# Convolutional Autoencoders In this Assignment, I will be using convolution layers to build your autoencoder. This usually leads to better results than dense networks and see it in action with the [EMNIST](https://www.tensorflow.org/datasets/catalog/emnist) dataset.

# **Prepare the Dataset**

**def map\_image(image, label):**

**'''Normalizes the image. Returns image as input and label.'''**

**image = tf.cast(image, dtype=tf.float32)**

**image = image / 255.0**

**return image, image**

**BATCH\_SIZE = 128**

**SHUFFLE\_BUFFER\_SIZE = 1024**

**train\_dataset = tfds.load('emnist', as\_supervised=True, split="train")**

**train\_dataset = train\_dataset.map(map\_image)**

**train\_dataset = train\_dataset.shuffle(SHUFFLE\_BUFFER\_SIZE).batch(BATCH\_SIZE).repeat()**

**test\_dataset = tfds.load('emnist', as\_supervised=True, split="test")**

**test\_dataset = test\_dataset.map(map\_image)**

**test\_dataset = test\_dataset.batch(BATCH\_SIZE).repeat()**

**Define the Model**

**# The encoder**

**def encoder(inputs):**

**'''Defines the encoder with two Conv2D and max pooling layers.'''**

**conv\_1 = tf.keras.layers.Conv2D(filters=64, kernel\_size=(3,3), activation='relu', padding='same')(inputs)**

**max\_pool\_1 = tf.keras.layers.MaxPooling2D(pool\_size=(2,2))(conv\_1)**

**conv\_2 = tf.keras.layers.Conv2D(filters=128, kernel\_size=(3,3), activation='relu', padding='same')(max\_pool\_1)**

**max\_pool\_2 = tf.keras.layers.MaxPooling2D(pool\_size=(2,2))(conv\_2)**

**return max\_pool\_2**

**# The bottleneck**

**def bottle\_neck(inputs):**

**bottle\_neck = tf.keras.layers.Conv2D(filters=256, kernel\_size=(3,3), activation='relu', padding='same')(inputs)**

**encoder\_visualization = tf.keras.layers.Conv2D(filters=1, kernel\_size=(3,3), activation='sigmoid', padding='same')(bottle\_neck)**

**return bottle\_neck, encoder\_visualization**

**#The decoder**

**def decoder(inputs):**

**conv\_1 = tf.keras.layers.Conv2D(filters=128, kernel\_size=(3,3), activation='relu', padding='same')(inputs)**

**up\_sample\_1 = tf.keras.layers.UpSampling2D(size=(2,2))(conv\_1)**

**conv\_2 = tf.keras.layers.Conv2D(filters=64, kernel\_size=(3,3), activation='relu', padding='same')(up\_sample\_1)**

**up\_sample\_2 = tf.keras.layers.UpSampling2D(size=(2,2))(conv\_2)**

**conv\_3 = tf.keras.layers.Conv2D(filters=1, kernel\_size=(3,3), activation='sigmoid', padding='same')(up\_sample\_2)**

**return conv\_3**

**# build the full Convolutional autoencoder functions.**

**def convolutional\_auto\_encoder():**

**inputs = tf.keras.layers.Input(shape=(28, 28, 1,))**

**encoder\_output = encoder(inputs)**

**bottleneck\_output, encoder\_visualization = bottle\_neck(encoder\_output)**

**decoder\_output = decoder(bottleneck\_output)**

**model = tf.keras.Model(inputs =inputs, outputs=decoder\_output)**

**encoder\_model = tf.keras.Model(inputs=inputs, outputs=encoder\_visualization)**

**return model, encoder\_model**