Assignment No. 3

Seminar And Technical Communication

Title: Analyze the topic and prepare technical details of the selected topic. This assignment may include contents like architecture details, different modules in detail, algorithms, and hardware details if any.

Topic of seminar : Autoencoders.

Name: Thorve Avishkar Shrikrushna.

Roll No.: 62

Exam Seat No: T1908402068

Subject Code: 317526

Date of Performance :

Date of Submission :

- **TITLE** : Analyse the topic and prepare technical details of the selected topic. This assignment may include contents like architecture details, different modules in detail, algorithms, and hardware details if any.

- **TECHNICAL DETAILS :**

  1.Understanding the Problem

  2.Data Preprocessing

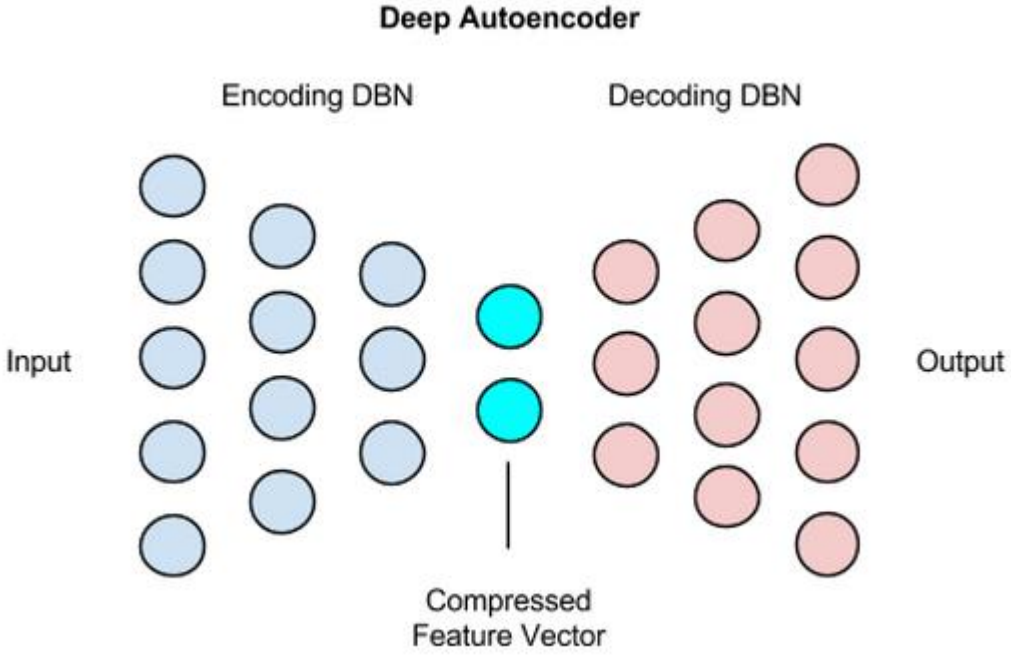
  3.Choosing a CNN Architecture

  4.Model Training

  5.Evaluation and Fine-Tuning

  6.Deployment

- **ARCHITECTURE DETAILS :**



➢ DIFFERENT MODULES :

1. Encoder

- **Function**: Compresses the input data into a lower-dimensional latent representation.
- **Components**: Multiple layers (fully connected, convolutional, etc.) with activation functions (ReLU, Sigmoid, Tanh) that map input data to a compact, abstract representation.

2. Latent Space (Bottleneck)

- **Function**: Represents the compressed, lower-dimensional encoding of the input data. It captures the essential features needed to reconstruct the input.
- **Components**: A reduced number of neurons compared to the input layer, often the narrowest part of the network.

3. Decoder

- **Function**: Reconstructs the input data from the latent space representation by gradually upscaling the compressed data back to its original size.
- **Components**: Layers mirroring the encoder, with activations that expand data dimensions toward the original input size.

4. Loss Function

- **Function**: Measures the difference between the original input and the reconstructed output. Minimizing the loss helps optimize the autoencoder.
- **Common Choices**:
    - Mean Squared Error (MSE) for regression tasks.
    - Binary Cross-Entropy (BCE) for binary data.

5. Regularization

- **Function**: Helps prevent overfitting and enforces desired properties, such as sparsity or robustness.
- **Types**:
    - L1/L2 Regularization.

- KL Divergence (used in Variational Autoencoders).

6. Optimization Algorithm

- **Function**: Optimizes the network's weights by minimizing the loss function.
- **Common Algorithms**:
  - Stochastic Gradient Descent (SGD).
  - Adam.

7. Noise Injection (Denoising Autoencoders)

- **Function**: Corrupts the input data by adding noise, and the model is trained to reconstruct the clean version of the data.
- **Components**: Layers that introduce noise during training, forcing the model to learn robust feature representations.

