INT - 246
Soft Computing
Project Report

DECEMBER 5

Lovely Professional University
Authored by: Ankit Devri

Denoising Autoencoder

Introduction

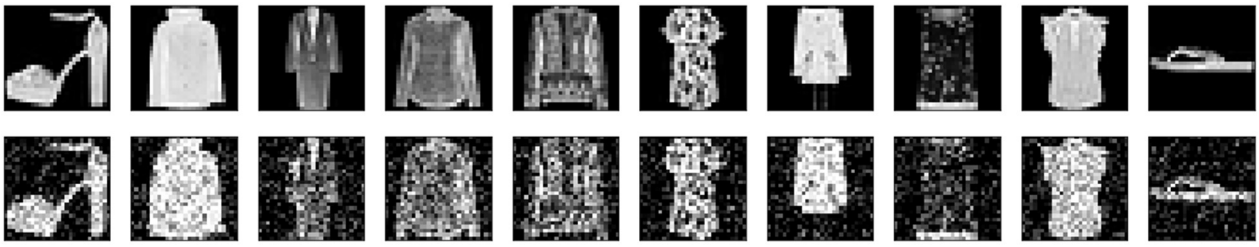
The denoising autoencoder is a type of autoencoder which removes noise from input to improve the quality of the data.

The Aim of the autoencoder is use black and white images and remove noise from them and produce clean image without noise.
The size of input data is 28 x 28 X 1 pixels and it generates the output of the same size filtered without noise.

Since the dataset consists of image data, the need of using a convolutional neural network was there. Thus, the denoising autoencoder was made using convolutional layers.

The dataset used in training the autoencoder is **MNIST** dataset and **FAISHON_MNIST** dataset a sample of the input with and without noise is given below.
The autoencoder model was coded in python with the help of **KERAS API** (running with **TensorFlow** at the backend), **NUMPY**, **MATPLOTLIB** libraries on **Jupyter Notebook.**

Clear Images



Noisy Images

The aim is to filter the noisy image given above to a clear image.

Student Contributions

Group - 27

Name	Contribution
Ankit Devri (Reg no. 11914065)	Coding Report Writing
Rudra Pratap (Reg no. 11903438)	
Siddharth (Reg no. 11906132)	

Auto Encoders

Introduction

An autoencoder is a type of artificial neural network used to learn efficient encodings of unlabeled data (unsupervised learning) through the encoder. The encoding is then processed by a decoder attempting to regenerate the input from the encoding.
The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data ("noise").

In simple terms, an autoencoder is a type of artificial neural network that encodes the input data and then modify it while decoding it to its original form to get the required output.

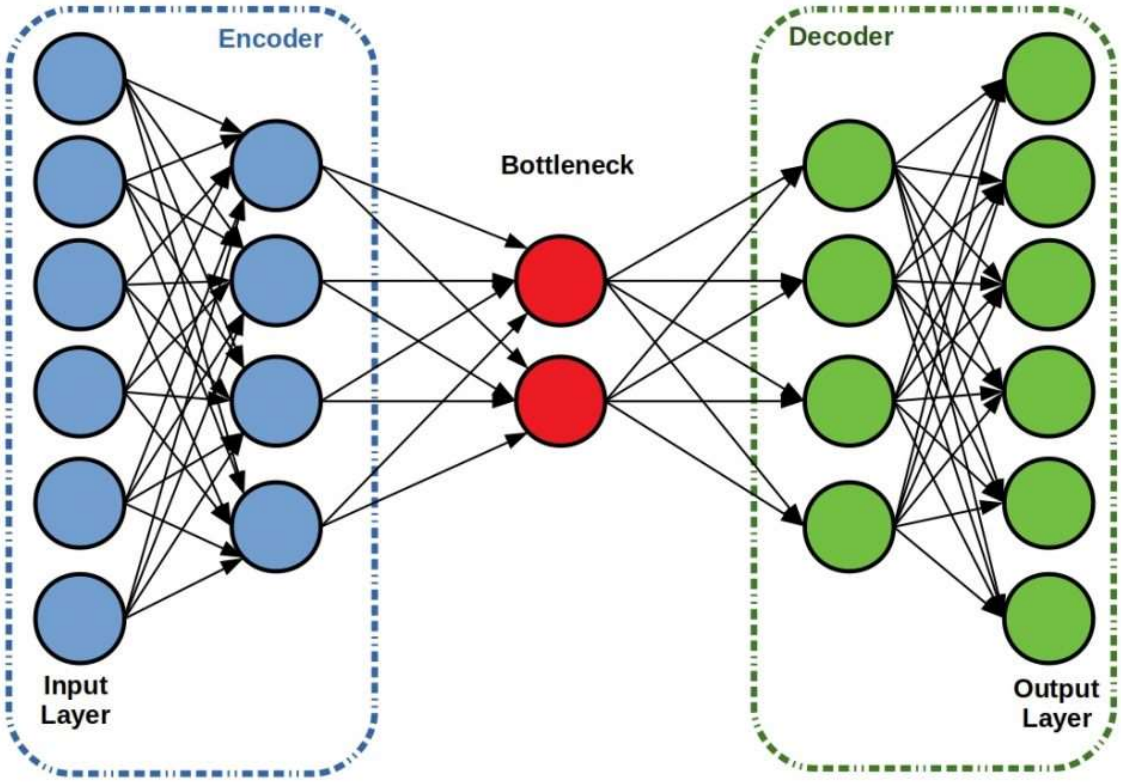
It's a simple example of unsupervised learning where the encodings generated over an input dataset are learned without specific supervision.

