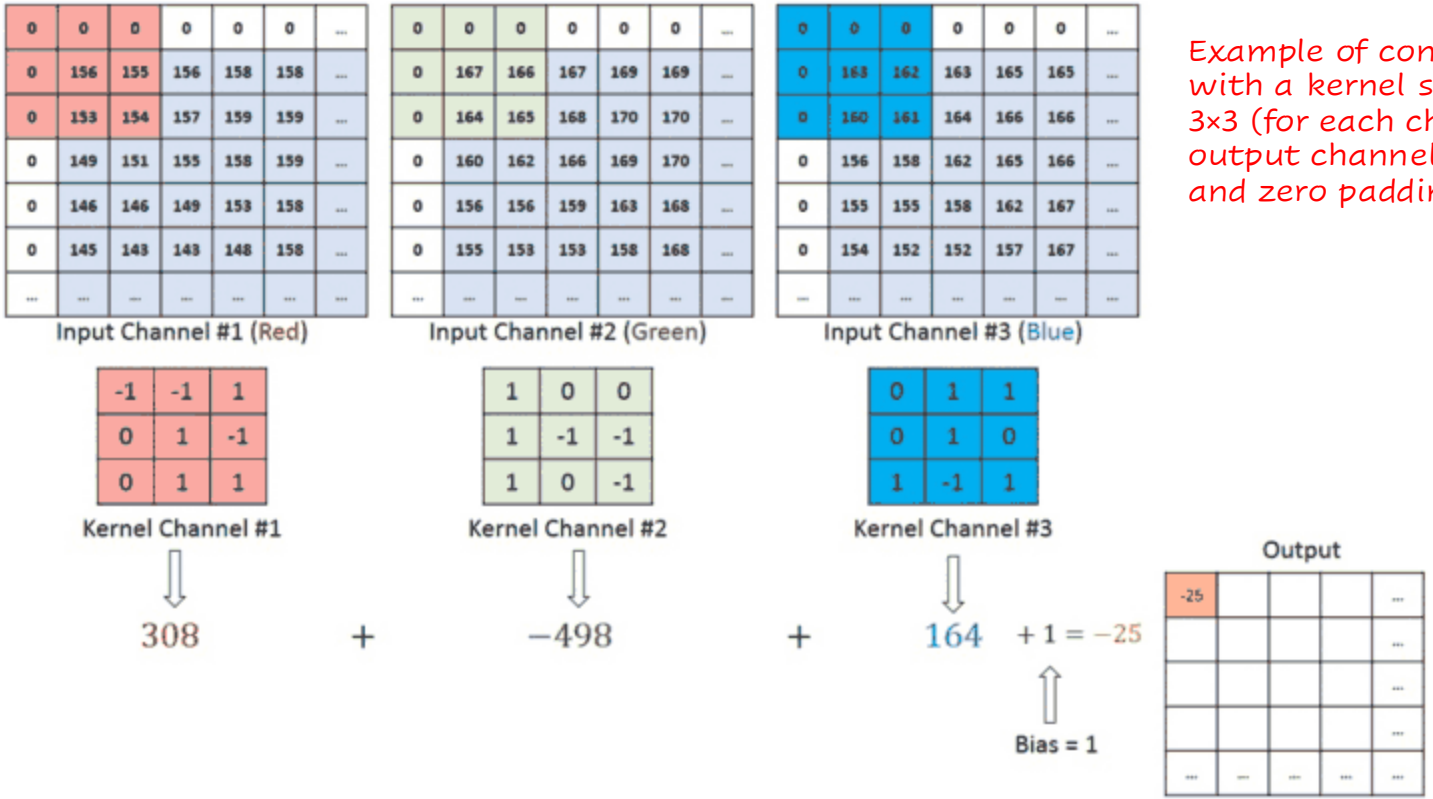




Convolution example with a 3x3 kernel, 1 output channel, stride=1 and without padding.

Image source: https://medium.com/nybles/a-brief-guide-to-convolutional-neural-network-cnn-642f47e88ed4

CNN's layers in a nutshell



Example of convolution with a kernel size of 3×3 (for each channel), 1 output channel, stride = 1, and zero padding.

Pooling layers

- Layers used to reduce the size of the input while extracting relevant features.
- Two main types of pooling:
  - **Max pooling** → Extracts the maximum value within the pooling window.
  - **Average pooling** → Extracts the average value within the pooling window.
- Key parameters include:
  - Filter size (pool size): The dimensions of the pooling window.
  - Stride: The step size for moving the pooling window across the input.
  - Padding: The method for handling border elements when the filter doesn't fit perfectly.

