Introduction to Large Language Models

Spring 2026

Multi-modal LLMs

(Some slides adapted from Ralph Grishman at NYU,
Yejin Choi at UWashington, N. Tomura at UDepaul, Jurafsky and Martin, CS224N,
CS224, CME295 at Stanford and other resourses on the web)

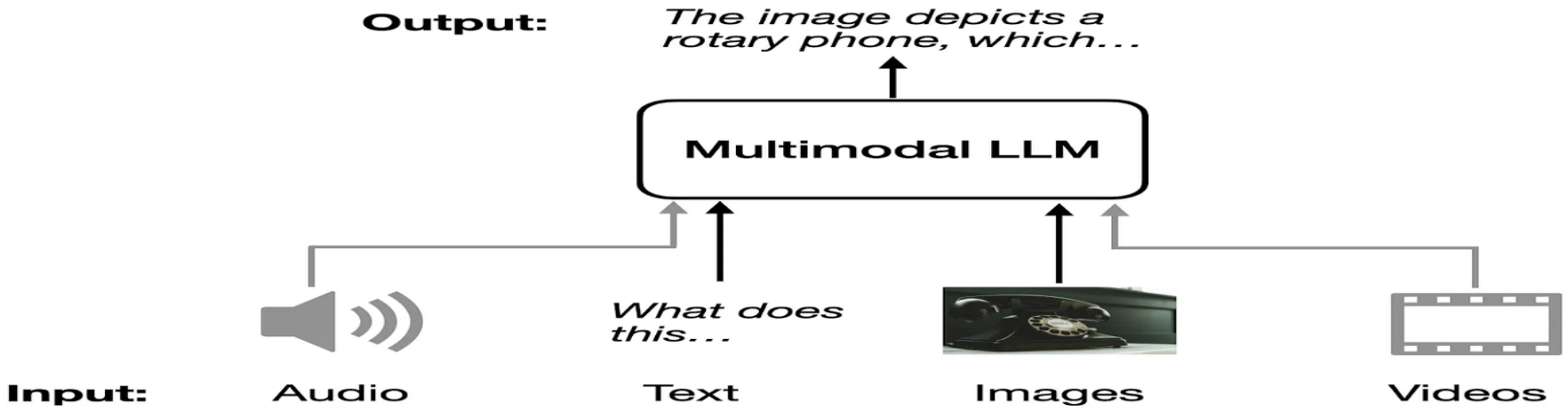
Multimodal Large Language Models (MLLMs)

A **Multimodal Large Language Model** is an AI system that can process, understand, and generate information across **multiple modalities** — not just text, but also images, audio, video, documents, and more.

Traditional LLMs (like early GPT models) were **unimodal** — they only understood and generated text. MLLMs extend this by integrating other sensory channels, much like how humans naturally combine sight, sound, and language to understand the world.

At its core, an MLLM must solve three fundamental challenges:

- **Perception** — How do you encode non-text inputs (images, audio, etc.) into a form the model can process?
- **Alignment** — How do you bridge the semantic gap between different modalities?
- **Generation** — How do you produce outputs in one or more modalities?



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

The Anatomy of an MLLM

Every MLLM, regardless of architecture, consists of some combination of these components:

[Non-text Input]

↓

[Modality Encoder] ← encodes raw input into feature vectors

↓

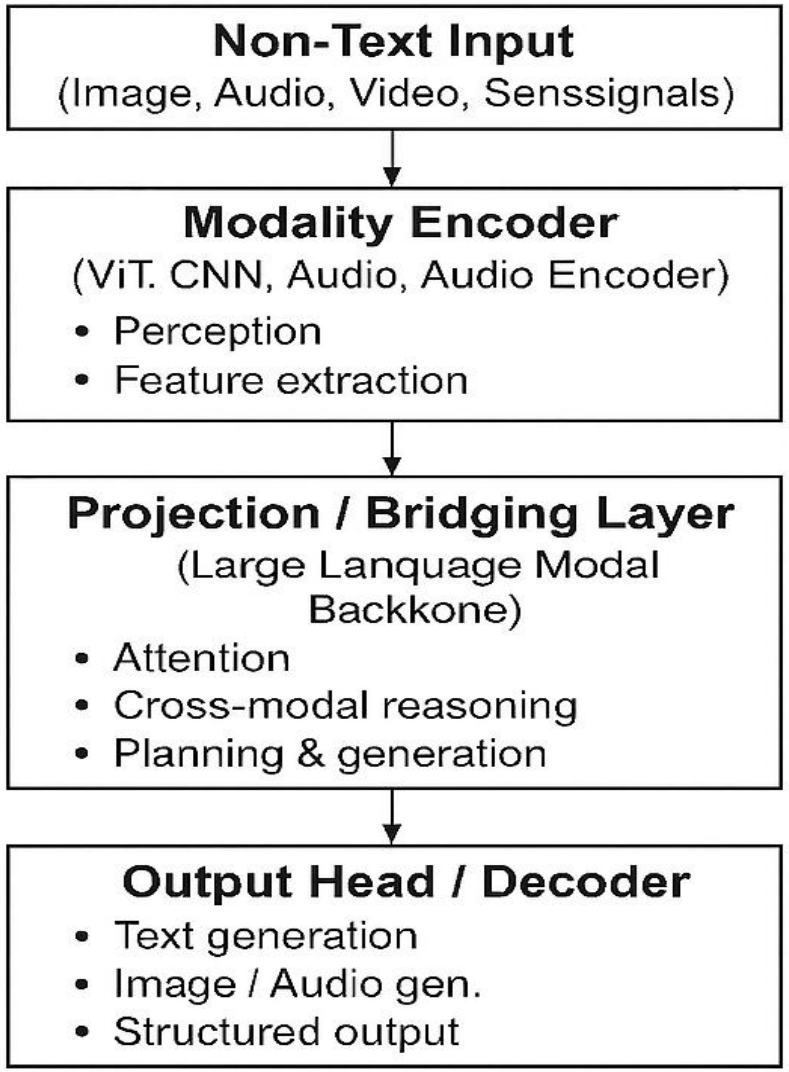
[Projection / Bridging Layer] ← aligns modality features with LLM space

↓

[Large Language Model Backbone] ← reasons, attends, generates

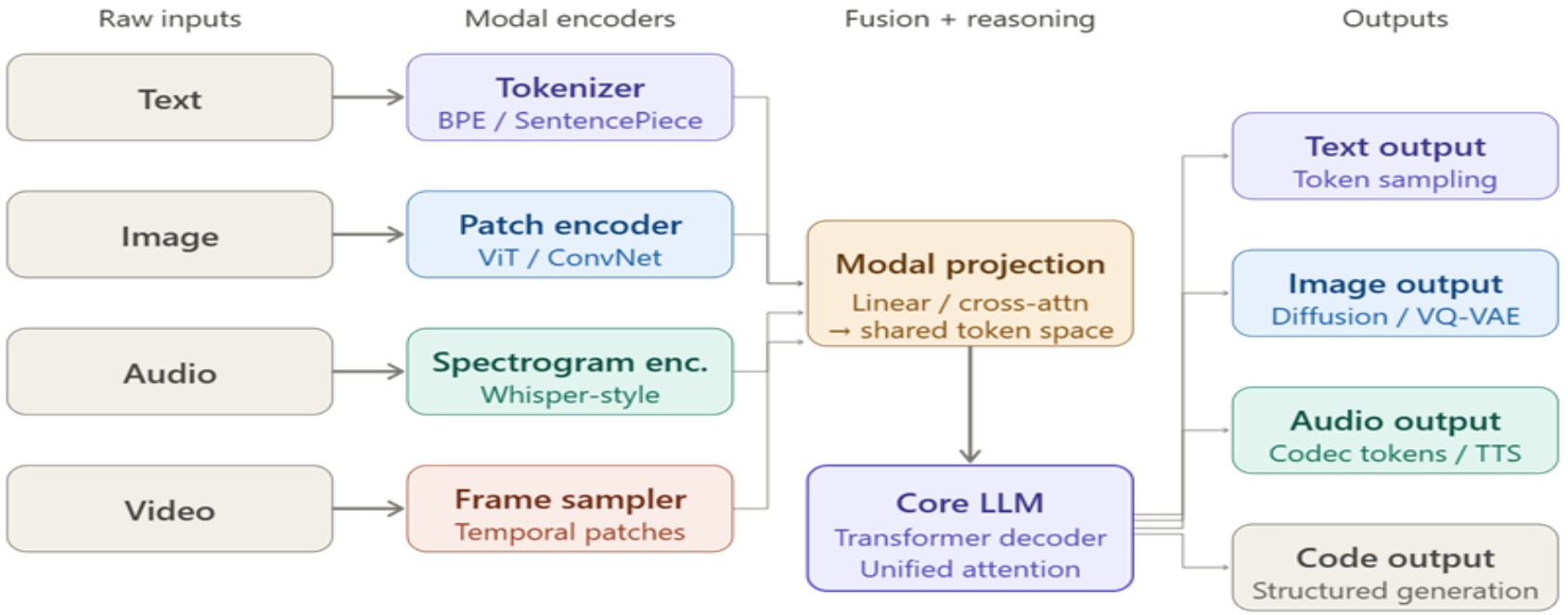
↓

[Output Head / Decoder] ← produces text, image, audio, etc.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

The Anatomy of an MLLM



Interleaved token sequence

[text tokens] [image patch tokens] [audio tokens] → all fed as one flat sequence to the core LLM

Key design challenges

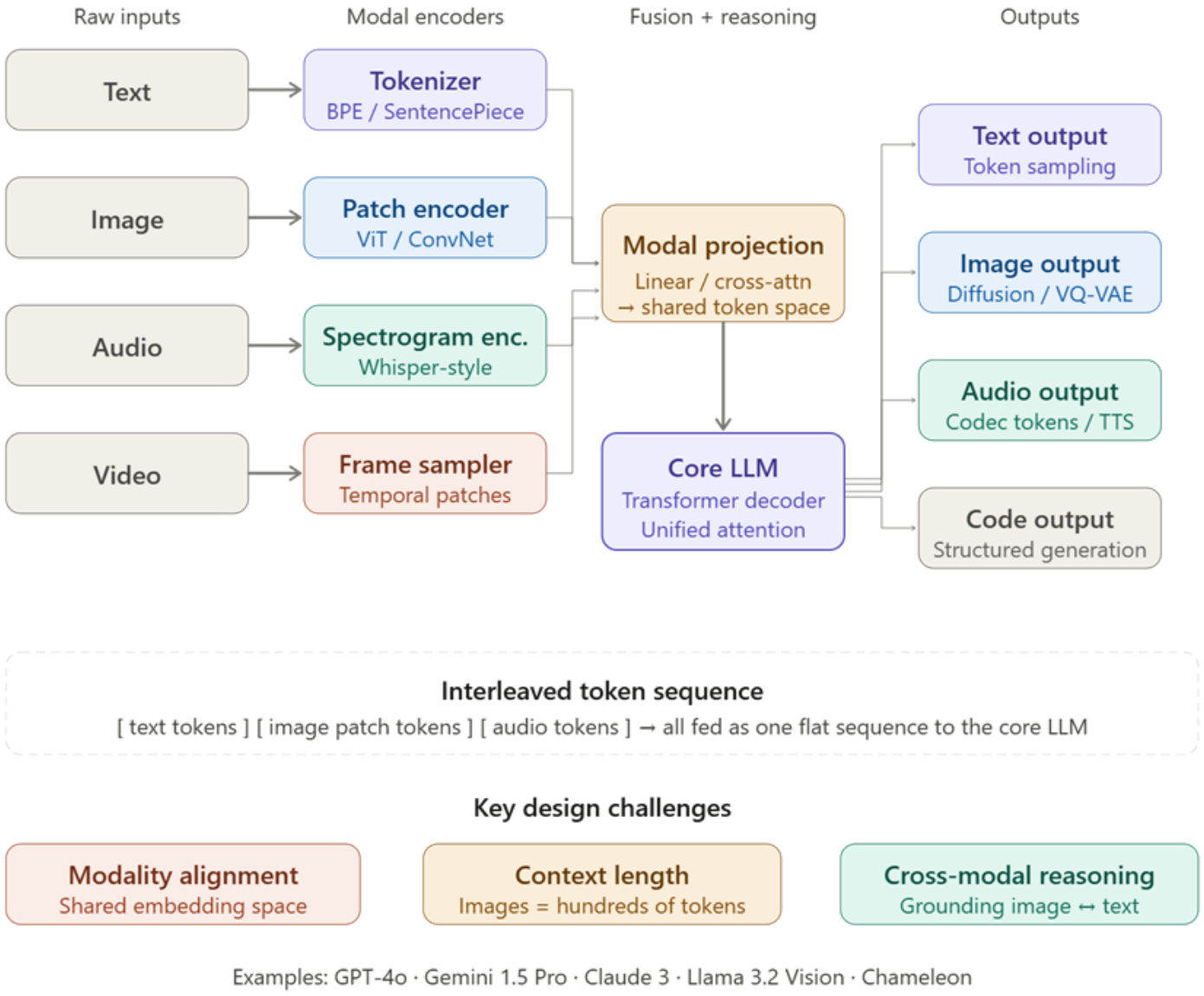
Modality alignment	Context length	Cross-modal reasoning
Shared embedding space	Images = hundreds of tokens	Grounding image ↔ text

Examples: GPT-4o · Gemini 1.5 Pro · Claude 3 · Llama 3.2 Vision · Chameleon

The Anatomy of an MLLM

Input encoders translate raw signals into vectors.

