**Durgesh Vishwakarma**

MSc IT Sem4 2021-22 | **Deep Learning**Practical 10 (**PSIT4P3a**)

VPM’s B. N. Bandodkar College of Science, Thane.

PRN: 2015430016

==============================================================================

**Aim:**Denoising of images using autoencoders.

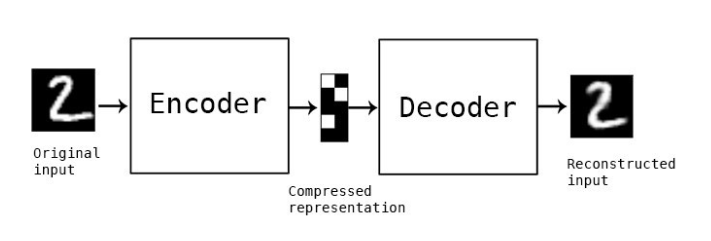
==============================================================================

# Table of Contents

* [What is an Autoencoder?](#_What_is_an)
* [Architecture of Autoencoder](#_Architecture_of_Autoencoder)
* [Implementation](#_Implementation:)
  1. [Importing libraries and dataset](#_Implementation:)
  2. [Adding Noise to MNIST Image dataset](#_2._Adding_Noise)
  3. [Building Autoencoder model using Keras](#_3._Building_Autoencoder)
  4. [Testing Autoencoder model](#_4._Testing_Autoencoder)
* [Conclusion](#_Conclusion:)

# What is an Autoencoder?

Autoencoder is an unsupervised artificial neural network that is trained to copy its input to output. Let’s consider that we are given an image, an autoencoder will first encode the image into a lower-dimensional representation, then decodes the representation back to the image.



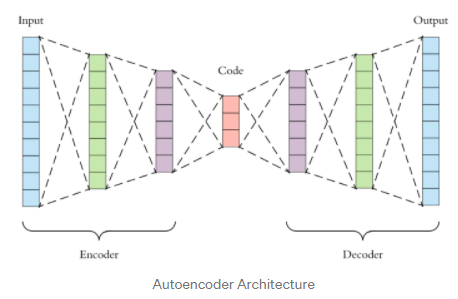
# Architecture of Autoencoder

There are mainly 3 parts in autoencoders

1. **Encoder**: In this part of the architecture the model compresses the input data to represent the compressed data in a reduced dimension.
2. **Code**: Also known as Bottleneck this part of the architecture represents the compressed data that is going to be fed to the decoder.
3. **Decoder**: This part reconstructs the encoded data as close to the input data as possible. The output from the decoder is a lossy reconstruction of the original data.

The goal of an autoencoder is to get an output that is identical to the input. The dimensionality of the input and output is similar as obviously the goal is to get the output as identical to the input we can get.

They are trained similarly to ANNs via backpropagation



# Implementation:

## 1. Importing libraries and dataset

First, we'll import all required libraries and MNIST image dataset.

|  |
| --- |
| #Importing libraries  from keras.layers import Input, Dense  from keras.models import Model  from keras.datasets import mnist  import keras.layers as layers  import numpy as np  import matplotlib.pyplot as plt  %matplotlib inline  (x\_train, \_), (x\_test, \_) = mnist.load\_data()  x\_train = x\_train.astype('float32') / 255.  x\_test = x\_test.astype('float32') / 255.  x\_train = np.reshape(x\_train, (len(x\_train), 28, 28, 1))  x\_test = np.reshape(x\_test, (len(x\_test), 28, 28, 1)) |

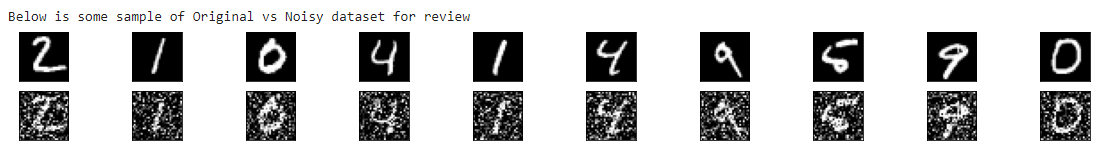
## 2. Adding Noise to MNIST Image dataset

We will add some noise to encode our original image into a noisy image dataset, which we'll send later as input to Autoencoders to decode or denoising it.

|  |
| --- |
| noise\_factor = 0.5  x\_train\_noisy = x\_train + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=x\_train.shape)  x\_test\_noisy = x\_test + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=x\_test.shape)  x\_train\_noisy = np.clip(x\_train\_noisy, 0., 1.)  x\_test\_noisy = np.clip(x\_test\_noisy, 0., 1.) |

|  |
| --- |
| print("Below is some sample of Original vs Noisy dataset for review")  n = 10  plt.figure(figsize=(20, 2))  for i in range(1, n + 1):      #Display original      ax = plt.subplot(2, n, i)      plt.imshow(x\_test[i].reshape(28, 28))      plt.gray()      ax.get\_xaxis().set\_visible(False)      ax.get\_yaxis().set\_visible(False)      #Display Encoded      ax = plt.subplot(2, n, i+n)      plt.imshow(x\_test\_noisy[i].reshape(28, 28))      plt.gray()      ax.get\_xaxis().set\_visible(False)      ax.get\_yaxis().set\_visible(False)  plt.show() |

