Table of Contents

[CNN 1](#_Toc3182223)

[MNIST 1](#_Toc3182224)

[The pooling operation 1](#_Toc3182225)

[Hyperparameters of the convolutional layer 2](#_Toc3182226)

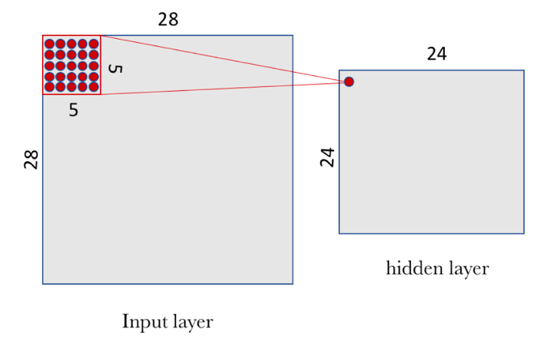
[Softmax 2](#_Toc3182227)

[RNN 2](#_Toc3182228)

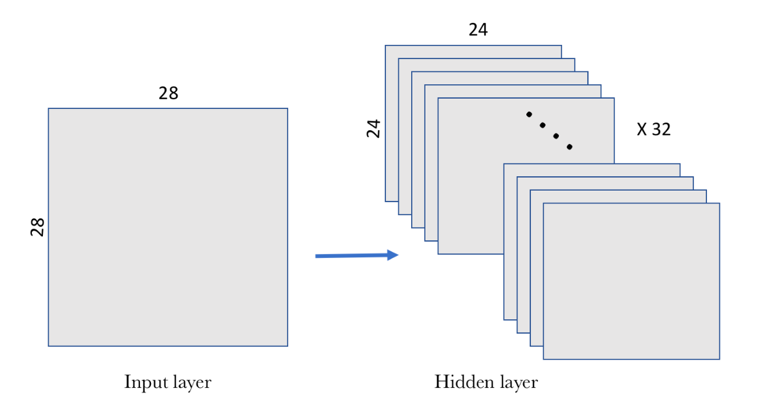
# CNN

## MNIST

In the case of MNIST, as input to our neural network we can think of a space of two-dimensional neurons 28×28 (height = 28, width = 28, depth = 1).



* to “connect” each neuron of the hidden layer with the 25 corresponding neurons of the input layer we will use a bias value b and a W-weights matrix of **size 5×5 that we will call filter (or kernel).** The value of each point of the hidden layer corresponds to the scalar product between the filter and the handful of 25 neurons (5×5) of the input layer.
* However, the particular and very important thing about convolutional networks is that we use the same filter (the same W matrix of weights and the same b bias) for all the neurons in the hidden layer: in our case for the 24×24 neurons (576 neurons in total) of the first layer. The reader can see in this particular case **that this sharing drastically reduces the number of parameters that a neural network would** have if we did not do it: it goes from 14,400 parameters that would have to be adjusted (5×5 × 24×24) to 25 (5×5) parameters plus biases b.
* But a filter defined by a matrix W and a bias b only allows detecting a specific characteristic in an image; therefore, in order to perform image recognition, it is **proposed to use several filters at the same time, one for each characteristic that we want to detect.** – need for several filters
* In our example we propose 32 filters, where each filter is defined with a W matrix of 5×5 and a bias b.

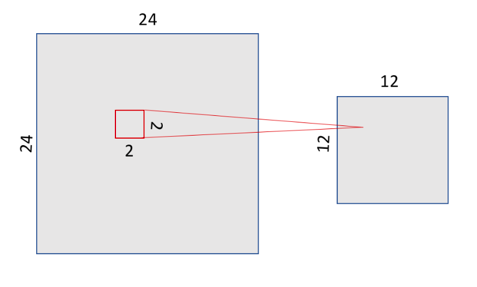


In this example, the first convolutional layer receives a size **input tensor (28, 28, 1)** and generates a size output **(24, 24, 32), a 3D tensor** containing the 32 outputs of 24×24 pixel result of computing the 32 filters on the input

### The pooling operation

In our MNIST example, we are going to choose a 2×2 window o

* the convolutional layer hosts more than one filter and, therefore, as we apply the max-pooling to each of them separately, the pooling layer will contain as many pooling filters as there are convolutional filters:



The result is, since we had a space of 24×24 neurons in each convolutional filter, after doing the pooling we have 12×12 neurons

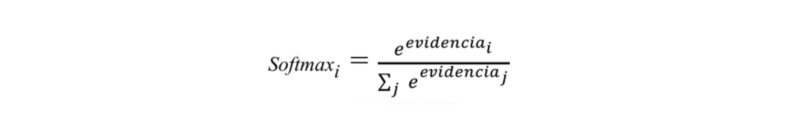
Max-pooling does not require parameters since it is a mathematical operation to find the maximum.

### Hyperparameters of the convolutional layer

* *Size and number of filters*
* *Stride:*which indicates the number of steps in which the filter window moves Large stride values decrease the size of the information that will be passed to the next layer
* Padding:But sometimes we want to obtain an output image of the same dimensions as the input and we can use the hyperparameter padding in the convolutional layers for this. With padding we can add zeros around the input images before sliding the window through it.

### Softmax

Once the evidence of belonging to each of the 10 classes has been calculated, these must be converted into probabilities whose sum of all their components add 1. For this, softmax uses the exponential value of the calculated evidence and then normalizes them so that the sum equates to one, forming a probability distribution. The probability of belonging to class *i* is:

Intuitively, the effect obtained with the use of exponentials is that one more unit of evidence has a multiplier effect and one unit less has the inverse effect. The interesting thing about this function is that a good prediction will have a single entry in the vector with a value close to 1, while the remaining entries will be close to 0. In a weak prediction, there will be several possible labels, which will have more or less the same probability.

# RNN