

# Autoencoder Convolucional(DAE's) para eliminar el ruido de imágenes (mejorar la calidad).

Ntory

# CONTENIDO

01

## INTRODUCCIÓN

Qué es Autoencoder?  
Convolucional  
Dataset empleado

02

## AUTOENCODER CONVUNCIONAL

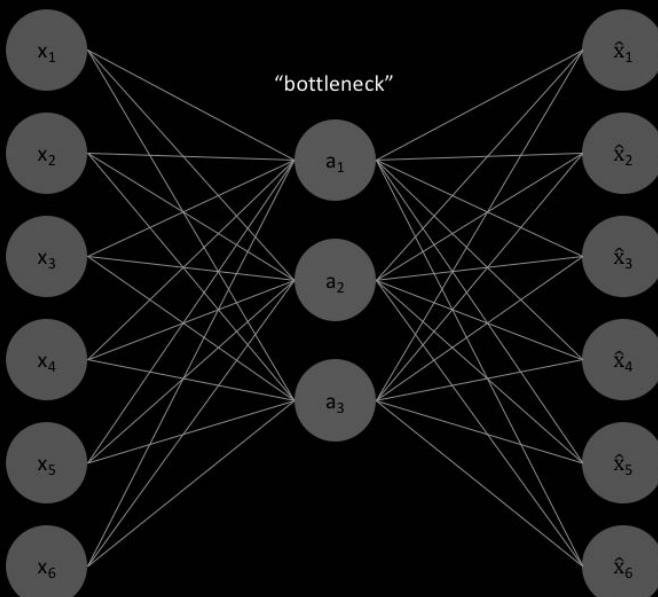
Estructura

03

## RESULTADOS

Resultados y conclusiones

Input layer      Hidden layer      Output layer



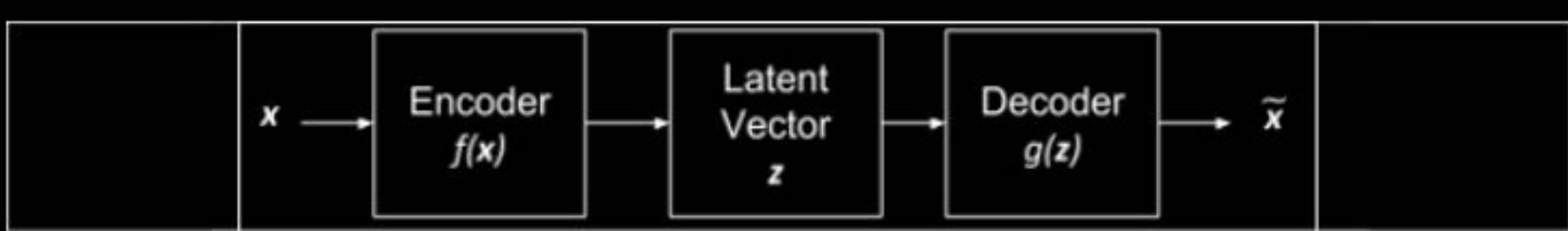
01

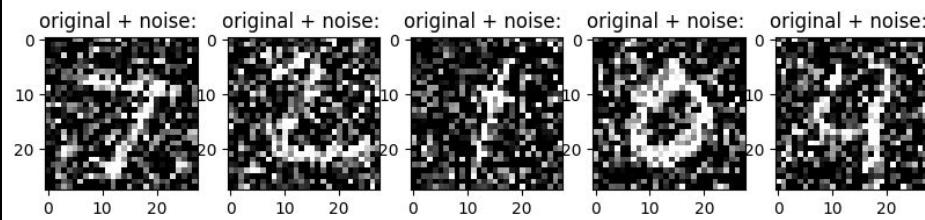
# Introduction

# INTRODUCCIÓN

Independientemente de las restricciones impuestas, todos los autoencoders comparten una arquitectura común, compuesta por dos partes fundamentales:

