Date
Page 1

# Deep Learning

## Unit 04: Autoencoders

### NOV-DEC 2022

Q3 a) Autoencoders are unsupervised learning approach. Justify the statement.

Ans. Autoencoders do not require any external labels such as class names (cat, dog, etc) during training.

- They learn only from the input data itself
- No target categories or labels are provided.
- They try to reconstruct the input by learning hidden patterns automatically.

Input (X) → Autoencoder → Output (X')

There is no labeled output like "This is digit 7."

The model simply learns the structure of the data.

Because they learn patterns without human-provided ~~labels~~ labels, they follow unsupervised learning.

- No labeled data - only raw input data is needed
- Learns ~~patterns~~ features automatically - finds patterns without supervision
- No class prediction - task is reconstruction, not classification
- self-learning - input acts as both input and target

e.g. Training autoencoder on 10,000 face images:

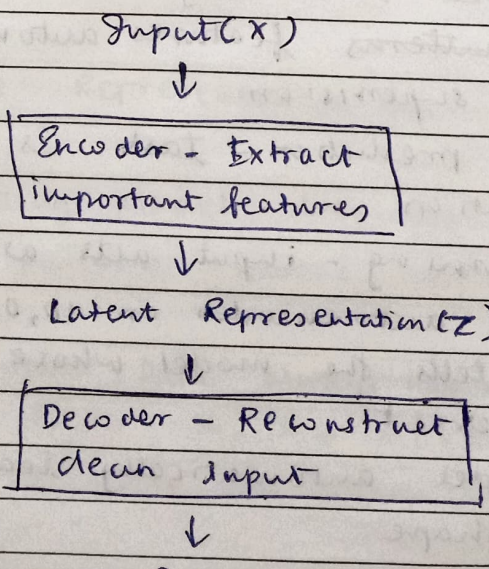
- No one tells the model whose face it is or what features exist.
- The model automatically learns useful features like
  - Eyes shape
  - Nose position

Date
Page

. Face outline
. No manual labels → unsupervised.

Q 3 b) Explain the concept of contractive autoencoder and its need.

Ans A contractive Autoencoder is a type of regularized autoencoder that is designed to learn features that do not change much when the input changes slightly.
. Even if the input image has small noise or it is slightly rotated or blurred, the model should still understand it as the same object.
So, contractive autoencoder focuses on making the latent representation stable.
. It is called contractive because it contracts (shrinks) the feature space around the data points.
Meaning:
. Similar inputs → very close representations in latent space
. Noise or tiny changes should not cause large changes in learned features
. Architecture



- Same structure as normal autoencoder, but during training extra penalty is added.

* Working:

1. Input data is passed into the Encoder
2. Encoder creates a latent representation (z)
3. Decoder reconstructs the original input
4. During training, the model calculates two losses:
   a) Reconstruction Loss: Make output close to input
   b) Contractive Penalty: Make z stable & less sensitive to changes

The contractive penalty is applied to keep the latent features from changing too much with tiny input changes.

* Loss Function

$$\text{Total Loss} = \underbrace{\|x - x'\|^2}_{\text{Reconstruction loss}} + \underbrace{\lambda \left\| \frac{\partial z}{\partial x} \right\|^2}_{\text{Contractive Penalty}}$$

Where:

- $\frac{\partial z}{\partial x}$ = sensitivity of latent space to input.
- If sensitivity is high → penalty increases. Forces the model to not change z much, even if x changes slightly

* Need

- To handle noise: Representation remains same even if input is noisy
- To avoid overfitting: Model does not memorise input
- To learn stable patterns: Latent space features do not change easily
- For real-world data: Data always has small variation
- For classification tasks: Similar images → similar codes
- It generalizes better to new unseen data

- It gives smooth, robust, and clean feature learning.

e.g. Let's say we train on images of digit 5. e.g.

If input image is slightly:

- rotated
- blurred
- has small noise

2.

Normal autoencoder →

- Might change the latent features a lot e.g.
- May think it's a different number

Contractive autoencoder →

- Latent representation stays almost same. 3.
- Model still understands it is 5

This is why it is very useful in

- Handwriting recognition
- Speech recognition
- Anomaly detection

* Advantages 4.

- Learns more robust and smooth features
- Reduces overfitting e.g.
- Good for data with noise and variations
- Helps classification after feature learning
- Better generalization to new samples. 5.

Q4a) State the applications of Autoencoders. Explain how the dimensionality feature of autoencoder is useful in information retrieval task. 6.

Ans. Autoencoders are useful in many machine learning and deep learning tasks because they learn compressed and meaningful representations of data. They help in denoising, feature extraction, anomaly detection, and generation. 7.