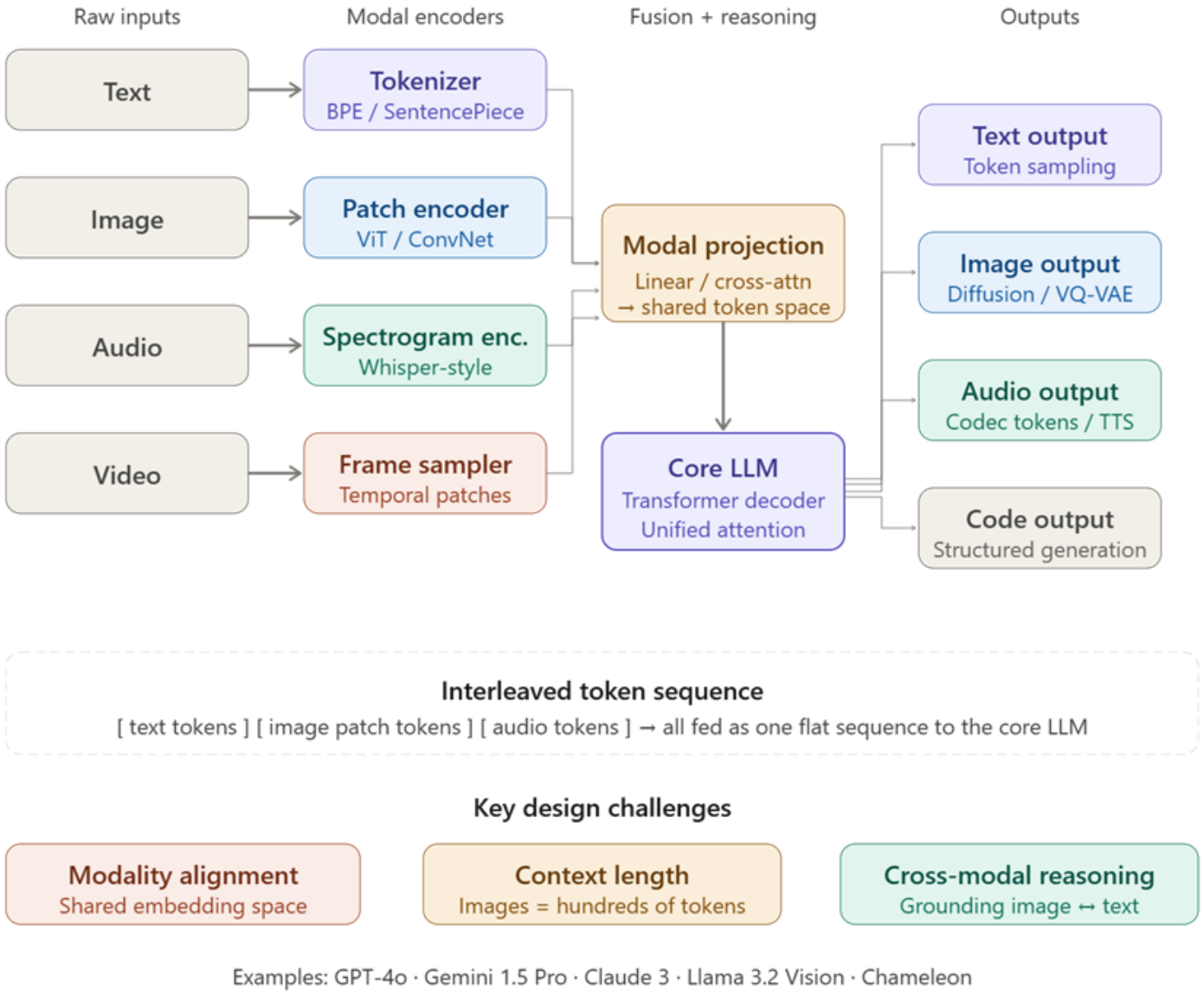
- Text uses a sub-word tokenizer (BPE or SentencePiece).
- Images are split into fixed-size patches and encoded by a vision transformer (ViT) or convolutional network.
- Audio is converted to a mel spectrogram and encoded similarly to how Whisper works.
- Video is treated as a sequence of temporal patches — essentially images with a time dimension.



The Anatomy of an MLLM

Modal projection is the critical glue layer.

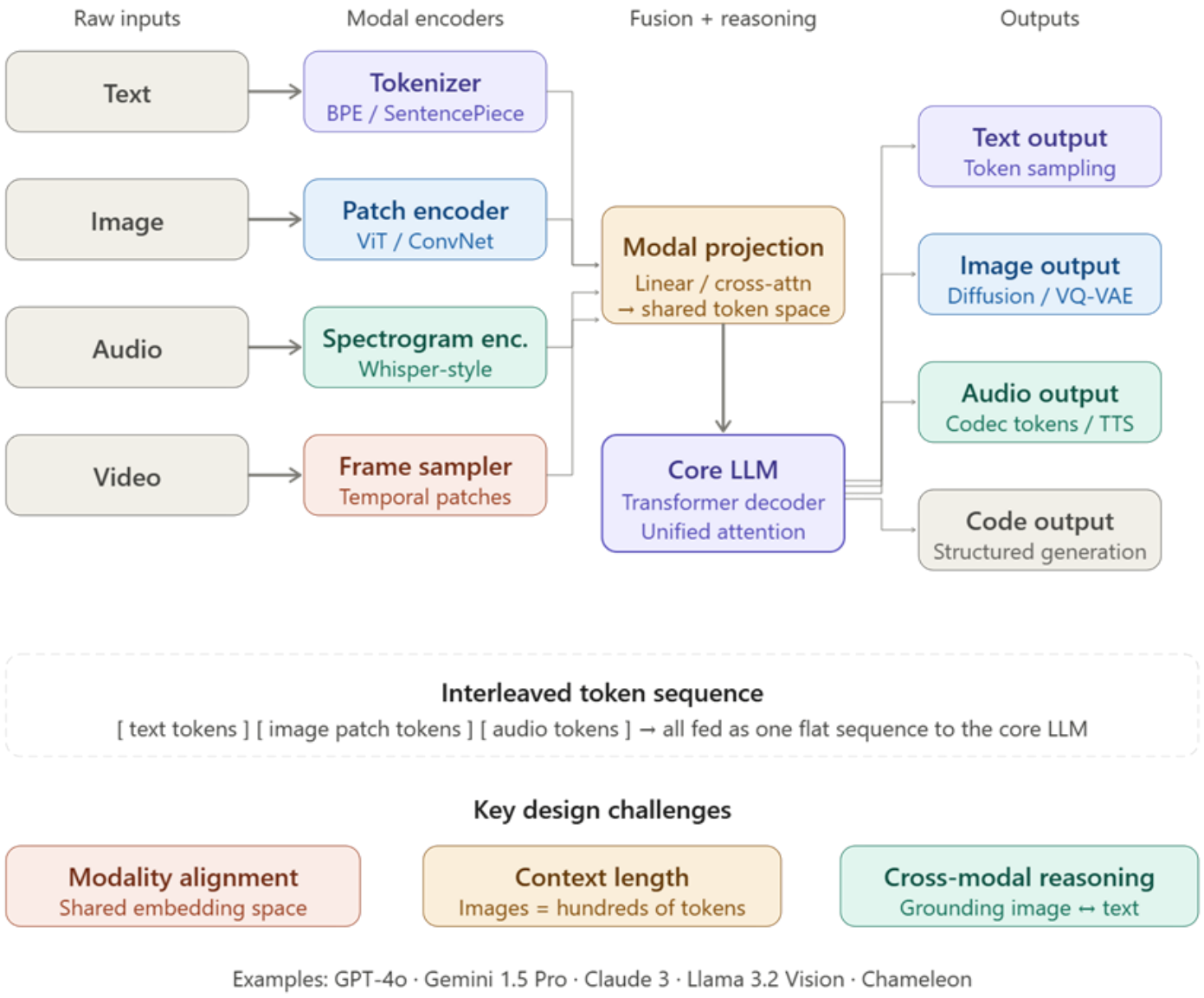
- Each encoder produces embeddings in its own space, so a learnable projection (usually a small linear layer or a cross-attention block) maps them all into the same dimensionality as the core LLM's token embeddings.
- The result is one flat sequence — text tokens and image/audio/video tokens interleaved side by side.



The Anatomy of an MLLM

The core transformer then treats this mixed sequence like ordinary language modelling.

- It has no special awareness that some tokens came from pixels rather than words — the modal alignment learning is baked in during pretraining and instruction-tuning.

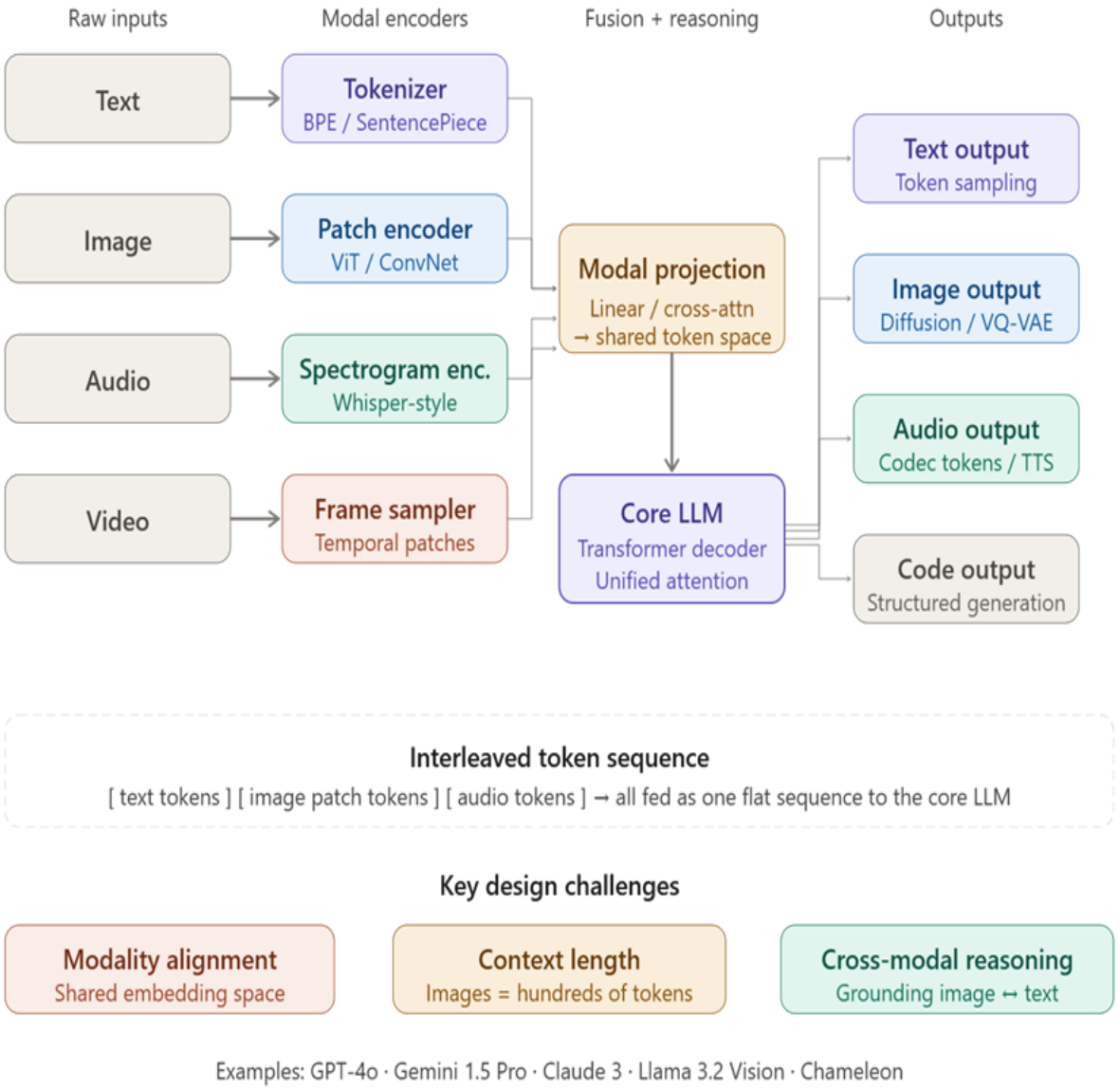


The Anatomy of an MLLM

Outputs are generated by routing the final hidden states through modality-specific decoders. Text is sampled normally. Images typically use a VQ-VAE codebook or a diffusion process. Audio output uses codec tokens (like EnCodec) or a TTS head.

