



# IIC 2433 Minería de Datos

<https://github.com/marcelomendoza/IIC2433>

# AUTOENCODERS

# Auto-encoders

Motivación: Trabajar sobre una representación de baja dimensionalidad y con alta capacidad informativa para aprendizaje automático. Útil para *data augmentation*.

Los primeros autoencoders se construyeron a partir de redes neuronales entrenadas para reconstruir la entrada.

Formalmente, el problema corresponde a aprender dos funciones:

Encoder:  $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$

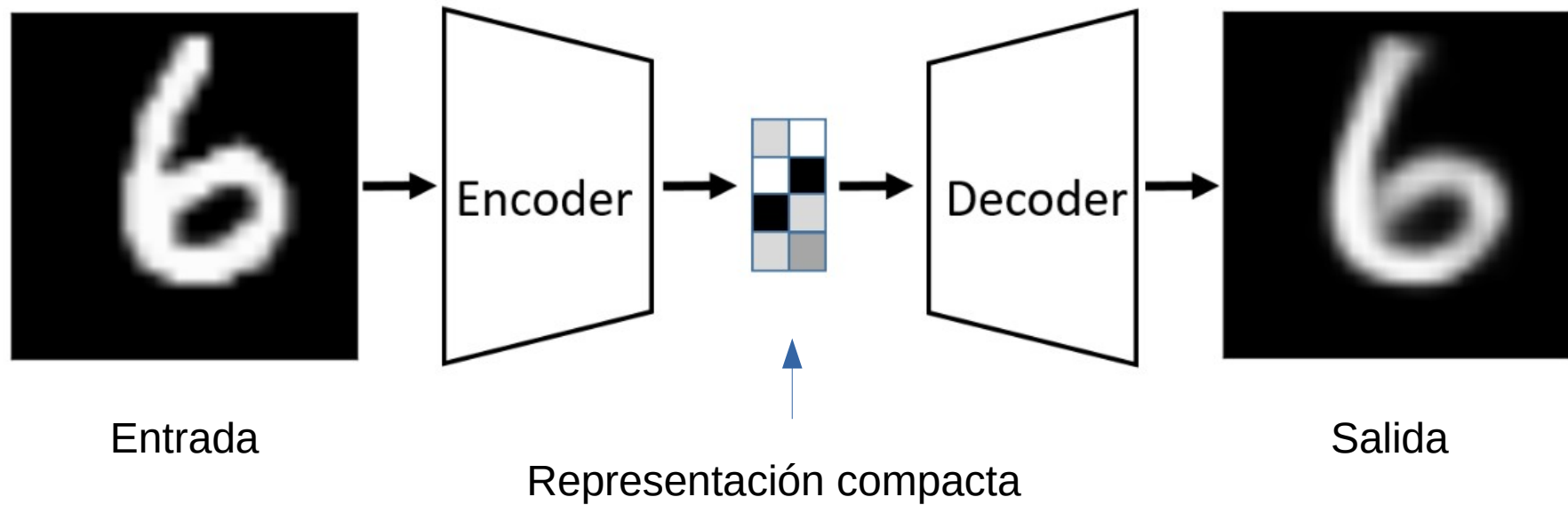
Composición de funciones

Decoder:  $B : \mathbb{R}^p \rightarrow \mathbb{R}^n$

de tal manera que:  $\arg \min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x}))]$

Error de reconstrucción  
(usualmente norma  $L_2$ )

## Auto-encoders



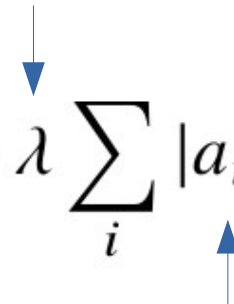
- Si A y B implementan redes feed-forward lineales (activación lineal), el autoencoder se denomina autoencoder lineal.
- Un autoencoder es una generalización de PCA.