1. Dimensionality Reduction: Autoencoders compress high-dimensional

data into fewer features.

e.g. Compressing 784-pixel MNIST images into a vector of 32 features

Better than PCA because autoencoders learn non-linear patterns.

2. Image Denoising: Denoising autoencoders remove noise from corrupted images.

e.g. Blurry or noisy photos → cleaned and restored images

Used in: camera enhancement, medical imaging (MRI, CT scan)

3. Anomaly Detection / Outlier Detection

Autoencoders learn only normal patterns of data.

- If abnormal data is given → reconstruction error is high
- Detect anomalies easily
- Used in: fraud detection in banking, industrial defect detection, network intrusion detection.

4. Data compression / storage: Autoencoders compress data into small latent vectors.

e.g. Face images compressed for storing in low memory devices. Used in: image compression systems, cloud storage optimization.

5. Feature Extraction for Machine Learning: Latent space features are more meaningful than raw data. Used for: Image recognition, text categorization, speech classification. Auto-encoders work as a pre-training step to improve accuracy.

6. Image Generation (Generative Models): Variational Autoencoders (VAEs) can create new synthetic images. e.g. Generate new faces that do not belong to any real person, Used in creative design and gaming.

7. Recommender systems: Autoencoders learn user behaviour and preferences. e.g. Movie and product recommendations (like Netflix, Amazon)

8. Speech and Audio Denoising: Removes background noise and enhances voice clarity. e.g. Call noise cancellation, hearing aid devices

9. Medical Applications: Used to detect diseases and reconstruct medical scans. e.g. ~~To~~ Tumor detection, MRI reconstruction

* How Dimensionality Reduction Helps in Information Retrieval

Autoencoders compress large data into a small latent vector (bottleneck representation). The representation contains only important features.

e.g. An image with 784 features is reduced to 32 features

- Still keeps the essential structure
- Removes noise and unwanted details
- Data becomes easy to store, search, and compare.
- Faster Searching: Only small feature vectors are compared instead of large data.
- Reduced storage: Database stores compressed features instead of full files.
- Better accuracy: Latent space keeps only meaningful patterns
- Handles noise: Retrieval is robust against small variations.

e.g. Image search system

1. A database stores latent feature vectors of 10,000 images
2. A user uploads a query image
3. Autoencoder converts ~~it~~ it into latent space.
4. System compares compressed vectors
5. Finds most similar images quickly

.. Retrieval is faster, efficient, and more accurate

Q.4 b) Explain denoising autoencoders with suitable figure

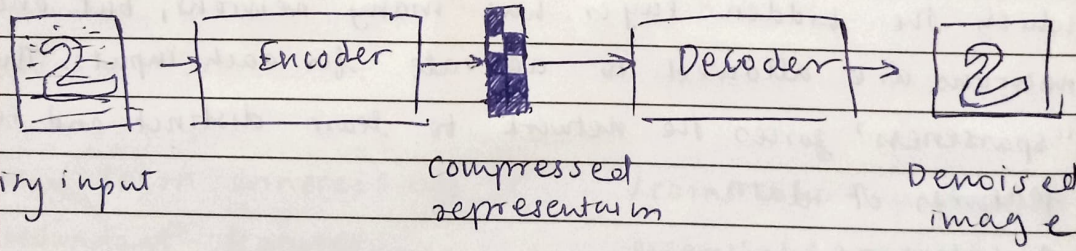
Ans. A Denoising Autoencoder (DAE) is a type of autoencoder that is trained to remove noise from the input data. Instead of learning to copy the input, it learns to reconstruct the clean/original data from a corrupted/noisy ~~version~~ version.

