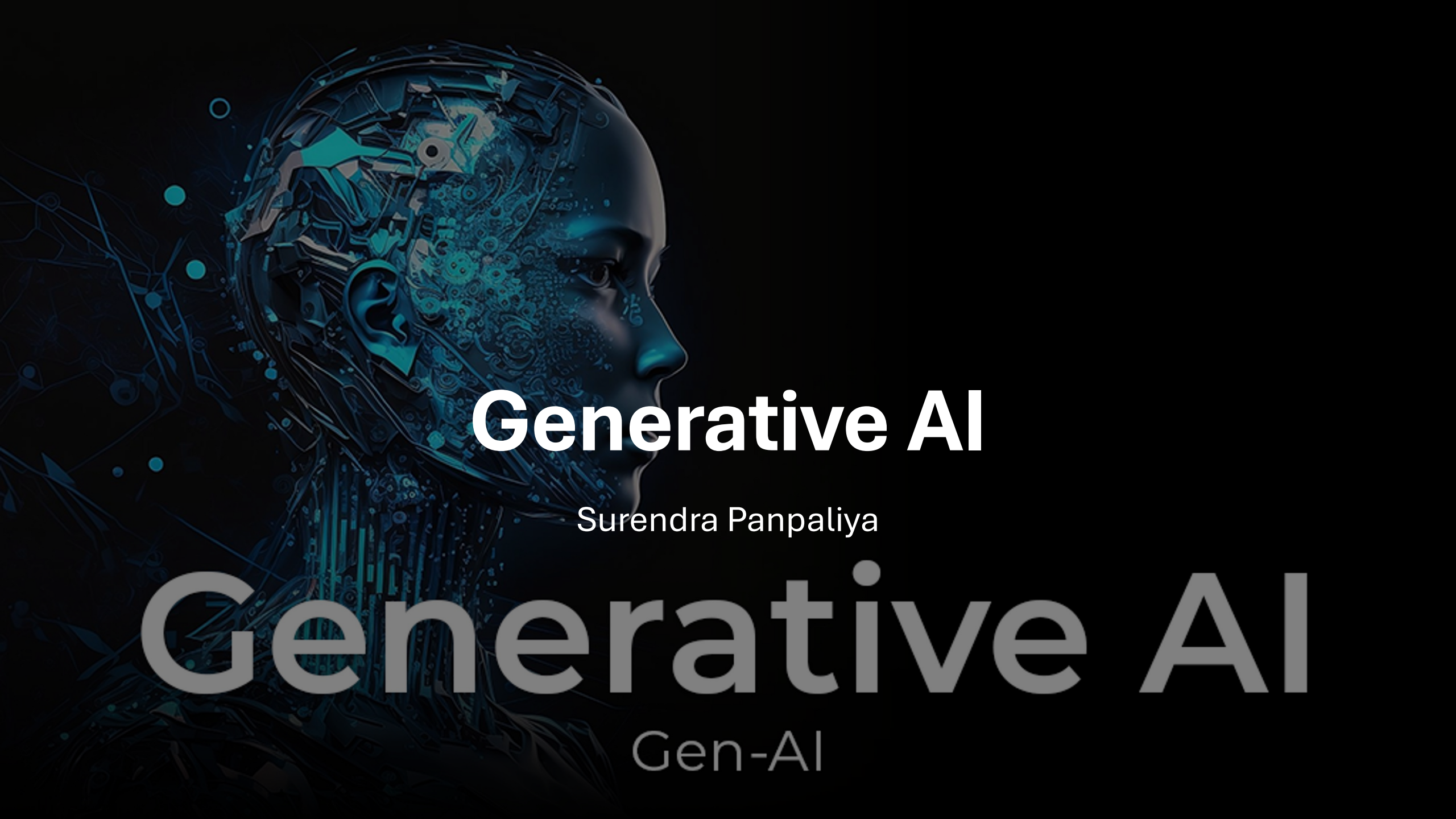


Generative AI

Surendra Panpaliya

PREREQUISITES

- Prior exposure to **Jupyter Notebooks** or **VS Code**
- Python for Data Science, Machine Learning
- Some experience with **LLMs** or **prompt engineering**

Lab Setup Requirements

Python **3.11 / 3.12**

Jupyter Notebook / VS Code

OpenAI Python SDK (openai)

OpenAI account with API key

Lab Setup Requirements

- pymilvus for Milvus Vector DB
- Streamlit
- GitHub for code submissions
- Milvus setup: Zilliz Cloud

Target Audience

Developers,

Data Scientists,

AI Enthusiasts, and

Tech Professionals interested in building

Agenda

DAY 1: Foundations of GPT-5 & Prompting Basics

DAY 2: Advanced Prompt Engineering & Control

DAY 3: Applications with GPT-5 + Milvus

DAY 4: Capstone & Enterprise Integration

Foundations of GPT-5 & Prompting Basics

Surendra Panpaliya

GKTCS Innovations

https://www.gktcs.com

Agenda

- Generative AI in Enterprise
- Evolution of GPT models
- Tokenization, embeddings,
- Context window in GPT-5

Agenda

- OpenAI Python SDK setup &
- API integration
- Basics of Prompting
- (system, user, assistant roles)

Hands-On

- Setup GPT-5 environment
- Generate text: Q&A,
- Summarization, translation
- Create a chatbot skeleton

Agenda

1. What is Generative AI?

2. Neural Generative Modeling

3. Large Language Models (LLMs)

4. Transformer Architecture & Attention Mechanism

What is Generative AI?

BRANCH OF ARTIFICIAL INTELLIGENCE

FOCUSES ON **CREATING NEW CONTENT**

TEXT, IMAGES, CODE, AUDIO, VIDEO

BY LEARNING PATTERNS FROM LARGE DATASETS.

What is Generative AI?

Large Language Models (LLMs)

Understand natural language,

Context, and intent, and

Produce human-like responses.

GenAI System

Powered by

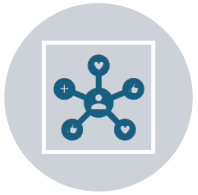
Deep learning,

Transformers,

Embeddings,

Reinforcement Learning

What is Generative AI?



Enabling them

to reason,

converse,

and create

with high accuracy.