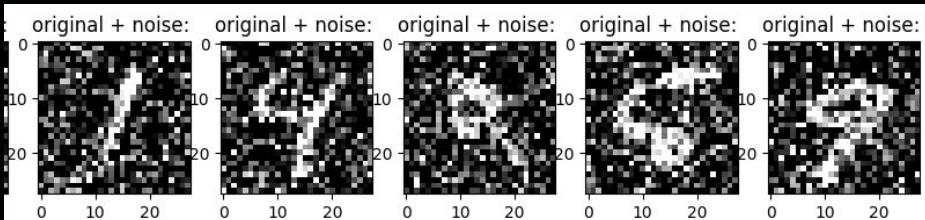
1. **Encoder** encargado de transformar la entrada  $x$ , en la representación latente, representada por el vector  $z = f(x)$ .
2. **Decoder**: cuyo trabajo consiste en la recuperación de los datos originales a partir de dicha representación latente,  $\tilde{x} = g(z)$





# Dataset: MNIST

Consiste en un conjunto de 70,000 imágenes de dígitos escritos a mano (0 al 9), cada una de 28x28 píxeles en escala de grises. Sin embargo nosotros ocupamos 60,000 imágenes con ruido para entrenar nuestro modelo y 10,000 para probar que tan bueno es nuestro modelo.



## 02

# Autoencoder Convucional

```
inputs = Input(shape=input_shape, name='Input_encoder')# Construir el modelo del codificador
x = inputs
# Capa de convolución 2D 1
x = Conv2D(filters=32, kernel_size=kernel_size, strides=2, activation='relu', padding='same')(x)
x = Dropout(0.20)(x) # Capa de dropout para regularización
# Capa de convolución 2D 2
x = Conv2D(filters=64, kernel_size=kernel_size, strides=2, activation='relu', padding='same')(x)
shape = K.int_shape(x) # Obtener la forma actual para la capa de aplanado

x = Flatten()(x) # aplanamos la salida para generar el latent vector
latent = Dense(latent_dim, name='Latent_vector')(x) # Capa densa para representar el vector latente
self.encoder = Model(inputs, latent, name='encoder') # Instanciar el modelo del codificador

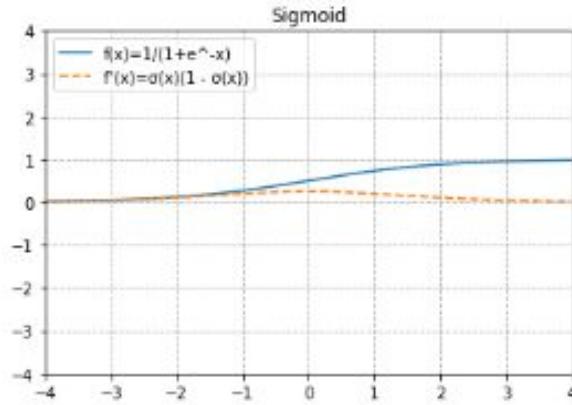
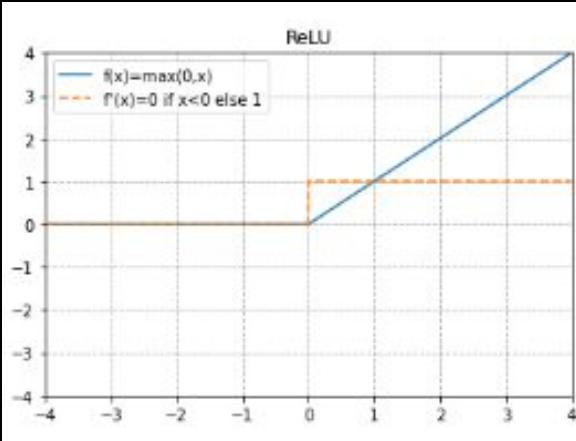
# Construir el modelo del decodificador
latent_inputs = Input(shape=(latent_dim,), name='decoder_input')
x = Dense(shape[1] * shape[2] * shape[3])(latent_inputs) # Capa densa para expandir el vector latente
# Obtenemos la forma adecuada para que nuestras capas que siguen puedan trabajar
x = Reshape((shape[1], shape[2], shape[3]))(x)

# 2 capas Conv2DTranspose(64)-Conv2DTranspose(32) que hacen lo contrario a las que ocupamos en el encoder
x = Conv2DTranspose(filters=64, kernel_size=kernel_size, strides=2, activation='relu', padding='same')(x)
x = Conv2DTranspose(filters=32, kernel_size=kernel_size, strides=2, activation='relu', padding='same')(x)
# Capa de salida para reconstruir la imagen original
outputs = Conv2DTranspose(filters=1, kernel_size=kernel_size,
                           padding='same', activation='sigmoid', name='decoder_output')(x)
self.decoder = Model(latent_inputs, outputs, name='decoder') # Instanciamos el modelo del decodificador
```