# Auto-encoders

- Reducir la dimensionalidad es útil para evitar over-fitting ¿Por qué?
- Surge un tradeoff: Por un lado queremos que la arquitectura obtenga un error de reconstrucción muy bajo. Por otro, queremos que la representación compacta descarte información no esencial.
- Una forma de abordar el tradeoff consiste en introducir sparsity en las activaciones de las capas ocultas. Existen dos estrategias, regularización  $L_1$  o uso de divergencia KL.

Requiere ajustar  
este parámetro

Sparse autoencoder con regularización  $L_1$ :

$$\arg \min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x}))] + \lambda \sum_i |a_i|$$


Activación de la  $i$ -th neurona oculta

# Auto-encoders

Sparse autoencoder basada en divergencia KL: asumimos que la activación de cada neurona actua como una variable Bernoulli con probabilidad  $p$ . Se controla el parámetro  $p$ .

Para cada batch, se estima la probabilidad y se calcula la diferencia, la cual es usada como regularizador.

Para cada neurona  $j$  la probabilidad empírica es:

$$\hat{p}_j = \frac{1}{m} \sum_i a_i(x)$$

Tamaño del batch 

Luego, la función de pérdida es:

$$\arg \min_{A,B} E[\Delta(\mathbf{x}, B \circ A(\mathbf{x}))] + \sum_j KL(p || \hat{p}_j)$$

# Auto-encoders variacionales

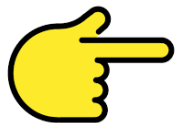
El VAE es un modelo generativo (enfoque Bayesiano) que describe el proceso generativo de los datos.

Dado un dataset  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ , el VAE asume la generación de cada

dato condicionada a una variable latente aleatoria  $\mathbf{z}_i$ . Este modelo describe al decoder (probabilístico).

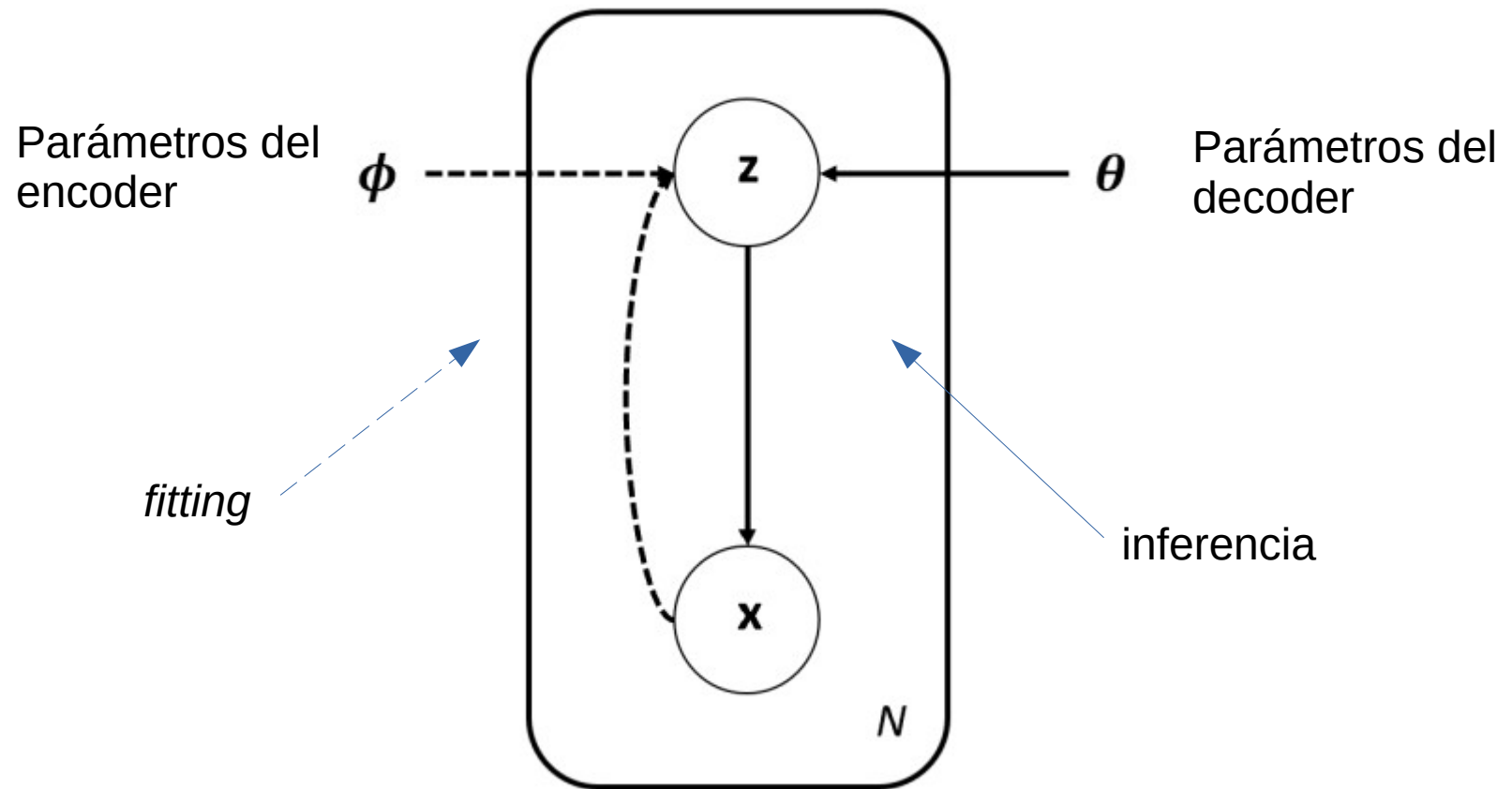
Análogamente, asumimos la existencia de una distribución a *posteriori* para generar las variables latentes a partir de los datos. Este modelo describe al encoder (probabilístico).

