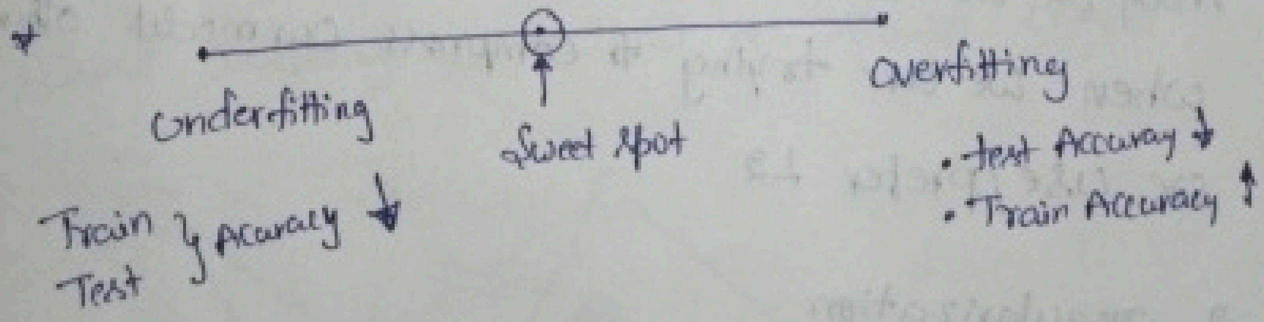
UNIT-V

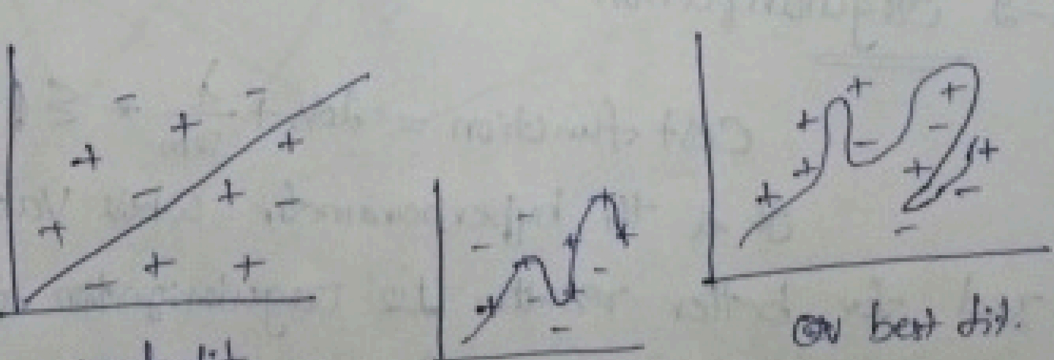
Regularization and Autoencoders

Regularization for Deep learning :-

Regularization is one of the most important concept of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

* Regularization in deep learning methods include L1 and L2 regularization, dropout, early stopping, and more.
* By applying regularization, models become more robust and better at making accurate predictions on unseen data.





Underfit

Overfit

Best fit.

'+' → Train data
'-' → Test data.

L1 - Regularization & L2 - Regularization

L_1 and L_2 are most Common types of regularization. These update the general cost function by adding another term known as regularization term

* The Values of weight matrices decrease because it assumes That a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce over-fitting to quite an extent.

L-1 regularization

(λ (0.01) - is a regularization parameter)

$$\text{cost function} = \text{loss} + \frac{\lambda}{2m} * \Sigma \|w\|$$

In This, The absolute Value of The weights. The weights may be reduced to zero here Hence, it is Very useful when we are trying to Compress our model. otherwise we use/prefer L2