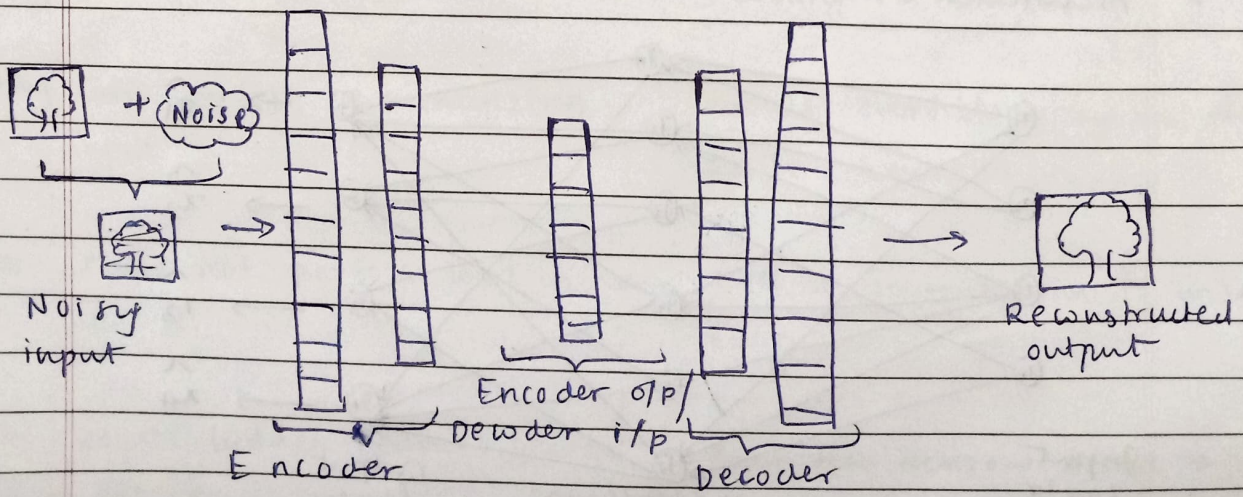
- Add noise to input → Encoder extracts useful features → Decoder reconstructs noise-free output

This makes the model robust, helps in better feature learning and improves generalization.

* Architecture / Working

1. Take original input (clean data): e.g. an image without noise
2. Add noises to input: (Gaussian noise, salt & pepper noise, dropout, etc.)
3. Encoder takes the noisy input and converts ~~it~~ it into a meaningful latent representation
4. Decoder reconstructs the clean version of the input
5. Model is trained to minimize the difference between:
   - Original clean input
   - Reconstructed output.



* Why Denoising?
- Learns more useful features
- Becomes robust to noise and distortions
- Improves accuracy in downstream tasks like classification
- Works better than simple autoencoder for real-world data

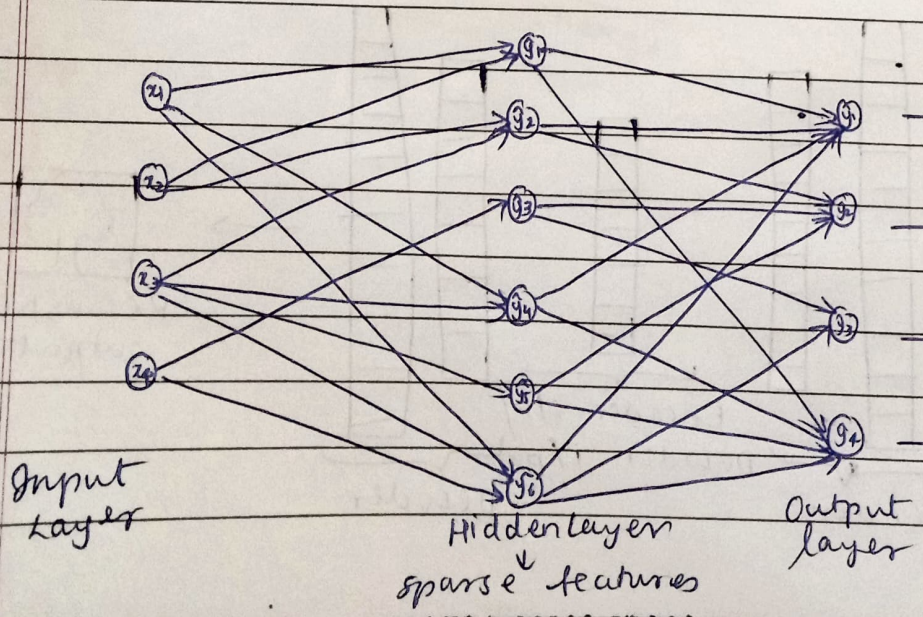
* Applications
- Image Processing: Removing camera noise, medical image denoising
- Audio Processing: Noise-free sound reconstruction
- Cybersecurity: Anomaly/fraud detection
- NLP: Remove spelling errors/noisy text
- Data Compression: Noise-invariant compression

May-Jun 2023

Q. 3a) Explain the architecture of sparse autoencoder with suitable diagram. What are the advantages of sparse encoder over usual autoencoder?

Ans. A sparse autoencoder is a special type of autoencoder in which the hidden layer has many neurons, but only a few neurons are allowed to activate for each input. This "sparseness" forces the network to learn distinct and important features of data.

* Architecture



sparsity constraint (Contractive

Even though hidden layer has more neurons, the network applies sparsity constraint so:

- Only a small number of neurons are active (value = 1)
- Most neurons remain inactive (value = 0)

This is done using:

Regularization like L1 or KL Divergence penalty

. Encouraging neurons to fire rarely.