The three hardest problems in building these systems are:

- **Modality alignment** — getting the projection layer to produce embeddings that are semantically comparable across modalities, usually requiring large-scale contrastive pretraining (CLIP-style).
- **Context budget** — a single image can consume 256–2000 tokens; video eats far more. Longer sequences mean higher attention cost (quadratic in the naive case).
- **Cross-modal grounding** — making the model reliably link a region of an image to specific words in the text, rather than just knowing about both separately.



Real-World Applications

These models enable tasks that were previously impossible for single-modality AI:

- **Image/Video Captioning**: Automatically describing visual content in natural language.
- **Visual Question Answering (VQA)**: Answering questions about an uploaded image or document.
- **Multimodal Search**: Finding products or information using a combination of voice, text, and photos.
- **Content Moderation**: Identifying harmful content by analyzing both the text and the accompanying video or audio cues.

Examples of multimodal models include **Gemini** from Google, **GPT-4o** from OpenAI, and open-source projects like **Macaw-LLM**.

Types of MLLMs by Modality

1. Vision-Language Models (VLMs)
The most common and mature category. Combine image understanding with language.

- **Input:** Image + Text
- **Output:** Text (or sometimes images)
- **Examples:** GPT-4V, LLaVA, Flamingo, BLIP-2, InstructBLIP, Qwen-VL, Claude 3/3.5/3.7

Capabilities: Image captioning, visual QA, OCR, diagram understanding, document parsing, scene understanding.

2. Audio-Language Models
Process speech or general audio alongside text.

- **Input:** Audio/Speech + Text
- **Output:** Text (or speech)
- **Examples:** Whisper (encoder), AudioPaLM, Qwen-Audio, Gemini (audio mode)

Capabilities: Speech recognition, speaker identification, audio event classification, spoken QA.

3. Video-Language Models

Extend vision-language models to handle temporal sequences of frames.

- **Input:** Video (sequence of frames) + Text
- **Output:** Text
- **Examples:** Video-LLaMA, VideoChat, TimeChat, Gemini 1.5 Pro (long video), InternVideo

Capabilities: Video captioning, temporal reasoning, action recognition, video QA.

Types of MLLMs by Modality

4. Document / OCR-Language Models

Specialized for understanding documents with rich layout, tables, and mixed text-image content.

- **Input:** Document image + Text
- **Output:** Text
- **Examples:** DocOwl, mPLUG-DocOwl, Donut, LayoutLMv3, Nougat

Capabilities: Form understanding, table extraction, invoice parsing, academic paper reading.

5. Any-to-Any Models (Omni Models)

The most ambitious category — handle any combination of input/output modalities.

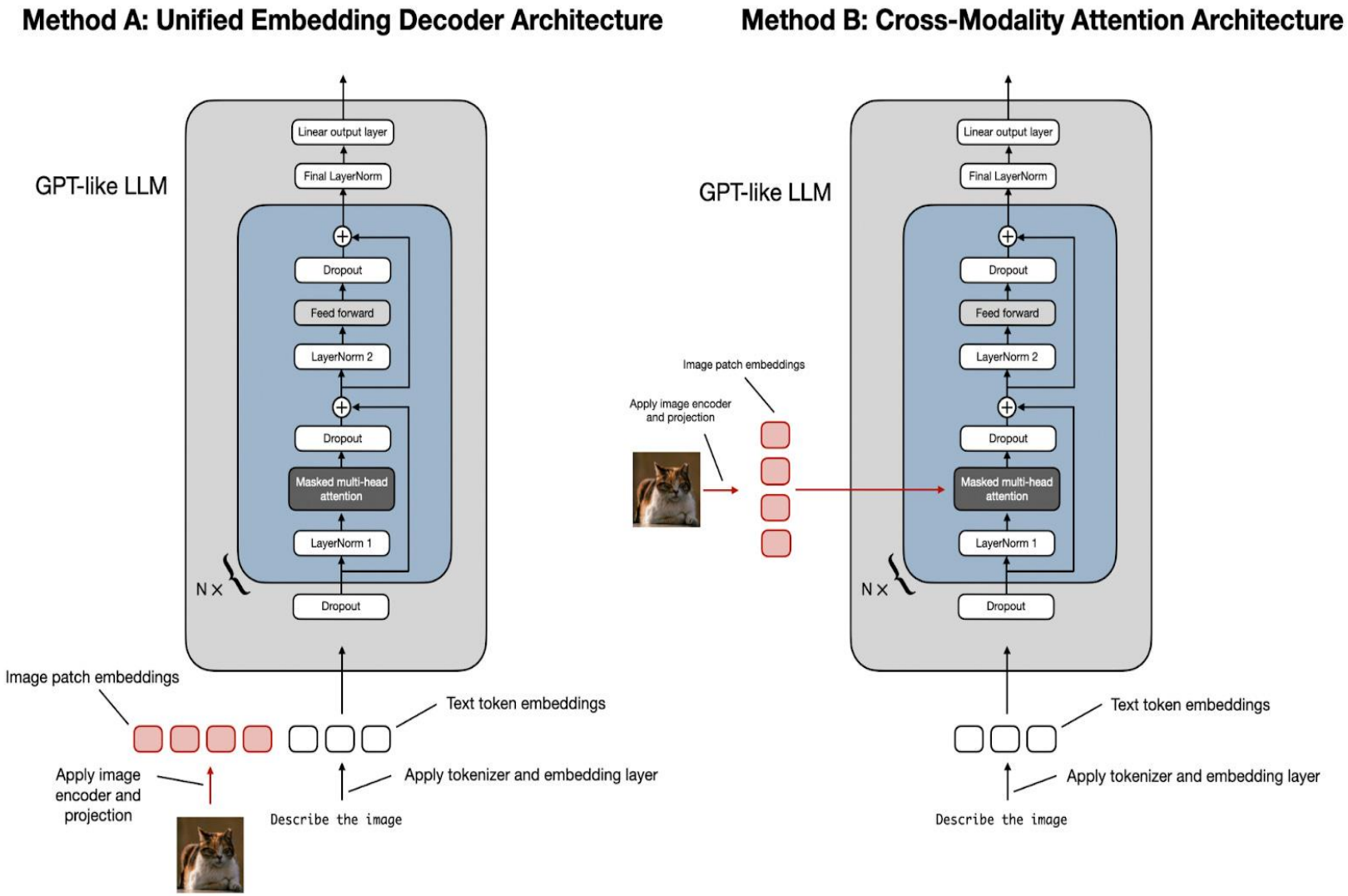
- **Input:** Text, Image, Audio, Video (any combination)
- **Output:** Text, Image, Audio (any combination)
- **Examples:** GPT-4o, Gemini 1.5/2.0, UnifiedIO, AnyGPT, CoDi

Capabilities: End-to-end multimodal conversation, real-time voice+vision interaction.

Common approaches to building multimodal LLMs

There are two main approaches to building multimodal LLMs:

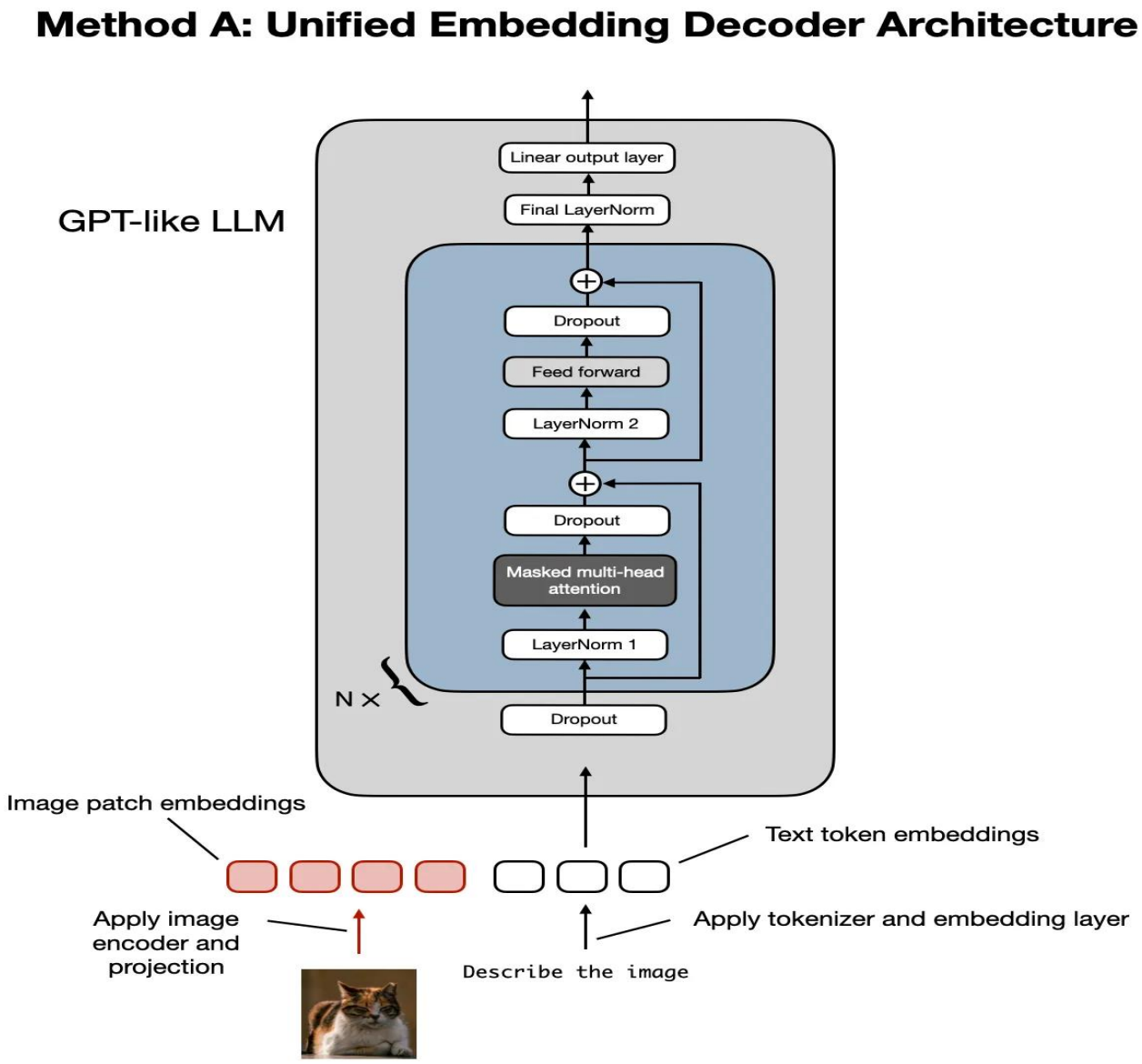
- Method A: Unified Embedding Decoder Architecture approach;
- Method B: Cross-modality Attention Architecture approach.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Method A: Unified Embedding Decoder Architecture