What is Generative AI?

https://www.youtube.com/watch?v=rwF-X5STYks

Neebal Technologies
www.neebal.com

Generative AI Applications

- Art and Design
- Music Composition
- Content Creation
- Virtual Worlds
- Fashion and Product Design
- Financial Forecasting
- Customer Behavior Analysis
- Demand Forecasting
- Healthcare Diagnosis and Prognosis
- Fraud Detection

Why is it transformative for enterprises?

Automates repetitive knowledge work

Enhances productivity across roles

Developer, Analyst, Project Manager, Leader

Why is it transformative for enterprises?

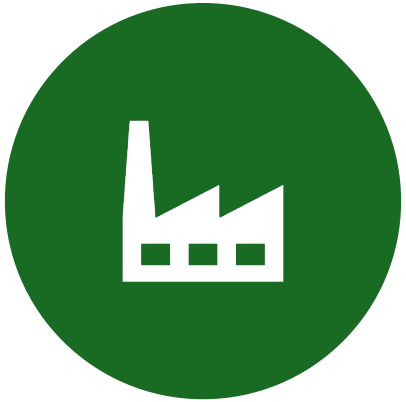
IMPROVES DECISION-MAKING WITH FASTER INSIGHTS

REDUCES TIME-TO-MARKET FOR DIGITAL SOLUTIONS

ENABLES PERSONALIZATION AT SCALE

Use Cases for Enterprises

1. BANKING & FINANCIAL SERVICES (BFSI)



2. MANUFACTURING

3. HEALTHCARE & LIFE SCIENCES

1. Banking & Financial Services (BFSI)

- Customer Support Copilots
- Fraud Detection Reports
- Risk & Compliance Automation
- Personalized Banking Advisors

Customer Support Copilots

AI-powered assistants

to handle loan queries,

credit card disputes, and

KYC FAQs.

Fraud Detection Reports

Automated analysis of

suspicious transaction patterns

with explanations in natural language.

Risk & Compliance Automation

Generating

compliance summaries,

audit reports, and

regulatory updates.

Personalized Banking Advisors

Recommending

financial products

based on customer profiles.

Example

AI generates a **weekly**

risk compliance dashboard

with insights for regulators,

saving analysts 10–15 hours per week.

2. Manufacturing

Predictive Maintenance Insights

Supply Chain Optimization

Digital Twins & Simulation

Workforce Training Copilots

Example

A **factory operations assistant**

that summarizes machine downtime logs and

suggests preventive actions.

3. Healthcare & Life Sciences

- Clinical Documentation
- Drug Discovery & Research
- Patient Engagement Chatbots
- Medical Imaging Analysis

Example

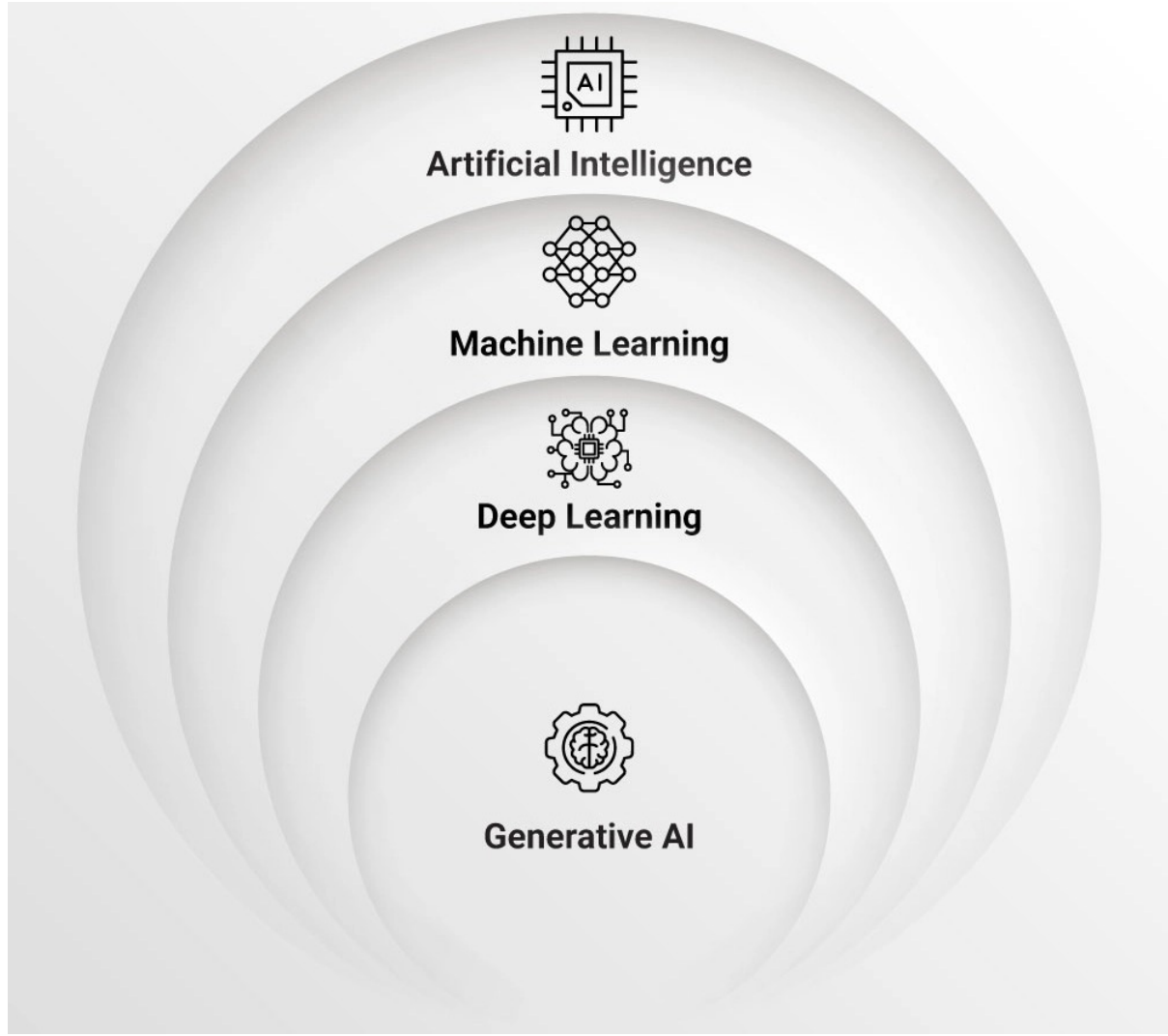
An **AI-powered patient summary generator**

that reads EHRs and produces

concise reports for doctors before consultations.