L-2 regularization

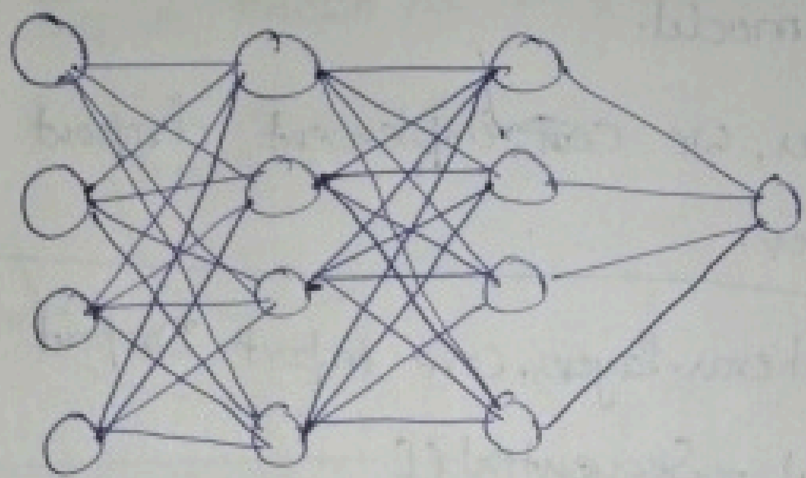
$$\text{Cost function} = \text{loss} + \frac{\lambda}{2m} * \Sigma \|w\|^2$$

It is the hyperparameter whose Value is optimized for better results. L2 regularization is also known as weight decay. as it forces the weights to decay towards zero (but not exactly zero) Ex: 0.01, 0.0001 etc.

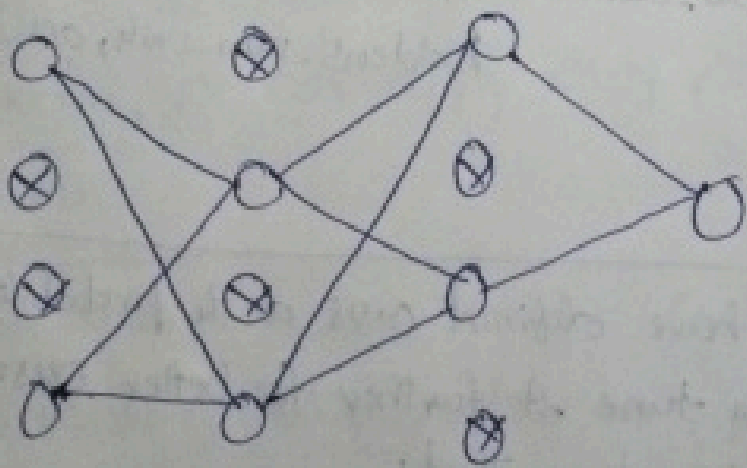
Dropout

* It is one of the important regularization technique. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.

Example



* At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below



So each iteration has a different set of nodes and this results in a different set of outputs. It can also be thought of as an ensemble technique in machine learning.

* Ensemble models usually perform better than a single model as they capture more randomness. Similarly dropout also performs better than a normal neural network model.

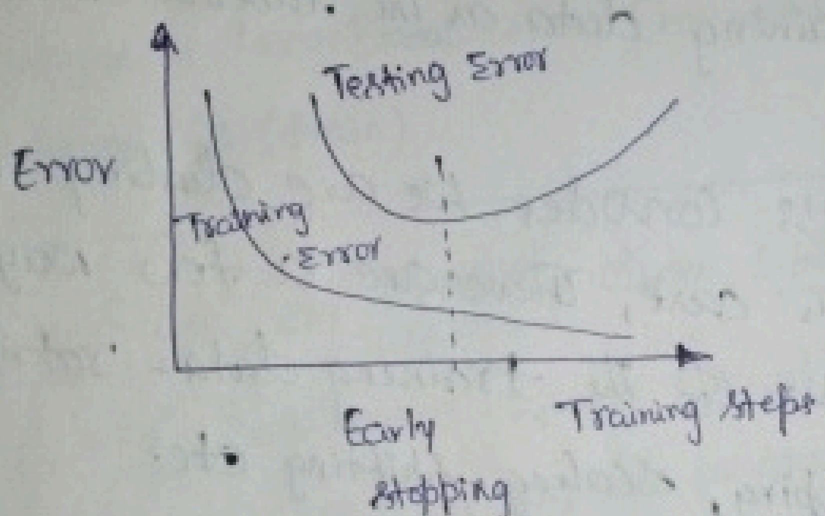
* In Keras, we can implement dropout using the Keras core layer.

```
from Keras.layers.core import Dropout
model = Sequential([
Dense(output_dim = hidden1_num_units, input_dim = input
                    _num_units, activation = 'relu'),

Dropout(0.25),

Dense(output_dim = output_num_units, input_dim =
        hidden5_num_units, activation = 'Softmax'),

])
```

We have defined 0.25 as the probability of dropping. We can tune it further for better results using the grid search method.