Las variables latentes tiene una distribución a priori denotada por  $p_{\theta}(\mathbf{z}_i)$ .



Diederik Kingma, Max Welling: Auto-Encoding Variational Bayes. ICLR 2014.

# Auto-encoders variacionales





# Auto-encoders variacionales

¿Qué hace el VAE?

Es un modelo de variable latente que usa redes neuronales (en específico perceptrón multicapa) para aproximar la *posterior* de

$q_\phi(z|x)$  y del modelo generativo  $p_\theta(x, z)$ .

Asumimos que la posterior aproximada es una Gaussiana multivariada. Los parámetros de esta distribución son calculados usando un MLP que toma los datos como entrada:

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma_\phi(x)\mathbf{I})$$

# Auto-encoders variacionales

¿Qué hace el VAE?

Es un modelo de variable latente que usa redes neuronales (en específico perceptrón multicapa) para aproximar la *posterior* de

$q_\phi(z|x)$  y del modelo generativo  $p_\theta(x, z)$ .

Asumimos que la posterior aproximada es una Gaussiana multivariada. Los parámetros de esta distribución son calculados usando un MLP que toma los datos como entrada:

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma_\phi(x)\mathbf{I})$$

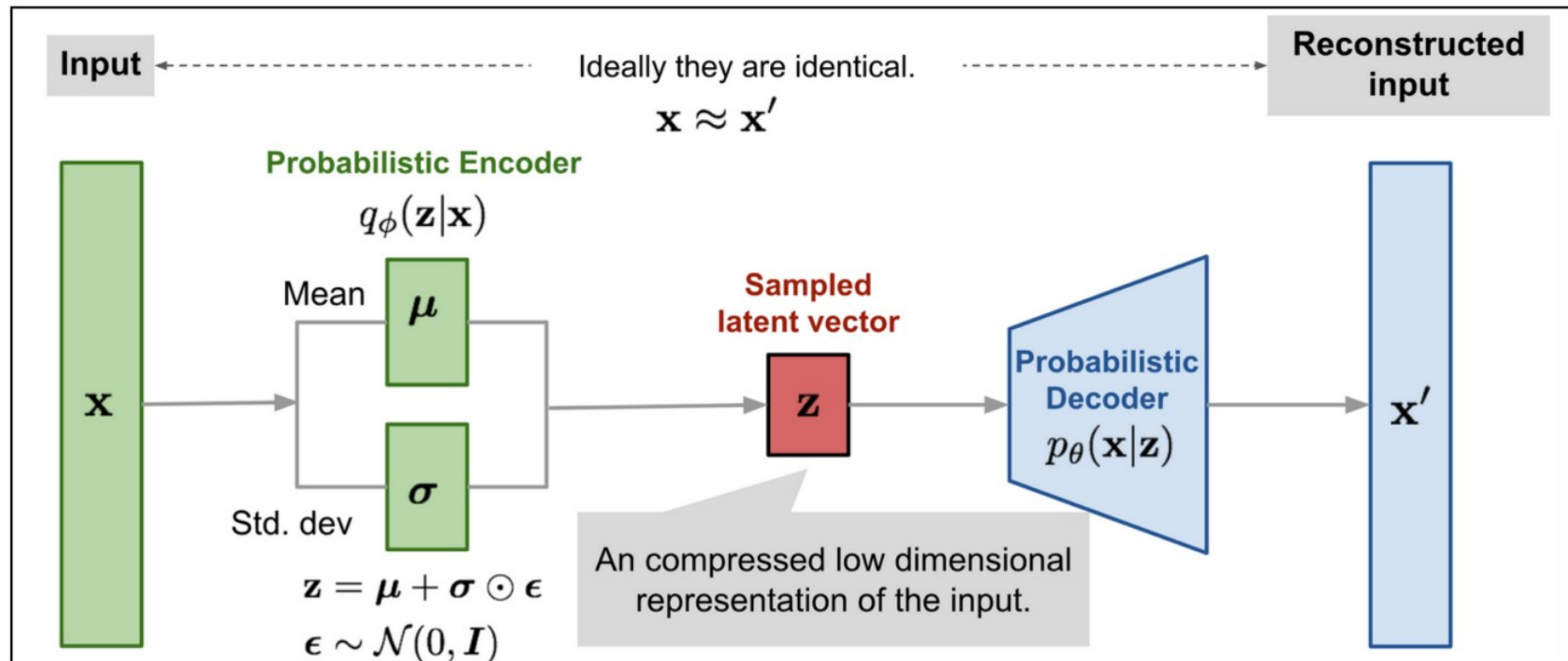
Para el *decoder*, se asume  $p(z)$  fijo:

$$p(z) = \mathcal{N}(0, \mathbf{I})$$

El modelo generativo dependerá del tipo de datos con los que trabajamos. Por ejemplo:

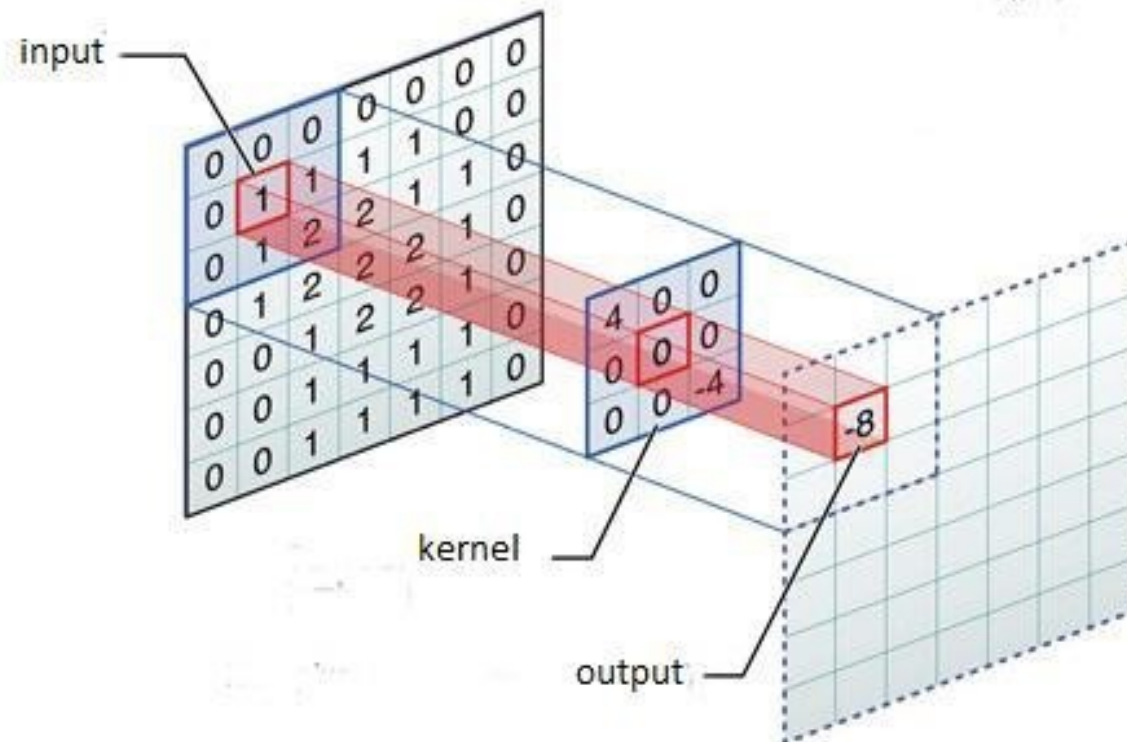
- Datos reales: Gaussiana multivariada
- Datos booleanos: Bernoulli

# Auto-encoders