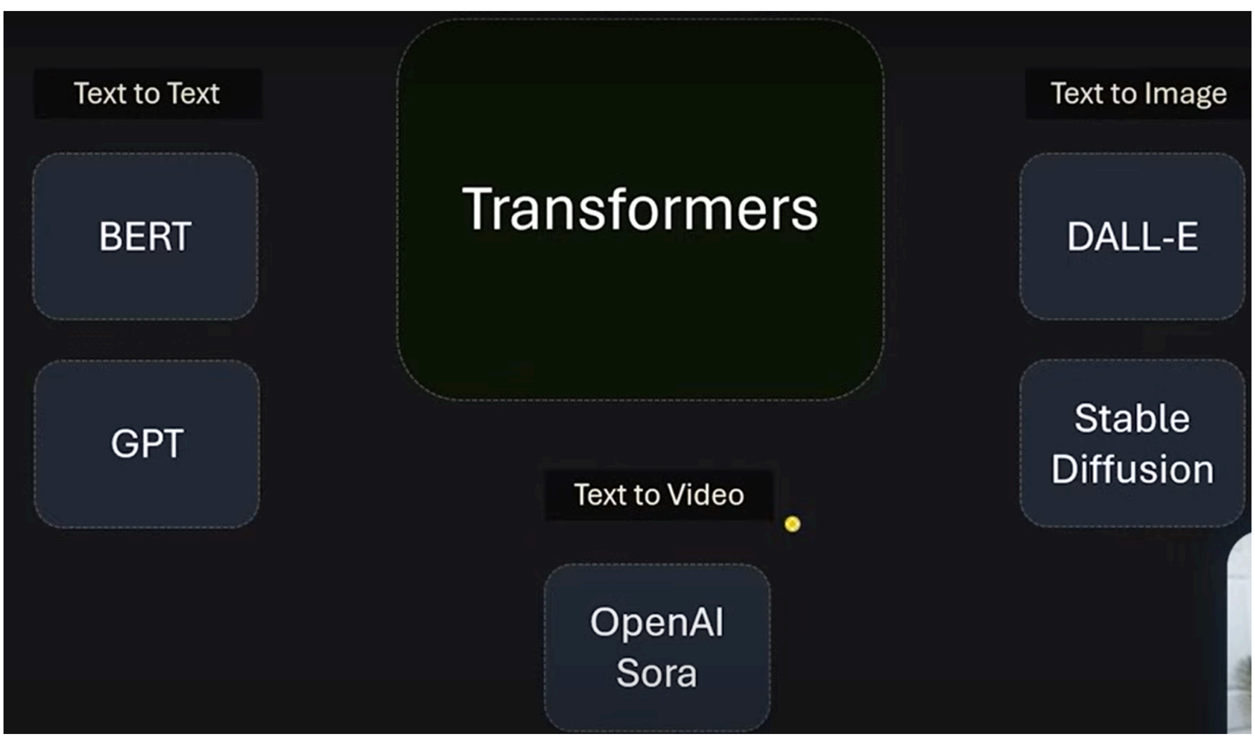
What is Generative AI?

Generative AI refers to a class of AI models that can **generate new content** like text, images, music, code, and more. It learns from existing data and produces new data that mimics the original.

Examples of GenAI Tools:

Tool/Model	Provider	Use Case
GPT-4	OpenAI	Text generation, chatbots, code
Gemini (Bard)	Google DeepMind	Multimodal GenAI, assistant
Claude	Anthropic	Safer chatbot, assistant use
LLaMA	Meta	Open-source language models
DALL·E	OpenAI	Image generation
Stable Diffusion	Stability AI	Image synthesis



Transformers

- Transformers are the foundation of modern language models like GPT, BERT, T5, Gemini, Claude, and LLa
- It is designed to handle sequential data, such as natural language, without relying on recurrence like RNNs or LSTMs
- They process sequences in parallel, making training faster.
- They handle long-range dependencies better than RNNs.
- They use a self-attention mechanism to focus on important parts of the input.

Basic Transformer Architecture Overview

Transformer has two main parts:

1. Encoder: Reads and understands the input text.
2. Decoder: Produces the output text, based on the encoder's understanding.

Each part contains multiple layers with:

- Multi-head self-attention mechanism
- Feed-forward neural networks
- Add and normalize layers

Encoder and Decoder Roles

Encoder

- Takes the entire input at once and processes it using self-attention.
- Outputs an encoded representation of the input sequence.