Early Stopping

* Early stopping is a kind of Cross Validation strategy where we keep one part of the training set as the validation set.
* When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.



* In the above image, we will stop training at the dotted line since after that our model will start over-fitting on the training data.
* In keras, we can apply early stopping using the callbacks function.

```
from keras.callbacks import EarlyStopping
EarlyStopping (monitor = 'val_err', patience = 5)
```

"monitor" denotes the quantity that needs to be monitored
"val_err" denotes the validation error.

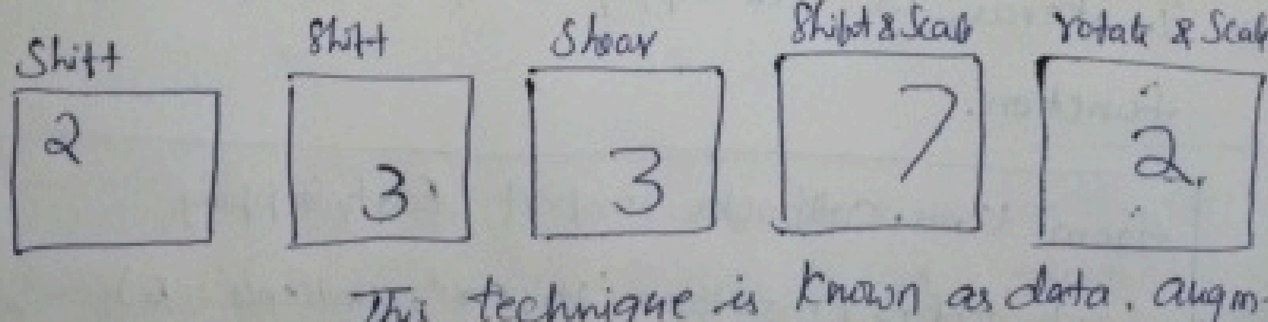
'patience' denotes the no. of epochs with no further improvement after which the training will be stopped.

Data Augmentation

* The simplest way to reduce overfitting is to increase the size of the training data.
* In machine learning, we were not able to increase the size of training data as the labeled data was too costly
* But, now let's consider we are dealing with images. In this case, there are a few ways of increasing the size of the training data - rotating the image, flipping, scaling, shifting etc.

Example

Some tranformation has been done on the Hand-written digit dataset



This technique is known as data augmentation.

* This usually provides a big leap in improving the accuracy of the model. It can be considered as a mandatory trick in order to improve our prediction

