Multimodal Large Language Models

Images, Audio, Video & Vector Retrieval Across Modalities

Course: Building Generative AI Business Solutions — MS Analytics & Data Science

Lecture Notes

Spring 2026

1. Introduction: From Unimodal to Multimodal Intelligence

Text-only large language models, despite their remarkable capabilities in reasoning, generation, and instruction following, suffer from fundamental perceptual limitations. They cannot reason about visual content — a photograph, a satellite image, a medical scan, or a schematic diagram is entirely opaque to a model that processes only tokenized text. When documents containing tables, charts, and embedded figures are converted to text via extraction pipelines, the spatial layout that carries meaning — column alignment, row grouping, relative positioning of labels to data — is destroyed. Audio signals, including speech prosody, environmental sounds, and musical structure, are similarly inaccessible. Video, which combines temporal visual dynamics with synchronized audio, represents the richest and most complex information channel that text-only models cannot perceive at all.

The core motivation for multimodal large language models (MLLMs) is straightforward: the world is multimodal, and intelligent agents must perceive it as such. Human cognition integrates visual, auditory, linguistic, and tactile information continuously. An agent that can only read text is fundamentally limited in the tasks it can perform — it cannot interpret a dashboard screenshot, analyze a medical image, understand a video tutorial, or read a scanned document with complex formatting. Multimodal perception transforms the capability frontier of AI agents from text-processing systems into genuine perceptual reasoners.

In the context of agentic AI — the architectural paradigm this course explores — multimodal perception maps directly to the perception stage of the agent loop (perception → planning → memory → execution). An agent that can see, read, and hear is qualitatively more capable than one that only reads text. It can receive a screenshot of a failing dashboard and diagnose the issue. It can ingest a PDF financial filing as rendered pages, extracting tables and charts without lossy text conversion. It can watch a video demonstration and extract procedural steps. These capabilities are not incremental improvements — they unlock entirely new categories of autonomous agent behavior.

The historical trajectory is instructive. Early computer vision systems (2012–2017) developed powerful image recognition through convolutional neural networks but operated in isolation from language understanding. Separately, transformer-based language models (2018–2023) achieved remarkable text reasoning but were entirely blind to visual input. The convergence of these two streams — connecting pre-trained vision encoders to pre-trained language models through learned projection layers — represents one of the most significant architectural innovations of the 2023–2025 period. This convergence was not merely additive; the combination of visual perception with linguistic reasoning produces emergent capabilities (spatial reasoning, chart interpretation, document understanding) that neither system possesses alone.

2. Modality Taxonomy

A modality is formally defined as a distinct channel of information characterized by its own signal structure, encoding requirements, and semantic density. Each modality imposes different computational demands on models that process it, and each carries information that cannot be fully reduced to another modality without loss. The following taxonomy covers the modalities relevant to current multimodal LLM research and deployment.

Text

Text is a discrete, sequential, symbolic modality. It is represented as a sequence of tokens drawn from a finite vocabulary, typically 32K–128K tokens for modern LLMs. Each token is mapped to a dense embedding vector of dimensionality d (commonly 4096 for models like LLaMA[1]). Text has the highest information density per token of any modality — a single token can encode a precise concept, entity, or logical operator. The sequential structure of text is processed via causal self-attention, where each token attends to all preceding tokens in the context window. Tokenization itself introduces a lossy compression: subword tokenizers like BPE[2] split rare words into fragments, potentially disrupting morphological reasoning. For multimodal models, the text modality serves as the "backbone" representation — all other modalities are projected into the text embedding space for unified processing.

[1] Meta AI, *LLaMA model architecture*. In the official LLaMA implementation, the model configuration defines the hidden embedding dimension as dim = 4096, which is used for token embeddings and all transformer layers. See the ModelArgs definition in the Meta LLaMA GitHub repository: [github.com]

[2] Sennrich, R., Haddow, B., & Birch, A. (2016). *Neural Machine Translation of Rare Words with Subword Units*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1715–1725. https://aclanthology.org/P16-1162/