Decoder

- Takes the previously generated output tokens and applies masked self-attention (to avoid looking ahead).
- Uses encoder-decoder attention to focus on relevant input parts.
- Predicts the next word one at a time.

Transformer Components

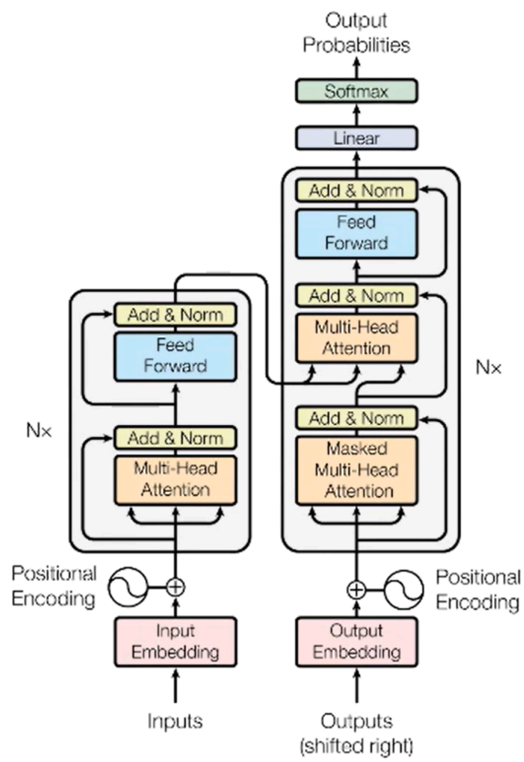


Figure 1: The Transformer - model architecture.

Components of the Transformer

A. Token Embedding

- Words are converted into numerical vectors (called embeddings).
- These vectors capture the semantic meaning of words.
  Example: The word "dog" might become a vector like [0.12, -0.31, 0.55, ...]

B. Positional Encoding

- Since Transformers do not process words in sequence order, positional encoding is added to give the model a sense of word order.
- These are mathematical patterns added to embeddings to indicate positions.

C. Multi-Head Self-Attention

This is the core of the Transformer.

- Each word (token) in a sentence can "attend to" every other word.
- The model calculates attention scores to determine which words are most relevant.

Steps:

1. Create three vectors from the input embeddings: Query, Key, and Value.
2. Compute attention scores by comparing the Query of one word with the Keys of all other words.
3. Use the scores to weigh the Values and produce a new representation of each word.

Multi-head attention allows the model to learn different relationships (such as grammar or meaning) in parallel.

D. Feed Forward Network

- After attention, the model uses a standard fully connected neural network to further process the output.
- It is applied individually to each token.

E. Add and Layer Normalization

- A shortcut connection is added between the input and output of each layer (residual connection).
- Layer normalization is used to stabilize training.
- A deep learning architecture used in GenAI models.
- Introduced in the 2017 paper: *"Attention is All You Need."*
- Works using **Self-Attention**, allowing the model to consider context from the whole sequence at once.

Transformer Workflow (Text Generation Example)

Input: "Once upon a time"

Steps:

1. Convert each word to embeddings.
2. Add positional encoding.
3. Pass through encoder layers.
4. Decoder starts with "<start>" token.
5. Decoder attends to encoder output and generates the next word.
6. Repeat step 5 until stop token is generated.

Transformer Flowchart:

Text

CopyEdit

```
Input Text
   ↓
Tokenizer → Embeddings
   ↓
Multi-head Self-Attention
   ↓
Feed Forward Neural Network
   ↓
Layer Normalization + Residual Connections
   ↓
Output Token Probabilities
```

Model Types

Model Type	Used For	Architecture
BERT	Sentiment, QnA, classification	Encoder-only
GPT (1 to 4)	Text generation, chat	Decoder-only
T5 / mT5	Translation, summarization	Encoder-Decoder
Gemini	Multimodal (text, image, audio)	Encoder-Decoder
Whisper	Speech-to-text	Decoder

Transformer Limitations

- Requires a lot of memory and compute power.
- Attention mechanism has quadratic complexity with sequence length.
- May hallucinate facts (especially in generative tasks).
- Prone to biases learned from data.

improvements in Modern Transformers

Technique	Purpose
FlashAttention	Speeds up attention computation
Sparse Attention	Reduces memory use for long texts
Low-Rank Adaptation	Allows lightweight fine-tuning
Positional Encoding Improvements	Better understanding of long texts
Multimodal Support	Adds image, video, audio support

Large Language Models

LLM stands for **Large Language Model**.
It is a **deep learning model** trained on massive amounts of text data to **understand and generate human language**.Example models: OpenAI GPT-3, GPT-4, Google's Gemini, Meta's LLaMA, Anthropic's Claude.