* In Keras, we can perform all of these transformations using

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(horizontal_flip=True)
datagen.fit(train)
```

Case Study on MNIST data with Keras

(Implement Simple Neural Networks for Hand written Images) → lab program.

Auto Encoders

We have seen the CNN/RNN architectures and the applications
they can be tagged to supervised learning.

* Shall we learn unsupervised learning technique.
"Auto Encoders".

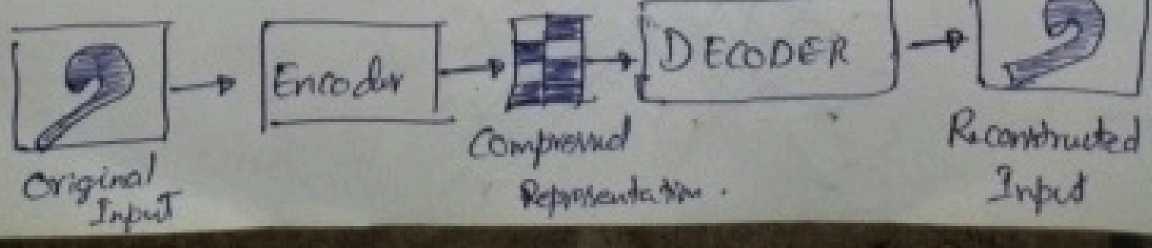
=> No Explicit labels required to train the model
=> Raw input is sufficient for the training

* An auto encoder is a simple ML algorithm which acquire input image and it will reconstruct the same. i.e the image is compressed. This is also called as dimensionality reduction.

* The dimensionality reduction certainly is used in the data preprocessing (reduce / compress)

* The process of dimensionality reduction reduce the dimensionality of the considered dataset.

* Auto encoders are meant for this Dimensionality Reduction.



* PCA - Principal Component Analysis (PCA). They do dim-
-sionality reduction. It is done with linear Transforma...
But, Auto encoders are using Non-linear transformati...

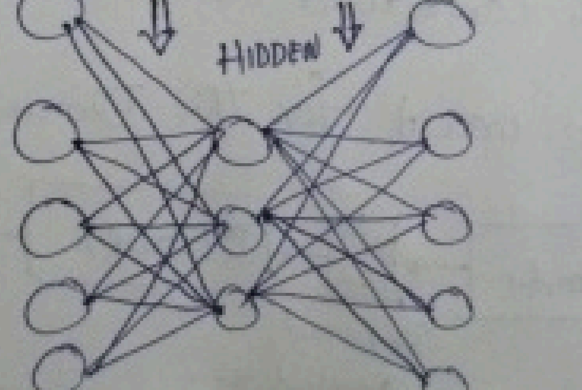
* Auto encoders are feedforward network (Remember)

* Auto encoders has a relatively simpler architecture and layers. Actually speaking, it has Three layers. But, we never count the input layer into account, we call auto encoders as two layered.

- Input layer
- Hidden layer (bottleneck layer)
- Output layer

* The output should always be the same as input.

Example: I/P → 11110000
O/P → 11110000



It has two layers. It has one hidden layer as you could see.

* An instance for a quicker reference. Assume we feed an image with 8 pixel values as input to the Auto encoder

* This is now compressed by the encoder, to 5 pixel at the hidden layer i.e bottleneck layer (or) middle layer.