Images

Images are continuous, two-dimensional signals represented as pixel arrays — specifically, RGB tensors of shape H × W × C, where H and W are spatial dimensions and C = 3 for color channels. Unlike text, images encode spatial relationships: the relative positions of objects, their sizes, textures, and colors all carry semantic meaning. For processing by transformer architectures, images are typically decomposed into non-overlapping patches of size P × P pixels (e.g., 14 × 14 or 16 × 16). A 224 × 224 image with P = 14 yields $(224/14)^2$ = 256 patch tokens. Each patch is linearly projected to a d-dimensional embedding, creating a sequence that can be processed by standard transformer layers. This patch representation preserves local spatial structure within each patch while relying on self-attention to capture global spatial relationships. The information density per patch token is substantially lower than per text token — a single image may require 256–9,216 tokens[3] to represent, whereas the equivalent textual description might require only 50–100 tokens. This asymmetry in token efficiency is a central design challenge for multimodal architectures.

Audio

Audio is a continuous, one-dimensional time-domain signal — a waveform representing air pressure variations over time, typically sampled at 16 kHz (speech) to 44.1 kHz (music). For model consumption, raw waveforms are frequently transformed to frequency-domain representations via the Short-Time Fourier Transform (STFT)[4] or Mel spectrograms[5], which produce 2D time-frequency matrices. These spectrograms can then be treated as single-channel images and processed by vision encoders — a key architectural insight exploited by models like Whisper. Audio carries rich information beyond lexical content: prosody (pitch, rhythm, emphasis), speaker identity, environmental context, and emotional tone. Phoneme structure provides fine-grained temporal alignment between acoustic signals and linguistic units. The Mel spectrogram transform applies a bank of triangular filters spaced according to the Mel scale, which approximates human auditory perception by using closer spacing at lower

[3] Vision Transformers convert images into sequences of fixed-size non-overlapping patches, producing one token per patch. For an image of resolution 1344 × 1344 with patch size 14, the token count is $(1344/14)^2 = 9{,}216$. See Dosovitskiy et al., *An Image Is Worth 16×16 Words*, ICLR 2021, and the CLIP ViT-L/14 model documentation. https://arxiv.org/abs/2010.11929; https://huggingface.co/openai/clip-vit-large-patch14

[4] The Short-Time Fourier Transform (STFT) is a windowed Fourier analysis method that produces a time–frequency representation of a signal by computing the Fourier transform over short, overlapping segments. See D. Gabor, *Theory of Communication*, 1946; and A. V. Oppenheim & R. W. Schafer, *Discrete-Time Signal Processing*, Pearson. [d.umn.edu], [books.google.com]

[5] Mel spectrograms apply a Mel-spaced filter bank to the short-time Fourier transform, compressing frequency resolution according to human pitch perception. The Mel scale was introduced by Stevens, Volkmann, and Newman (1937), and Mel-frequency spectral representations are standard in speech and audio processing (O'Shaughnessy, *Speech Communication*). See also the modern definition in *librosa* documentation. doi/10.1121/1.1915893 http://librosa.org/ [jstor.org], [librosa.org]

frequencies. For a signal with sampling rate f_s, the spectrogram has frequency resolution up to $f_s/2$ (Nyquist limit) and temporal resolution determined by the STFT window size and hop length.

Video

Video is the most information-dense modality: a temporal sequence of image frames (typically 24–30 fps) synchronized with an audio track. Processing video requires both spatial attention (within each frame) and temporal attention (across frames), creating a computational challenge that scales quadratically with both spatial resolution and temporal duration. Practical approaches subsample frames — selecting 1–4 fps or key frames via scene detection — and process each frame through a vision encoder before aggregating temporal information through cross-frame attention or temporal pooling. The synchronized audio track provides complementary information: speech, sound effects, and music that contextualize the visual content. Models like Gemini 2.5 Pro natively process interleaved video frames and audio within a single context window of up to 1M tokens. A 1-hour video at 1 fps with 256 patches per frame generates 921,600 visual tokens alone — illustrating why extreme context lengths and aggressive token compression are essential for video understanding.