Generative AI vs Traditional AI

Feature	Traditional AI	Generative AI
Goal	Make predictions or classifications	Generate new content: text, images, code, etc.
Input → Output	Input → Fixed output (e.g., yes/no, number)	Input → Creative output (e.g., paragraph, summary)
Training	Rule-based or statistical	Trained on vast internet-scale datasets


AI
Artificial Intelligence
Machine Learning
Deep Learning
Generative AI

Artificial Intelligence

Broad concept of machines

doing tasks that typically require

human intelligence like

understanding language,

recognizing images, decision making

Machine Learning

ML IS A **SUBSET OF AI** WHERE

MACHINES **LEARN FROM DATA**

INSTEAD OF BEING EXPLICITLY PROGRAMMED.

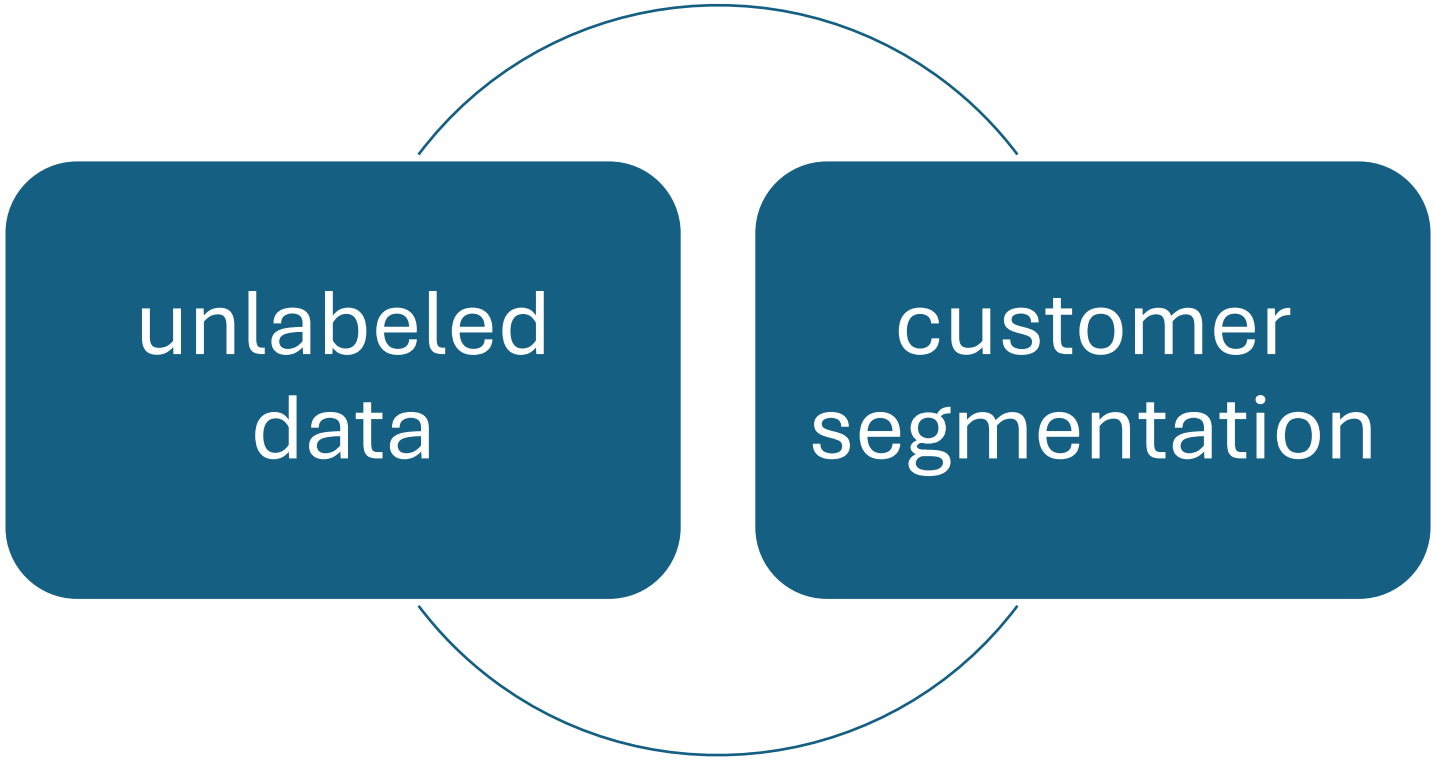
How it works?

FEED IN DATA

TRAIN A MODEL

MAKE PREDICTIONS
ON NEW DATA

Unsupervised



Reinforcement

LEARNING VIA REWARDS

DYNAMIC PRICING BOTS

Deep Learning (DL) – Neural Networks with Many Layers

Subset of ML that

uses **artificial neural networks**

(like the human brain)

with **multiple layers**

to solve complex problems.

Strengths

Handles huge datasets

Great for unstructured data (text, images, speech)

Generative AI

A branch of DL that focuses on

generating new content

like text, images, music, or code

by learning from existing data.

Large Language Models

MASSIVE GENERATIVE AI MODELS

TRAINED ON **BILLIONS OF TEXT EXAMPLES**

TO UNDERSTAND AND GENERATE

HUMAN-LIKE LANGUAGE.

Examples

GPT (OpenAI)

Claude (Anthropic)

Gemini (Google)

LLaMA (Meta)

AI vs ML vs DL vs GA vs LLMs

Feature	AI	ML	DL	GA	LLMs
Definition	Machines mimicking human intelligence	Learning from data	Neural networks with layers	Generate content from data	Giant models trained on text
Input	Rules / logic	Data + Labels	Lots of data (text, images)	Text, images, music	Massive text datasets
Output	Actions / decisions	Predictions / labels	Features / patterns	New content (text, image)	Fluent, context-aware language

Short Video

- 🎬 AI vs ML vs DL vs Generative AI – Quick Visual Explanation
- (Useful for learners and corporate teams)