While an auto encoder supports a lot of functions such as noise reduction, reconstruction, dimensionality reduction, anomaly detection, image processing and much more. This time the project is based on the noise reducing function of autoencoder.

Its architecture is divided into three separate components:

1. **An Encoder**
2. **The latent Space (Bottleneck)**
3. **The Decoder**

Architecture of Autoencoder



The three major components of an Autoencoder as shown above are

1. **ENCODER:** The main function of this encoder is to extract out the core characteristics of the image and encode them into the latent space while reducing the dimensions of the image, this is where noise is filtered out.

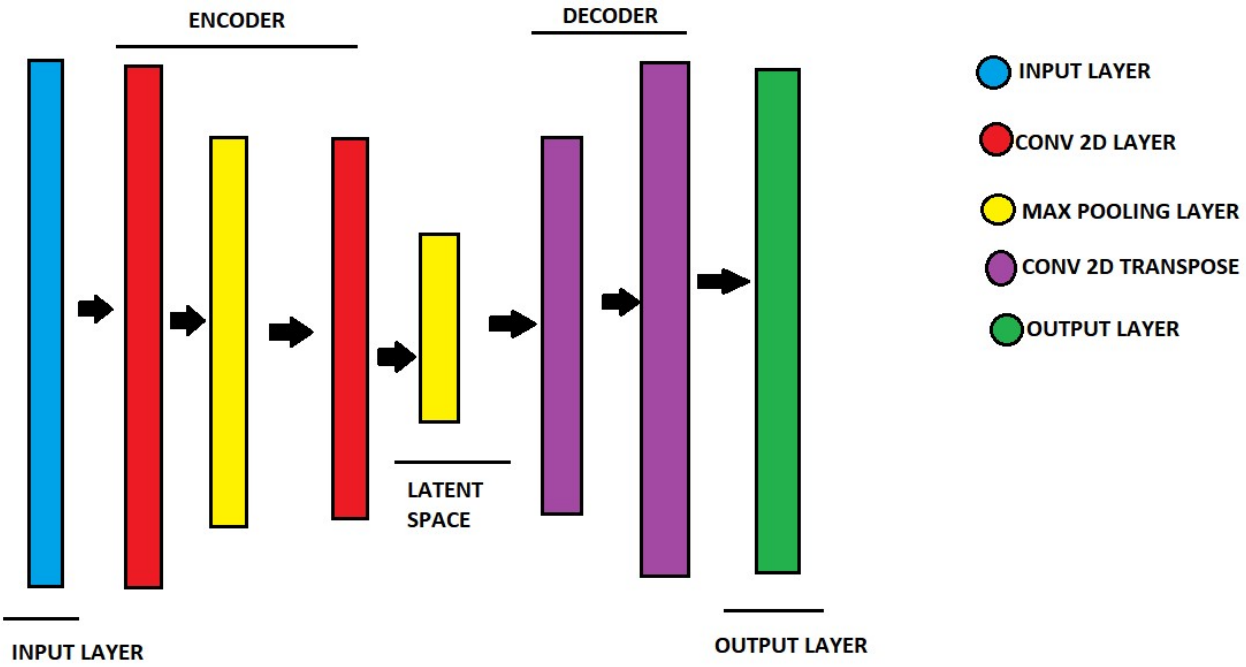
2. **BOTTLENECK (Latent Space):** The latent space or the bottleneck is space where the encoded values from the encoder are stored. This is the point from where the reconstruction of image begins.