* Working

1. Input is passed to encoder
2. Encoder produces hidden representation but with sparse activation
3. Decoder tries to reconstruct the original input.
4. Loss = Reconstruction error + sparsity penalty

$$L = \|x - x'\|^2 + \beta \cdot (\text{sparsity penalty})$$

e.g. Input size = 10

Hidden layer neurons = 50

But only 2-3 neurons become active for any input → learns useful patterns

* Advantages of Sparse Autoencoder over Usual Autoencoder

| Feature/Point | Usual Autoencoder | Sparse Autoencoder (Advantage) |
|---|---|---|
| Feature learning | May learn unnecessary or redundant features. | Learns only important and meaningful features |
| Overfitting | Can overfit by memorizing input | Avoids overfitting by limiting active neurons |
| Generalization | Might not perform well on new data | Better generalization to unseen data |
| Neuron Activation | Most hidden neurons become active | Only few neurons active → Sparse representation |

| | | |
|---|---|---|
| Interpretability | Features are less interpretable | Features are more clear and understandable |
| Pattern Detection | Struggles to detect rare patterns | Detects rare and useful hidden patterns |
| Representational Power | Not always efficient | Efficient and powerful feature representation |

Q.4 a) Explain the structure of regularized autoencoders. What is the purpose of sparsity constraint in sparse autoencoders?

Ans. A Regularized Autoencoder is an autoencoder in which we add extra constraints or penalties during training so that the encoder learns better features and the model does not simply memorize the input.

Regularized autoencoders solve a major problem of basic autoencoders: Basic autoencoders may just copy input → output without learning useful features.

To avoid this, we add regularization (extra rules) to force the model to learn meaningful patterns.

* General Structure:

A regularized autoencoder has three main parts:

1. Encoder
   - Takes high-dimensional input data
   - Applies neural network layers
   - Reduces it to a compressed / meaningful form
   - Learns useful patterns instead of simply copying.
2. Latent Space (Hidden / Bottleneck Layer)
   - Contains encoded or compressed information
   - Here, regularization constraints are applied such as:
     - Sparsity constraint (Sparse Autoencoder)
     - Noise constraint (Denoising Autoencoder)

. Sensitivity constraint (Contractive Autoencoder)

3. Decoder

Takes latent representation

- Tries to reconstruct the original input
- Learns to rebuild the important features ~~retr~~ retained by the encoder

- Regularized autoencoders have the same basic structure as a normal autoencoder:

Input → Encoder → Latent space → Decoder → Output

2 Loss function

Regularized autoencoders modify the normal reconstruction loss to include penalty terms.

Total Loss = $\underbrace{||X - X'||^2}_{\text{Reconstruction loss}}$ + $\underbrace{\text{Regularization Penalty}}_{\text{Constraint}}$

Examples of penalties:

- Sparse Autoencoder → Sparsity penalty (KL divergence)
- Denoising Autoencoder → Noise removal loss
- Contractive Autoencoder → Jacobian penalty

Regularization forces the auto encoder to learn robust, general, & useful features.

1 Types of Regularized Autoencoders

1. Sparse Autoencoder: forces only a few neurons to be active.
2. Denoising Autoencoder: Adds noise to input and learns to reconstruct data
3. Contractive Autoencoder: makes latent space stable to small changes in input.
4. Variable Autoencoder (VAE): Adds probabilistic constraints

Purpose of Sparsity Constraint in Sparse Autoencoder

The sparsity constraint forces the autoencoder to activate

only a small number of neurons in the hidden layer for any given input.

This is done by adding a penalty term to the loss function so that:

- If a neuron fires too often → it gets penalized
- So it learns to stay OFF most of the time
- Only when a specific feature appears → that neuron activates

e.g.

* Why sparsity?

1. Prevents the model from copying

If all hidden neurons activate:

- The network may simply "copy" the input
- This gives no useful feature learning

Sparsity stops this by allowing only a few neurons to fire.

2. Each neuron learns a unique feature

Because only a few neurons activate:

- Neuron 1 may learn vertical edges
- Neuron 2 may learn horizontal edges
- Neuron 3 may learn curves
- Neuron 4 may learn diagonal patterns

Thus, the network becomes a feature detector.