* What are the things required to build or construct an auto encoder!

* Simple
  - → An encoder (Encoding method)
  - → A Decoder (Decoding method)
  - → A loss function (To compare the output with target, we need this)

Properties / features of Auto encoders

* Data Specific Behaviour
* Lossy Compression nature
* Unsupervised in nature.

Data Specific: This can work only on the data which are similar to what the system is already trained on.

For instanc, if an autoencoder is trained on Comp
cat images may not work fine with donkey images.

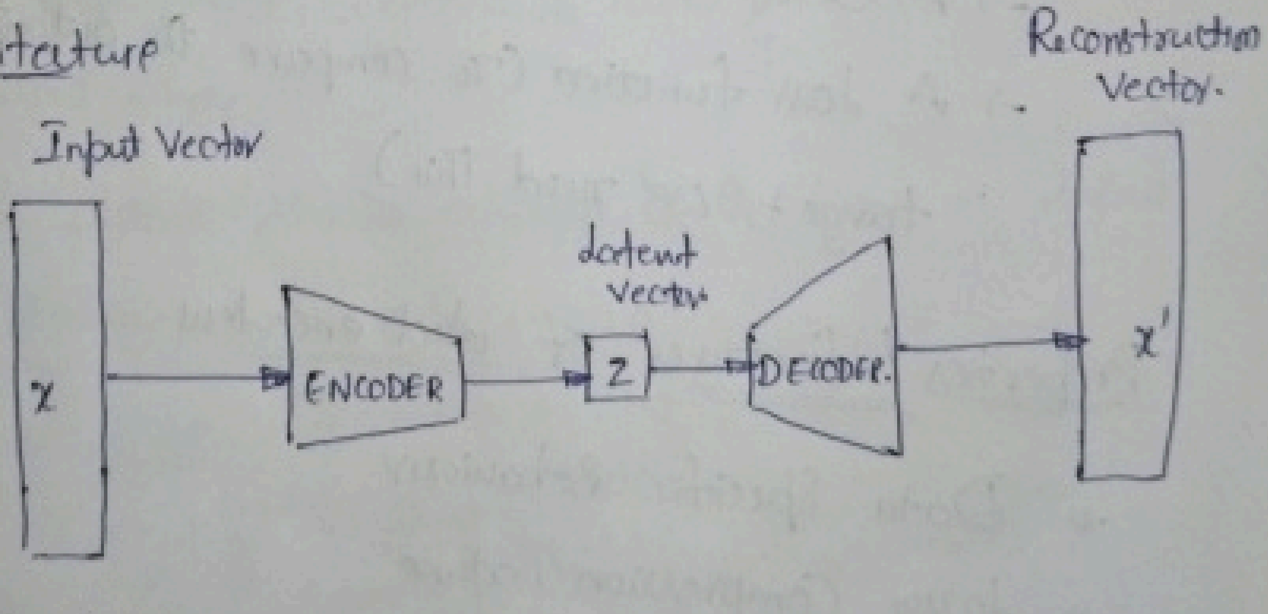
Lossy Compression nature

The Expectation may not always happen. Same is the case with Auto encoders. The output may not be exact as input, But it will be a very closer ones.

Unsupervised / Self-Supervised :

We can call This Unsupervised as we need not do anything other than feeding the raw input. No explicit labeling required.

Architecture



Applications

* Dimensionality Reduction
* Anomaly Detection (IOT, Sensors, Images & others)
* Image Processing. * Denoising I/P * Dealing with RawData

Types of Autoencoders

* Under Complete Autoencoders
* Sparse Autoencoders
* Denoising Autoencoders
* Variational Autoencoders.

Sparse Autoencoders

* Sparse autoencoders are controlled by changing the no. of nodes at each hidden layer
* These are work by penalizing the activation of some neurons in hidden layers
* It means That a penality directly proportional to the no. of neurons activated is applied to the loss function.
* Using L1 or L-2 Regularization.

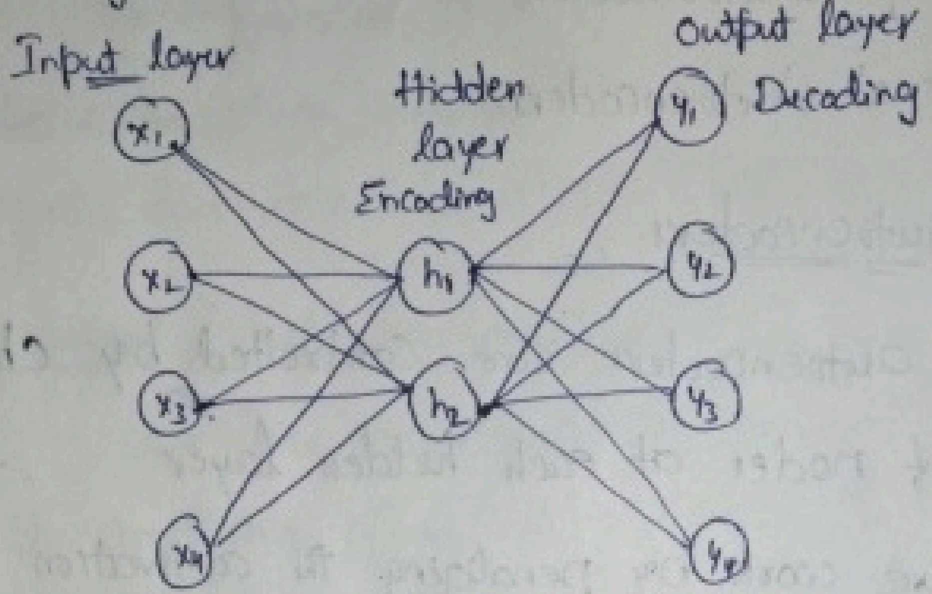
$$\text{Cost function} = (y-\hat{y}) + \lambda \sum_i (a^{h(i)})$$

λ - 0.01

a - activation.

$h(i) \rightarrow$ hidden neuron.

* Sparse Autoencoders are important because they can learn useful features from unlabeled data, which can be used for tasks such as anomaly detection, denoising, and dimensionality reduction.

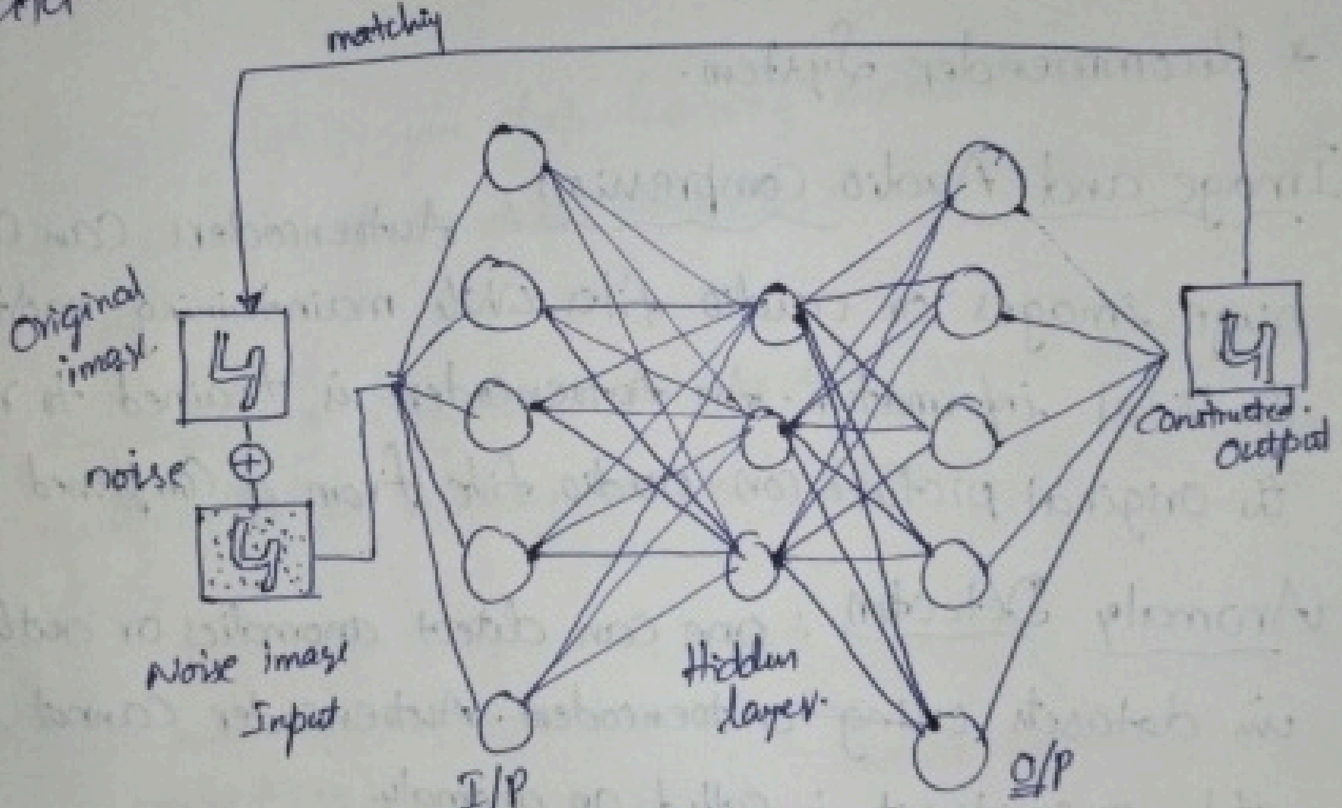