**Output:**



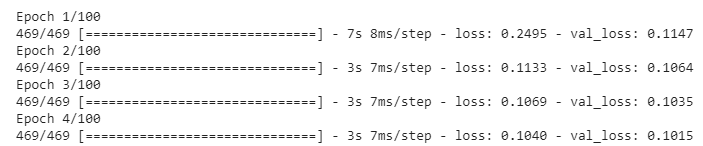
## 3. Building Autoencoder model using Keras

Here we will build Autoencoder model using Keras and train it with 100 epochs for better output. It may take few minutes to execute and produce output.

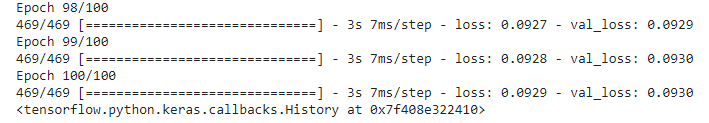
|  |
| --- |
| input\_img = Input(shape=(28, 28, 1))  x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(*input\_img*)  x = layers.MaxPooling2D((2, 2), padding='same')(x)  x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x)  **encoded** = layers.MaxPooling2D((2, 2), padding='same')(x)  x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(*encoded*)  x = layers.UpSampling2D((2, 2))(x)  x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x)  x = layers.UpSampling2D((2, 2))(x)  **decoded** = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)  autoencoder = Model(input\_img, decoded)  autoencoder.compile(optimizer='adam', loss='binary\_crossentropy') |

|  |
| --- |
| autoencoder.fit(x\_train\_noisy, x\_train, epochs=100, batch\_size=128, shuffle=True, validation\_data=(x\_test\_noisy, x\_test)) |

**Output:**



**….**



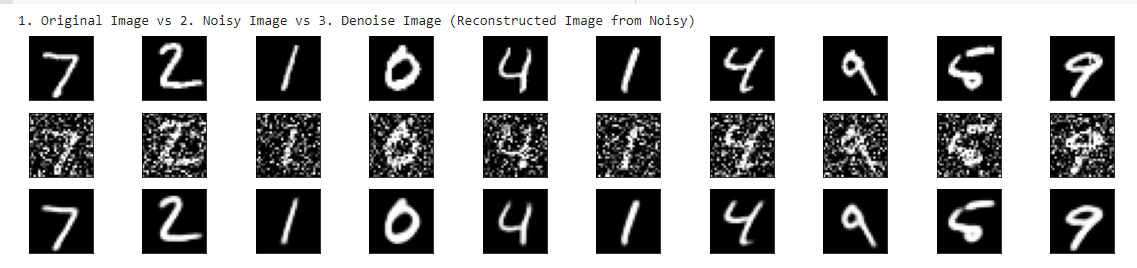
## 4. Testing Autoencoder model

As our Autoencoder model is ready now. We will compare below to test the model.

1. Original Image
2. Noisy Image
3. Denoise Image

|  |
| --- |
| decoded\_imgs = autoencoder.predict(x\_test)  n = 10  print("1. Original Image vs 2. Noisy Image vs 3. Denoise Image (Reconstructed from Noisy Image)")  plt.figure(figsize=(20, 4))  for i in range(n):      # display original      ax = plt.subplot(3, n, i + 1)      plt.imshow(x\_test[i].reshape(28, 28))      plt.gray()      ax.get\_xaxis().set\_visible(False)      ax.get\_yaxis().set\_visible(False)      #Display Encoded      ax = plt.subplot(3, n, i+1+n)      plt.imshow(x\_test\_noisy[i].reshape(28, 28))      plt.gray()      ax.get\_xaxis().set\_visible(False)      ax.get\_yaxis().set\_visible(False)      # display reconstruction      ax = plt.subplot(3, n, i+1+n+n)      plt.imshow(decoded\_imgs[i].reshape(28, 28))      plt.gray()      ax.get\_xaxis().set\_visible(False)      ax.get\_yaxis().set\_visible(False)  plt.show() |

**Final Output:**



# Conclusion:

* We have successfully performed following actions in Denoising of images using autoencoders.
  + Pulled Original image dataset from TensorFlow’s MNIST dataset.
  + Transformed Original Image into Noisy Image
  + Encoded Noisy Image into code using Keras’s autoencoders model.
  + Decoded Code to reconstruct Original Image using Keras’s autoencoders model.
* For more details please review PPT file of this.