3. Reduces overfitting

With sparse activation:

- Network becomes simpler internally
- It avoids memorizing the training data
- Generalizes better to new data

4. Produces meaningful latent representations

Sparse representation:

- Is more interpretable
- Captures the underlying structure of data
- Useful for downstream tasks (classification, clustering)

5. Inspired by Biological Neurons

... amount of compression

In the human brain:

- Only a few neurons fire at a time
- Sparse coding is energy-efficient and more meaningful
- Sparse autoencoder follows the same principle.

e.g. Assume hidden layer has 50 neurons.

- Sparsity constraint says: "For each input, only 3 neurons should fire."

Now when you show digit 0:

- Neuron 8 (circle detector) → fires
- Neuron 14 (round curve detector) → fires
- Neuron 20 (edge closure detector) → fires

All other 47 neurons remain OFF.

Thus the model learns:

- Only key patterns needed to represent each input.

Q. 4b) Explain architecture of autoencoder with neat diagram. Explain the hyperparameters that must be set before training of autoencoders.

Ans An autoencoder is a type of Artificial Neural Network used to compress the input data into a small representation and then reconstruct the original data back from it. It is mainly used for dimensionality reduction, noise removal, and feature learning.

* Architecture

An autoencoder has three main components:

A) Encoder

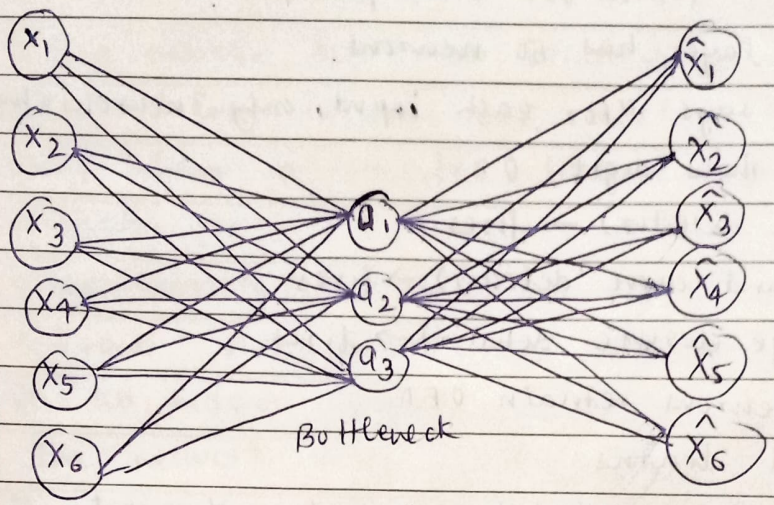
- Takes high-dimensional input (images/data)
- Learns important and useful features
- Converts input into compressed form

B) Latent Space / Bottleneck Layer.

- Middle layer with very few neurons
- Stores compressed / key information
- Forces network to learn only important features
- Representation here is called latent code/ feature vector

C) Decoder

- Takes compressed features and reconstructs original input
- Tries to make output as similar as possible to input.




* Working

1. Input is given to Encoder
2. Encoder extracts important patterns → produces latent code
3. Decoder uses latent code to reconstruct input
4. A loss function measures difference between Input (X) and Output (X')
5. Model adjusts weights to reduce reconstruction error
6. After training → Encoder learns meaningful features of data

* Loss Function used

Autoencoder tries to minimize:

$$Loss = ||X - X'||^2$$

Means the output must look like the input

* Hyperparameters - to be set before Training

Hyperparameters = settings chosen before traing, which affect model performance. They are:

- Learning rate: Controls how fast weights update
- Number of hidden Layers: More layers = deeper feature learning

- Neurons in Latent Layer: Controls amount of compression
- Activation Functions: ReLU, Sigmoid, Tanh for non-linear learning
- Batch Size: Number of samples used per training step
- Epochs: Number of times model sees whole dataset
- Loss Function: Measures reconstruction quality (MSE etc.)
- Optimizer: Used to update weights (Adam, SGD, etc.)
- Regularization: Prevent overfitting (Dropout, Sparsity, Noise)
- Weight Initialization: Helps fast and stable learning

These must be chosen properly for a good autoencoder model.

