FINAL TASK: DEEP LEARNING COURSE

GOAL:
Develop an encoder-decoder and build a scatter plot with the latent vectors generated. Use the MNIST dataset to develop this exercise.

STEPS:

- Remember to “normalize” the input data.
- Build a neural network with these characteristics:
  - ⇒ The input layer corresponds to the (normalized) MNIST images (Flattened).
  - ⇒ The first hidden layer will have 100 neurons and activation function “selu”.
  - ⇒ The mid-hidden layer (that corresponds to the latent space) will have 30 neurons.
  - ⇒ The third hidden layer will have 100 neurons again and activation function “selu”.
  - ⇒ The output layer will have 28*28 neurons.
  - ⇒ Add a Reshape layer at the end to recover the shape (28,28) of the MNIST images.
  - ⇒ Layers 2, 3, 4 and 5 will be Dense layers.

- Compile the model using a mae loss and Adam optimizer with learning rate of 0.001.
- Train the model with the training data, and 10 epochs. Remember to gather the fitting results to build the graph of train losses.
- Build the graph of train losses.
- Build a neural network similar to the encoder and copy the corresponding weights from the trained autoencoder.
- Compile the model (without parameters because we only want to predict not train it).
- Obtain the latent vectors by applying the predict() function of the encoder over the training set.
- Use the next couple of functions to reduce the dimensionality of the latent vectors from 30 to 2. This will allow the representation of the vectors in a bidimensional space (this action takes approximately 15 minutes):
  - ⇒ `tsne = TSNE()`
  - ⇒ `X_validas = tsne.fit_transform(<vectores_latentes_producidos_por_el_encoder>)`

- Build a scatter plot with the vector contained in X_validas (or whatever variable that you have used to gather the tsne.fit_transform() results). Use the parameter c=y_train to paint each sample class in a different color. One possible implementation could be:
  - ⇒ `scatter = plt.scatter(X_validas[:,0], X_validas[:,1], c=y_train, s=10, cmap="tab10")`
  - ⇒ `plt.legend(handles=scatter.legend_elements()[0], labels = ["0","1","2","3","4","5","6","7","8","9"])`
  - ⇒ `plt.show()`