Image source: https://medium.com/nybles/a-brief-guide-to-convolutional-neural-network-cnn-642f47e88ed4

Fully Connected Layer

- A fully connected layer is simply a layer where **all neurons are connected to the neurons in the next layer**.

- The image is “flattened” to feed into the fully connected layer.
- This type of layer is useful, for example, for performing image classification tasks.

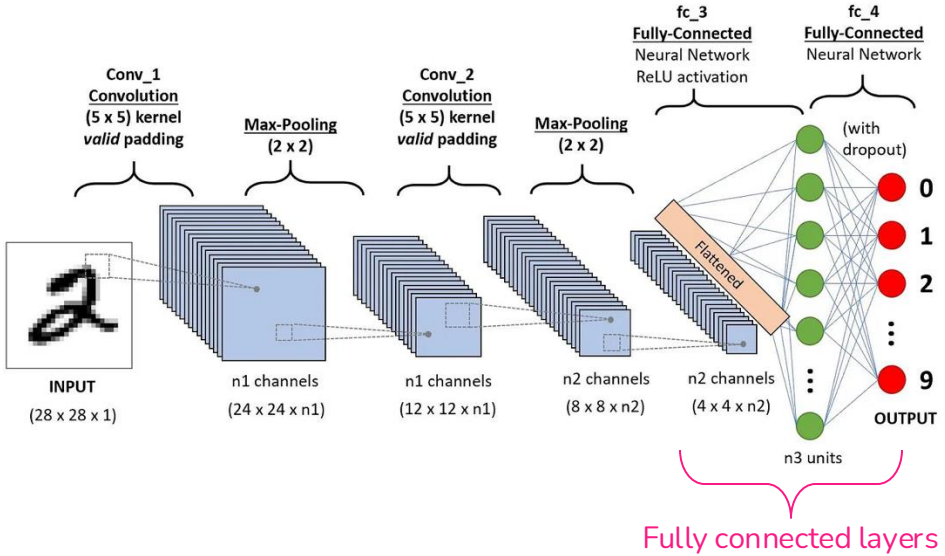
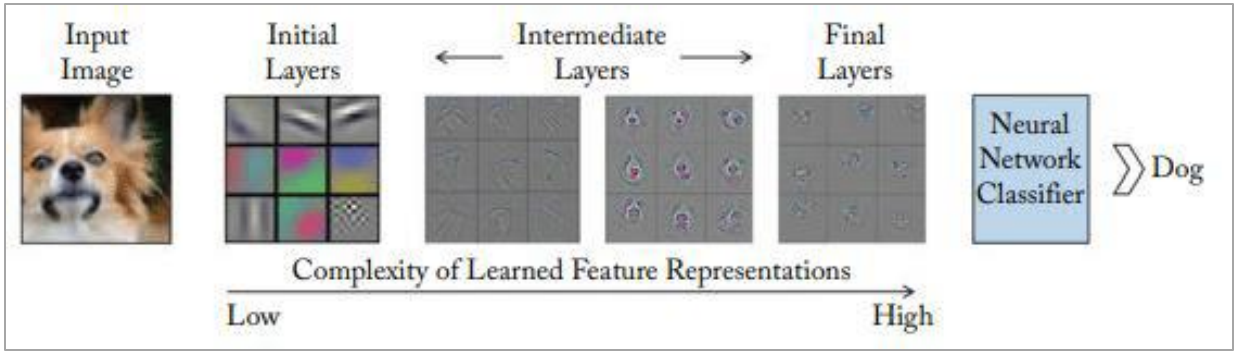


Image source https://medium.com/@alejandro.itoaramendia/convolutional-neural-networks-cnns-a-complete-guide-a803534a1930

Pattern extraction

- CNNs learn to detect hierarchical features in images.
- Low-Level Features: Detected by the initial layers (e.g., edges, corners, textures).
- High-Level Features: Detected by the deeper layers (e.g., shapes, objects, faces).



Early layers extract simple patterns
Deeper layers capture more abstract representations (e.g., progression from edges → textures → objects).

Image source: Sayeed, Azam. (2020). Detecting Malaria from Segmented Cell Images of Thin Blood Smear Dataset using Keras from Tensorflow. International Journal for Research in Applied Science and Engineering Technology. 8. 597-607. 10.22214/ijraset.2020.1109.

CNN using Python


```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

model.summary()
```

Example of CNN

- GitHub repository:

https://github.com/rpezoa/ML-HEP-School/blob/main/notebooks/Deep_Learning_CNN.ipynb

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

CCTVal
CENTRO CIENTÍFICO
TECNOLÓGICO
DE VALPARAÍSO

¡Gracias!

Acknowledgements: ANID PIA/APOYO AFB230003