* Applications

- Image compression, denoising audio & images, anomaly detection (fraud / defects detection), feature extraction for ML models, medical image reconstruction

NOV-DEC 2023

Q4 b) State and explain, Undercomplete Autoencoders

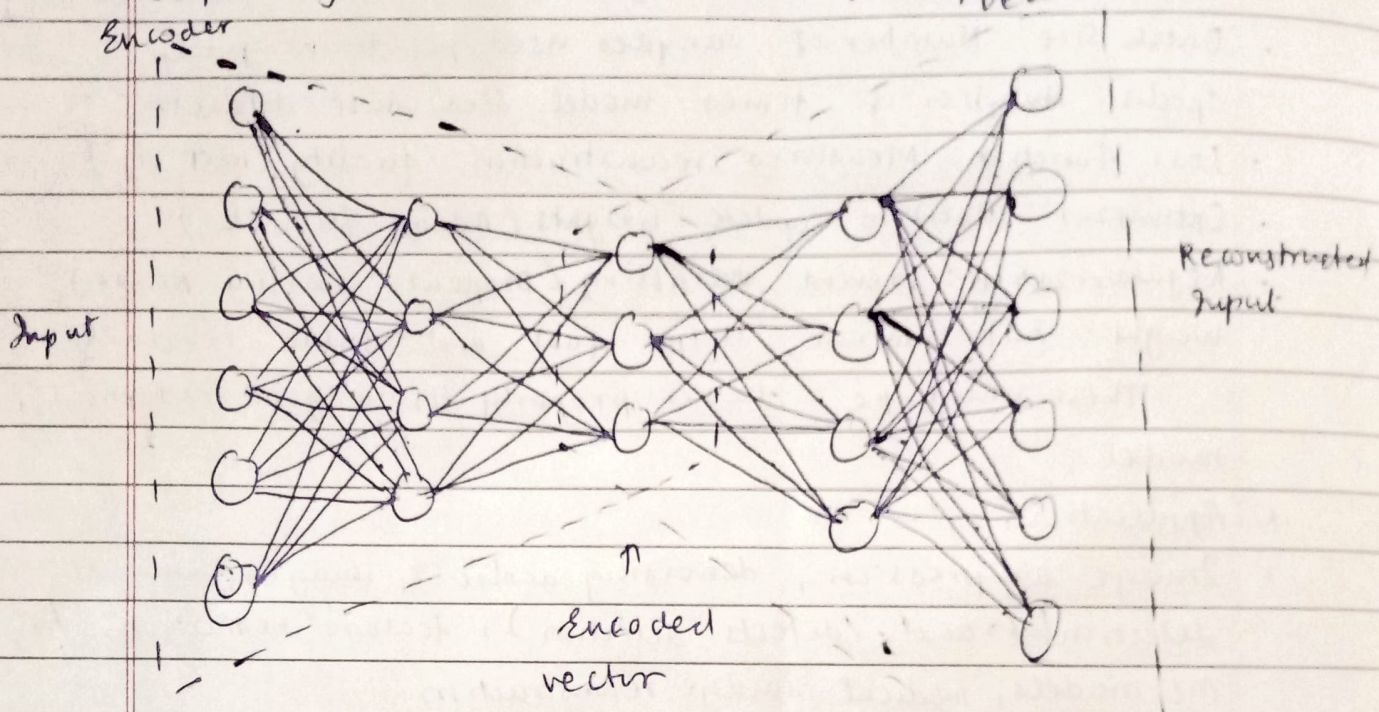
Ans. Autoencoder is a type of Artificial Neural Network used to learn compressed representation of data. It tries to reconstruct the input at the output. When the latent layer (middle layer) has fewer neurons than the input layer, it is called an Undercomplete Autoencoder.

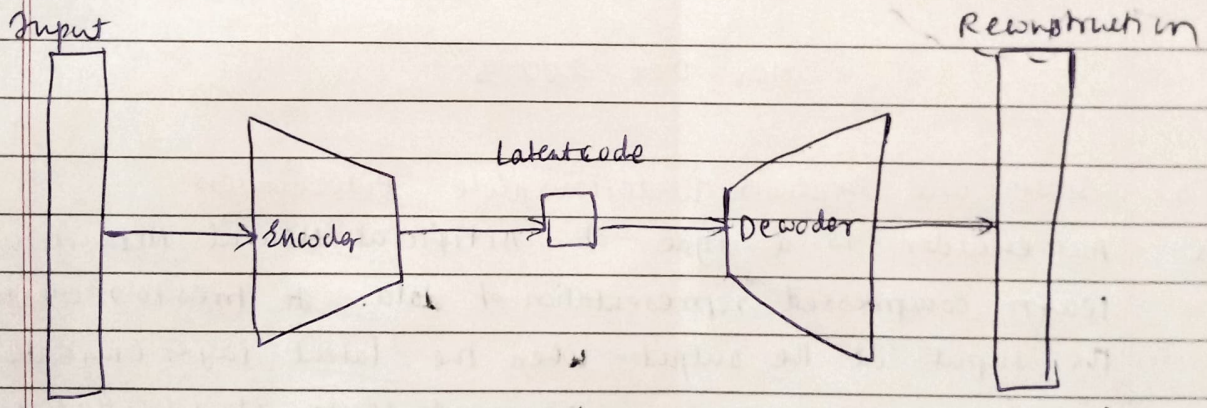
It is called undercomplete because the number of features in the bottleneck layer is less than input features → network is forced to learn important patterns only.

