PVP 20

Code: 20AM4701A, 20DS4701A

IV B.Tech - I Semester – Regular / Supplementary Examinations
DECEMBER 2025

GENERATIVE ARTIFICIAL INTELLIGENCE
(Common for AIML, DS)

Duration: 3 hours Max. Marks: 70

Note: 1. This paper contains questions from 5 units of Syllabus. Each unit carries 14 marks and have an internal choice of Questions.
2. All parts of Question must be answered in one place.

BL – Blooms Level CO – Course Outcome

			BL	CO	Max. Marks
		UNIT-I			
1	a)	Differentiate between Generative and Discriminative models with suitable examples.	L2	CO1	7 M
	b)	Discuss the ethical considerations and risks associated with the deployment of Generative AI.	L2	CO1	7 M
		OR			
2	a)	What is transfer learning? Explain its significance in the context of fine-tuning Large Language Models (LLMs).	L2	CO1	7 M
	b)	Explain the phenomenon of "hallucination" in Large Language Models (LLMs).	L2	CO1	7 M

		UNIT-II			
3	a)	Make use of encoder-decoder structure of a Vanilla Autoencoder and list two of its applications.	L3	CO2	7 M
	b)	Explain the role of the Generator and Discriminator in a Generative Adversarial Network (GAN).	L2	CO1	7 M
		OR			
4	a)	Apply the concept of the Reparameterization Trick in Variational Autoencoders (VAEs). Why is it necessary?	L3	CO2	7 M
	b)	Analyze the components of the loss function used to train a VAE.	L4	CO4	7 M
		UNIT-III			
5	a)	Explain the Self-Attention mechanism with a step-by-step example.	L3	CO2	7 M
	b)	How does Multi-Head Attention extend Self-Attention mechanism to improve model performance?	L2	CO1	7 M
		OR			
6	a)	Compare and Contrast the Transformer architecture with traditional RNN/LSTM models for handling long-range dependencies.	L4	CO4	7 M
	b)	What is positional encoding? Justify its necessity in the Transformer architecture.	L2	CO1	7 M

		UNIT-IV			
7	a)	Describe the BERT architecture and list out any two applications of it.	L2	CO1	7 M
	b)	Compare and Contrast the various GPT models.	L4	CO4	7 M
		OR			
8	a)	Examine how a Text-to-Text Transfer Transformer (T5) model formulates various NLP tasks into a unified framework.	L4	CO4	7 M
	b)	Illustrate any two examples in NLP tasks that are suitable to apply T5 model.	L3	CO3	7 M
		UNIT-V			
9	a)	Apply the reverse diffusion process to describe how a noisy image is gradually denoised to generate a new sample.	L3	CO3	7 M
	b)	List any three key application areas for diffusion models beyond image generation.	L2	CO1	7 M
		OR			
10	a)	Make use of Forward and reverse process in describing diffusion models architecture.	L3	CO3	7 M
	b)	Analyse how DALL-E2 combines concepts from Transformers and Diffusion models to generate images from text prompts.	L4	CO4	7 M

PVP20

Code No:20AM4701A,20DS4701A

IV B.Tech - I Semester - Regular / Supplementary Examinations DECEMBER 2025
GENERATIVE ARTIFICIAL INTELLIGENCE
(Common for AIML, DS)

Duration: 3 Hours **Max. Marks: 70**

Note:

1. This question paper contains questions from 5 units of Syllabus. Each Unit carries 14 Marks and have an internal choice of Questions.
2. All parts of Question paper must be answered in one place.

5 x 14 = 70 Marks

1(a)Differentiate between Generative and Discriminative models with suitable examples.

- **Generative Models:**

Learn the joint probability distribution and can generate new data samples.**[3 Marks]**

- **Discriminative Models:**

Learn the conditional probability and focus on classification or prediction.**[3 Marks]**

- **Example Explanation:**

Generative model can generate handwritten digits, while a discriminative model only classifies digits. **[1 Mark]**

1(b) Discuss the ethical considerations and risks associated with the deployment of Generative AI (7M) ---(ANY 3 valid points) [1+2+2+2 Marks]

Generative AI systems raise significant **ethical, legal, and social concerns** due to their ability to create human-like content at scale.

ethical considerations and risks:

- Misinformation:
    - Models can generate convincing yet false news, deepfakes, phishing content, hate speech, or self-harm instructions.
    - Risk: Erosion of trust.
- Bias:
    - Training data may encode societal biases; models can amplify stereotypes in text or images.
    - Risk: Discriminatory outputs in hiring, lending, or content ranking.

- Privacy:
    - Training on personal data or scraped web data may leak sensitive information in outputs.
    - Risk: Privacy violations, re-identification, synthetic profiles resembling real people.
- Accountability & transparency:
    - Difficulty assigning responsibility for harms from AI-generated content.
    - Need for disclosure that content is AI-generated, logging, auditing, and governance.

2(a) What is transfer learning? Explain its significance in fine-tuning Large Language Models (LLMs)

- **Definition:**

Transfer learning adapts a pre-trained model to a new task. **[2 Marks]**

- **Significance:**

Reduces training time, data requirement, and computational cost. **[2 Marks]**

- **Steps in Fine-tuning:**

    - Select pre-trained model
    - Prepare dataset
    - Freeze base layers
    - Train task-specific layers
    - Unfreeze and fine-tune
    - Evaluate performance **[3 Marks]**

2(b) Explain the phenomenon of "hallucination" in Large Language Models (LLMs)

- **Definition:**
  Hallucination is the generation of confident but incorrect or fabricated information. **[4 Marks]**
- **Risks:**
  Dangerous in healthcare, legal, and financial domains. **[3 Marks]**

3(a) Explain the encoder–decoder structure of a Vanilla Autoencoder and list two applications

- **Encoder:**
  Compresses input into lower-dimensional representation. **[2 Marks]**
- **Latent Space:**
  Bottleneck storing essential features. **[2 Marks]**
- **Decoder:**
  Reconstructs original input from latent space. **[2 Marks]**
- **Applications (Any two):**
  Dimensionality reduction, noise removal, feature learning. **[1 Mark]**

3(b) Explain the role of Generator and Discriminator in a GAN

- **Generator:**
  Generates fake data from random noise. **[3 Marks]**
- **Discriminator:**
  Distinguishes real data from fake data. **[3 Marks]**
- **Adversarial Training:**
  Both trained in minimax game to improve generation quality. **[1 Mark]**

4(a)Apply the Reparameterization Trick in VAEs. Why is it necessary?