variacionales



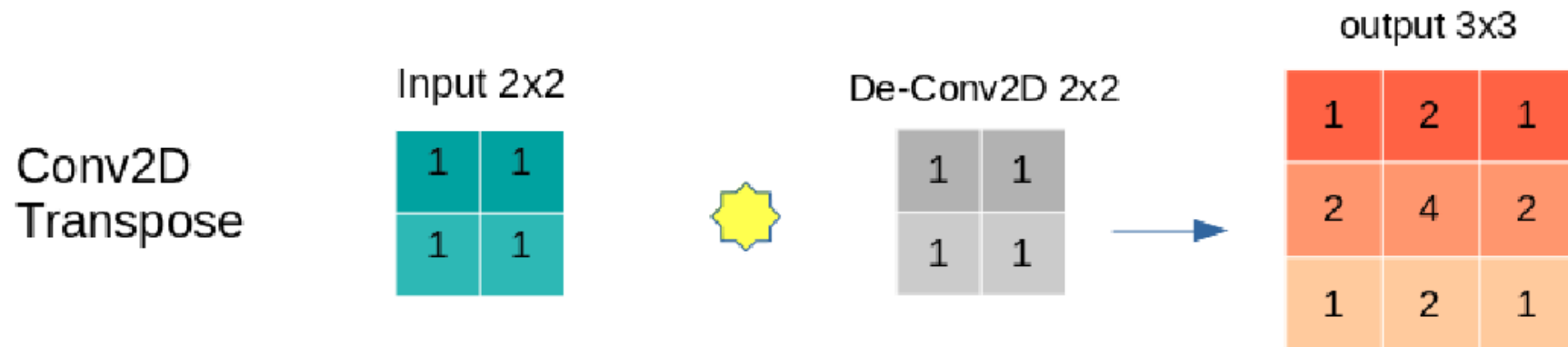
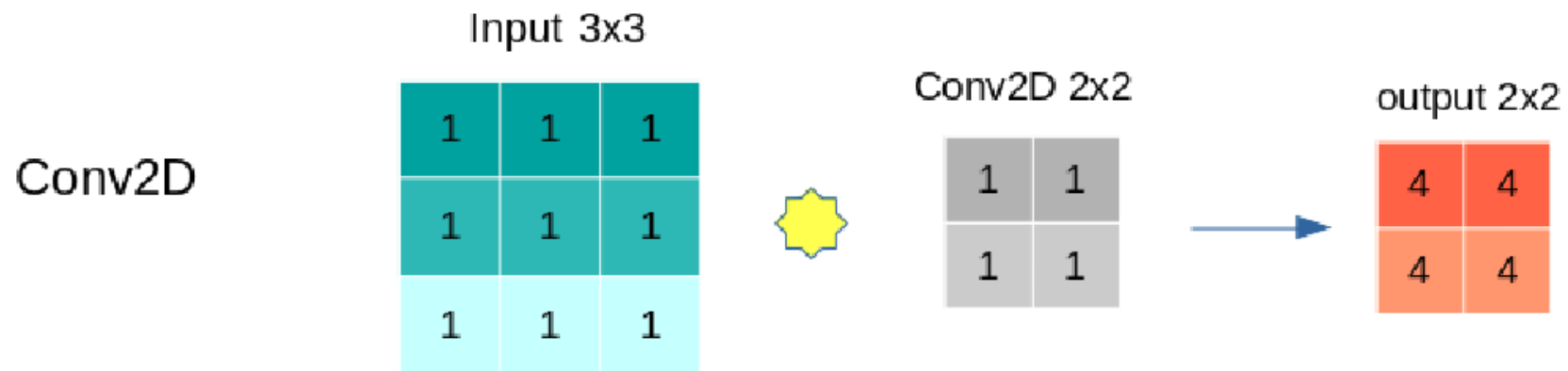
El VAE puede generar datos nuevos