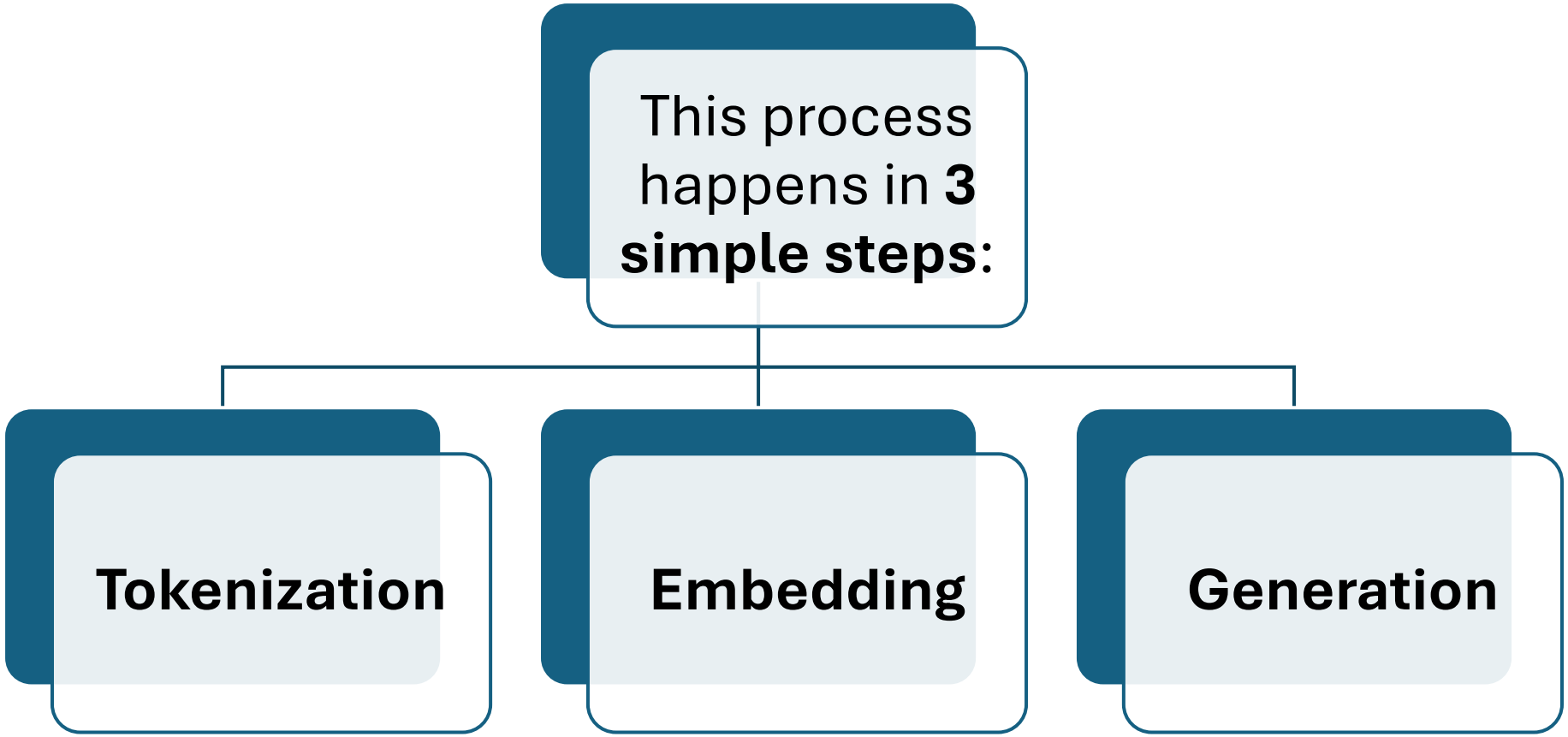
Neural Generative Modeling

TOKENIZATION: BREAKING DOWN INPUT TEXT

EMBEDDING: REPRESENTING LANGUAGE IN VECTOR SPACE

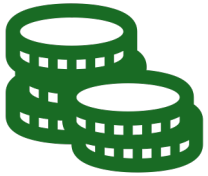
GENERATION: SAMPLING AND DECODING TECHNIQUES

What is Neural Generative Modeling?



Step 1: Tokenization

Break the input sentence



into **smaller units (tokens)**

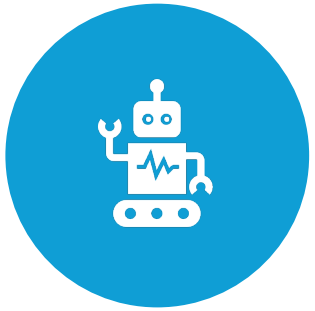
like words or parts of words

that a computer can understand.

Step 2: Embedding

CONVERT EACH TOKEN INTO A

VECTOR (A LIST OF NUMBERS)



SO THE MODEL CAN “UNDERSTAND”

ITS **MEANING AND CONTEXT** MATHEMATICALLY.

Step 3: Generation

The model generates a smart response

using **decoding techniques** like:

Greedy Search

choosing most probable word at each step

3. Large Language Models (LLMs)

Overview of popular models:

- GPT (OpenAI)
- Claude (Anthropic)
- Gemini (Google)
- LLaMA (Meta)

What Are LLMs?

AI SYSTEMS TRAINED

ON MASSIVE
AMOUNTS OF DATA

BOOKS, MANUALS,
WEB CONTENT, CODE

The Battle of AI Models: GPT-4 vs Gemini vs Claude vs LLaMA

GPT-4 Gemini Claude LLaMA

GPT vs. Gemini vs. Claude vs Llama

Which AI is better in 2025?

https://www.youtube.com/watch?v=5KiDabAa9JY

Core Strengths & Use Cases

ChatGPT-5

Best overall performer in coding,

creative writing, and

multimodal interactions.

Excelled in calculus

https://platform.openai.com/docs/overview

Core Strengths & Use Cases

Google Gemini (2.5 Pro/Flash)

Strong in multimodal context

(text+image+audio/video),

reasoning-heavy tasks, and

cost-effective free usage.

https://gemini.google.com/app

Core Strengths & Use Cases

Anthropic Claude (4 Sonnet / Opus)

Tops safety, ethical alignment,

long-context reasoning

best accuracy in reading comprehension

without hallucinations.

https://claude.ai/onboarding?returnTo=%2F%3F

Core Strengths & Use Cases

Meta Llama 4 (Scout & Maverick)

Competitive open-source option,

excels in reasoning and

massive-context tasks,

rivaling GPT-4o in benchmarks.