- **Need for Reparameterization:**
  Direct sampling is non-differentiable. **[2 Marks]**
- **Concept:**
  Separates randomness from parameters. **[2 Marks]**
- **Formula:**
  z=μ+σϵz = \mu + \sigma \epsilonz=μ+σϵ **[1 Mark]**
- **Benefits:**
  Enables backpropagation and stable training. **[2 Marks]**

4(b)Analyze the components of the loss function used to train a VAE.

- **Reconstruction Loss:**
  Ensures similarity between input and output. **[3 Marks]**
- **KL Divergence:**
  Regularizes latent space and enforces smoothness. **[3 Marks]**
- **Total Loss:**
  Sum of reconstruction loss and KL divergence. **[1 Mark]**

5(a) Explain the Self-Attention mechanism with a step-by-step example

- Uses **Query, Key, Value vectors [2 Marks]**
- Computes attention using **dot-product + softmax [2 Marks]**
- Produces **context-aware representations [2 Marks]**
- Captures **long-range dependencies [1 Mark]**

5(b)How does Multi-Head Attention extend Self-Attention to improve performance?

- Extension of **self-attention**
- Uses **Query, Key, Value vectors [2 Marks]**
- Computes attention using **dot-product + softmax [2 Marks]**
- Produces **context-aware representations [2 Marks]**
- Captures **long-range dependencies [1 Mark]**

6(a)Compare Transformer architecture with RNN/LSTM models for long-range dependencies

- Sequential vs Parallel processing **[2 Marks]**

- Vanishing gradient problem in RNNs **[2 Marks]**
- Self-attention in Transformers **[2 Marks]**
- Better scalability and performance **[1 Mark]**

6(b) What is positional encoding? Justify its necessity in Transformers

- Definition of positional encoding **[2 Marks]**
- Transformers lack sequence awareness **[2 Marks]**
- Added to embeddings **[1 Mark]**
- Helps preserve word order and context **[2 Marks]**

7(a) Describe the BERT architecture and list any two applications

- Bidirectional Transformer encoder **[3 Marks]**
- Pre-training using MLM and NSP **[2 Marks]**
- Applications (Any two): QA, sentiment analysis, NER **[2 Marks]**

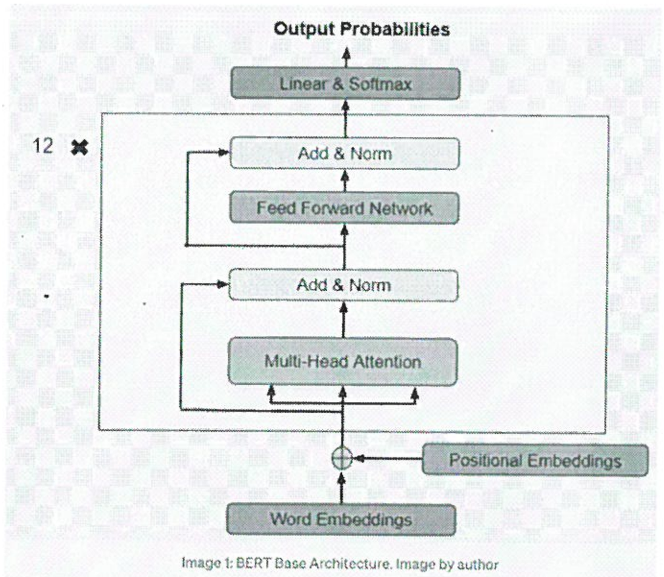


Image 1: BERT Base Architecture. Image by author

7(b) Compare and contrast various GPT models.

- Transformer decoder-based architecture **[2 Marks]**
- Evolution from GPT-1 to GPT-4 **[3 Marks]**
- Improvements in scale, reasoning, safety **[2 Marks]**

- Each version improves **scale, accuracy, and capability**

  - **Model size:** Increases significantly from GPT-1 to GPT-4
  - **Context understanding:** Improves with scale and training methods
  - **Learning ability:** Moves from fine-tuning to few-shot and instruction learning

- **Safety & alignment:** Stronger in later models
- **Applications:** From basic NLP → conversational AI → multimodal systems

8(a)Explain how the T5 model unifies various NLP tasks into a text-to-text framework

- Text-to-Text Transfer Transformer concept **[3 Marks]**
- Encoder–decoder architecture **[2 Marks]**
- Enables multitask and transfer learning **[2 Marks]**

8(b) Illustrate any two NLP tasks suitable for the T5 model

- T5 uses **text-to-text framework(any 2 task)-3.5 for each task**
    - Machine translation
    - Text summarization
    - Sentiment analysis
    - Question answering

Q9(a) Explain the reverse diffusion process used to generate an image from noise (7 Marks)
Marks

- Start from noise **[1½ Marks]**
- Noise prediction using neural network **[2 Marks]**
- Iterative denoising **[2 Marks]**
- Final image generation **[1½ Marks]**

1. **Starting from Noise**
    - The process begins with a completely random image made of Gaussian noise.
    - This noisy image contains no meaningful structure.
2. **Noise Prediction**
    - A neural network (usually U-Net) is trained to **predict the noise** present in the image at each step.
3. **Noise Removal**
    - Once noise is predicted, it is subtracted from the image.
    - This produces a slightly clearer image.
4. **Iterative Denoising**
    - The above steps are repeated many times.
    - Each step removes a small amount of noise.
5. **Final Image Generation**
    - After several iterations, the noisy image gradually turns into a **clear and realistic image.**