# Auto-encoders variacionales con imágenes



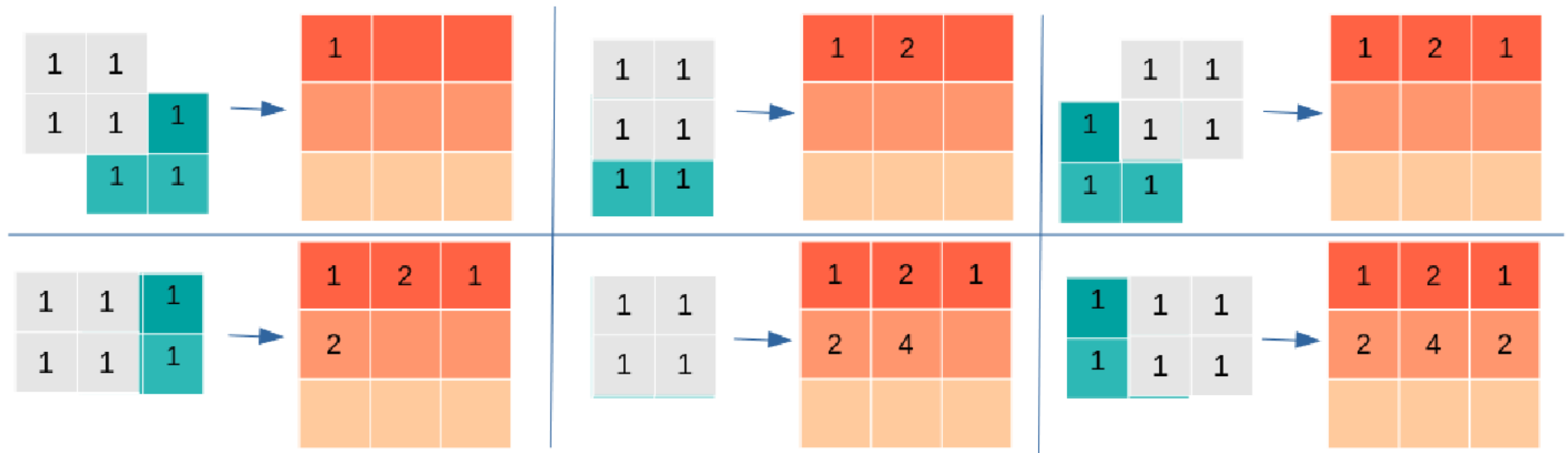
# Auto-encoders variacionales con imágenes