Meta Llama 4 (Scout & Maverick)

Meta_UI Interface

https://www.meta.ai/?utm_source=ai_meta_site&utm_medium=web&utm_content=AI_nav&utm_campaign=06112025_moment

Evolution of GPT Models → GPT-5

Generative Pretrained Transformers (GPT)

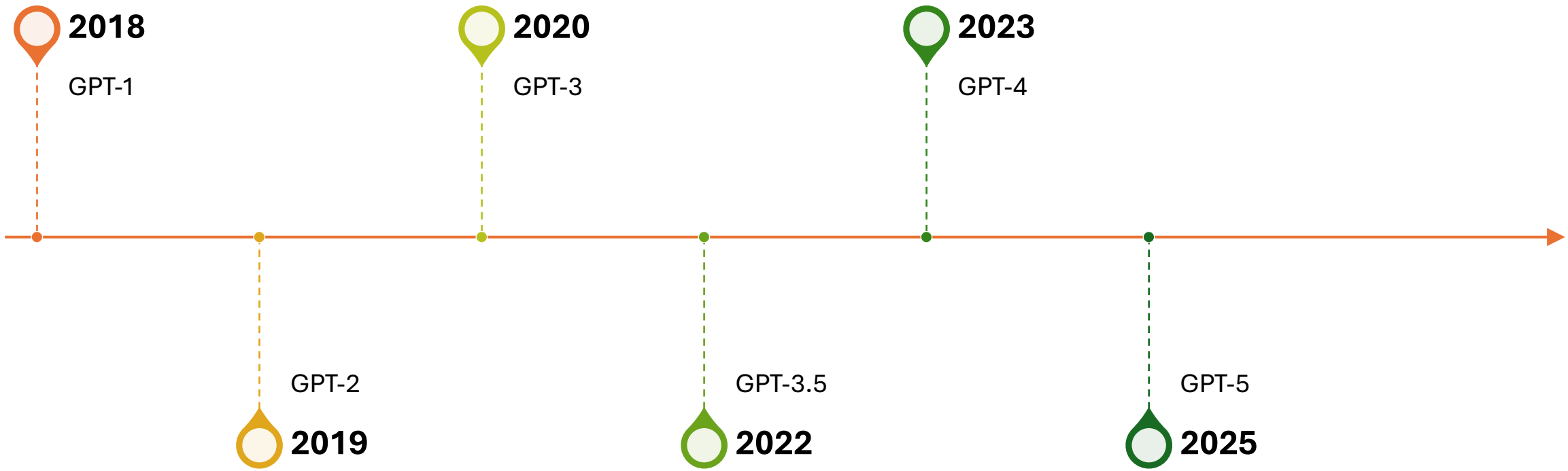
are a family of large language models

developed by **OpenAI**.

Evolution of GPT Models

Each new version represents a leap in **scale, architecture, reasoning, and enterprise applicability.**

Evolution of GPT Models



GPT-1 (2018)

The proof-of-concept.

Introduced the transformer architecture

for natural language understanding.

Trained on ~117M parameters.

GPT-2 (2019)

Showed strong text generation capabilities.

1.5B parameters.

Famous for being partially withheld

due to concerns about misuse.

GPT-3 (2020)

Breakthrough in size (175B parameters).

Demonstrated zero-shot, few-shot learning.

Powered the first wave of real-world LLM applications.

GPT-3.5 (2022)

Optimized variant of GPT-3

with reinforcement learning from human feedback (RLHF).

Provided better dialogue handling

the engine behind **ChatGPT’s early versions**.

GPT-4 (2023)

Huge improvement in reasoning,

multi-modal inputs (text + images),

reduced hallucinations,

stronger coding and enterprise reliability.

GPT-5 (2025)

Current state-of-the-art.

Expanded reasoning depth, memory,

multi-modal

text, images, audio, video

GPT-5 (2025)

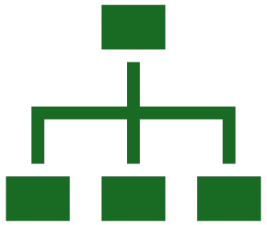
IMPROVED **FACTUAL ACCURACY**,

TOOL ORCHESTRATION,

AGENTIC AI SUPPORT.

GPT-5 (2025)

Designed for
enterprise integration



with **compliance, governance, and**

security in mind.

Comparative Chart: GPT Evolution

Feature / Model	GPT-3 (2020)	GPT-3.5 (2022)	GPT-4 (2023)	GPT-5 (2025)
Parameters	175B	~175B (optimized)	~1T est.	Multi-trillion scale (sparse/expert models)
Core Capability	Few-shot learning	Conversational fine-tuning (RLHF)	Advanced reasoning, multimodal	Deep reasoning, planning, multi-modal (text, images, audio, video)

Comparative Chart: GPT Evolution

Feature / Model	GPT-3 (2020)	GPT-3.5 (2022)	GPT-4 (2023)	GPT-5 (2025)
Context Window	2K–4K tokens	8K tokens	32K–128K	**Up to millions (long-context memory)**
Use Cases	Chatbots, Q&A, apps	Conversational AI	Enterprise copilots, coding, multimodal	**Agentic AI, enterprise copilots, compliance-ready AI assistants**

Comparative Chart: GPT Evolution

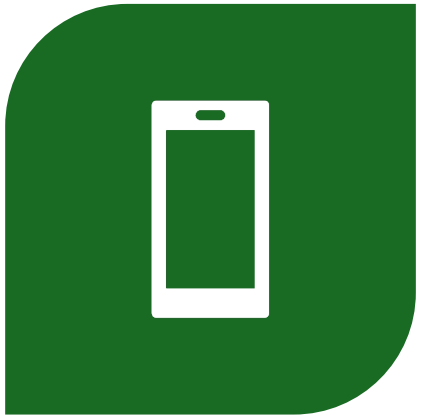
Feature / Model	GPT-3 (2020)	GPT-3.5 (2022)	GPT-4 (2023)	GPT-5 (2025)
Hallucinations	Moderate	Lower than GPT-3	Reduced further	**Minimal with retrieval + reasoning**
Fine-Tuning	Yes	Optimized	Yes, domain-specific	**Advanced domain adaptation with guardrails**

Comparative Chart: GPT Evolution

Feature / Model	GPT-3 (2020)	GPT-3.5 (2022)	GPT-4 (2023)	GPT-5 (2025)
Enterprise Focus	Startup adoption	Developer adoption	Enterprise pilots	**Enterprise-first (governance, observability, integration)**

Key Takeaway

GPT-5 REPRESENTS A SHIFT FROM



SMART ASSISTANT TO

ENTERPRISE-READY AGENT.