Key Features of LLMs

- **Large Scale**: Trained on billions of words and documents.
- **Context Understanding**: Understands sentence context and relationships between words.
- **Multi-task Learner**: Can answer questions, summarize, translate, code, write essays, and more.
- **Pretrained**: Trained first on general language, then fine-tuned for specific tasks.

How LLMs Work (High-Level Steps)

	Description
1	**Input Prompt** – You give a sentence/question (e.g., "Explain gravity")
2	**Tokenization** – The text is broken into tokens (e.g., words or word-parts)
3	**Model Inference** – Neural network processes the tokens to predict next words
4	**Decoding** – Tokens are converted back into human-readable text
5	**Output Generation** – Model gives a full answer/sentence based on your input

LLM Architecture (Behind the Scenes)

LLMs are based on **Transformer Architecture**, which includes:

- **Self-Attention** – Focuses on relevant parts of the sentence.
- **Feedforward Layers** – Layers that process each token’s context.
- **Positional Encoding** – Maintains word order understanding.
- **Layers and Parameters** – GPT-3 has 175 billion parameters, GPT-4 even more.

Types of Language Models

Type	Description
GPT (Generative Pre-trained Transformer)	Trained to generate next word/token
BERT (Bidirectional Encoder)	Focuses on understanding context in both directions
T5, BART	Used for summarization, translation, etc.

Applications of LLMs

- **Text Generation** – Chatbots, emails, articles
- **Code Generation** – Copilot, ChatGPT for developers
- **Question Answering** – Virtual assistants, customer service bots
- **Text Summarization** – Summarizing articles, legal docs
- **Translation** – Multilingual support
- **Sentiment Analysis** – Analyzing customer reviews

Popular LLM Platforms

Platform	Model
OpenAI	GPT-3, GPT-4
Google	Gemini (formerly Bard)
Meta	LLaMA 2

Anthropic	Claude
Cohere	Command R+

Limitations of LLMs

- May generate **incorrect or biased outputs**
- Doesn’t understand meaning like a human
- Computationally expensive
- Needs prompt tuning to behave correctly
- Can be exploited to generate harmful content if not filtered

Future of LLMs

- **Smaller, efficient models** (like LLaMA or Mistral)
- **Multimodal LLMs** – Combine text, image, video understanding
- **Personalized AI** – Fine-tuned on user-specific data
- **Agentic AI** – LLMs performing step-by-step actions automatically

Tokens

A **token** is a basic unit of text that a language model understands and processes. When you input a sentence into a model like ChatGPT or GPT-4, it breaks the sentence down into tokens before doing any computation.smallest unit of input/output processed by GenA

Depending on the tokenizer being used, a token can be:

- A whole word (for simple tokenizers)
- A part of a word (subword units)
- A punctuation mark
- A space or special character

Example: "ChatGPT is awesome!" → ["Chat", "G", "PT", " is", " awesome", "!"]

Types of Tokenizers

There are several methods used to split text into tokens:

1. **Whitespace Tokenization**
   Splits text by spaces.
   Example: "I love AI" becomes ["I", "love", "AI"]

2. **WordPiece (used by BERT)**
   Splits rare words into smaller subword pieces.
   Example: "unhappiness" becomes ["un", "##happiness"]

3. **Byte Pair Encoding (used by GPT)**
   Merges frequent pairs of bytes to form tokens.
   Efficient for languages with large vocabularies.

4. **SentencePiece (used by T5, mT5)**
   Treats input as a raw string and uses unsupervised learning to generate tokens

Prompts

- Instructions or questions you give to a GenAI model.

- Example: `"Write a poem about the moon in Shakespearean style."`

Prompt Engineering

Prompt engineering is the practice of designing effective prompts that guide the model to generate desired results.

Well-designed prompts can:

- Improve model accuracy
- Control output tone and structure
- Avoid ambiguity

Prompt Formats

Zero shot prompting

No examples given.
Prompt:
"Summarize the paragraph below."