Las funciones de activación son el secreto que hace que las redes neuronales sean capaces de resolver problemas complejos. Ocupamos ReLU y sigmoid

En todo ocupamos 'same' (para mantener el tamaño de la entrada)

Para compilar ocupamos el optimizador Adam junto con la callback ReduceLROnPlateau y Means Square Error como función de perdida,

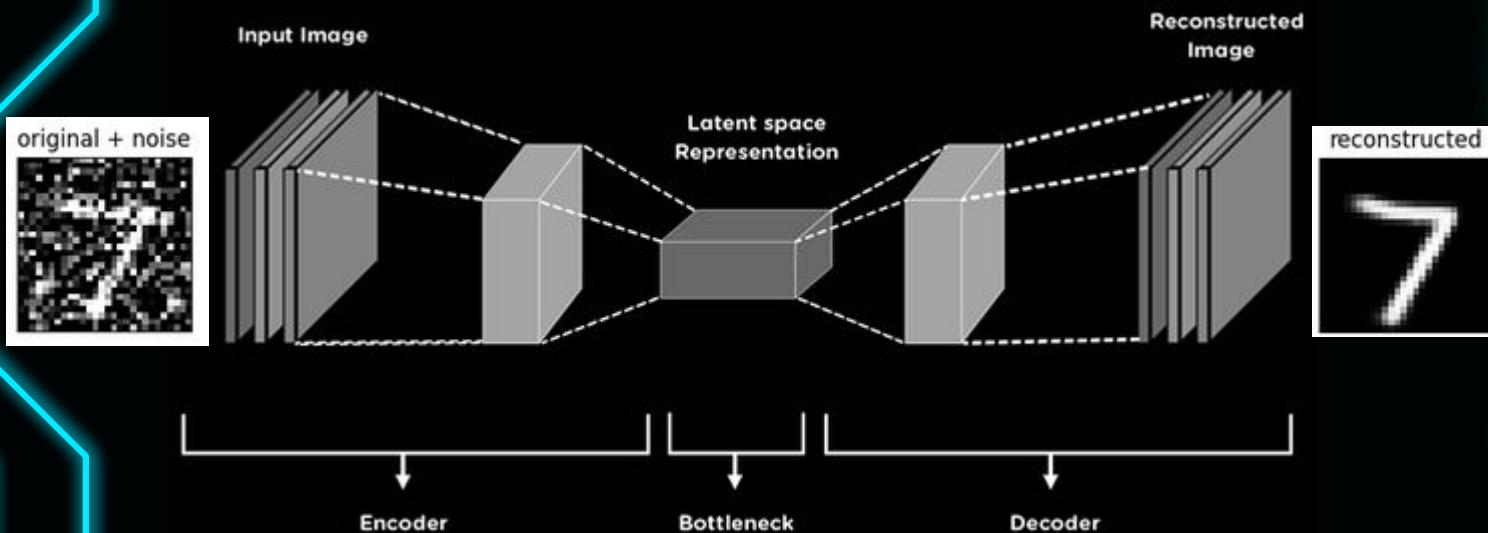


Cabe recalcar que naturalmente una reducción de la dimensionalidad implica que se pierde información por el camino, que estos datos ya no están completos.