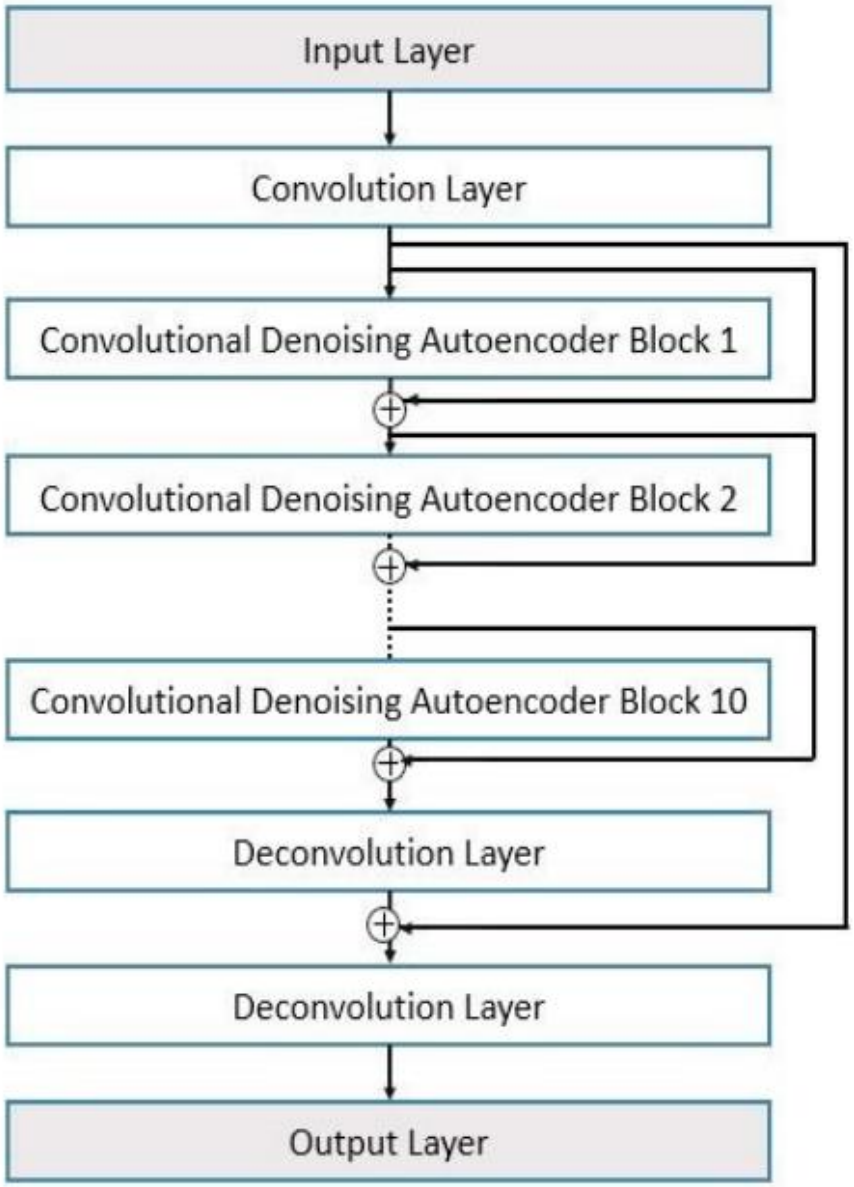
8. Graph Module (Graph Autoencoders)

- **Function**: Processes data with graph structures by learning node embeddings while preserving graph topology.
- **Components**: Graph convolution layers to handle graph-structured data.

9. Generative Module (Variational Autoencoders, Adversarial Autoencoders)

- **Function**: Learns a distribution over the data in the latent space to generate new samples resembling the input data.
- **Components**: Layers for sampling from a latent distribution, often paired with regularization methods like KL Divergence or adversarial networks.

- **ALGORITHMS**:


Input Layer
Convolution Layer
Convolutional Denoising Autoencoder Block 1
Convolutional Denoising Autoencoder Block 2
Convolutional Denoising Autoencoder Block 10
Deconvolution Layer
Deconvolution Layer
Output Layer

➢ HARDWARE DETAILS :

1. Processing Units:

- **CPU (Central Processing Unit)**: For small datasets or simpler models, a CPU can handle the training and inference processes. However, it's slower for deep learning tasks, especially as the complexity and size of the autoencoder increase.
- **GPU (Graphics Processing Unit)**: GPUs are more efficient than CPUs for training large-scale autoencoders because they can handle parallel computations. This is essential when working with high-dimensional data or deep architectures. Popular GPUs include NVIDIA Tesla and RTX series.
- **TPU (Tensor Processing Unit)**: TPUs are specialized hardware accelerators developed by Google, optimized for tensor operations and deep learning tasks. They are highly effective for training large models like autoencoders in TensorFlow.

2. Memory:

- **RAM**: Sufficient RAM is needed to load the dataset and manage the autoencoder's weights and activations during training. For small to medium datasets, 16GB–32GB is common, but larger datasets may require up to 64GB or more.
- **GPU Memory**: When using GPUs, the memory on the card (e.g., 8GB, 16GB, or more) is important. Larger models require more GPU memory to handle the high volume of computations efficiently.

3. Storage:

- **SSD (Solid State Drive)**: Faster access to data stored on SSDs improves the speed of loading datasets and checkpointing during training. SSDs are preferred over HDDs for better data transfer rates, reducing training bottlenecks.
- **Cloud Storage**: For large datasets, cloud storage solutions like Google Cloud, AWS, or Azure are often used in combination with cloud-based GPUs or TPUs.

4. Network:

- **High Bandwidth Connectivity**: When using cloud services, high-speed internet is crucial for uploading large datasets, syncing model checkpoints, and handling distributed training.

Conclusion : Hence we have successfully analysed the topic "Autoencoders" through various models used, algorithms, hardware details, technical details and system architecture.

Cognitive	Psychomotor	Affective	Total	Signature