## Filtros convolucionales 2D

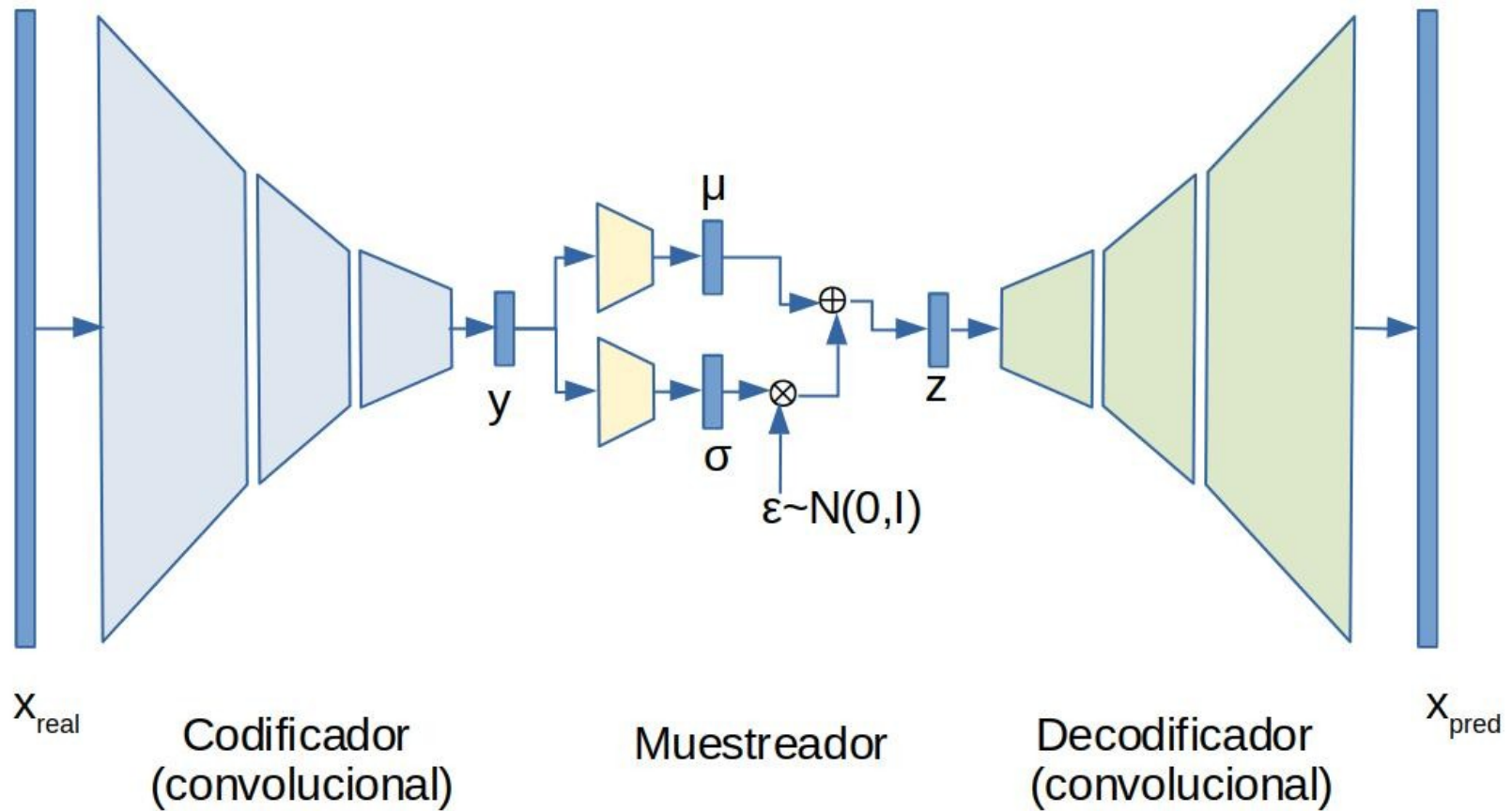


# Auto-encoders variacionales con imágenes

## Filtros convolucionales 2D transpose



# Auto-encoders variacionales con imágenes



## Data augmentation en base al VAE preentrenado