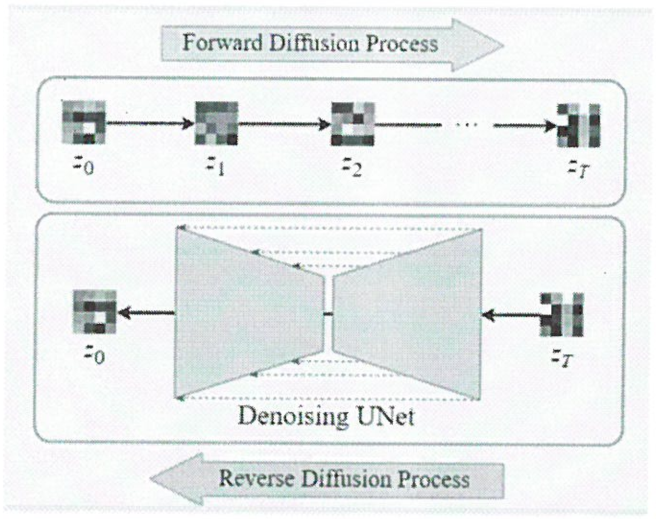
Q9(b) List any three applications of diffusion models other than image generation (7 Marks)

Applications (Any 3 Optional)(2.5+2.5+2 Marks)

- Medical data generation
- Text generation
- Image super-resolution
- Data Augmentation
- Audio Generation
- Video Generation
- Drug and Molecule Design

Q10(a) Forward and Reverse Processes in Diffusion Models

- **Forward Process (Noising):** [3M]
  - Start with real data → add Gaussian noise step by step → becomes pure noise
  - **Purpose:** Teach model how data becomes noise
- **Reverse Process (Denoising):** [3M]
  - Start with random noise → neural network predicts and removes noise step by step → generates realistic data
  - **Purpose:** Learn to recover data from noise



[1M]

Q10(b) Explain how DALL·E 2 uses Transformers and Diffusion models to generate images from text

Text Encoding: Transformer converts text prompt → embeddings [2 M]

- Aligns text and image representations
- **Diffusion Generation:** Starts from noise → gradually generates image using embeddings [2 M]
- **Example:** Prompt: "A dog wearing glasses and reading a book" → realistic matching image [3 M]

PVP20

Code No:20AM4701A, 20DS4701A,

IV B.Tech - I Semester - Regular / Supplementary Examinations DECEMBER 2025
GENERATIVE ARTIFICIAL INTELLIGENCE
(Common for AIML, DS)

Duration: 3 Hours **Max. Marks: 70**

Note:

1. This question paper contains questions from 5 units of Syllabus. Each Unit carries 14 Marks and have an internal choice of Questions.
2. All parts of Question paper must be answered in one place.

5 x 14 = 70 Marks

1(a) Differentiate between Generative and Discriminative models with suitable examples (7M)

Models are broadly classified into generative models and discriminative models based on how they learn from data and what type of problems they are designed to solve. This distinction is very important in the field of Generative Artificial Intelligence, where the goal is not only prediction but also data generation. Generative and discriminative models differ in their learning objectives, probability modeling, and applications.

Definition of Generative Models
A generative model learns the joint probability distribution of the input data and the output labels. Mathematically, it models P(X, Y), where X represents the input features and Y represents the class labels. By learning how the data is generated, a generative model can

Examples of Generative Models
A generative model trained on handwritten digits can generate new digit images that look similar to real ones.

Definition of Discriminative Models
A discriminative model learns the conditional probability of the output label given the input features. It models P(Y|X) and focuses on learning a direct mapping from inputs to outputs. The main objective of a discriminative model is accurate classification or prediction rather than data generation.

Examples of Discriminative Models
A discriminative model trained on digit images can classify an image as 0 to 9 but cannot generate a new digit image.

Comparison Between Generative and Discriminative Models
Generative models can generate new data samples, while discriminative models cannot. Generative models attempt to model the entire data distribution, whereas discriminative models focus only on separating classes. Generative models are useful when data is incomplete or unlabeled, while discriminative models generally require fully labeled datasets and perform better in pure classification tasks.

1(b) Discuss the ethical considerations and risks associated with the deployment of Generative AI (7M) (ANY 3)

Bias and Fairness
One of the major ethical concerns of Generative AI is bias. These models are trained on large datasets collected from the internet, which often contain social, cultural, and historical biases. As a result, the generated content may reinforce stereotypes related to gender, race, religion, or socioeconomic status. Biased outputs can lead to unfair treatment of individuals and groups, especially in sensitive applications such as hiring, education, healthcare, and law enforcement. Ensuring fairness requires careful dataset curation, bias detection, and regular auditing of model outputs.

Misinformation and Fake Content
Generative AI can easily produce realistic but false information in the form of text, images, or videos. This poses a significant risk of spreading misinformation, disinformation, and fake news. Deepfake technology, which generates realistic fake videos and audio, can be misused for political manipulation, fraud, or defamation. The large-scale deployment of Generative AI increases the difficulty of distinguishing between real and AI-generated content, which can undermine public trust in digital information.

Privacy and Data Security
Generative AI models are often trained on vast amounts of data that may include personal or sensitive information. There is a risk that the model may unintentionally memorize and reproduce private data. This raises serious concerns related to data privacy and confidentiality. If not properly managed, Generative AI systems could violate data protection laws and expose personal information. Strong data governance policies, anonymization techniques, and privacy-preserving training methods are required to reduce this risk.

Intellectual Property and Copyright Issues
Another ethical challenge is the use of copyrighted material in training datasets. Generative AI models may generate content that closely resembles existing copyrighted works, leading to intellectual property disputes. Artists, writers, and content creators may feel that their work is being used without consent or proper compensation. This raises questions about ownership, originality, and fair use of AI-generated content. Clear legal frameworks and transparent data usage policies are needed to address these concerns.