Documents

Documents represent a fused modality — they combine text, spatial layout, embedded images, tables, and typographic formatting into a single artifact. A PDF page is not merely text; the spatial arrangement of elements on the page carries semantic meaning. Column boundaries separate distinct content streams. Table cells are defined by their spatial coordinates, not by explicit delimiters. Headers are distinguished by font size and weight, not by markup tags. Layout-aware encoders like LayoutLM and DocFormer process documents by combining text token embeddings with 2D positional embeddings derived from bounding box coordinates, enabling the model to understand that two text spans appearing at the same y-coordinate but different x-coordinates belong to different table columns. An alternative approach, increasingly prevalent in 2025–2026, treats each document page as a rendered image and processes it directly through a VLM — this bypasses the need for explicit layout parsing and handles arbitrary formatting, including handwritten annotations, stamps, and non-standard layouts.

Sensor, Tabular, and Time-Series Data

Beyond the primary modalities of text, image, audio, and video, multimodal systems increasingly incorporate structured numerical data from IoT sensor streams, medical devices (ECG, EEG, pulse oximetry), financial time series, and tabular datasets. These modalities have regular temporal or structural patterns that differ fundamentally from natural language. Integrating them with LLMs typically involves either serializing the data as formatted text (with significant precision and context-length costs) or encoding them through specialized numerical encoders whose outputs are projected into the LLM's embedding space — analogous to vision encoders for images. Meta's ImageBind demonstrates that even inertial measurement unit (IMU) data can be embedded in a shared multimodal space.

3. Vision Language Model Architecture

3.1 The Three-Stage Pipeline

Modern vision language models (VLMs) share a common three-stage architecture: a vision encoder that converts images to token sequences, a projection or adapter layer that aligns the visual representation with the LLM's embedding space, and the LLM backbone that performs joint reasoning over visual and text tokens. This modular design enables leveraging pre-trained components for each stage and aligning them through relatively lightweight training.

Stage 1: Vision Encoder (ViT / CLIP)

The vision encoder transforms a raw image into a sequence of dense feature vectors. The dominant architecture is the Vision Transformer (ViT), which operates by splitting an input image of resolution H × W into a grid of non-overlapping patches, each of size P × P pixels. For a standard ViT-L/14 configuration with input resolution 224 × 224 and patch size P = 14, this produces N = (H/P) × (W/P) = 16 × 16 = 256 patches. Each patch is flattened into a vector of dimension $P^2 \times C = 14^2 \times 3 = 588$ and linearly projected to a d-dimensional embedding (1024 for ViT-L). Learnable position embeddings are added to each patch token to encode spatial location. A special [CLS] token is prepended to the sequence to aggregate global image semantics. The resulting sequence of N + 1 tokens is processed through L transformer encoder layers with multi-head self-attention, producing a final representation where each patch token encodes both local visual features and global context.

The patch count formula is central to understanding computational cost: N = (H / P) ×

(W / P)

For high-resolution images (e.g., 1344 × 1344 with P = 14), this yields N = 9,216 patch tokens — a substantial addition to the LLM's context window. Self-attention over these tokens has $O(N^2)$ complexity, making high-resolution processing computationally expensive. Several strategies address this cost: dynamic resolution (resize images to the minimum resolution needed for the task), patch merging (combine adjacent patches in later layers to reduce token count), and windowed attention (restrict attention to local windows with periodic global attention). Pre-training of the vision encoder is typically performed via contrastive learning on large-scale image-text pair datasets, as in CLIP.

Stage 2: Projection / Adapter Layer

The vision encoder outputs embeddings in its own dimensionality (e.g., 1024d for ViT-L/14, 1408d for EVA-CLIP), which must be projected to match the LLM backbone's token embedding space (e.g., 4096d for LLaMA-7B, 5120d for LLaMA-13B). Two dominant approaches exist for this alignment.

Linear MLP Projection. A simple two-layer MLP with a GELU[6] activation maps each visual token independently from the encoder's dimensionality to the LLM's dimensionality. This approach, used in LLaVA, is lightweight (~20M parameters for a 1024→4096 projection), fast to train, and preserves the one-to-one correspondence between patch tokens and projected tokens. The transformation can be expressed as:

$$h = W_2 \cdot \text{GELU}(W_1 \cdot v + b_1) + b_2$$

where v is the vision encoder output for a single patch, W_1 and W_2 are learned weight matrices, and h is the projected token in the LLM's embedding space. The trade-off is that it performs no cross-token compression — all 256+ patch tokens are passed to the LLM, consuming context window capacity.