Denoising Autoencoders

* These are similar to regular autoencoders in that they take an input and produce an output. However, they differ because they don't have the input image as their ground truth. Instead, they use a noisy version.

* Denoising autoencoders can learn more robust features compared to standard autoencoders. The approach is unsupervised, which means it does not require labeled data for training.

The training objective of a denoising autoencoder involves minimizing the difference between the clean input and the reconstructed output. Common loss function for This task include mean squared error (MSE) or binary cross-entropy, depending on the nature of the data.



* A Denoising Autoencoders is trained by minimizing This loss Through the use of backpropagation, which involves updating the weights of both encoder and decoder components.

* Applications of Denoising Autoencoders span a Variety of domains, including Computer Vision, Speech Processing, and Natural language Processing.

Applications of Autoencoders

* Image and Audio Compression
* Anomaly Detection
* Dimensionality Reduction
* Data Generation
* Denoising
* Recommender System.

Image and Audio Compression :- Autoencoders can compress huge images or audio files while maintaining most of the vital information. An Autoencoder is trained to recover the original picture (or) audio file from a compressed representation.

Anomaly Detection :- one can detect anomalies or outliers in datasets using autoencoders. Autoencoder cannot accurately reconstruct is called an anomaly.

Dimensionality Reduction :- lower the dimensionality of high dimensional linear (or) non linear dataset

Data Generation :- Employ autoencoders to generate new data similar to the training data.

Denoising :- One can utilize autoencoders to reduce noise from data

Recommender System :- we can use user's preferences to generate personalized suggestions.