Job Displacement and Economic Impact
The deployment of Generative AI can automate tasks traditionally performed by humans, especially in creative, customer service, and content-generation roles. This can lead to job displacement and economic inequality if workers are not reskilled or supported during the transition. While Generative AI also creates new job opportunities, the short-term impact on

employment can be significant. Ethical deployment requires investing in education, upskilling, and workforce transition programs.

Security and Malicious Use
Generative AI can be exploited for harmful purposes such as generating phishing emails, malicious code, or automated cyberattacks. Attackers can use AI to scale up criminal activities with minimal effort. The ability to generate convincing human-like content increases the effectiveness of social engineering attacks. Developers and organizations must implement safeguards, usage restrictions, and monitoring systems to prevent misuse.

Lack of Transparency and Accountability
Many Generative AI models operate as black-box systems, making it difficult to understand how decisions or outputs are produced. This lack of transparency reduces accountability, especially when AI-generated content causes harm. Users may not know whether content was created by a human or an AI system. Ensuring explainability, clear documentation, and accountability mechanisms is essential for ethical deployment.

Environmental Impact
Training and deploying large Generative AI models require significant computational resources, leading to high energy consumption and carbon emissions. This environmental cost raises ethical concerns about sustainability. Organizations must consider energy-efficient model design, green computing practices, and responsible resource usage when deploying large-scale AI systems.

2(a) What is transfer learning? Explain its significance in fine-tuning Large Language Models (LLMs). (7M)

Transfer learning adapts a pre-trained model to a new task by reusing learned features, reducing training time and computational cost. This is especially important for **diffusion models and vision transformers**, where training from scratch is expensive.

1. Select a Pre-trained Model
Choose a model trained on a large, related dataset for custom text-to-image generation ensures relevant feature reuse.

2. Prepare the Target Dataset
Collect and preprocess task-specific data. Data augmentation is useful when data is limited.

3. Freeze Base Layers (Feature Extraction)
Freeze early layers that capture generic features like edges and textures. Add or replace task-specific layers This preserves general knowledge while allowing specialization.

4. Initial Training (Head Only)
Train only the newly added layers using a **low learning rate**. This helps the model adapt to the new task without overfitting or forgetting pre-trained features.

5. Fine-tuning (Unfreeze Layers)
Gradually unfreeze higher layers (top 20–50%) and fine-tune the model with a very small learning rate (e.g., 1e-5). This adapts high-level representations to the target domain.

6. Evaluation and Iteration
Evaluate performance using task-specific metrics. Adjust hyperparameters, freezing strategy, or regularization if required.

Transfer learning improves efficiency, reduces data requirements, and enables high-performance models to be adapted effectively for new tasks.

2(b) Explain the phenomenon of "hallucination" in Large Language Models (LLMs) (7M)

Hallucination in Large Language Models (LLMs) refers to the phenomenon where the model generates **confident, fluent, but factually incorrect or non-existent information.** The response may sound plausible, yet it is not grounded in verified data or reality. LLMs are trained to **predict the next most probable word** based on patterns in training data, not to verify truth. As a result, when faced with ambiguous, incomplete, or unseen queries, the model may "fill in gaps" with likely-sounding content.

Major Causes of Hallucination(Any 2)

1. **Training Data Limitations:** Incomplete, outdated, or biased datasets lead to incorrect outputs.
2. **Lack of Real-Time Knowledge:** LLMs do not inherently access live databases unless integrated with tools.
3. **Overgeneralization:** Models may extend patterns beyond valid contexts.
4. **Prompt Ambiguity:** Vague or misleading prompts increase hallucination risk.
5. **Model Confidence Bias:** The model generates text confidently even when uncertain.

Examples:

- Claiming a non-existent research paper with fake authors.
- Providing incorrect medical or legal advice with high confidence.

Hallucinations can cause **misinformation**, especially in sensitive domains like healthcare, finance, or law, leading to loss of trust and potential harm.

3(a) Explain the encoder–decoder structure of a Vanilla Autoencoder and list two applications(7M)

A Vanilla Autoencoder is a basic unsupervised neural network used to learn efficient representations of data. The main objective of an autoencoder is to reconstruct the input data at the output layer with minimum error. Unlike classification models, autoencoders do not require labeled data. They are widely used for feature learning, dimensionality reduction, and noise removal. A vanilla autoencoder consists of two main components, namely the encoder and the decoder, along with a latent space in between.

The vanilla autoencoder follows an encoder–decoder architecture where the input data is first compressed and then reconstructed. The flow of data is

Input → Encoder → Latent Space → Decoder → Reconstructed Output.

The model is trained such that the reconstructed output is as close as possible to the original input.

Encoder
The encoder is the first part of the autoencoder. Its main function is to transform the high-dimensional input data into a lower-dimensional representation called the latent vector or code. The encoder consists of one or more neural network layers that gradually reduce the number of neurons. Each layer applies a linear transformation followed by a non-linear activation function. The encoder forces the network to retain only the most important features of the input data.

Latent Space
The latent space, also known as the bottleneck layer, is the compressed representation of the input data. It contains fewer dimensions compared to the input layer. This bottleneck prevents the network from simply copying the input to the output and encourages meaningful feature learning. The quality of the latent space determines how well the autoencoder can reconstruct the input.

Decoder
The decoder is the second part of the autoencoder. Its role is to reconstruct the original input from the latent representation produced by the encoder. The decoder consists of neural network layers that gradually increase the dimensionality of the data. It applies transformations that mirror the encoder process. The decoder aims to produce an output that closely matches the original input.

Application 1: Dimensionality Reduction
One major application of vanilla autoencoders is dimensionality reduction. They are used to convert high-dimensional data into compact feature representations while preserving important information. Unlike traditional methods such as PCA, autoencoders can capture non-linear relationships in data. This is useful in data visualization, data compression, and feature extraction tasks.