* Main Components

- Input Layer: Original data given to the network
- Encoder: Reduces the dimension and compresses the data
- Latent / Bottleneck layer: Stores compressed and most important features

- Decoder: Expands the data back to original dimensions
- Output Layer: Reconstructed version of input




* Working

1. Input data (example: an image with many pixels) is given.
2. Encoder compresses it into a small feature vector (latent vector)
3. The latent layer contains only important and useful information
4. Decoder tries to rebuild original data from compressed features.
5. Network learns by reducing reconstruction error:

   $Loss = ||x - x'||^2$

   (Difference between input and output)

It cannot copy input directly, so it learns patterns, not noise.

Consider handwritten digit images (784 pixels = input features)

| Layer | Neurons |
|---|---|
| Input | 784 |
| Encoded hidden layer | 128 |
| Latent layer | 32 |
| Decoded hidden layer | 128 |
| Output | 784 |

The network learns how a digit looks (generally) only using 32 important features

Why Undercomplete: dimensionality reduction, removes noise, finds hidden useful features, prevents overfitting

Applications: Image compression, noise reduction (denoising autoencoder), anomaly detection (abnormal data will reconstruct poorly), feature extraction for machine learning models

May-Jun 2024

) Explain the architecture of undercomplete autoencoder. What is the difference between undercomplete autoencoder and sparse autoencoder.

| Undercomplete Autoencoder | Sparse Autoencoder |
|---|---|
| Compress input by using fewer neurons in bottleneck layer | Use many hidden neurons but activate only a few |
| By reducing latent dimension size | By applying sparsity constraint (penalty) |
| Smaller than input dimension | Can be equal or larger than input dimension. |

| | | |
|---|---|---|
| Activation Behavior | All neurons in bottleneck are active | Only a few neurons become active for each input |
| Regularization | Not required | Required (L1, KL divergence) |
| Purpose | Direct dimensionality reduction | Learn unique, important features |
| Learning Focus | Compress data as much as possible | Make features specialized and interpretable. |

(Q6) What is a Bottleneck in autoencoder and why is it used?

Ans. The Bottleneck is the middle layer of an autoencoder where the data is highly compressed into a small number of neurons. It stores only the most important features of the input.

- Input → Encoder → Bottleneck → Decoder → Output
- Bottleneck layer = Latent space / code.

It is called "Bottleneck" because:

- It has very few neurons compared to input
- It forces the model to squeeze data through a narrow path.
- Just like a bottle neck restricts flow, this layer restricts information so only essential patterns can pass.

Purpose

- Dimensionality reduction: removes unnecessary information and reduces feature size
- Feature learning: Learns the most important patterns in data
- Avoid overfitting: Model cannot memorize everything, must generalize.
- Efficient storage: Stores data in compressed form

Better Reconstruction Quality forces decoder to use important details only.

e.g. Input image = 784 pixels
Bottleneck = 32 neurons
Encoder reduces 784 features → only 32 important features
Decoder reconstructs image using only these 32 features.
→ If the autoencoder still reconstructs image correctly →
It means 32 features were enough to represent main structure.

Input Layer (large) → Encoder → Bottleneck (very few neurons) → Decoder → Output layer (like input)

~~Q. 3 b~~

Q 3 b) Nov Dec 2024

How do autoencoders differ from other unsupervised learning techniques like PCA or clustering algorithms?

| Any feature | Autoencoders | PCA (Principal Component Analysis) | Clustering (e.g., K-Means) |
|---|---|---|---|
| Learning Type | Neural network-based | Statistical linear technique | Groups data into clusters |
| Output Purpose | Reconstruct input from compact features | Reduce dimensionality | Assign data to groups |
| Feature Extraction | Non-linear features | Only linear features | No feature extraction (just grouping) |
| Architecture | Encoder + Decoder | No neural network | Mathematical distance-based |

| | | | |
|---|---|---|---|
| Representation Type | Latent vector in bottleneck | Principal components | Cluster centres |
| Flexibility | Can model complex patterns | Limited to simple structure | No reconstruction capability |
| Uses | Image / audio compression, denoising, anomaly detection | Dimensionality Reduction | Grouping similar data |

Autoencoders are more powerful because they:

- can capture non-linear relationships in data
- can be designed for special tasks (sparse, denoising, contractive, VAE)
- can generate new data (Generative Models)
- Better feature learning for ML models
  PCA & clustering cannot do these