Key Takeaway

It not only generates but

reasons, plans, and

integrates securely

into enterprise ecosystems.

GPT-4 vs.GPT-5 parameters

GPT-4 Parameters (Developer-Facing Controls)

GPT-5 Parameters (Higher-Level Controls)

GPT-4 parameters

- Temperature
- Top-p (nucleus sampling)
- Max tokens
- Frequency penalty / Presence penalty

Temperature

Controls randomness in output.

Range: 0 → 2

lower = deterministic,

higher = more creative

Temperature

temperature=0

factual, repeatable answers

temperature=1.2

more diverse text.

Top-p (nucleus sampling)

Probability mass of tokens

When an LLM generates text,

it looks at the **probability distribution**

of all possible next words.

Top-p (nucleus sampling)

Top-p small (0.3–0.5) →

only the *very top words* are considered

More deterministic.

Top-p (nucleus sampling)

Top-p large (0.9–1.0)

wider set of possible words

More variety.

Top-p (nucleus sampling)

Top-p = 1.0

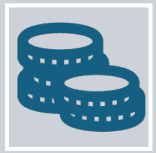
Means “no cutoff,” i.e.,

consider all tokens

(similar to default).

Max tokens

Hard cap on how long the model’s output can be.



Example: max_tokens=500

ensures the reply won’t exceed 500 tokens.

Frequency penalty / Presence penalty

Frequency penalty

Reduces repeated phrases.

Presence penalty

Encourages introducing *new topics*.

GPT-5 Parameters (Higher-Level Controls)

- Verbosity
- Directness / Formality
- Reasoning depth
- Context weighting

Verbosity

Controls the *level of detail*.

verbosity=low → concise summary,

verbosity=high → step-by-step breakdown with explanations.

Directness / Formality

Lets you specify tone

Conversational vs. Professional

Directness / Formality

directness=formal
for enterprise docs

directness=casual
for friendly
explanations.

Reasoning depth

Influences how much structured reasoning, decomposition, and self-explanation the model applies before responding.

Reasoning depth

Example:

reasoning=shallow for quick answers vs.

reasoning=deep for full problem-solving trace.

Context weighting

Ability to prioritize parts of input

system vs user instructions,

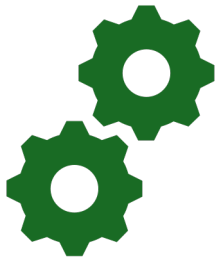
recency vs long-term memory

GPT-4 (Classic) vs GPT-5 (Next-Gen)

Aspect	GPT-4 (Classic)	GPT-5 (Next-Gen)
Creativity Control	temperature, top_p	creativity (abstracted, tuned with verbosity + reasoning)
Length Control	max_tokens	verbosity (style-based length, not just a cap)
Repetition Control	frequency_penalty, presence_penalty	Integrated into discourse management (automatic coherence)
Tone/Style	Manual prompting	Parameters like formality, directness
Reasoning	Emergent, prompt-engineered	Explicit: reasoning=deep vs reasoning=fast
Context Handling	Fixed token window	Weighted prioritization + extended memory alignment

Summary

GPT-4 is like controlling a car



with manual gears

(temperature, max tokens).

Summary

GPT-5 gives you assisted driving modes

verbosity, reasoning depth, tone

more semantic,

less low-level tweaking.

4. Transformer Architecture & Attention Mechanism

Self-Attention and its role in LLMs

Encoder-only, decoder-only, and encoder-decoder variations

Use cases and model examples for each type

What is a Transformer in LLM?

A Deep learning model architecture

introduced by Vaswani et al.

in the paper

“Attention Is All You Need” (2017)

Self-Attention: The Core Mechanism

WHAT IT DOES?

EACH WORD EXAMINES ALL OTHER WORDS

IN A SENTENCE TO DECIDE WHAT MATTERS MOST.

WORDS BECOME **QUERIES**, **KEYS**, AND **VALUES**.

Transformer Variants

a) Encoder-Only

Structure: Stack of self-attention layers (like BERT)

Best for:

Understanding tasks—classification,

search embedding, document summarization.

Transformer Variants

b) Decoder-Only

Structure:

Masked self-attention + feed-forward layers (like GPT)

Best for:

Generating text, chatbots, email copy.

Transformer Variants

c) Encoder-Decoder

Structure:

Encoder processes input →

Decoder uses both self- and cross-attention

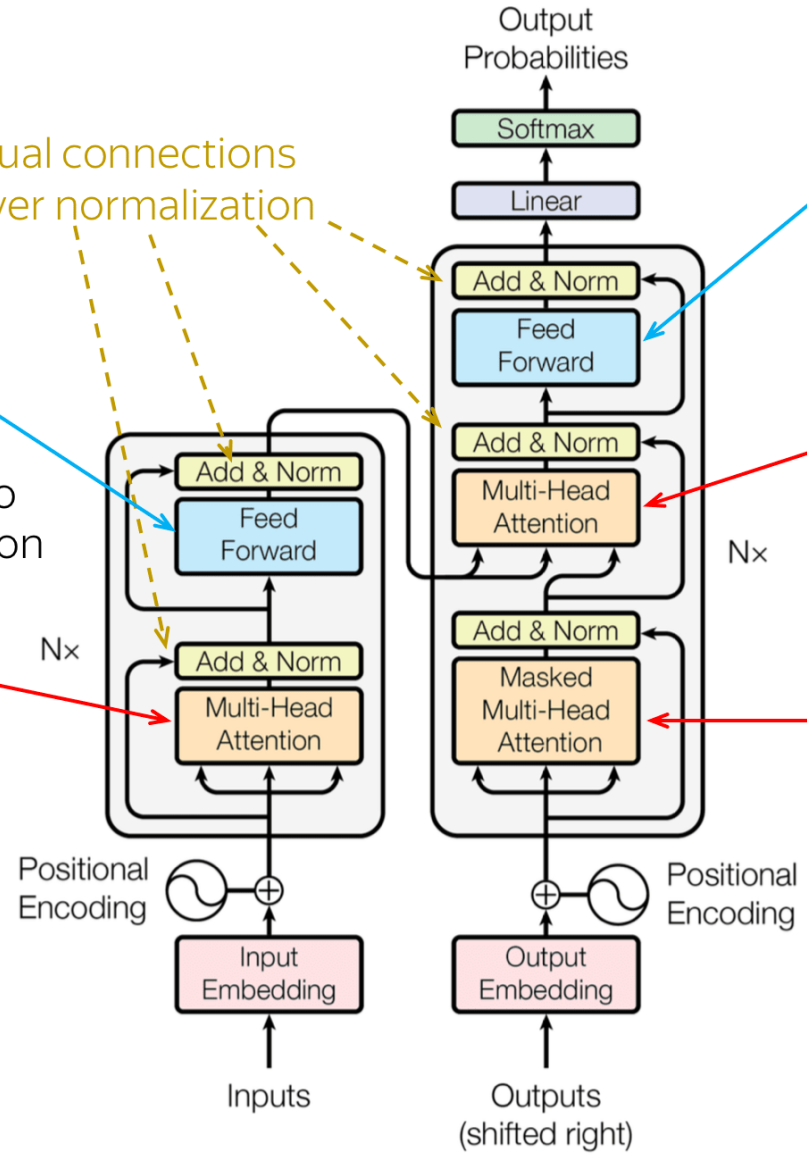
(original “Attention is All You Need”)