Application 2: Denoising
Another important application of vanilla autoencoders is denoising. In this case, the model is trained to reconstruct clean data from noisy input. The autoencoder learns to ignore noise and focus on the underlying structure of the data. This technique is commonly used in image processing, audio signal enhancement, and sensor data cleaning.

3(b) Explain the role of Generator and Discriminator in a GAN (7M)

A **Generative Adversarial Network (GAN)** consists of two neural networks—the **Generator (G)** and the **Discriminator (D)**—that are trained simultaneously in a competitive (adversarial) framework. Their interaction enables the model to generate realistic synthetic data.

Role of the Generator (G):
The Generator's task is to **create fake data samples** (such as images, audio, or text) that resemble real data. It takes **random noise (z)** as input and transforms it into data samples. The Generator aims to **fool the Discriminator** into classifying its outputs as real. Over time, it learns the underlying data distribution and produces increasingly realistic samples.

Role of the Discriminator (D):
The Discriminator acts as a **binary classifier** that distinguishes between real data (from the training dataset) and fake data (produced by the Generator). It outputs a probability indicating whether a given sample is real or generated. The Discriminator's objective is to **correctly identify real vs. fake samples**, thereby providing feedback to the Generator.

- The Generator and Discriminator are trained alternately.
- The Discriminator is trained to maximize its classification accuracy.
- The Generator is trained to minimize the Discriminator's ability to detect fake samples.
  This interaction is formulated as a **minimax game**, where the Generator minimizes the loss while the Discriminator maximizes it.

As training progresses, the Generator improves its ability to generate realistic data, while the Discriminator becomes better at detection. Ideally, training reaches an equilibrium where the Discriminator cannot distinguish real from fake samples (probability ≈ 0.5).

Applications:
GANs are widely used in **image generation, image super-resolution, style transfer, data augmentation, and deepfake detection.**

4(a)Apply the Reparameterization Trick in VAEs. Why is it necessary? (7M)

A **Variational Autoencoder (VAE)** is a generative model that learns a probabilistic latent space to generate new data. Unlike standard autoencoders, VAEs model the latent variables as **probability distributions** instead of fixed values. The **reparameterization trick** is a key technique that enables efficient training of VAEs using gradient-based optimization.

Reparameterization Trick – Concept:
The reparameterization trick solves this problem by **separating randomness from the model parameters**. Instead of sampling z directly, we sample from a standard normal distribution and transform it deterministically:

$$z=\mu+\sigma\odot\epsilon,$$

Here, randomness is introduced through ε, while μ and σ remain differentiable.

- Enables **backpropagation** through the stochastic sampling step
- Makes learning a smooth and continuous latent space possible

VAE During training:

1. The encoder outputs μ and σ
2. ε is sampled from a standard normal distribution
3. z is computed using the reparameterization trick
4. z is passed to the decoder to reconstruct input data
5. Loss is computed as **reconstruction loss + KL divergence**

The reparameterization trick allows VAEs to efficiently learn meaningful latent representations and generate high-quality new samples.
The reparameterization trick is essential for VAEs because it enables gradient-based learning despite stochastic sampling, making practical VAE training feasible.

4(b)Analyze the components of the loss function used to train a VAE. (7M)

A **Variational Autoencoder (VAE)** is trained using a composite loss function that balances **accurate data reconstruction** with **regularization of the latent space**. This loss function is consists of two main components: **Reconstruction Loss** and **KL Divergence Loss.**

1. Reconstruction Loss

The reconstruction loss measures how well the decoder reconstructs the input data from the latent variable **z.** It encourages the model to generate outputs that are close to the original inputs.

- For continuous data (e.g., images):
  Mean Squared Error (MSE) is commonly used.
- For binary data:
  Binary Cross-Entropy (BCE) is used.

Role:

- Ensures high-quality reconstruction
- Forces latent variables to capture meaningful features

2. KL Divergence Loss (Regularization Term)

The KL divergence term measures how much the learned latent distribution deviates from a chosen prior distribution usually standard normal.

Role:

- Regularizes the latent space
- Encourages smooth and continuous representations
- Prevents overfitting

- Enables meaningful sampling from the latent space

3. Combined VAE Loss Function

The total loss is the sum of reconstruction loss and KL divergence:

$$L_{VAE}=L_{recon}+L_{KL}$$

Importance of Balancing Both Terms

- High reconstruction loss → poor output quality
- High KL loss → latent space collapse
- Proper balance ensures **good reconstruction and generalization**

5(a) Explain the Self-Attention mechanism with a step-by-step example(7M)

- **Self-Attention mechanism** Captures **long-range dependencies**
- It Enables **parallel processing**
- Improves understanding of **context and meaning**

Step 1: Input Embeddings

Each word is converted into a vector using word embeddings (plus positional encoding).

Step 2: Create Query, Key, and Value Vectors

For each word embedding, three vectors are generated using learned weight matrices:

- **Query (Q):** What the word is looking for
- **Key (K):** What the word offers
- **Value (V):** The actual information to be passed

Step 3: Compute Attention Scores

The attention score between two words is calculated using the dot product of their Query and Key vectors:

$$\text{Score}(Q,K)=Q\cdot K$$

Step 5: Contextualized Word Representation

The result is a **context-aware representation** of each word, enriched with information from relevant words in the sentence.

Step 6: Multi-Head Attention (Brief)

In practice, multiple self-attention heads run in parallel to capture different types of relationships (syntax, semantics, position).

5(b)How does Multi-Head Attention extend Self-Attention to improve performance? (7M)

Multi-Head Attention (MHA) is an extension of the self-attention mechanism used in Transformer models. Instead of computing a single attention function, multi-head attention runs **multiple self-attention operations in parallel**, allowing the model to capture different types of relationships in the data simultaneously. This significantly improves model performance and representational power.