3. **DECODER:** The function of the decoder is to capture the core characteristics from image and then it does the opposite of encoder it enlarges the dimension of the image, it's the mirror opposite of encoder, thus the output generated has the same size of the input image.

Architecture used in Denoising Autoencoder

Input Layer

The input layer accepts the input with dimensions 28 x 28 x 1 in form of **NUMPY** array



Architecture of the Encoder

The Encoder consist of two sublayers, each sublayer includes a **Convolutional** layer and a **Max Pooling** layer. Here Conv2D layer of **Keras API** is used for image dataset.
Each Convolutional layer has about 32 filters each having size of 4 x 4 (with padding applying as same).
Each Max Pooling layer has size of 2 x 2 (with padding applying as same).

There are 2 sets of above layers in the encoder.
The first set output reduces dimension from 28 x 28 x32 to 14 x 14 x 32
While the second set reduces the dimension from 14 x 14 x32 to 7 x 7 x 32.

Latent Space dimension hence becomes **7 x 7 x 32.**

Architecture of the Decoder

The decoder consists of only 2 layers opposite to the 2 sublayers of the encoder. The layers used in decoder are transposed convolutional layers which help in upsampling the core data in the latent Space using strides of 2 it undoes the compression operation with respect to each convolution layer and max pooling layer. The first set output increase dimension from 7 x 7 x 32 to 14 x 14 x 32 While the second set increase the dimension from 14 x 14 x 32 to 28 x 28 x 32.

NEURAL NETWORK PSEUDO CODE

```
# Training Model

# Input
input = layers.Input(shape=(28, 28, 1))

# Encoder
x = layers.Conv2D(32, (4, 4), activation="relu", padding="same")(input)
x = layers.MaxPooling2D((2, 2), padding="same")(x)
x = layers.Conv2D(32, (4, 4), activation="relu", padding="same")(x)
x = layers.MaxPooling2D((2, 2), padding="same")(x)


# Decoder
x = layers.Conv2DTranspose(32, (4, 4), strides=2, activation="relu",
padding="same")(x)
x = layers.Conv2DTranspose(32, (4, 4), strides=2, activation="relu",
padding="same")(x)

# Output
x = layers.Conv2D(1, (4, 4), activation="sigmoid", padding="same")(x)

# Autoencoder
autoencoder = Model(input, x)
autoencoder.compile(optimizer="adam", loss="binary_crossentropy")
autoencoder.summary()
```

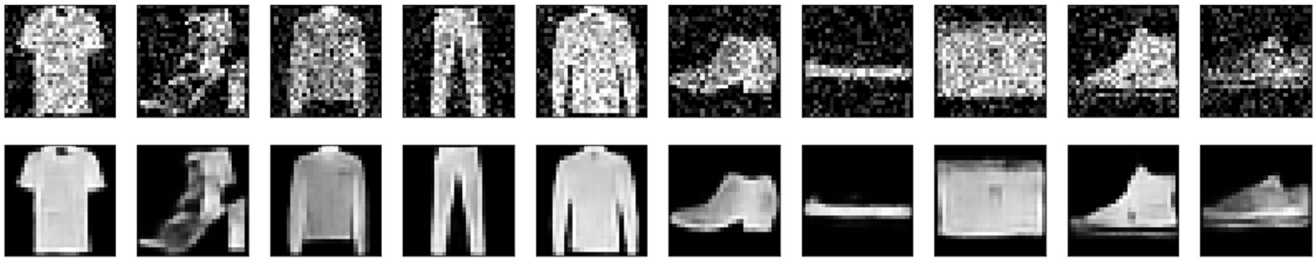
This is the code of the training model used for the autoencoder.

Results and Observations

The Autoencoder was trained with 60000 sample images from the **MNIST** dataset and the 60000 sample images from the **FASHION MNIST** dataset, with batch size of 128 images and 65 epochs.
The loss function used was **binary cross entropy** and optimizer used was **Adam**.

After Training with only **MNIST** dataset:
The same autoencoder was up to 93.12% accurate in results

After Training with only **FAISHON MNIST** dataset:
The same autoencoder was up to 74.25% accurate in results



After Training with both **MNIST** and **FAISHON MNIST** dataset combined:
The same autoencoder was up to 74% accurate in results