Q-Former (Querying Transformer). Introduced in BLIP-2 and used in InstructBLIP, the Q-Former is a cross-attention module that uses a fixed set of N_q learnable query tokens (typically 32–64) to attend to the full set of visual feature tokens via cross-attention.

The queries extract the most salient information from the visual features and compress the representation from 256+ tokens down to N_q tokens. This dramatically reduces the number of tokens injected into the LLM context, at the cost of potential information loss for fine-grained spatial details. The Q-Former is pre-trained in two stages: vision-language representation learning (image-text matching, contrastive, and generative objectives) and vision-to-language generative learning. The cross-attention mechanism computes:

$$\text{Attention}(Q, K, V) = \text{softmax}(Q \cdot K^T / \text{sqrt}(d_k)) \cdot V$$ where Q comes from the learnable query tokens and K, V come from the vision encoder output. This allows the Q-Former to selectively attend to the most informative visual patches.

Stage 3: LLM Backbone and Token Fusion

The projected visual tokens are introduced into the LLM's input sequence alongside text tokens. In the simplest formulation, visual tokens are prepended to the text token sequence: the LLM receives [V_1, V_2, ..., V_N, T_1, T_2, ..., T_M] as its input, where V_i are visual tokens and T_j are text tokens. The LLM's causal self-attention mechanism then operates over this unified sequence — every text token can attend to all visual tokens and all preceding text tokens. Generation is autoregressive: the model predicts the next text token conditioned on all visual tokens and all previously generated text tokens. This fusion mechanism requires no architectural modification to the LLM itself — only the input representation changes. The training procedure typically freezes the vision encoder, trains the projection layer on

[6] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs).* arXiv:1606.08415. The GELU activation is defined as $\text{GELU}(x) = x\Phi(x)$, where Φ is the standard Gaussian cumulative distribution function, and is widely used in Transformer architectures such as BERT and GPT. https://arxiv.org/abs/1606.08415 [arxiv.org]

image-caption alignment data, and then fine-tunes the entire system (or the LLM with LoRA adapters) on visual instruction-following data.

Training Data and Curriculum

The training curriculum for VLMs typically follows a two-phase approach. Phase 1: Pre-training alignment. The vision encoder is frozen and only the projection layer is trained on large-scale image-caption pairs (e.g., 558K filtered pairs from CC3M in LLaVA). The objective is to align the visual feature space with the LLM's token space — teaching the projection layer to produce tokens that the LLM interprets as meaningful visual descriptions. Phase 2: Visual instruction tuning. The projection layer and LLM (or LoRA adapters on the LLM) are jointly fine-tuned on curated instruction-following datasets that pair images with multi-turn question-answer conversations. This phase teaches the model to follow complex visual instructions: "Describe the trend in this chart," "What is unusual about this X-ray?" or "Extract all entities from this document page." The quality and diversity of Phase 2 data is the primary differentiator between models — GPT-4o's advantage comes largely from the scale and curation of its instruction-tuning corpus rather than from architectural innovations.

3.2 CLIP: Contrastive Language–Image Pre-Training

CLIP, introduced by Radford et al. (2021), established the foundational approach to learning aligned visual and textual representations through contrastive pre-training. The architecture consists of two parallel encoders: a Vision Transformer (or ResNet variant) that maps images to a d-dimensional embedding, and a text transformer that maps captions to the same d-dimensional space. Both encoders are trained simultaneously on a dataset of 400 million (image, caption) pairs collected from the internet.

The training objective is the InfoNCE contrastive loss. Given a batch of B (image, caption) pairs, CLIP computes the cosine similarity between every image embedding and every caption embedding in the batch, producing a B × B similarity matrix. The loss encourages the diagonal entries (matching pairs) to have high similarity while off-diagonal entries (non-matching pairs) have low similarity. Formally, for image embedding z_i and text embedding z_t, the loss for the image-to-text direction is:

$L_{i2t} = -\log(\exp(\text{sim}(z_i, z_t) / \tau) / \Sigma_j \exp(\text{sim}(z_i, z_j) / \tau))$ where $\text{sim}(a, b) = (a \cdot b) / (||a|| \cdot ||b||)$ is the cosine similarity and τ is a learned temperature parameter that controls the sharpness of the distribution. The symmetric text-to-image loss is computed analogously, and the total loss is the average of both directions. This contrastive objective creates a shared embedding space where semantically related images and texts are nearby in cosine distance, enabling zero-shot classification (embed class names as text, find nearest image), cross-modal retrieval, and — critically for VLMs — high-quality visual features that are already linguistically aligned.