Working of Multi-Head Attention:

1. The input embeddings are linearly projected into **multiple sets of Query (Q), Key (K), and Value (V)** vectors using different learned weight matrices.
2. Each set corresponds to an individual **attention head**, and self-attention is computed independently for each head.
3. Each head focuses on **different relationships** (e.g., subject–verb, word order, semantic similarity).
4. The outputs of all attention heads are **concatenated.**
5. A final linear projection combines these outputs into a single representation.

Mathematically:

$$\text{MultiHead}(Q,K,V)=\text{Concat}(\text{head}1,\ldots,\text{head}h)$$

Advantages of Multi-Head Attention:

- Captures **diverse contextual relationships** simultaneously
- Improves modeling of **long-range dependencies**
- Enhances representation quality without increasing sequence length
- Allows attention at **different subspaces** of the embedding dimension

Multi-head attention improves accuracy in tasks such as **machine translation, text summarization, and question answering**, forming the backbone of Transformer-based models like BERT and GPT.
By enabling parallel attention over multiple representation subspaces, multi-head attention significantly enhances the expressive power and performance of self-attention mechanisms.

6(a)Compare Transformer architecture with RNN/LSTM models for long-range dependencies(any 4) --(7M)

Long-range dependencies refer to a model's ability to capture relationships between elements that are far apart in a sequence. Transformer architectures and RNN/LSTM models handle this challenge in fundamentally different ways.

1. Architecture

- **RNN/LSTM:** Process sequences **sequentially**, where each time step depends on the previous hidden state. LSTMs improve RNNs by using gates to control information flow.
- **Transformer:** Uses **self-attention** and processes the entire sequence **in parallel**, without recurrence.

2. Handling Long-Range Dependencies

- **RNN:** Suffers from **vanishing/exploding gradients**, making it hard to retain information over long sequences.
- **LSTM:** Mitigates this issue using memory cells and gates, but still struggles with very long contexts.
- **Transformer:** Self-attention directly connects every token to every other token, enabling **efficient modeling of long-range dependencies**.

3. Parallelism and Training Efficiency

- **RNN/LSTM:** Sequential nature prevents parallel computation, resulting in **slower training**.
- **Transformer:** Fully parallelizable, leading to **faster training on large datasets**.

4. Context Representation

- **RNN/LSTM:** Context is compressed into a single hidden state, which may lose distant information.
- **Transformer:** Each token has a **context-aware representation** influenced by all other tokens.

5. Scalability

- **RNN/LSTM:** Difficult to scale for very long sequences.
- **Transformer:** Scales well with large datasets, though attention has quadratic complexity.

6. Performance in Practice

- **RNN/LSTM:** Effective for short to medium sequences (speech, time series).
- **Transformer:** Superior performance in NLP tasks like translation, summarization, and LLMs.

6(b)What is positional encoding? Justify its necessity in Transformers. (7M)

Positional Encoding is a technique used in Transformer architectures to **inject information about the order and position of tokens** in a sequence. Since Transformers rely on self-attention and process input tokens **in parallel**, they do not inherently capture sequence order like RNNs or CNNs. Positional encoding provides this missing sequential information.

Why Positional Encoding Is Needed:
Self-attention treats the input as a **set of tokens**, meaning that without positional information, sentences like *"dog bites man"* and *"man bites dog"* would appear identical to the model. Therefore, positional encoding is necessary to help the model understand **word order, distance, and relative positions.**

How Positional Encoding Works:
Positional encodings are vectors added to the input embeddings at the bottom of the Transformer. These vectors encode the position of each token in the sequence.

Positional encoding is crucial in Transformers because it compensates for the lack of recurrence or convolution, allowing the model to understand token order while preserving parallelism.

7(a) Describe the BERT architecture and list any two applications-(7M)

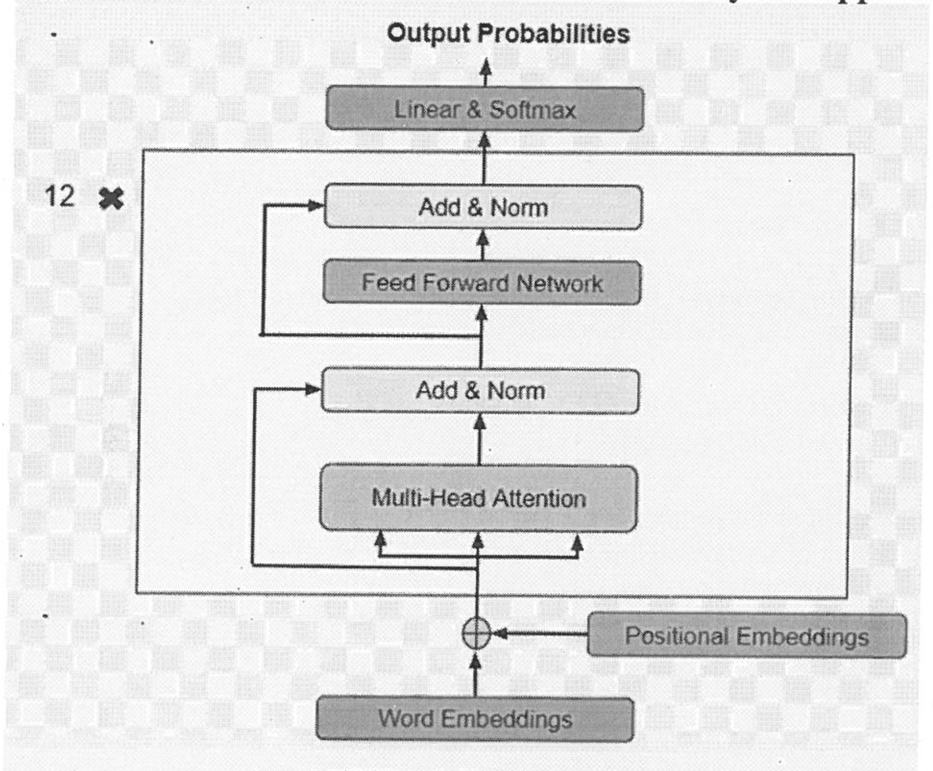


Image 1: BERT Base Architecture. Image by author

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google that is designed to understand the **context of words bidirectionally**, i.e., using both left and right context simultaneously. It is based entirely on the **Transformer encoder architecture**.

