REPORTE DE PRÁCTICA

Seminario De Sistemas Basados En Conocimiento

**Nombre Del Alumno** **Código Fecha**

|  |  |  |
| --- | --- | --- |
| Christopher Ceballos Jimenez | 219750442 | 09/Mayo |

**NOMBRE DE LA PRÁCTICA/ NUMERO DE LA PRÁCTICA**

|  |
| --- |
| Práctica 7. Implementación de un Autoencoder utilizando el dataset MNIST |

**MARCO TEÓRICO**

|  |
| --- |
| Los autoencoders son un tipo de redes neuronales ampliamente empleadas en Deep learning, destacando en la reducción de ruido en imágenes, como el ruido sal y pimienta. Este tipo de ruido se caracteriza por la presencia aleatoria de píxeles blancos y negros dispersos en la imagen, imitando la apariencia de sal y pimienta esparcida. Los autoencoders funcionan codificando la imagen en una representación de menor tamaño, para luego decodificarla y reconstruir la imagen original, pero con el ruido eliminado. A lo largo de este proceso, la red aprende a atenuar las variaciones aleatorias de píxeles causadas por el ruido, concentrándose en las características principales de la imagen.  La arquitectura estándar de un autoencoder consta de dos componente: el codificador y el decodificador. El codificador reduce la dimensión de la entrada y la transforma en un conjunto de características, mientras que el decodificador reconstruye la entrada original a partir de estas características. Al entrenar un autoencoder con imágenes libres de ruido, la red desarrolla la capacidad de identificar y preservar las características esenciales, lo que le permite luego eliminar el ruido de nuevas imágenes al no considerarlo relevante. |

**IMPLEMENTACIÓN (CÓDIGO)**

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  import pandas as pd  import tensorflow as tf  from sklearn.metrics import accuracy\_score, precision\_score, recall\_score  from sklearn.model\_selection import train\_test\_split  from tensorflow.keras import layers, losses  from tensorflow.keras.datasets import fashion\_mnist  from tensorflow.keras.models import Model  """# Resultado aproximado de nuestro Autoencoder  <image src="./salt1.png" alt="Sal y pimienta test1">  """  #data de fashion mnist  (x\_train, \_), (x\_test, \_) = fashion\_mnist.load\_data()  x\_train = x\_train.astype('float32') / 255.  x\_test = x\_test.astype('float32') / 255.  x\_train = x\_train[..., tf.newaxis]  x\_test = x\_test[..., tf.newaxis]  print(x\_train.shape)  """## Fn para agregar ruido sal y pimienta"""  def add\_salt\_n\_pepper(img, noise\_factor):  img\_out = tf.identity(img)  treshold = 1 - noise\_factor  noise = tf.random.uniform(shape=tf.shape(img), minval=0., maxval=1.)  #sal  img\_out = tf.where(noise < noise\_factor, 1.0, img\_out)  #pimienta  img\_out = tf.where(noise > treshold, 0.0, img\_out)  return img\_out  noise\_factor = 0.05  x\_train\_noisy = add\_salt\_n\_pepper(x\_train, noise\_factor)  x\_test\_noisy = add\_salt\_n\_pepper(x\_test, noise\_factor)  x\_train\_noisy = tf.clip\_by\_value(x\_train\_noisy, clip\_value\_min=0., clip\_value\_max=1.)  x\_test\_noisy = tf.clip\_by\_value(x\_test\_noisy, clip\_value\_min=0., clip\_value\_max=1.)  """### Look at imgs with noise"""  n = 10  plt.figure(figsize=(20, 2))  for i in range(n):  ax = plt.subplot(1, n, i + 1)  plt.title("original + noise")  plt.imshow(tf.squeeze(x\_test\_noisy[i]))  plt.gray()  plt.show()  """### We define a convolutional autoencoder"""  class Denoise(Model):  def \_\_init\_\_(self):  super(Denoise, self).\_\_init\_\_()  # encoding part  self.encoder = tf.keras.Sequential([  layers.Input(shape=(28, 28, 1)),  layers.Conv2D(16, (3, 3), activation='relu', padding='same', strides=2),  layers.Conv2D(8, (3, 3), activation='relu', padding='same', strides=2)])  # decoding part  self.decoder = tf.keras.Sequential([  layers.Conv2DTranspose(8, kernel\_size=3, strides=2, activation='relu', padding='same'),  layers.Conv2DTranspose(16, kernel\_size=3, strides=2, activation='relu', padding='same'),  layers.Conv2D(1, kernel\_size=(3, 3), activation='sigmoid', padding='same')])  def call(self, x):  encoded = self.encoder(x)  decoded = self.decoder(encoded)  return decoded  autoencoder = Denoise()  autoencoder.compile(optimizer='adam', loss=losses.MeanSquaredError())  # 3 epocas porque se quema mi compu  autoencoder.fit(x\_train\_noisy, x\_train,  epochs=10,  shuffle=True,  validation\_data=(x\_test\_noisy, x\_test))  encoded\_imgs = autoencoder.encoder(x\_test\_noisy).numpy()  decoded\_imgs = autoencoder.decoder(encoded\_imgs).numpy()  n = 10  plt.figure(figsize=(20, 4))  for i in range(n):  # display original + noise  ax = plt.subplot(2, n, i + 1)  plt.title("original + salt and pepper")  plt.imshow(tf.squeeze(x\_test\_noisy[i]))  plt.gray()  ax.get\_xaxis().set\_visible(False)  ax.get\_yaxis().set\_visible(False)  bx = plt.subplot(2, n, i + n + 1)  plt.title("reconstructed")  plt.imshow(tf.squeeze(decoded\_imgs[i]))  plt.gray()  bx.get\_xaxis().set\_visible(False)  bx.get\_yaxis().set\_visible(False)  plt.show() |

**RESULTADOS/CONCLUSIONES**

|  |
| --- |
| El proceso de creación y uso de los autoencoders demuestran ser una herramienta eficaz para la edición de imágenes y la eliminación de ruido en éstas. Permiten recrear las imágenes con una fidelidad impresionante incluso cuando ha sido corromida. |