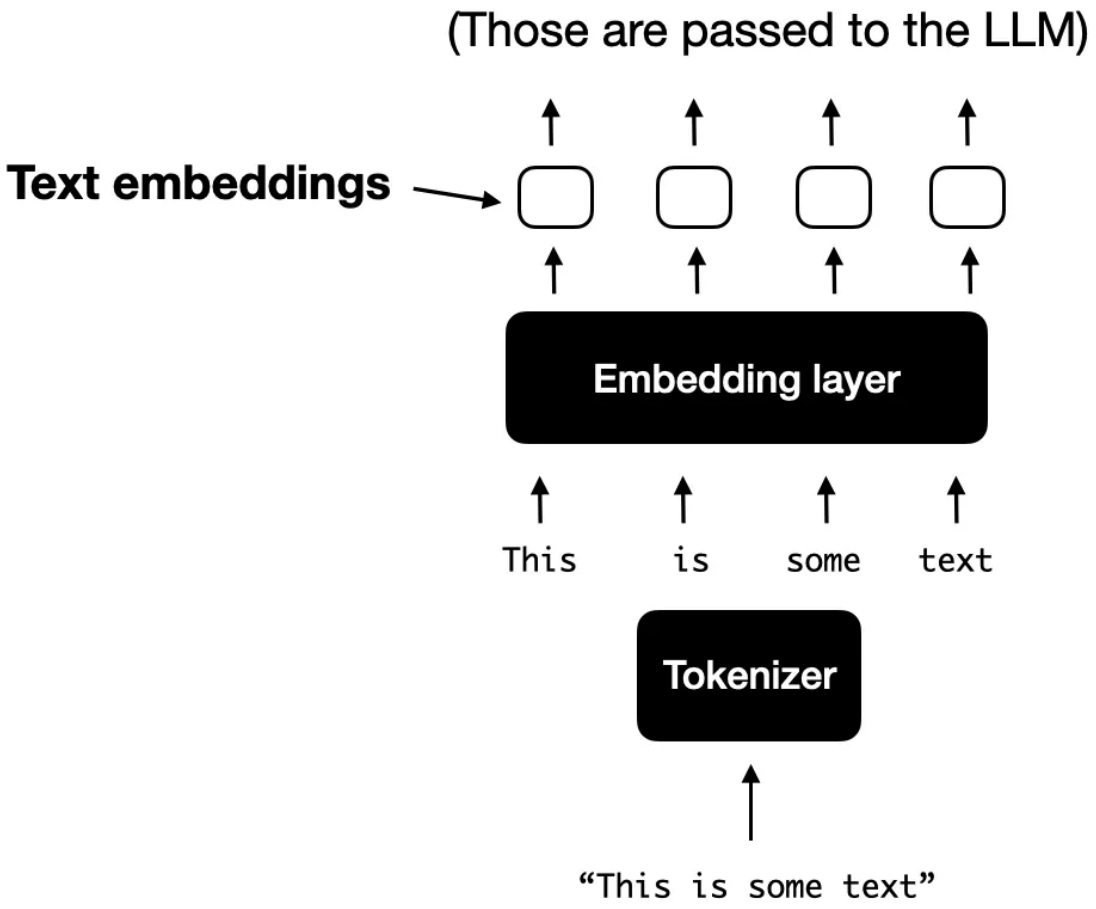
- The ***Unified Embedding-Decoder Architecture*** utilizes a single decoder model, much like an unmodified LLM architecture such as GPT-2 or Llama 3.2.
- In this approach, images are converted into tokens with the same embedding size as the original text tokens, allowing the LLM to process both text and image input tokens together after concatenation.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Tokenizing text and converting it into token embedding vectors

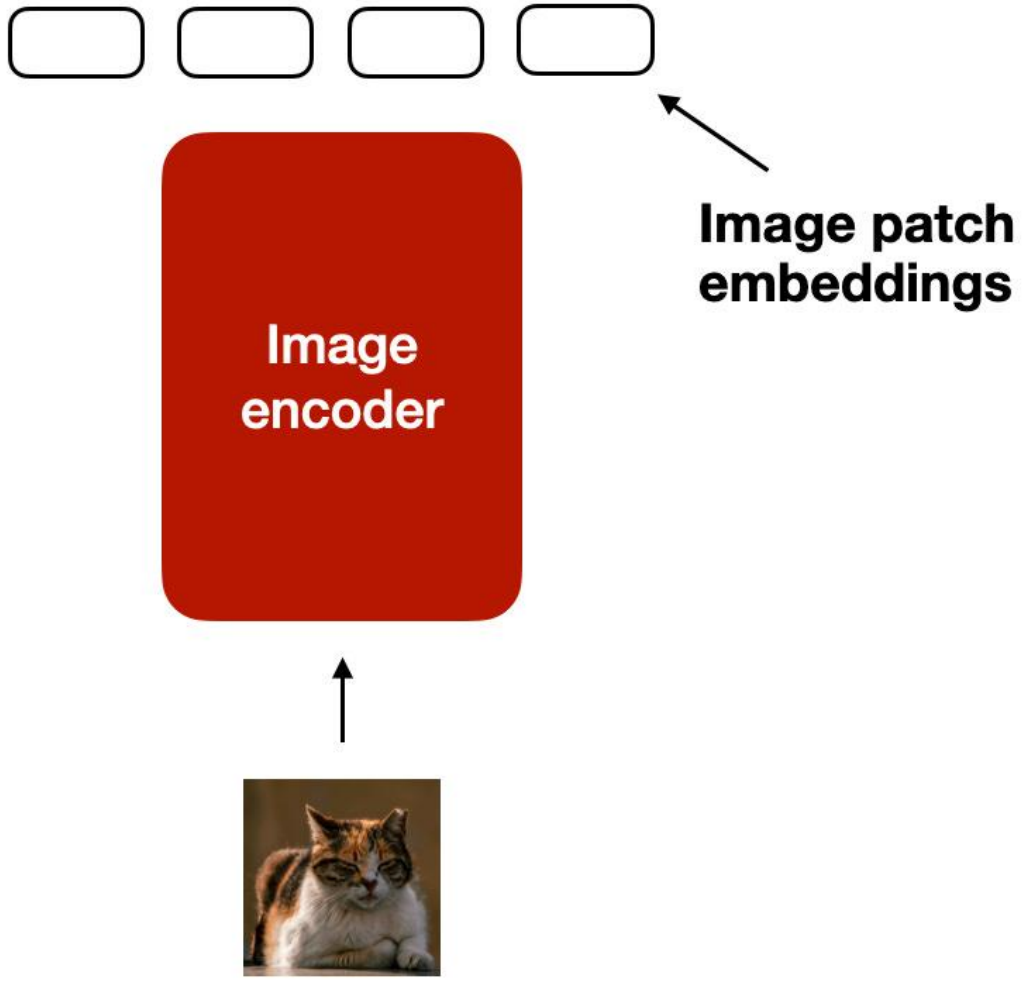
- For a typical text-only LLM that processes text, the text input is usually tokenized (e.g., using Byte-Pair Encoding) and then passed through an embedding layer, as shown in the figure below.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Understanding Image encoders

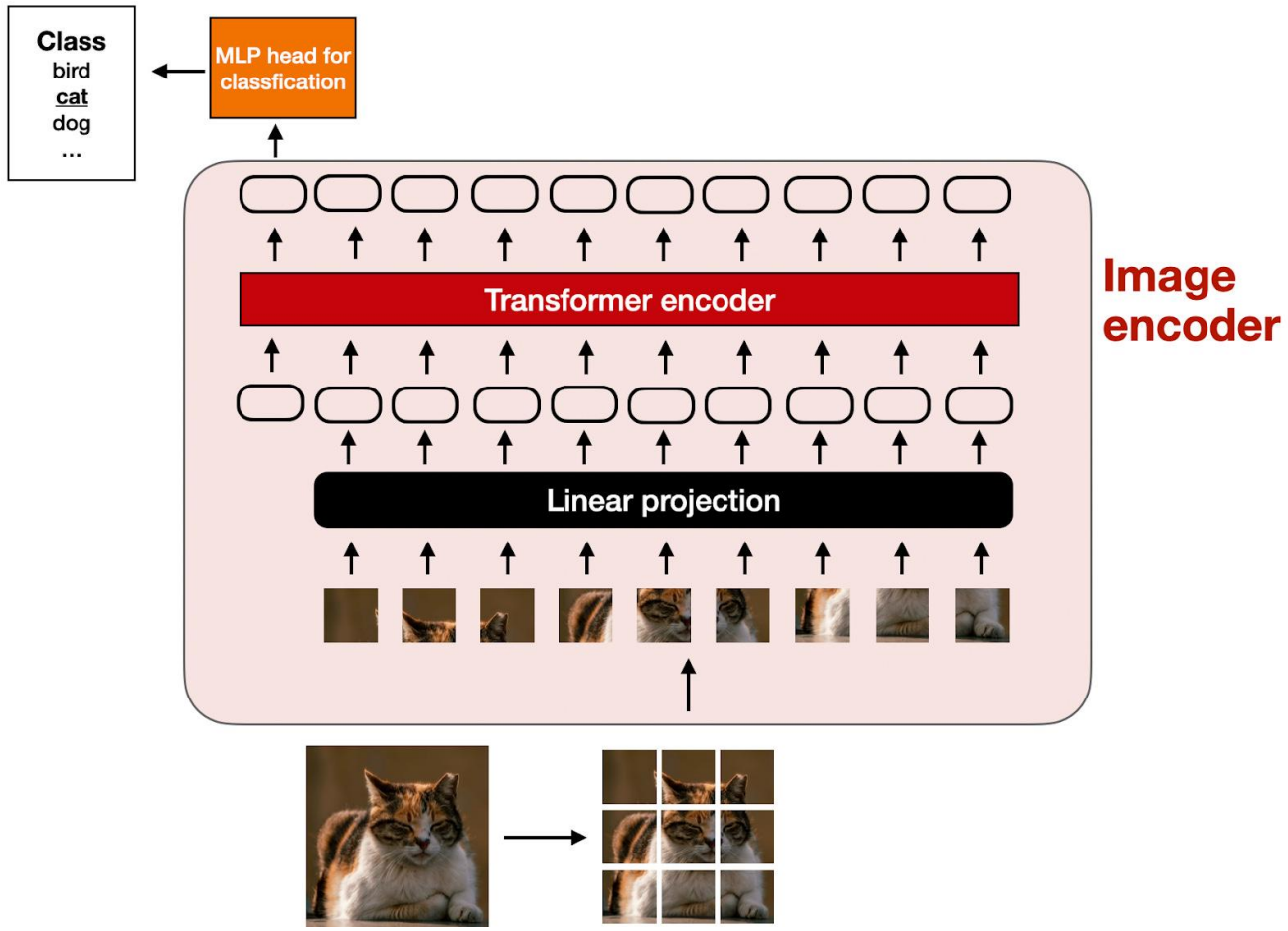
- Analogous to the tokenization and embedding of text, image embeddings are generated using an image encoder module (instead of a tokenizer), as shown in the figure below.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Understanding Image encoders

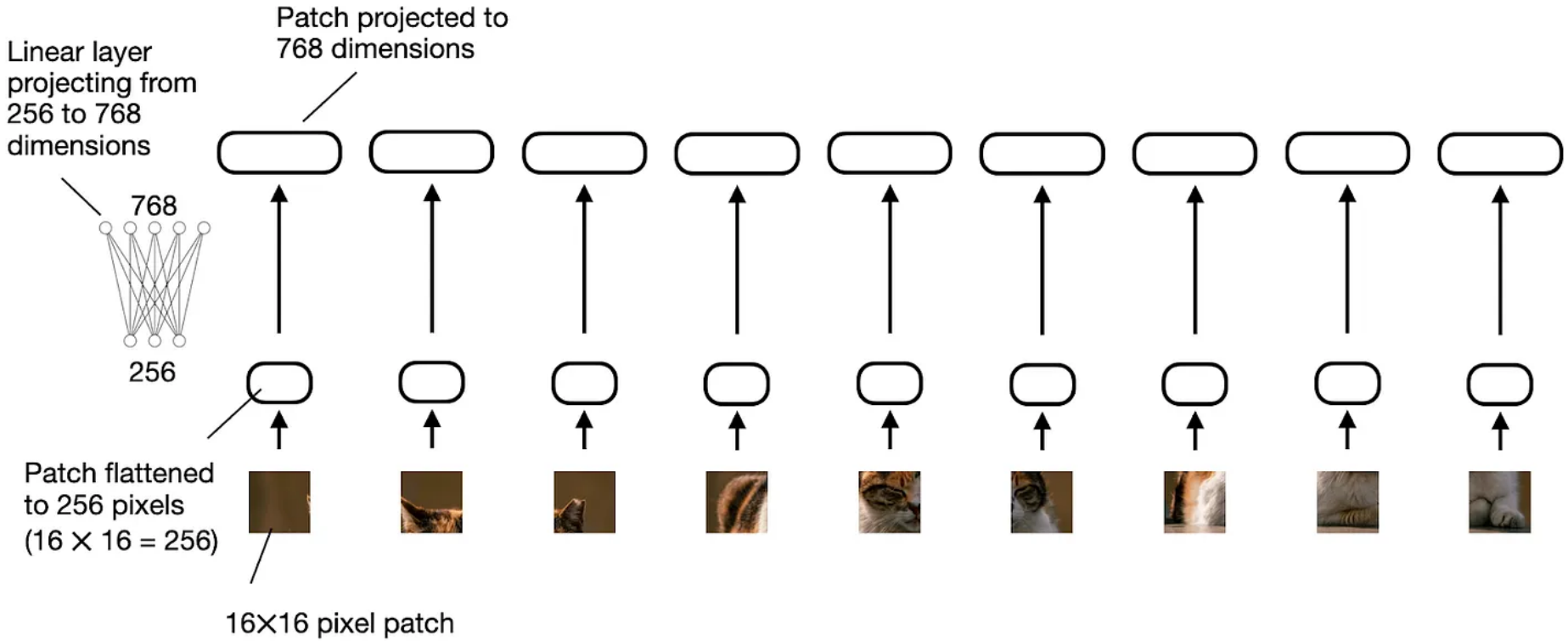
- What happens inside the image encoder shown above? To process an image, we first divide it into smaller patches, much like breaking words into subwords during tokenization. These patches are then encoded by a pretrained vision transformer (ViT), as shown in the figure
- Note that ViTs are often used for classification tasks, so I included the classification head in the figure above. However, in this case, we only need the image encoder part.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