BERT Architecture:
BERT consists of multiple stacked **Transformer encoder layers**. Each encoder layer contains two main sub-components:

1. **Multi-Head Self-Attention:**
   This allows each word in a sentence to attend to all other words, capturing deep contextual relationships in both directions.
2. **Position-wise Feed-Forward Network:**
   A fully connected neural network applied independently to each token to transform representations.

- **Input Embeddings:** Combination of token embeddings, positional embeddings, and segment embeddings
- **Layer Normalization and Residual Connections:** Improve training stability and convergence
- **Bidirectional Context:** Unlike GPT, BERT reads the entire sentence at once

Pre-training Objectives:

- **Masked Language Modeling (MLM):** Randomly masks tokens and predicts them
- **Next Sentence Prediction (NSP):** Learns sentence relationships

Model Variants:

- **BERT-Base**
- **BERT-Large**

Applications of BERT (Any Two):

1. **Question Answering Systems** – Extracts precise answers from passages
2. **Text Classification** – Used for sentiment analysis, spam detection

7(b) Compare and contrast various GPT models.-- (7M)
(comparison of any 3 GPT model)

GPT (Generative Pre-trained Transformer) models are a family of autoregressive language models developed by OpenAI. They are based on the **Transformer decoder architecture** and differ in size, training data, capabilities, and applications.

1. GPT-1 (2018):

- First GPT model with **117 million parameters**
- Demonstrated the effectiveness of **pre-training + fine-tuning**
- Limited contextual understanding and generation capability
- Used mainly for basic NLP tasks

2. GPT-2 (2019):

- Much larger (up to **1.5 billion parameters**)
- Improved text coherence and long-form generation

- Raised concerns about misuse due to realistic text generation
- Still lacked strong reasoning and instruction-following abilities

3. GPT-3 (2020):

- Extremely large with **175 billion parameters**
- Enabled **few-shot and zero-shot learning**
- Strong performance across translation, summarization, and QA
- High computational and deployment cost

4. GPT-3.5:

- Optimized version of GPT-3
- Improved conversational abilities
- Used as the base for early ChatGPT versions
- Better instruction-following than GPT-3

5. GPT-4 / GPT-4.x:

- Enhanced **reasoning, factual accuracy, and safety**
- Supports **multimodal inputs** (text + images)
- Better handling of complex tasks and long contexts
- Used in advanced AI assistants and professional applications

8(a) Explain how the T5 model unifies various NLP tasks into a text-to-text framework--(7M)

T5 (Text-to-Text Transfer Transformer) is a Transformer-based model proposed by Google that introduces a unified framework where **all NLP tasks are converted into a text-to-text format.** In this approach, both the input and the output of every task are represented as plain text strings, simplifying model design and training.

Traditional NLP systems require different architectures or output layers for different tasks such as translation, summarization, classification, and question answering. T5 removes this complexity by framing **every task as text generation**, enabling a single model to handle multiple NLP tasks.

Text-to-Text Formulation:

- **Input:** Task description + input text
- **Output:** Desired result in text form

Input: "summarize: [document]"
Output: "[summary]"

Architecture:
T5 uses a **Transformer encoder–decoder architecture:**

- **Encoder:** Processes the input text (task + data)
- **Decoder:** Generates the output text token by token

- Unified model for diverse NLP tasks
- Simplifies fine-tuning and deployment
- Enables **transfer learning across tasks**
- Improves generalization and consistency

By converting all NLP problems into a text-to-text format, T5 provides a simple yet powerful unified framework that achieves strong performance across a wide range of language tasks.

8(b) Illustrate any two NLP tasks suitable for the T5 model. (7M)

The **T5 (Text-to-Text Transfer Transformer)** model converts every Natural Language Processing (NLP) task into a **text-to-text format**, where both input and output are plain text. This unified framework allows T5 to handle a wide range of NLP tasks using the same architecture. Four suitable NLP tasks are illustrated below.

(Any 2 Optional)

1. Machine Translation

Task: Convert text from one language to another.

- **Input:**
  translate English to German: Artificial Intelligence is powerful
- **Output:**
  Künstliche Intelligenz ist leistungsstark

Explanation:
The task prefix instructs the model to translate between languages.

2. Text Summarization

Task: Generate a concise summary of a long document.

- **Input:**
  summarize: Artificial Intelligence is transforming healthcare, education...
- **Output:**
  AI is transforming multiple sectors.

Explanation:
T5 identifies key information and generates an abstractive summary.

3. Sentiment Analysis

Task: Determine the sentiment of a given text.

- **Input:**
  sentiment: The movie was absolutely amazing
- **Output:**
  positive

Explanation:
Classification is handled as text generation.

4. Question Answering

Task: Answer questions based on a given passage.

- **Input:**
  question: Who invented the telephone? context: Alexander Graham Bell invented the telephone.
- **Output:**
  Alexander Graham Bell

Explanation:
T5 generates answers directly in text form.

By framing translation, summarization, sentiment analysis, and question answering as text-to-text tasks, T5 provides a flexible and unified approach to NLP.

Q9(a) Explain the reverse diffusion process used to generate an image from noise--(7M)

1. **Starting from Noise**
   - The process begins with a completely random image made of Gaussian noise.
   - This noisy image contains no meaningful structure.
2. **Noise Prediction**
   - A neural network (usually U-Net) is trained to **predict the noise** present in the image at each step.
3. **Noise Removal**
   - Once noise is predicted, it is subtracted from the image.
   - This produces a slightly clearer image.
4. **Iterative Denoising**
   - The above steps are repeated many times.
   - Each step removes a small amount of noise.
5. **Final Image Generation**
   - After several iterations, the noisy image gradually turns into a **clear and realistic image.**

Q9(b) List any three applications of diffusion models other than image generation. (7 Marks)

Applications (Any 3 Optional)

- Medical data generation
- Text generation
- Image super-resolution
- Data Augmentation
- Audio Generation
- Video Generation

Diffusion models are powerful generative models that learn to transform noise into meaningful data through a gradual denoising process. While they are widely known for image generation, diffusion models are also successfully applied in several **non-image domains.** Three important applications are described below.