One shot prompting

One example provided.
Prompt:
"Translate to Hindi: 'I am hungry.' → मैं भूखा हूँ
Translate to Hindi: 'Good night.' →"

Few shot prompting

Multiple examples provided.
Prompt:
"Q: What is the capital of India?
A: New Delhi
Q: What is the capital of France?
A:"

Chain of thought prompting

Prompt encourages the model to show reasoning.
Prompt:
"If there are 10 apples and 4 people, how many apples does each person get? Think step by step."

Real Time Use Cases of Prompts

1. **Chatbots**
   Prompt: "You are a helpful assistant. Answer the user's questions politely."

2. **Code generation**
   Prompt: "Write a Python function to check if a number is prime."

3. **Customer service**
   Prompt: "Write a response to a customer complaining about late delivery."

4. **Creative writing**
   Prompt: "Write a poem about space in the style of Shakespeare."

Core Components of GenAI Model:

Component	Role
Dataset	Pretraining data like books, websites, dialogues

Model Architecture	Typically Transformer-based
Training	Predict next token (language modeling)
Fine-Tuning	Task-specific tuning (e.g., ChatGPT from GPT)
Inference	Generating output from input prompt

GenAI Pipeline:

```
User Prompt
   ↓
Tokenizer (Text → Tokens)
   ↓
Model (Transformer)
   ↓
Next Token Prediction (Tokens → Text)
   ↓
Generated Response
```

Hands-On Examples

Using OpenAI GPT via API (Python)

```
import openai

openai.api_key = "your_api_key_here"

response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[
        {"role": "system", "content": "You are a helpful tutor."},
        {"role": "user", "content": "Explain black holes simply."}
    ]
)

print(response['choices'][0]['message']['content'])
```

Using Hugging Face Transformers

```
from transformers import pipeline

generator = pipeline("text-generation", model="gpt2")
result = generator("Once upon a time in Bangalore,", max_length=50,
num_return_sequences=1)

print(result[0]['generated_text'])
```

Prompt Engineering Examples

Basic Prompt:

> "Write a story about a dog who becomes a space explorer."

Instructional Prompt:

> "Summarize the following article in 3 bullet points."

Role Prompt:

> "You are a stand-up comedian. Write a short monologue about coffee."

Use Cases of GenAI

Domain	Use Case Example
Education	Automated tutoring, question generation
Healthcare	Medical text summarization, symptom analysis
Software Dev	Code generation (e.g., GitHub Copilot)
Marketing	Ad copy creation, content writing
Design & Art	AI-generated images, logos
Customer Support	AI chatbots, automated ticket responses

Ethics and Limitations

Challenges

- **Bias** in training data
- **Hallucination** (producing wrong info)
- **Misuse** for misinformation
- **Lack of explainability**

Responsible Use

- Transparency in output
- Human-in-the-loop verification
- Ethical prompt design

Advanced Concepts

Fine-Tuning vs Prompt-Tuning

Type	Description
Fine-Tuning	Modifying weights using custom data
Prompt-Tuning	Adjusting prompts without changing weights

Multimodal GenAI (e.g., Gemini, GPT-4o)