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention: target token looks at the source

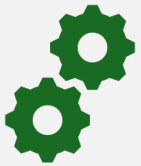
queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states

ENCODER (Left Side)

Takes the input sentence and



processes it into

an abstract representation

Input Embedding + Positional Encoding

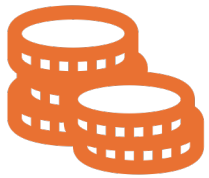
Words are converted into

vectors (Input Embedding).

Positional information is added

so the model knows the order of words.

Encoder Self-Attention (Multi-Head Attention)



Each word (token)

attends to all other words

in the sentence.

Helps understand context.

Encoder Self-Attention Example

in “he saw a bat,”

“bat” can mean an animal or

a sports item depending

on the context.

Why Is It Called Multi-Head Attention?

Multi-head means

We **split the attention into**

Multiple parallel “heads”

Each head learns to focus on

different types of relationships or patterns.

Add & Norm

Output of attention is added back

to the input (residual connection).

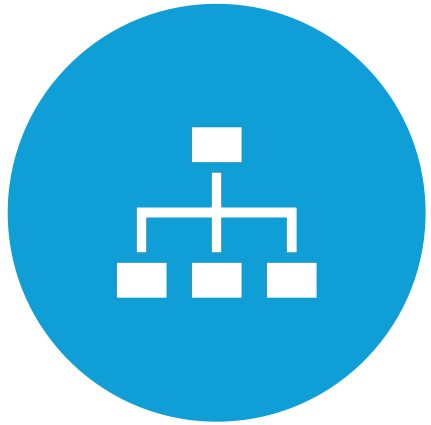
Layer normalization is applied

to stabilize training.

Feed-Forward Network (FFN)

A SMALL NEURAL NETWORK APPLIED

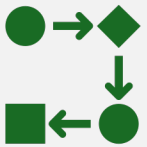
INDEPENDENTLY TO EACH TOKEN.



PROCESSES THE INFORMATION FURTHER.

Add & Norm

Another residual connection and normalization.



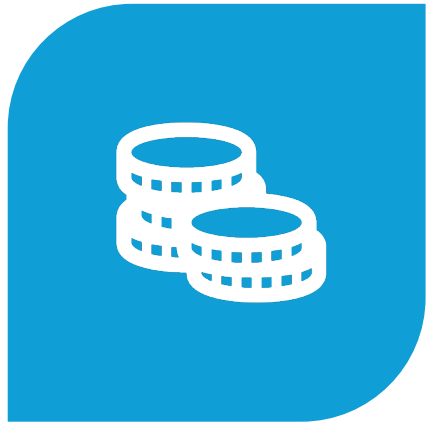
This is **repeated N times**,

producing contextualized token representations.

DECODER (Right Side)

GENERATES THE OUTPUT

LIKE A TRANSLATED SENTENCE OR



NEXT TOKEN IN LLM.

Output Embedding + Positional Encoding

Previous output tokens are embedded

shifted right for training.

Positional encoding is added.

Masked Multi-Head Self-Attention

Like encoder attention,

but **masked** to prevent future tokens

from being seen

Add & Norm

Decoder-Encoder Attention

The decoder token now **attends to encoder outputs.**

This is where the decoder “looks”

at the encoded input sentence

to decide what to generate.

Add & Norm

Feed-Forward Network

Further processes the information

(same as encoder’s FFN).

Add & Norm

Also repeated N times.

FINAL STEP: Output Probabilities

A **Linear layer + Softmax** converts

the final decoder output

to probabilities over vocabulary.

FINAL STEP: Output Probabilities

The token with

the highest probability

is selected as

output (e.g., next word).

Encoder Flow Summary

Input → Embedding → Positional Encoding →

Self-Attention → Add & Norm →

Feed Forward → Add & Norm →

Output of Encoder Layer

Decoder Flow Summary

Output Shifted Right → Embedding →

Positional Encoding →Masked Self-Attention →

Add & Norm →Encoder-Decoder Attention →

Add & Norm →Feed Forward → Add & Norm →

Final Output → Linear + Softmax

Attention Flavors

Self-Attention:

Words attend to words

within the same sentence

(question or policy).

Attention Flavors

Masked Self-Attention:

Decoder-only models only

look at **prior words**

keeps output coherent in auto-generation

Attention Flavors

Cross-Attention:

Decoder aligns each output token

with the encoded input

from the encoder

Simplified Architecture Flow

Encoder-Only Flow:

Input → Tokenize →

Embed → Self-Attention →

Feed-Forward →

Output embeddings.

Simplified Architecture Flow

Decoder-Only Flow:

Prompt → Masked Self-Attention →

Feed-Forward →

Generate next word.

Simplified Architecture Flow

Encoder-Decoder Flow:

Input → Encoder → Context →

Decoder + Cross-Attention →

Output.

What is Prompt Engineering?

Prompt Engineering is the skill of

designing and structuring inputs (prompts)

to guide Large Language Models (LLMs)

like GPT-5 toward desired outputs.

What is Prompt Engineering?

It’s not just “asking a question”

it’s about **framing context, tone,**

style, constraints, and tasks

so the model delivers exactly what you need.

What is Prompt Engineering?

Think of it as

Programming the model

with natural language.