1. Audio and Speech Generation

Diffusion models are used to generate **high-quality audio signals**, including speech and music. They can model complex temporal dependencies and produce realistic waveforms.

Examples:

- Text-to-speech systems
- Music synthesis
- Speech enhancement and noise reduction

2. Text and Language Modeling

Diffusion models have been adapted for **text generation, text infilling, and language modeling** by gradually denoising token representations.

Examples:

- Text completion
- Controlled text generation
- Language-based diffusion transformer

3. Scientific and Molecular Data Generation

Diffusion models are used to generate **molecular structures, protein conformations, and chemical compounds** in scientific research.

Examples:

- Drug discovery
- Protein structure prediction
- Material science simulations.

Q10(a) Explain the forward and reverse processes in diffusion models (7M)

Diffusion models are based on two important processes:

1. Forward Process – Adding noise
2. Reverse Process – Removing noise

Together, these processes enable data generation.

Forward Process (Noising Process)

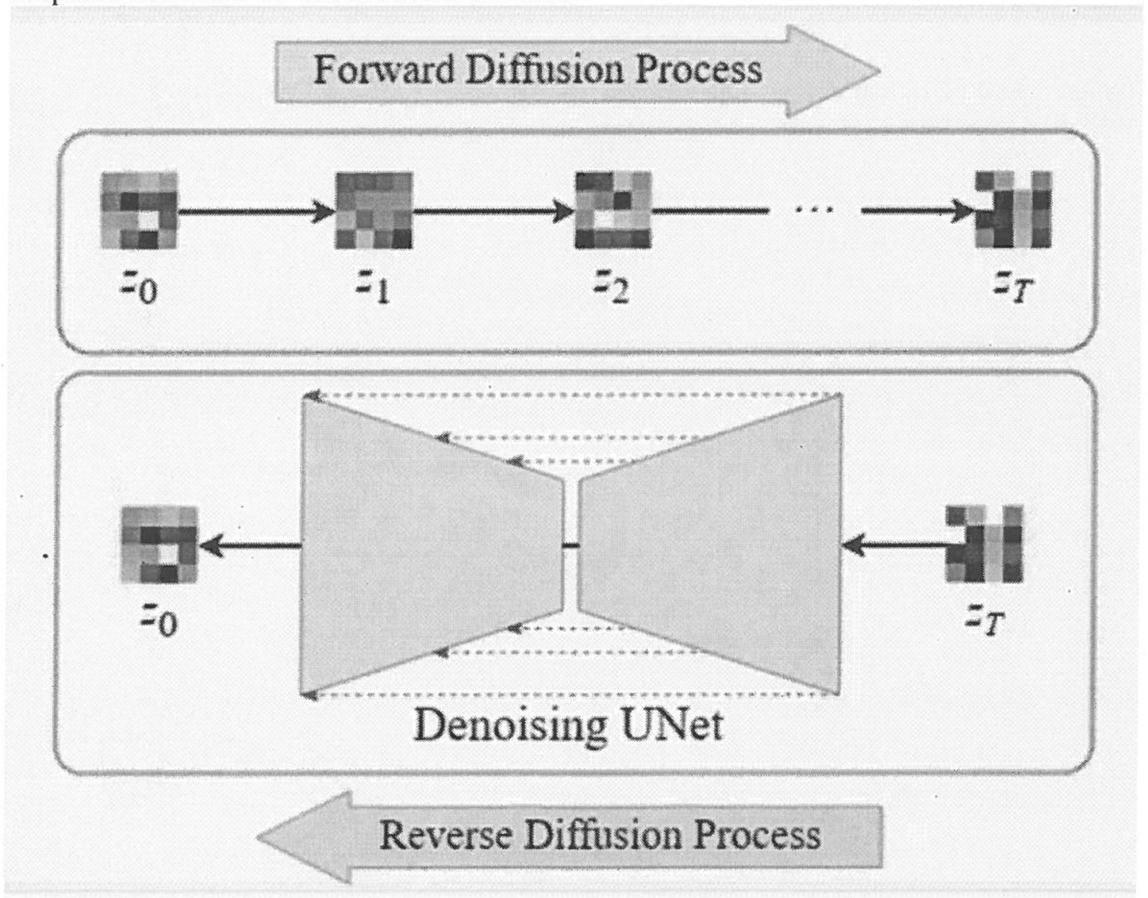
- Starts with a real image
- Small Gaussian noise is added step by step
- After many steps, the image becomes pure noise
- This process is fixed and does not require learning

Purpose: Teach the model how data becomes noise

Reverse Process (Denoising Process)

- Starts with random noise
- A neural network predicts the noise at each step
- Noise is removed gradually
- Finally produces a realistic image

Purpose: Learn how to recover data from noise



Q10(b) Explain how DALL·E 2 uses Transformers and Diffusion models to generate images from text

(7 Marks)

DALL·E 2 is a text-to-image generation system developed by OpenAI.

It combines:

- **Transformers** for text understanding
- **Diffusion models** for image generation

Step-by-Step Working

1. **Text Encoding**
   - The input text prompt is processed using a Transformer-based model.
   - Converts text into meaningful embeddings.
2. **CLIP Model**
   - CLIP aligns text and image representations.
   - Ensures the generated image matches the text meaning.
3. **Diffusion Image Generation**
   - The diffusion model starts with noise.
   - Gradually generates an image based on text embedding.

Example

Prompt: *"A dog wearing glasses and reading a book"*
DALL·E 2 generates a relevant, realistic image matching the description.

[illegible]