Q.3 c) Explain the following listed hyperparameters and methods which are used to optimize the training of autoencoders:

Ans i) ~~Activation functions~~ To train an autoencoder properly, we must set some important hyperparameters. These help the model learn better features and reduce errors.

i) Activation functions

These functions decide whether a neuron should fire or not.

e.g. ReLU → good for hidden layers (only positive values activate neurons)

- Sigmoid → used for image pixel values between 0-1.
  e.g. if input image is grayscale (0-1), sigmoid activation is used in output layer.

ii) Loss functions

Loss tells us how different the reconstructed output is from input

e.g. MSE (Mean Squared Error) → for normal images
. Binary Cross Entropy → for black & white images
e.g. If input digit "5" looks different after reconstruction, the loss increases. Goal → Minimize loss.

iii) Learning Rate

Controls how fast the network learns
- Too high → model learns wrong things
- Too low → training becomes very slow

e.g. Learning rate = 0.001 (Commonly used for auto encoders)

iv) Batch size

Number of training samples processed at once
- Small batch → better learning but slow
- Large batch → fast but may overfit

e.g. Batch size = 32 images per training step

v) Early Stopping

Stops ~~early~~ training automatically when the model starts overfitting.

e.g. If validation error increases after 50th epoch, training stops at 50, not 100. This saves time & gives better results.

vi) Initialization

Initial weights should not be random bad values. Good initialization helps model start learning in the right direction

e.g. Xavier or He initialization is used so activations don't vanish or explode

May - Jun 2025

Q4 a) Write a short note on Stochastic autoencoders and decoders.

Ans. A stochastic Autoencoder is a type of autoencoder where we add randomness in the hidden layer or output. This means → the hidden representation or output is not fixed every time.

- It does not create only one answer
- It samples different possible answers.
- Helps model uncertain or noisy data

It learns to handle data that has randomness or variations in real life

* Working

1. Input is given to encoder
2. Encoder produces a probability distribution instead of fixed values (like mean and variance)
3. A random sample is taken from this distribution
4. Decoder uses this sample to reconstruct the output.

So, for the same input, output can change a little each time.

e.g. Suppose input image is a digit "3". A normal autoencoder always gives the same "3" back.
Stochastic autoencoder: May generate slightly different styles of "3" (thicker or thinner, little curve difference, etc.). Good for generating new samples, & not only copying input.

Input (x) → Stochastic Encoder → Latent Distribution (z) (Random Sampling) → Stochastic Decoder → Output (x')

* Applications

- Generating new images: because it can create different styles / variations
- Speech / music generation: natural variation in sound.
- Handling noisy data: Can understand uncertainty
- Variational Autoencoders (VAE): VAEs use stochastic encoder & decoder
- Data augmentation: Creates new examples from learned patterns.

* Advantages

✓ More flexible & realistic results

✓ Good for uncertainty handling
✓ Produces many variations of data
✓ Better generalization than normal autoencoders

Q4b) Differentiate between denoising autoencoder and contractive autoencoder

| Feature | Denoising Autoencoder (DAE) | Contractive Autoencoder (CAE) |
|---|---|---|
| Purpose | Remove noise from input | Make latent features stable |
| Training method | Add noise to input data & learn to reconstruct clean data | No noise added - apply penalty to keep features from changing |
| How Robustness is achieved | By learning to ignore added noise | By reducing sensitivity to input changes |
| Loss function | Reconstruction Loss (X vs X') | Reconstruction loss + Contractive penalty |
| Latent space Behavior | Focus on cleaning and restoring input | Focus on smooth and invariant feature space |
| Focus | Data recovery | Feature robustness and stability |
| Good For | Image / Audio denoising | Classification & feature representation |

GoodLuck | Page No.
Date

Example visualization: Noisy 7 → Clean 7 | Tilted 7 → Still understood as 7

eg: Denoising Autoencoder

Noisy input → Encoder → Decoder → Clean output

e.g. Input: Image of number "3" with noise (dots, blur)
Output: clean "3"
Model learns to remove noise & restore clarity.

Applications: Image cleaning (camera enhancements), speech/audio enhancement (noise removal), MRI/CT scan noise reduction, data pre-processing for ML

Contractive Autoencoder

Input → Encoder (contractive penalty) → Decoder → Stable output

e.g. Input: Image of "3" slightly tilted or slightly blurred. Latent features: Remain very similar → still recognized as 3. Small changes in input do not confuse the model.

Applications: Handwriting and face recognition, Robust feature extraction, classification tasks, anomaly detection, industrial fault detection