The role of the linear projection module

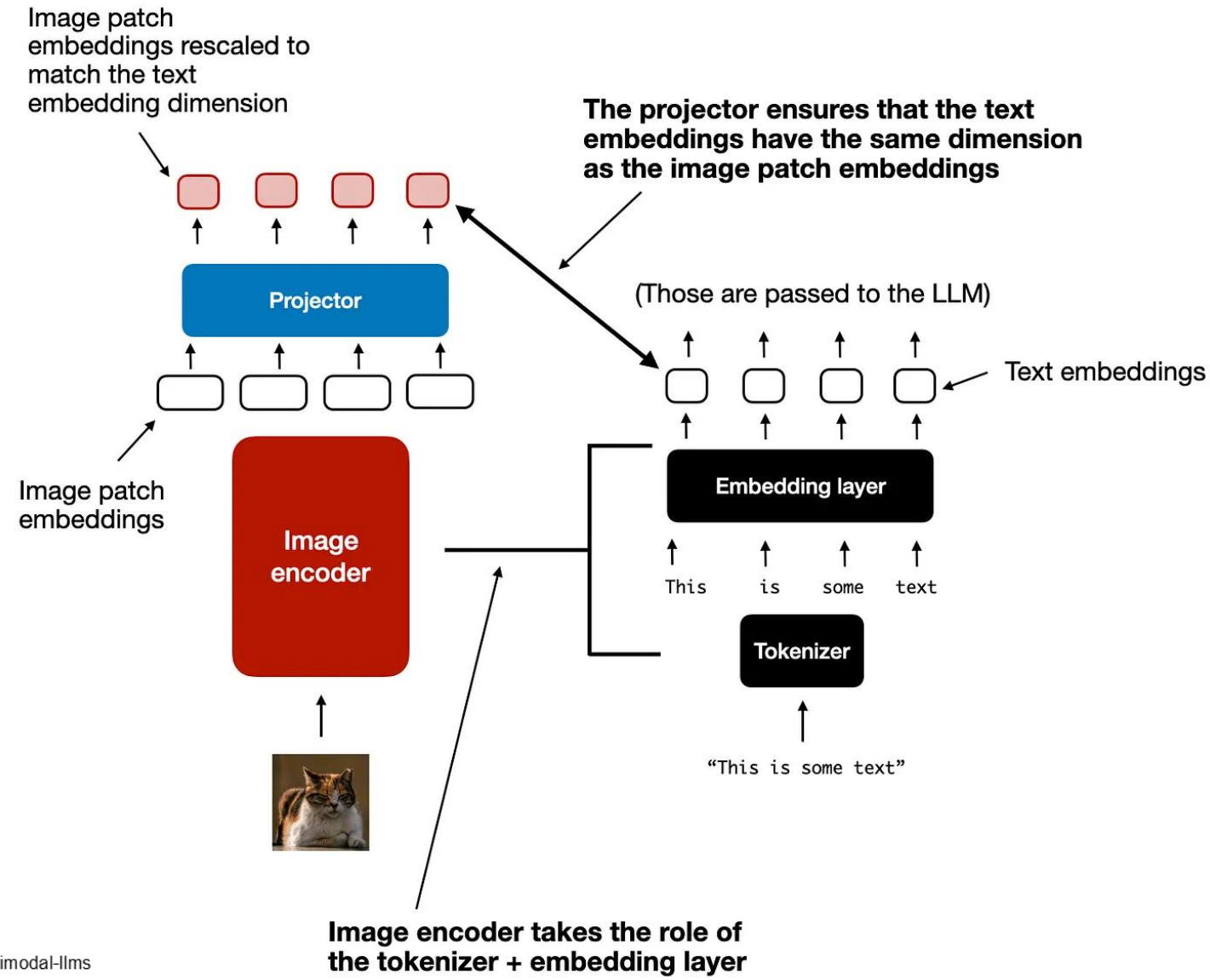
- The "linear projection" shown in the previous figure consists of a single linear layer (i.e., a fully connected layer). The purpose of this layer is to project the image patches, which are flattened into a vector, into an embedding size compatible with the transformer encoder. This linear projection is illustrated in the figure below. An image patch, flattened into a 256-dimensional vector, is up-projected to a 768-dimensional vector.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Image vs text tokenization

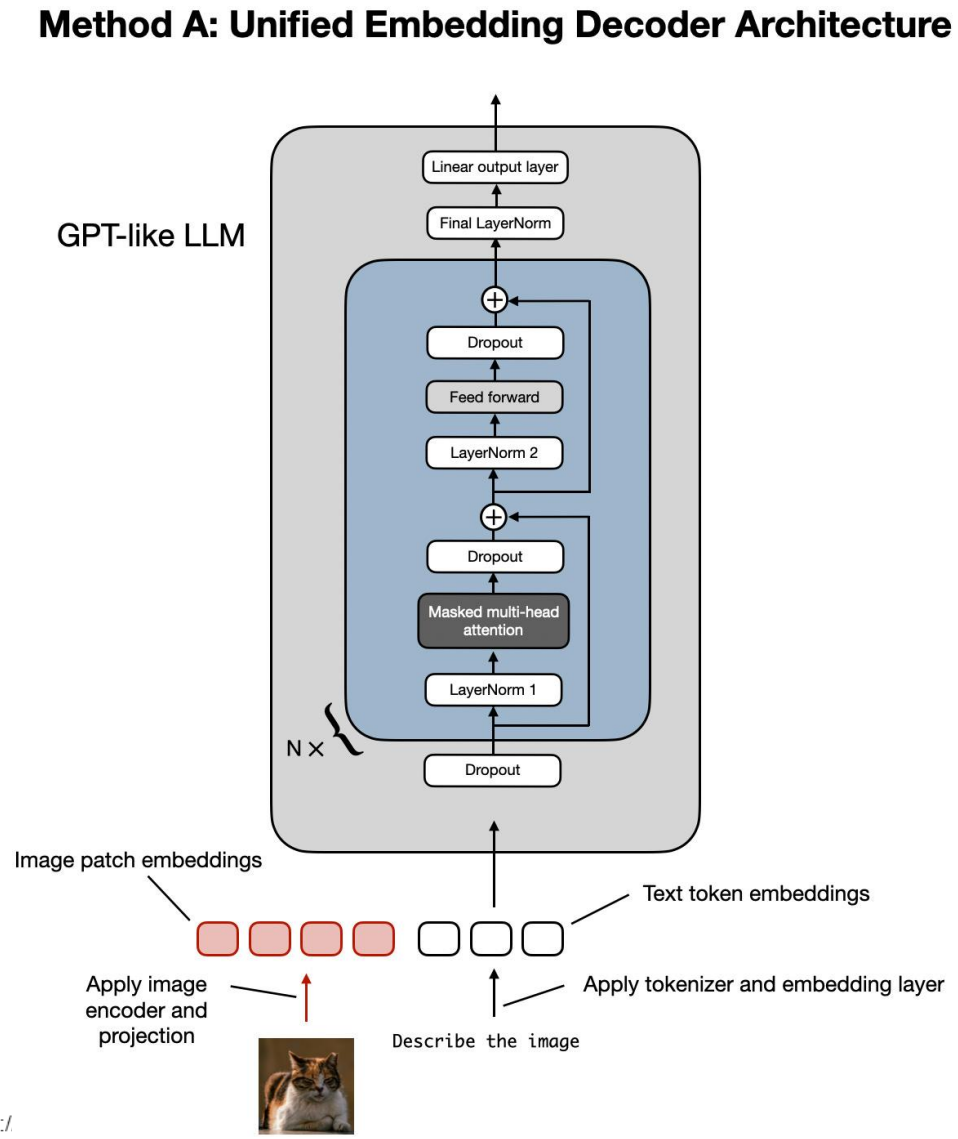
- An additional ***projector*** module that follows the image encoder. This *projector* is usually just another ***linear projection*** layer that is similar to the one explained earlier.
- The purpose is to project the image encoder outputs into a dimension that matches the dimensions of the embedded text.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Method A: Unified Embedding Decoder Architecture

- Now that the image patch embeddings have the same embedding dimension as the text token embeddings, we can simply concatenate them as input to the LLM.
- The image encoder we discussed in this section is usually a pretrained vision transformer. A popular choice is CLIP or OpenCLIP.