Types of Prompts

Instructional Prompts

Few-Shot Prompts

Zero-Shot Prompts

Types of Prompts

Chain-of-Thought (reasoning prompts)

Role-based Prompts

Persona + Style Prompts

Roles in GPT-5 Prompts

System Role

User Role

Assistant Role

System Role

Defines **rules, personality,**

and behavior of the assistant.

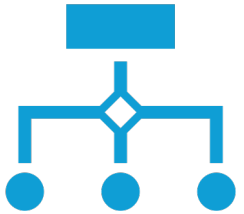
Highest priority

it sets the context that persists.

System Role

Sets the *governing policy* for the model:

persona, tone, boundaries, format.



Think of it as the “SOP” the model must follow.

User Role

Represents the **end user's input or request**.

The business request/question.

This is where you pass the task plus any fresh context.

Assistant Role

Can pre-seed examples of *ideal answers*

or tool-use traces.

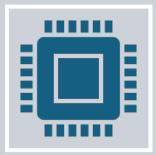
Powerful for **few-shot** prompting

and enforcing style/format.

Mental model

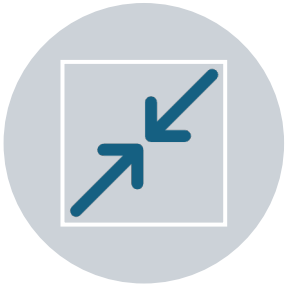
System = “Who you are + how to behave.”

Assistant = “Here’s how a great answer looks.”

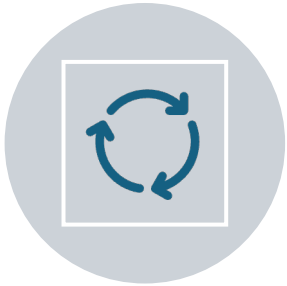


User = “Do *this* job now with this data.”

Prompt Patterns



Zero-shot:



system + user (fastest; good when rules are clear).

Few-shot:

add assistant examples to demonstrate style/structure.

Prompt Patterns

Structured output:

ask for JSON; validate downstream.

Persona switching:

tailor system role for

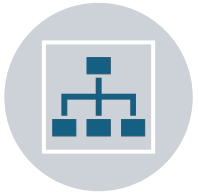
Developer / Team Lead / PM.

Summary

Prompt Engineering = designing instructions.

Types = Zero-shot, Few-shot, Instruction,

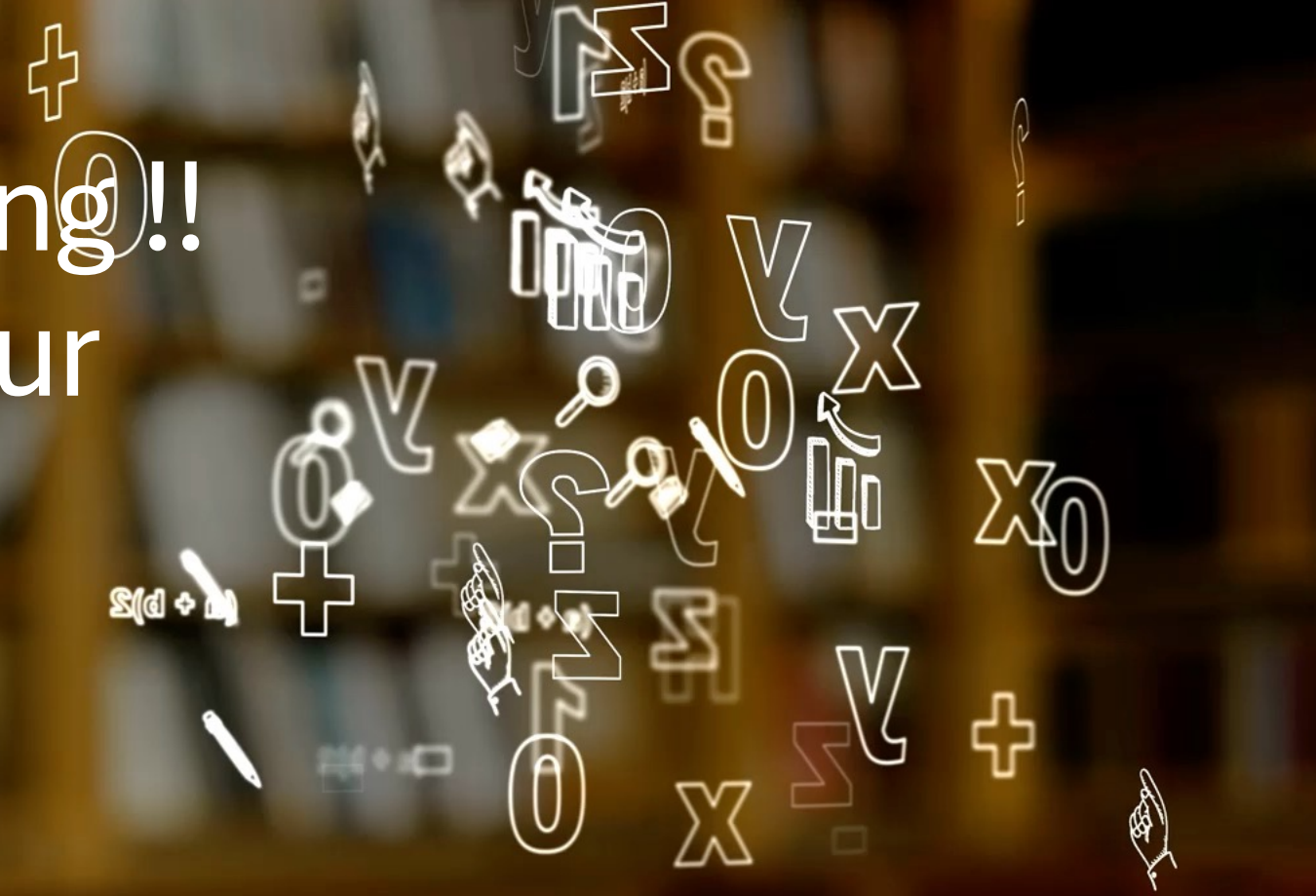
Chain-of-Thought, Role-based, Persona/Style.



GPT-5 Roles = System (rules),

User (queries), Assistant (replies/history).

Happy Learning!! Thanks for Your Patience ☺



Surendra Panpaliya

GKTCS Innovations