

Difference Between

LLM vs Fine-Tuned LLM vs RAG vs CAG



| LLIVI vs Fine-Tuned LLIVI vs RAG vs CAG | | | | |
|---|-----|----------------|-----|--|
| Feature | LLM | Fine-Tuned LLM | RAG | |

Task-specific NLP

Requires task

specific data

High

Moderate

Domain-specific

QA (BioBERT,

ClinicalBERT,

FinBERT, etc)

Fact-based NLP

Knowledge base

High (with data)

High

Enterprise search

CAG

Cache-optimized

NLP

Cache mechanism

High (with cache)

High

Performance

scaling

(E-commerce,

Gaming Chat Bots)

linkedin.com/in/anchitpancholi/

General NLP

None

(already trained on vast

amounts of diverse data)

Moderate

Low

Chatbots (ChatGPT,

Gemini, Llama,

Antropic, etc.)

Purpose

Accuracy

Complexity

Use Case /

Example

Training Need

Implementation

LLM (Large Language Model)

Large Language Models are pre-trained on vast amounts of diverse data to perform general-purpose natural language processing (NLP) tasks. They work out-of-the-box for many tasks like text generation, summarization, translation, and more. **Example: ChatGPT, Gemini, Llama, Antropic, etc.**



When to Use



Pros



Cons



Quick solutions

Applications where

- domain-specific Wide range of
- knowledge isn't critical.
- computational resources or time to

Scenarios with limited

train a custom model.

- No need for task-

specific data.

applicability.

- No additional training
 - cost (initial training cost is very high)

- Limited accuracy for domain-specific tasks.
- May generate
- **incorrect** outputs.
- Black-box nature makes it harder to interpret

irrelevant or factually

results



- **Right Use Case**
- Chatbots
- Email automation
- Basic summarization tasks.



linkedin.com/in/anchitpancholi/

Fine-Tuned LLM

Fine-tuned LLMs are Large Language Models adapted to specific tasks or domains by further training on a smaller dataset tailored to the target task. **Example: BioBERT, ClinicalBERT, FinBERT, etc.**



When to Use

- When the general-purpose model isn't sufficient for domain-specific accuracy.
 - higher precision in niche
 tasks (e.g., medical or legal
 language understanding).

Applications requiring

 Scenarios with task-specific labeled data



Pros

- Higher accuracy for specialized tasks.
 - Improved performance for specific domains.
- Flexibility in adapting to organizational needs.



Cons

- Requires labeled data
 - for fine-tuning.
- Higher computational and time costs

compared to using a

- base LLM.
- Risks of overfitting on limited data.



Right Use Case

- Sentiment analysis for customer feedback in
- Fraud detection.

specific industries.

- Since these are target task specific so can be
 - apply most of places.



linkedin.com/in/anchitpancholi/

RAG (Retrieval-Augmented Generation)

RAG combines a generative language model with a retrieval mechanism. The model retrieves relevant information from a knowledge base or external documents and uses this context to generate accurate and factual responses.



When to Use

- When the task involves
 answering questions based
 on up-to-date or domain-specific knowledge.
- Applications requiring factual correctness and interpretability.



Pros

• Highly factual responses

Can leverage up-to-date

- and external information on vector & graph
- More efficient for tasks requiring detailed knowledge.

databases.

 Easy & very quick to implement



Cons

- Requires maintaining and indexing a knowledge base.
- Latency can increase due to retrieval steps from
- Extra stroage usage, same data get converted into vector embedding

vector database

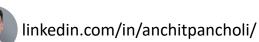
Continues data ingestion

depended



Right Use Case

- Enterprise Q&A systems.
- Document
 summarization with
- references.Knowledge discovery
- in research domains.



CAG (Cache-Augmented Generation)

CAG leverages a caching mechanism to store frequently used or generated outputs, reducing the need for redundant computation and enhancing efficiency in generation tasks.



When to Use

- When tasks involve repetitive queries or require optimized performance.
- Applications where latency
 is critical and cached
 outputs can improve
 response time.



Pros

- Faster responses by leveraging cached data.
- Reduced computational overhead for repeated queries.
- Enhanced scalability for high-demand systems.



Cons

- Requires efficient cache management and invalidation strategies.
- Risk of outdated or irrelevant cache data.
- Added complexity in integrating caching mechanisms.



Right Use Case

- Real-time chat systems
 with repeated user queries.
- High-traffic APIs needing rapid responses.
- Large-scale systems with cost optimization requirements.



linkedin.com/in/anchitpancholi/



Anchit Pancholi

Follow to know more about latest technology trends

www.linkedin.com/in/anchitpancholi/