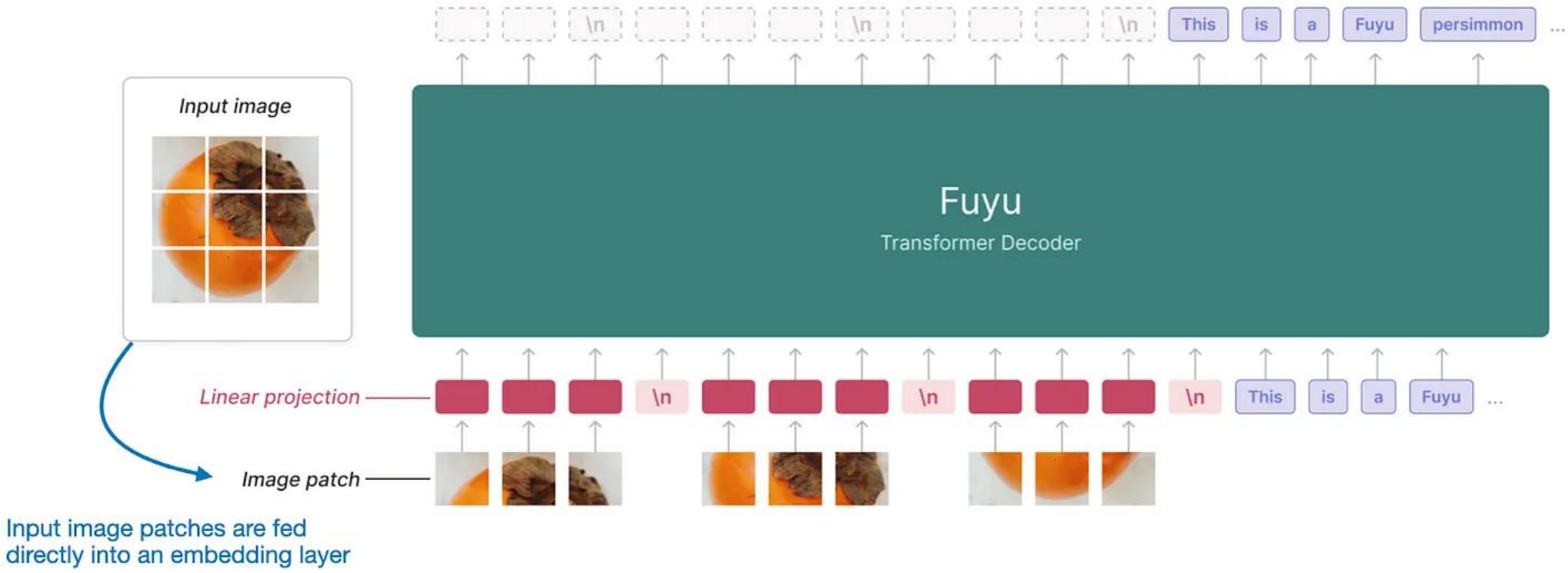


https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

https:/

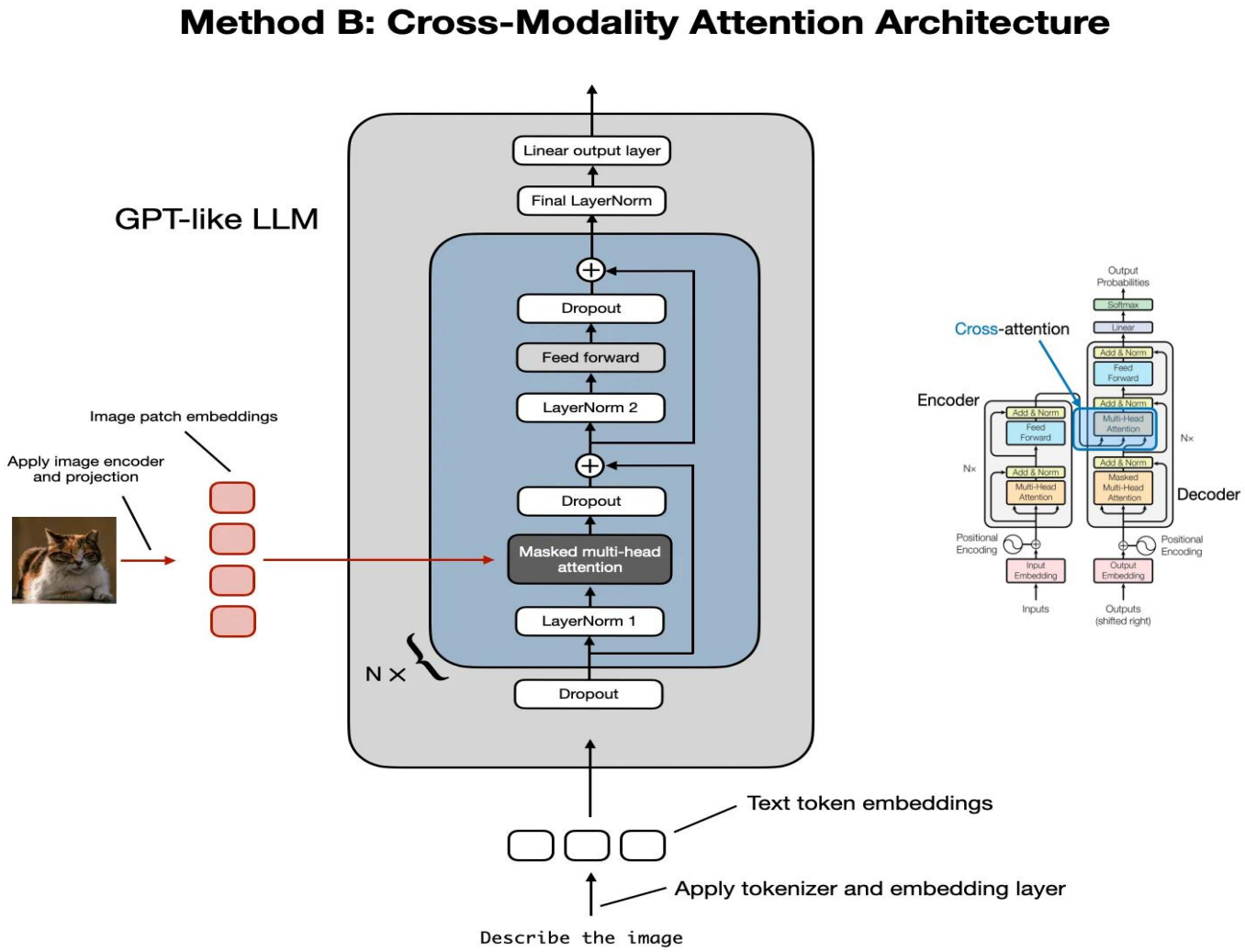
Fuyu multimodal LLM that operates directly on the image patches without image encoder

- there are also versions of Method A that operate directly on patches, such as Fuyu, which is shown in the figure below.



Method B: Cross-Modality Attention Architecture

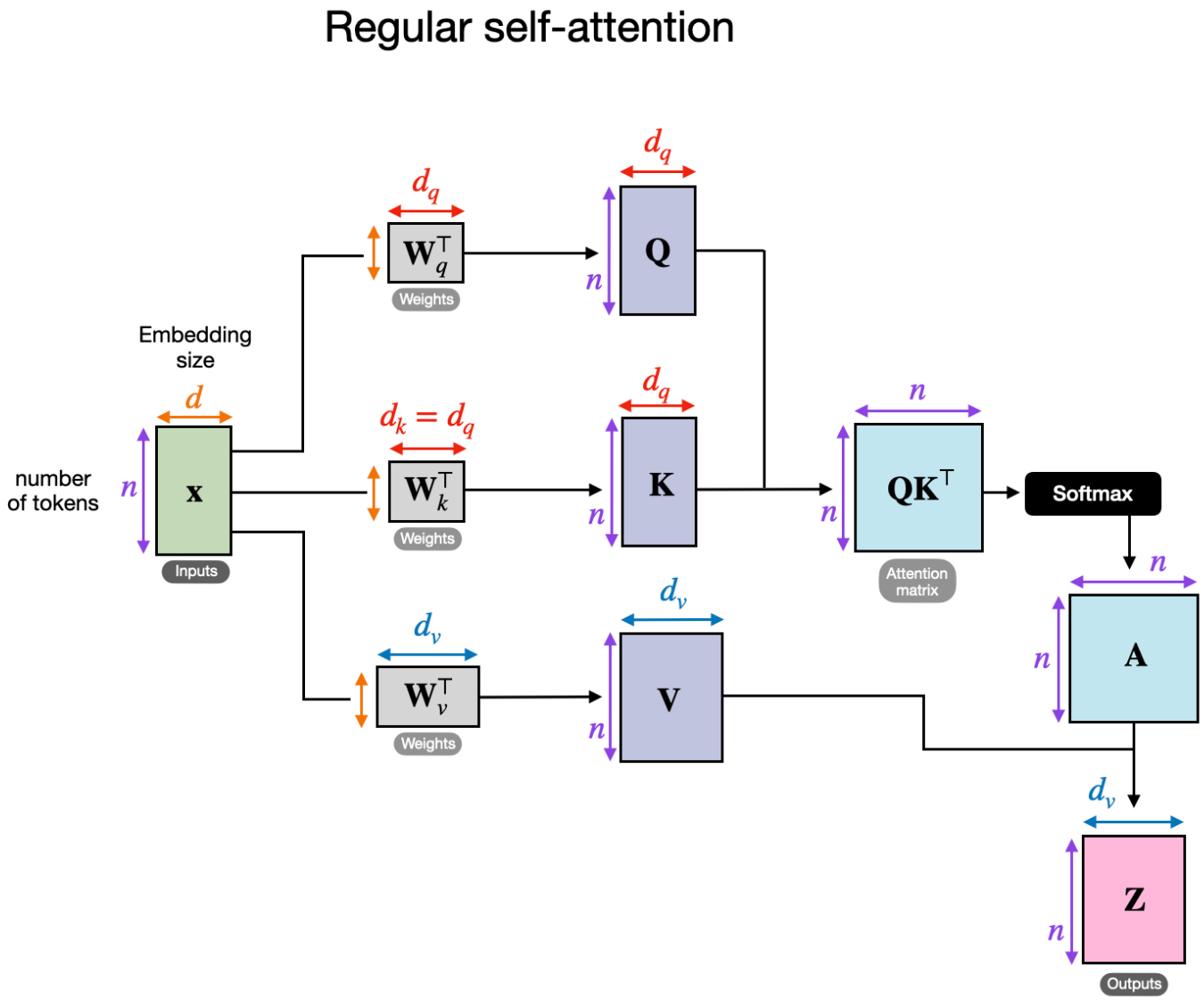
- In the Cross-Modality Attention Architecture method depicted in the figure above, we still use the same image encoder setup we discussed previously.
- However, instead of encoding the patches as input to the LLM, we connect the input patches in the multi-head attention layer via a cross-attention mechanism.
- In the context of multimodal LLM, the encoder is an image encoder instead of a text encoder, but the same idea applies.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Regular self-attention mechanism

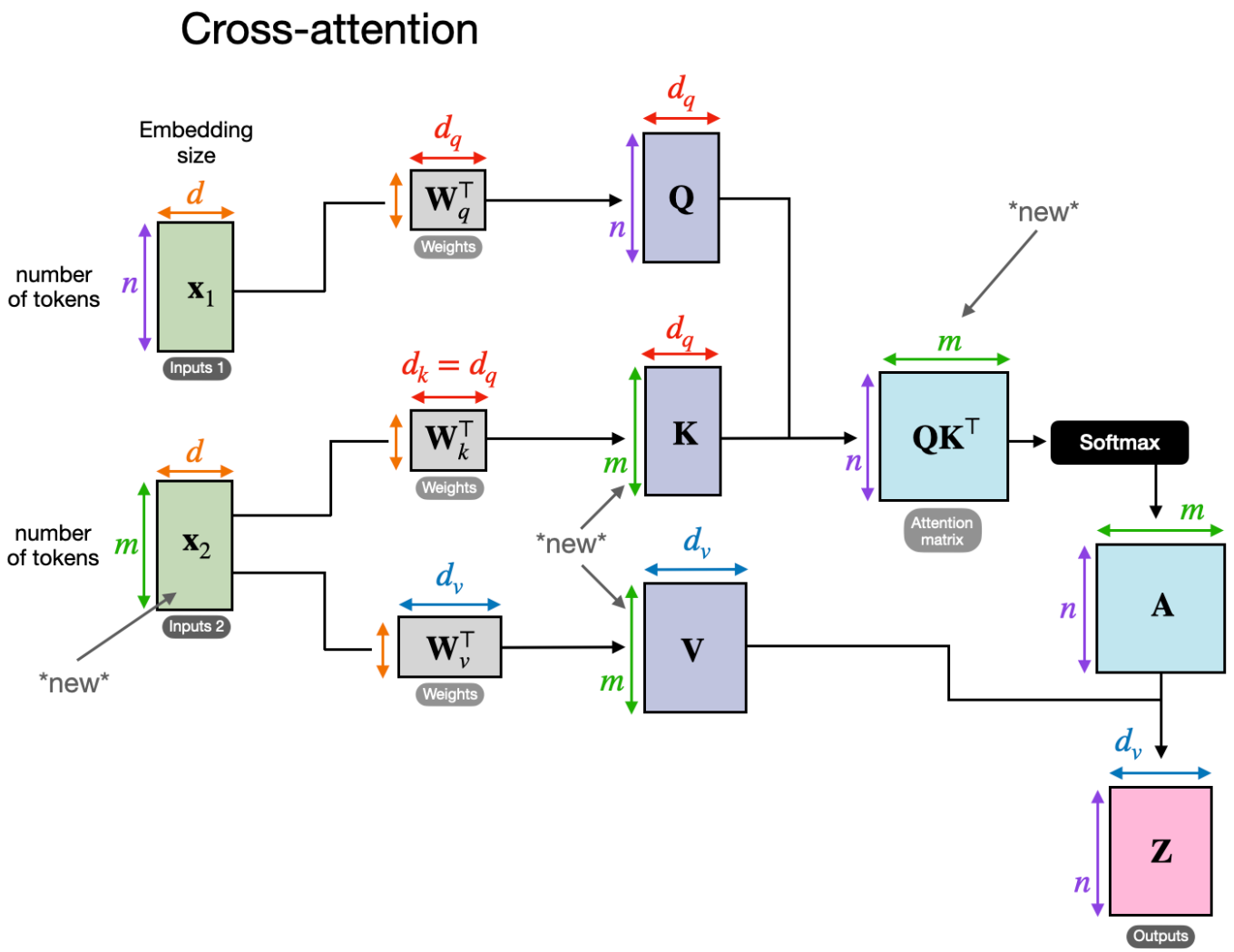
- x is the input, and W_q is a weight matrix used to generate the queries (Q). Similarly, K stands for keys, and V stands for values. A represents the attention scores matrix, and Z are the inputs (x) transformed into the output context vectors.