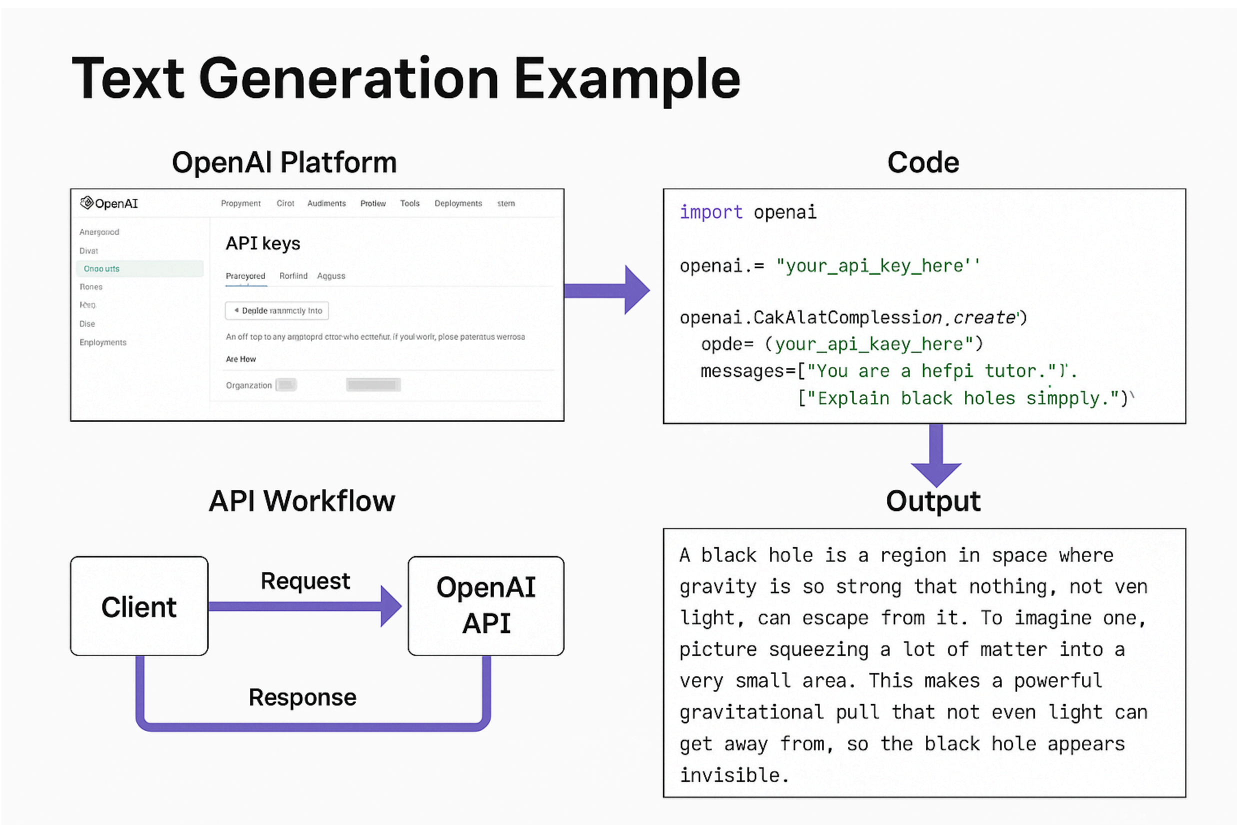
- Accepts multiple input types: **text + image + audio + video**
- Used in vision-language tasks, e.g., **image captioning**, **video Q&A**

RLHF (Reinforcement Learning from Human Feedback)

- Used in ChatGPT's fine-tuning
- Human feedback is used to train reward models
- Output is adjusted to align with human preferences

Flowchart: End-to-End GenAI Interaction

```
USER
  ↓
[ Prompt ]
  ↓
Tokenizer (Splits prompt into tokens)
  ↓
Transformer Model (Processes input)
  ↓
Generated Tokens (Predicted one-by-one)
  ↓
Detokenizer (Tokens → Text)
  ↓
Response
```



Title: Text Generation Using a Pre-trained Language Model (e.g., GPT)

Step 1: User Prompt / Input
The process starts when the user types a prompt or question, such as:
"Explain black holes simply."
This text acts as a seed or starting context for the model to generate further text.

Step 2: Tokenization (Input → Tokens)
The input sentence is broken down into tokens using a tokenizer.
Example:
"Explain black holes simply." → [50256, 2082, 1299, 17598, 13]
This step is required because the model understands numerical tokens, not plain text.

Step 3: Contextual Encoding in Neural Network
The tokens are passed through multiple layers of a Transformer neural network.
The model understands the meaning of each token, relationships between tokens, and the full context of the prompt using self-attention mechanisms.

Step 4: Probability Distribution Prediction
The model calculates probability scores for possible next tokens.
For example:
"is": 0.27
"are": 0.15
"are formed": 0.09
"by": 0.04
This helps the model decide the most appropriate next word.

Step 5: Token Selection and Decoding
Based on the probabilities, the model selects one token using decoding strategies like greedy decoding, sampling, or top-k sampling.
Then the selected tokens are converted back into human-readable text.

Step 6: Loop Until Stopping Criteria
Steps 3 to 5 are repeated in a loop, adding one token at a time.
The loop stops when a maximum length is reached, a stop token appears, or any user-defined condition is met.

Step 7: Output Display
The final generated text is shown to the user.
Example output: "Black holes are regions in space where gravity is so strong that not even light can escape."

Suggested Hands-On Exercises for Trainers

Exercise	Tools
Generate a poem using OpenAI API	Python + OpenAI API
Summarize news using Hugging Face models	`transformers`, `pipeline`
Create chatbot with prompt templates	OpenAI ChatCompletion
Try prompt-tuning with examples	Prompt design lab sessions
Compare GPT-2 vs GPT-4 outputs	Hugging Face + OpenAI