# 03

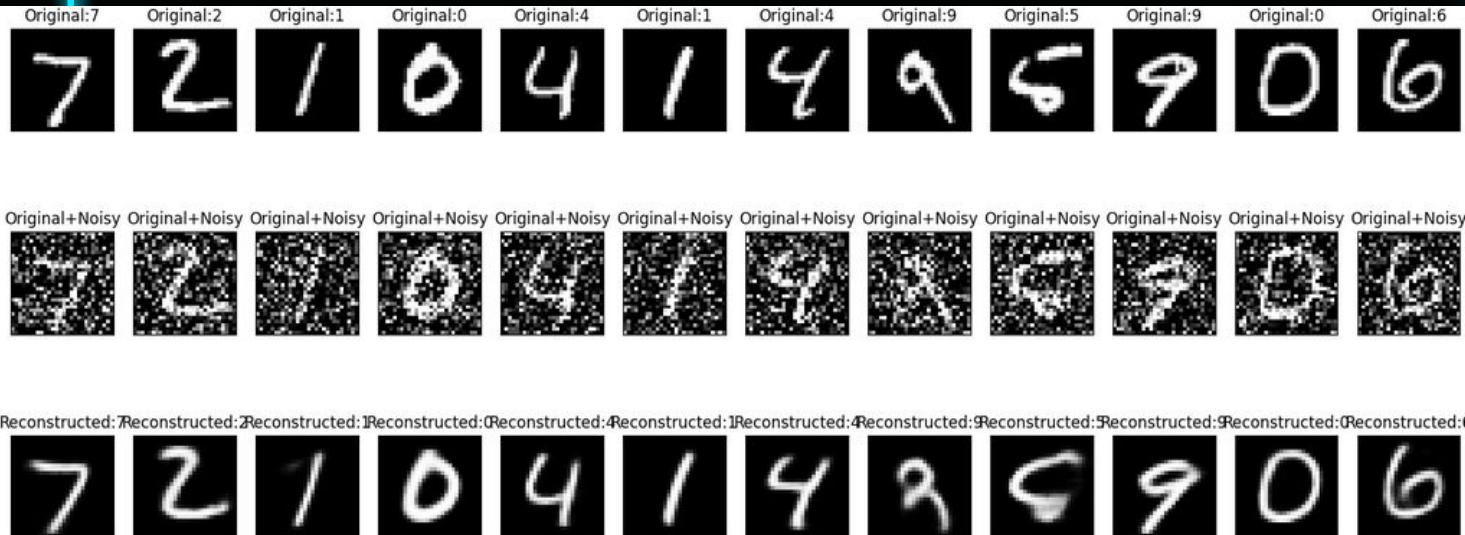
# Resultados y Conclusión

Mean Squared Error (MSE) : 0.013719023205339909



## 03

# Resultados y Conclusión



# Referencias

- [0] Material visto y proporcionado en clase(en particular lo visto en la clase 21).
- [1] Sosa-Costa, A. (2019, octubre 1). Reducción de ruido en imágenes utilizando Autocodificadores. Cursos de Programación de 0 a Experto Garantizados. Recuperado el 7 de diciembre de 2023, de:  
<https://unipython.com/reduccion-de-ruido-en-imagenes-utilizando-autocodificadores/>
- [2] Jesús. (2019, enero 19). Autoencoders. DataSmarts. Recuperado el 9 de diciembre de 2023, de:  
<https://www.datasmarts.net/autoencoders/>
- [3] Rowel Atienza, 2020, *\*Advanced Deep Learning with TensorFlow 2 and Keras\** (Second Edition), Packt Publishing Ltd. Recuperado el 10 de diciembre de 2023, de:  
<https://www.hlevkin.com/hlevkin/45MachineDeepLearning/Keras/Advanced%20Deep%20Learning%20with%20TensorFlow%20and%20Keras.pdf>
- [4] Rodríguez Ferrero, Ignacio, *\*Uso de autoencoders en la compresión de imágenes\**(Trabajo Fin De Grado, UNIVERSITAT POLITÈCNICA DE VALÈNCIA) Recuperado el 10 de diciembre de 2023, de:  
<https://riunet.upv.es/bitstream/handle/10251/185803/Rodriguez%20-%20Uso%20de%20autoencoders%20en%20la%20compresion%20de%20imagenes.pdf?sequence=1>

OCUPE MÁS pero estas son las más importantes.

**GRACIAS POR SU ATENCIÓN!  
CUÍDALA!!!**