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Cross-attention

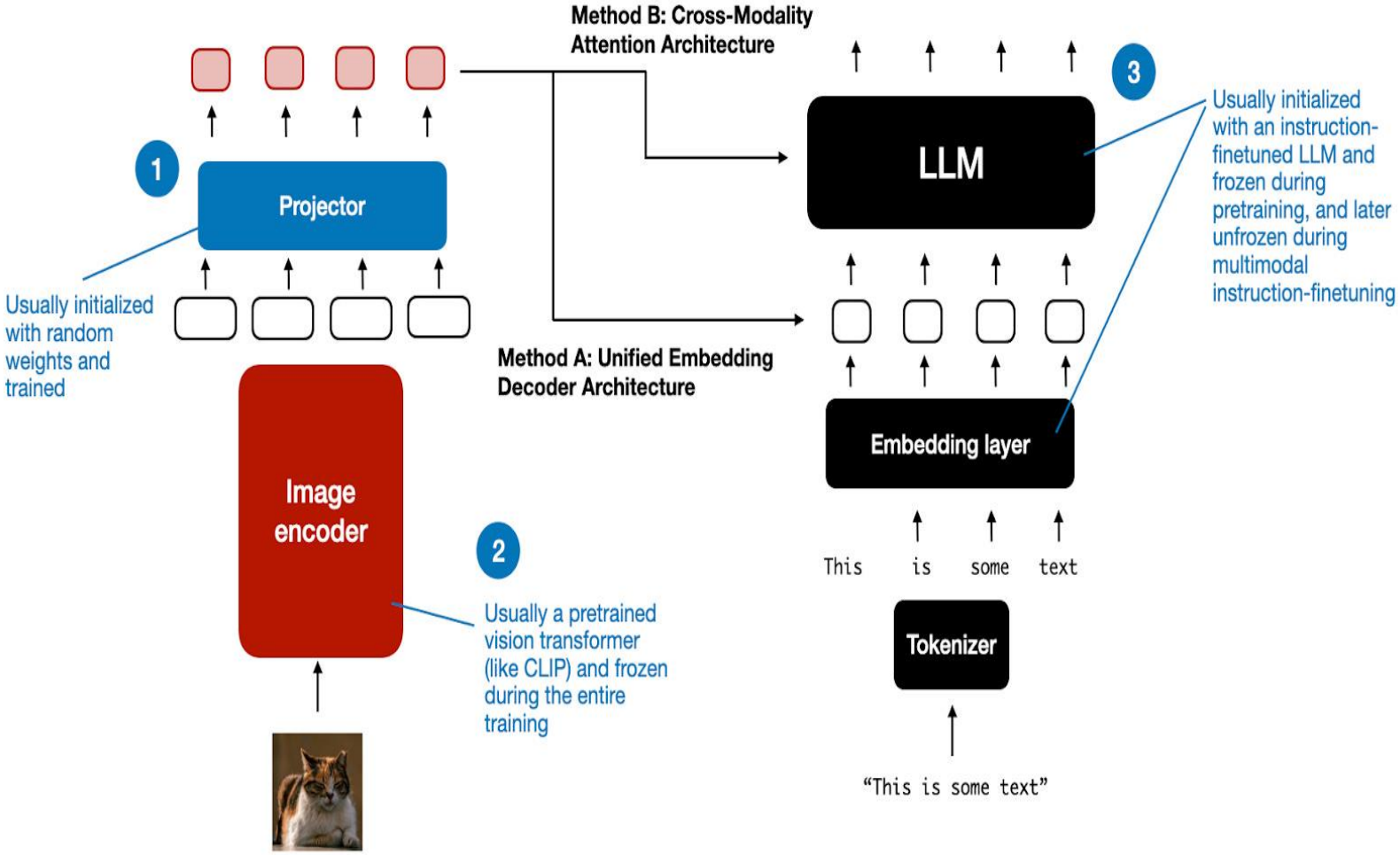
- In cross-attention, in contrast to self-attention, we have two different input sources, as illustrated in the following figure.
- As illustrated in the previous two figures, in self-attention, we work with the same input sequence. In cross-attention, we mix or combine two different input sequences.



https://magazine.sebastianraschka.com/p/understandi

Unified decoder and cross-attention model training

- Similar to the development of traditional text-only LLMs, the training of multimodal LLMs also involves two phases: pretraining and instruction finetuning.
- However, unlike starting from scratch, multimodal LLM training typically begins with a pretrained, instruction-finetuned text-only LLM as the base model.
- For the image encoder, CLIP is commonly used and often remains unchanged during the entire training process
- Keeping the LLM part frozen during the pretraining phase is also usual, focusing only on training the projector—a linear layer or a small multi-layer perceptron. Given the projector's limited learning capacity, usually comprising just one or two layers, the LLM is often unfrozen during multimodal instruction finetuning (stage 2) to allow for more comprehensive updates



https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

Method A: Unified Embedding Decoder Architecture vs Method B: Cross-modality Attention Architecture

Which method is more effective. The answer depends on specific trade-offs.

- The Unified Embedding Decoder Architecture (Method A) is typically easier to implement since it doesn't require any modifications to the LLM architecture itself.

- The Cross-modality Attention Architecture (Method B) is often considered more computationally efficient because it doesn't overload the input context with additional image tokens, introducing them later in the cross-attention layers instead. Additionally, this approach maintains the text-only performance of the original LLM if the LLM parameters are kept frozen during training.

Recent multimodal models and methods

- Recent multimodal models are converging toward a few core capabilities:
  - unified text-image-audio-video reasoning,
  - long-context memory,
  - agent/tool use,
  - real-time interaction,
  - and native multimodal generation

Frontier Multimodal Models

GPT-4.1 / GPT-5

Key characteristics:

- text + image + tool integration,
- extremely long context windows,
- strong coding and instruction following,
- increasingly agentic workflows.

GPT-4.1 introduced:

- 1M-token context,
- better multimodal reasoning,
- stronger code synthesis. (TechCrunch)

GPT-5 expands toward:

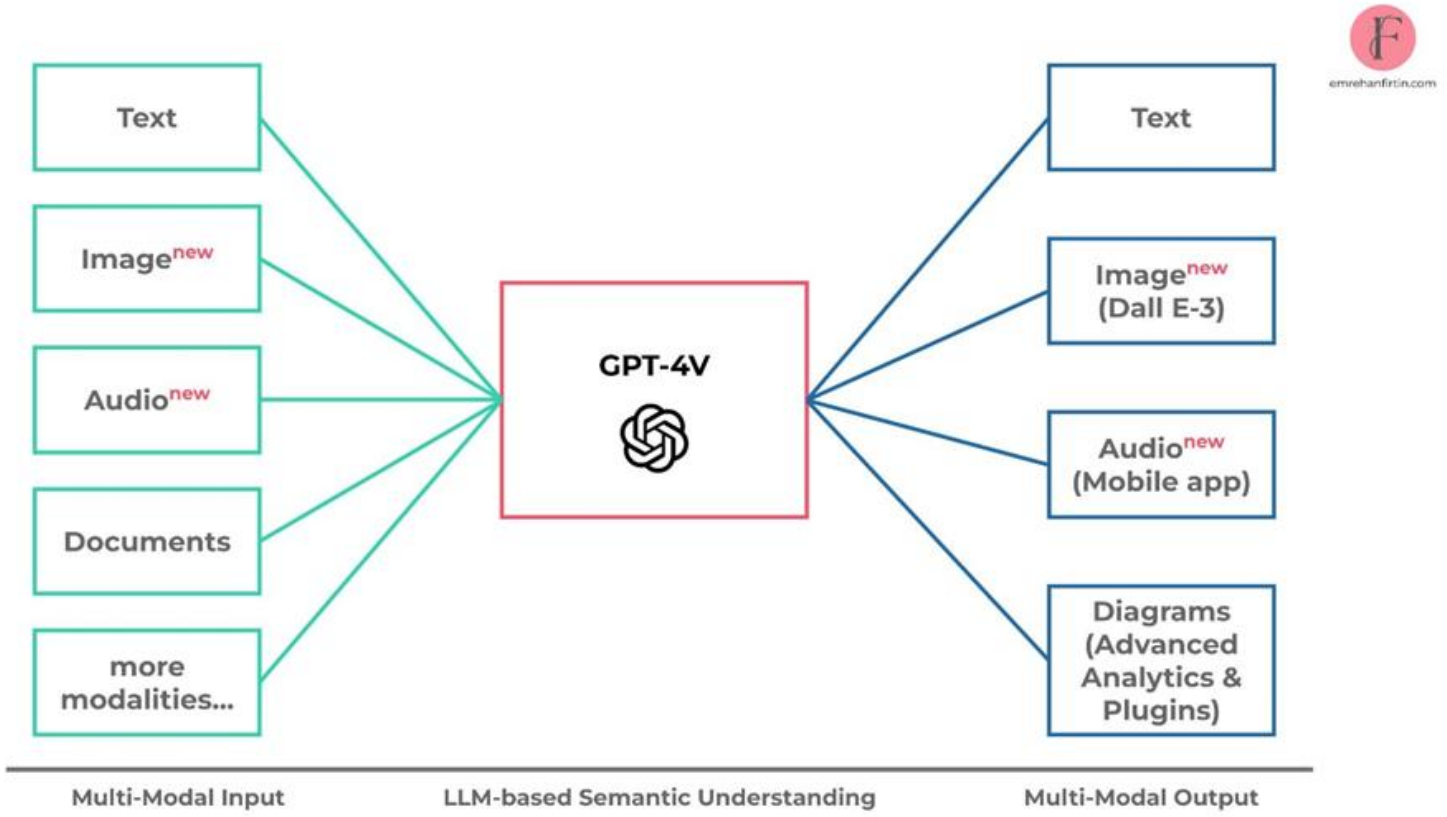
- unified reasoning systems,
- stronger multimodal planning,
- integrated tool execution,
- persistent agent behavior. (Wikipedia)

OpenAI trend:

moving from “chat model” → “general multimodal agent.”

Resources:

- OpenAI GPT Models



Frontier Multimodal Models

Gemini 2.5 Pro / Gemini 3 family

Google's Gemini line emphasizes:

- native multimodality,
- huge context windows,
- video understanding,
- spatial reasoning.

Notable directions:

- multimodal chain-of-thought,
- native video/audio processing,
- cross-modal retrieval,
- real-time interaction.

Gemini 2.5 Pro became notable for:

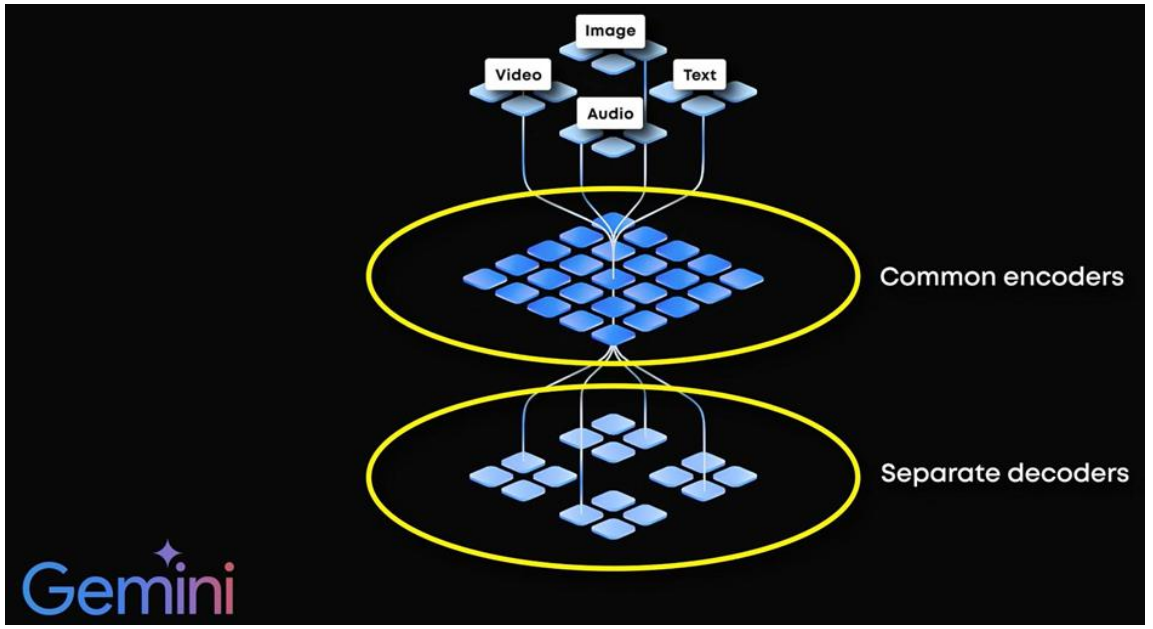
- strong reasoning,
- high-context processing,
- multimodal performance parity with top models. (TechCrunch)

Community benchmarks in late 2025 highlighted Gemini 3 Pro as especially strong in:

- video reasoning,
- spatial understanding,
- visual benchmarks. (Reddit)

Resources:

- Google DeepMind Gemini



Frontier Multimodal Models

Claude Opus 4 / Claude Sonnet 4.x

Anthropic's direction:

- long-horizon reasoning,
- safe agentic execution,
- coding,
- computer-use workflows.

Claude 4.x models are especially associated with:

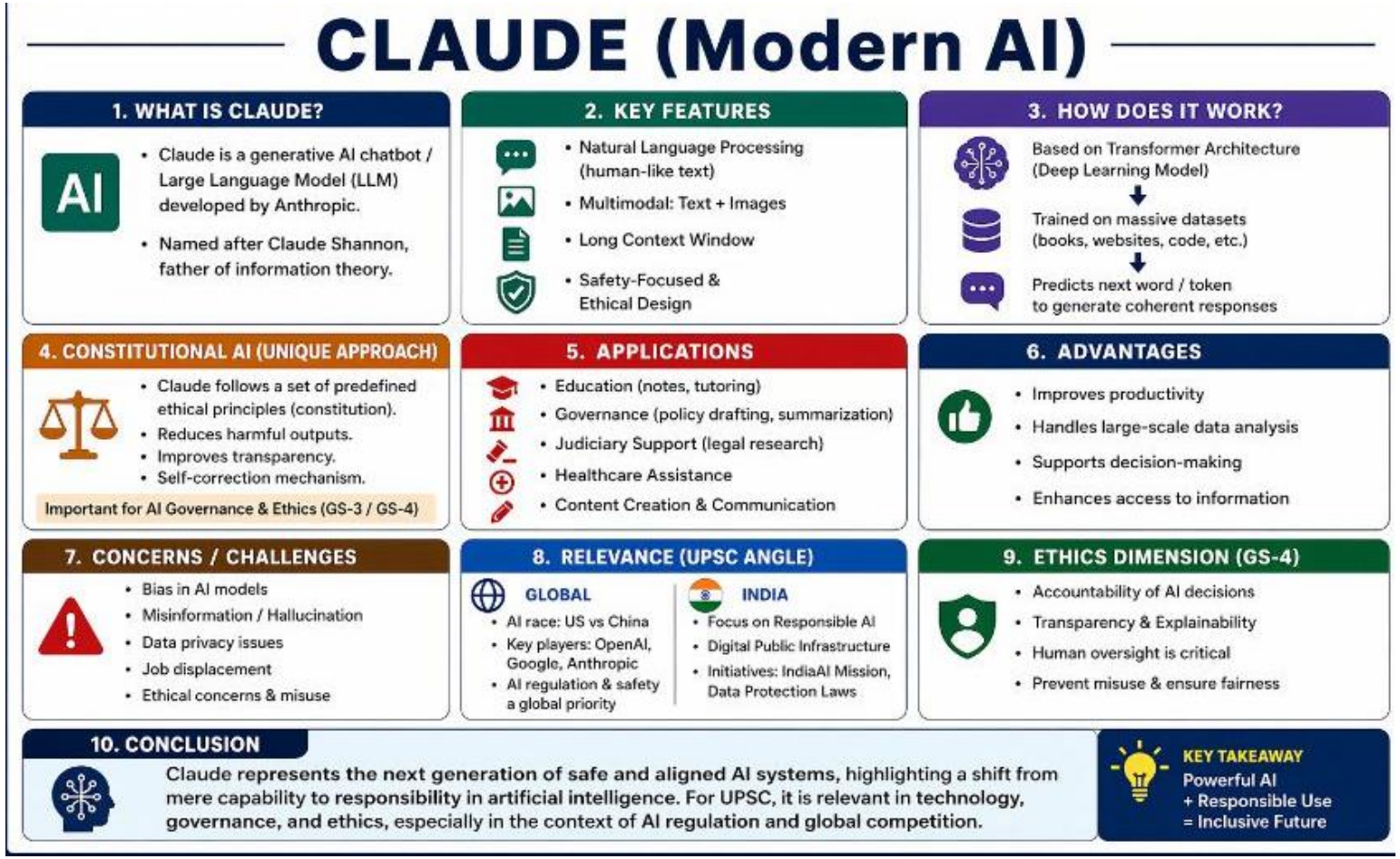
- tool-augmented agents,
- "computer use,"
- software engineering tasks,
- enterprise reliability. (TechRadar)

Anthropic strongly emphasizes:

- constitutional AI,
- alignment,
- controllable autonomy.

Resources:

- Anthropic Claude

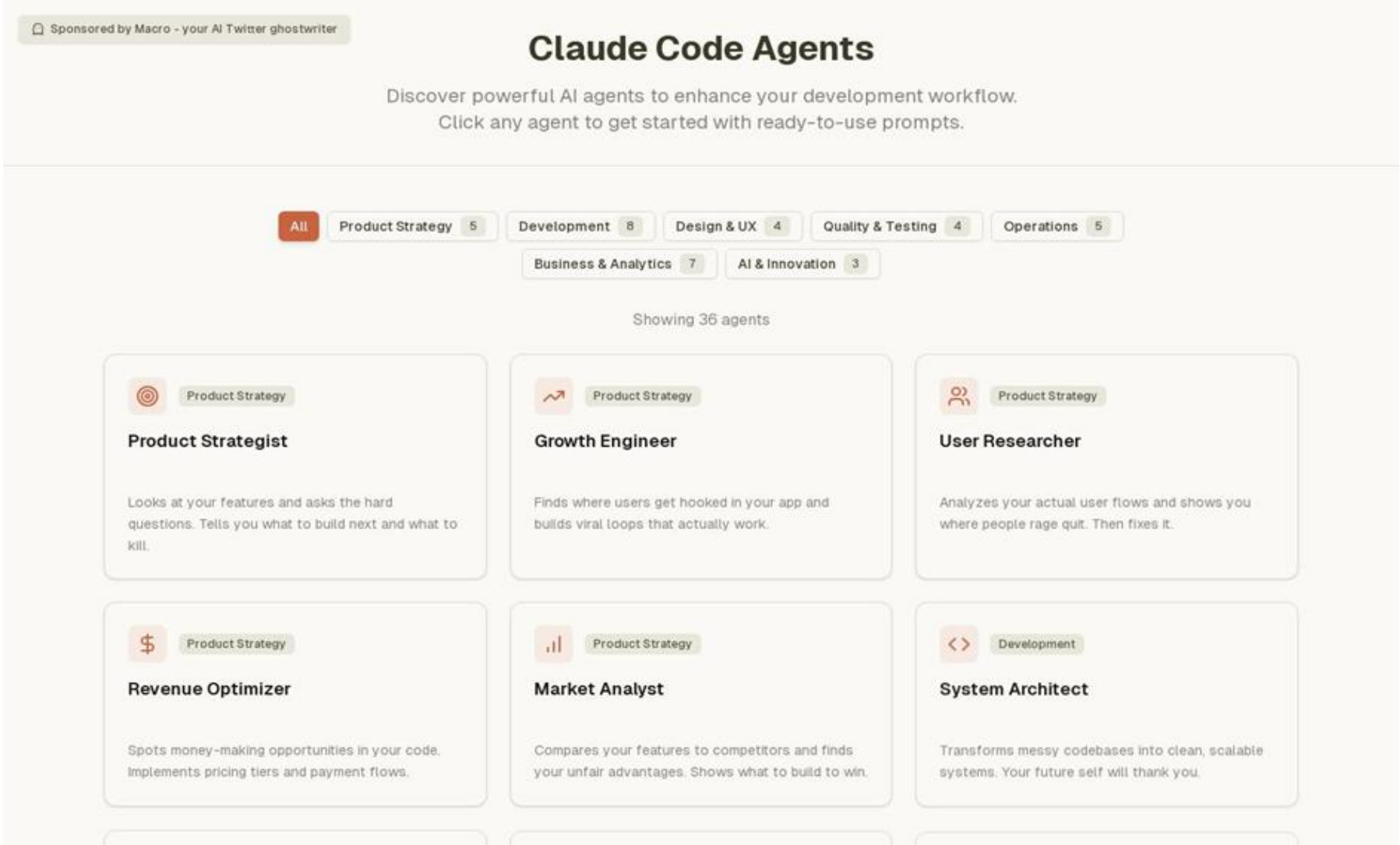


https://www.vaidicslucknow.com/current-affair/what-is-claude-what-is-large-language-model-llm/

Frontier Multimodal Models

A code agent is an AI system that can:

- Understand coding tasks from natural language
- Plan multi-step solutions
- Write and modify code
- Execute tasks (sometimes via tools or APIs)
- Iterate based on feedback or errors



Open-Source Multimodal Models

Qwen2-VL

Strong open multimodal family:

- OCR,
- visual grounding,
- multilingual understanding.

Llama 3.2 Vision

Meta's multimodal extension of Llama:

- image understanding,
- mobile deployment,
- open ecosystem focus.

Gemma 3

Google's lightweight multimodal open models:

- local inference,
- efficient deployment,
- image-text reasoning. (LumiChats)

Mistral Small 3.1

Focus:

- efficient multimodal inference,
- open deployment,
- compact architecture. (LumiChats)

Important Recent Methods

1. Vision-Language Models (VLMs)

Core paradigm:

- encode images/video,
- fuse with language representations,
- autoregressively reason/output text.

Examples:

- GPT-4o,
- Gemini,
- Qwen2-VL,
- Claude Vision.

Key advancement:

unified token spaces for text + vision.

2. Mixture-of-Experts (MoE)

Used heavily in frontier models.

Idea:

- activate only subsets of parameters,
- scale efficiently,
- improve specialization.

Benefits:

- lower inference cost,
- better scaling,
- modular specialization.

Important Recent Methods

3. Retrieval-Augmented Generation (RAG)

Still foundational.

Modern multimodal RAG includes:

- image retrieval,
- video retrieval,
- document grounding,
- multimodal embeddings.

Trend:

from “chatbot memory” → “multimodal knowledge systems.”

4. Agentic Tool Use

Very important recent direction.

Models increasingly:

- browse,
- write code,
- call APIs,
- manipulate GUIs,
- orchestrate workflows.

Connected technologies:

- Model Context Protocol,
- tool calling,
- computer-use agents,
- autonomous planning.

Important Recent Methods

5. Native Audio Models

Recent systems increasingly integrate:

- speech recognition,
- speech synthesis,
- emotional prosody,
- real-time dialogue.

Shift:

pipeline systems → end-to-end speech models.

6. Video Understanding & Generation

Major 2025 trend.

Capabilities:

- temporal reasoning,
- procedural understanding,
- long-video comprehension,
- video generation.

Strong activity from:

- Google,
- OpenAI,
- Runway,
- Pika,
- Kling.

Research Trends

Multimodal Chain-of-Thought

Models reason across modalities:

- inspect image,
- infer relationships,
- produce intermediate reasoning,
- solve tasks step-by-step.

Embodied AI

Integration with:

- robotics,
- simulators,
- desktop control,
- physical-world agents.

Important for:

- humanoid robotics,
- autonomous assistants.

World Models

Increasing focus on:

- temporal prediction,
- physical simulation,
- latent environment modeling.

Important labs:

- DeepMind,
- NVIDIA,
- Tesla,
- OpenAI research.

Benchmark Trends

Recent evaluations show:

- reasoning models outperform specialists on many generalized tasks,
- but still lag specialized CV systems in precise geometric perception. (arXiv)

Models now achieve:

- near-expert performance on professional exams,
- strong multimodal medical reasoning,
- advanced coding performance. (arXiv)

Current Architectural Direction

The field is converging toward:

Unified Multimodal Foundation Model

+

Tool Use / Agents

+

Long Context Memory

+

Retrieval Systems

+

Real-Time Interaction

The major competitive axes now are:

- reasoning quality,
- multimodal grounding,
- context length,
- inference speed,
- agent reliability,
- safety/alignment,
- deployment efficiency.

Current Architectural Direction

Multimodal capabilities have become standard across frontier models, and efficiency improvements are delivering GPT-4-level performance at dramatically lower costs. A few forces are shaping the space:

- **Reasoning + vision fusion**: Models now combine visual perception with chain-of-thought reasoning rather than treating them separately.
- **Agentic use**: Visual agents that can operate UIs, read screenshots, and take actions are becoming practical.
- **Open-source closing the gap**: Open-weight families like DeepSeek-V3.2 and Qwen3 are narrowing the gap with closed models at a fraction of the cost.
- **Specialization**: Domain-specific multimodal systems for medical imaging, scientific research, and document processing are maturing alongside general-purpose models.

No single model dominates every task — the best choice depends on whether you prioritize video understanding (Gemini), agentic coding (Claude), versatility (GPT-5.5), or cost efficiency (open-source options).

THANK YOU FOR YOUR ATTENTION