The temperature parameter τ plays a crucial role: too high a temperature produces a uniform distribution where all pairs appear equally similar; too low a temperature produces a distribution that is too peaked, making training unstable. CLIP learns τ as a log-parameterized scalar initialized to 1/0.07.

Large batch sizes (32,768 in the original CLIP) are essential because the number of negative pairs scales as B^2 - B, providing more discriminative signal per gradient step.

CLIP's text encoder has a notable limitation: a 77-token maximum context length, which constrains the complexity of text descriptions it can encode. LLM2CLIP (Microsoft Research, 2024) addresses this by replacing CLIP's text encoder with a full LLM fine-tuned via caption contrastive loss, extending the effective text context to thousands of tokens while maintaining compatibility with CLIP's visual encoder. This enables richer, more detailed image-text alignment for retrieval and generation tasks.

3.3 Token Fusion Strategies

The mechanism by which visual and textual token sequences are combined within the model has significant implications for computational cost, cross-modal reasoning depth, and scalability. Four primary fusion strategies are employed in current architectures.

Strategy	Mechanism	Pros	Cons	Used In
Early Fusion	Concatenate all visual + text tokens into single sequence before first layer	Richest cross-modal attention; strong spatial reasoning	$O((V+T)^2)$ cost; limits resolution and multi-image	LLaVA, Qwen-VL
Late Fusion	Separate processing through most layers; combine near output	Efficient; parallel processing of modalities	Weak cross-modal reasoning; coarse visual understanding only	Early CLIP-based systems
Cross-Attention (Flamingo)	Dedicated cross-attn heads at intervals attend to compressed visual representation	Efficient; scalable to video; fixed visual token count	Visual info only accessible at cross-attn layers	Flamingo, IDEFICS
Hybrid	Early fusion in lower layers with sparse cross-attention in higher layers	Balanced cost and reasoning depth	Complex implementation; harder to train	GPT-4o, Gemini (likely)

Early Fusion concatenates all patch tokens from the vision encoder with text tokens into a single sequence before the first LLM layer. Every attention head in every layer computes full cross-modal attention between all visual and all textual tokens. This provides the richest possible interaction between modalities but has a major cost: for a high-resolution image producing 4,096 patch tokens combined with a 4,096-token text prompt, the attention computation scales as $O((4096 + 4096)^2) = O(67M)$ per layer per head. Early fusion is used in LLaVA and yields strong spatial reasoning, but limits practical image resolution and multi-image inputs.

Late Fusion processes visual and textual tokens independently through most of the model's layers, with cross-modal interaction occurring only in the final few layers or at the output head. This is computationally efficient — the two modality streams can be processed in parallel — but produces weaker cross-modal reasoning because the model has limited capacity to ground text in visual details. Late fusion is suitable for tasks where coarse visual understanding suffices but inadequate for fine-grained visual question answering.

Cross-Attention Fusion (Flamingo-style) augments the LLM backbone's transformer layers with dedicated cross-attention heads that attend to a compressed visual representation (e.g., from a Perceiver Resampler that reduces hundreds of patch tokens to a fixed-size set of visual tokens). These cross-attention layers are inserted at regular intervals (e.g., every 4th layer) and are the only components that directly attend to visual features. Standard self-attention layers continue to operate over text tokens only. This design is efficient and scalable — particularly for video, where multiple frames must be processed — and the Perceiver Resampler ensures that visual token count remains fixed regardless of input resolution.

Hybrid Fusion is employed by modern production models, combining strategies. Lower layers may use early fusion for a small number of critical visual tokens (e.g., the [CLS] token and high-salience patch tokens), while higher layers use sparse cross-attention to the full visual feature set. This balances computational cost with cross-modal reasoning depth. Gemini and GPT-4o are believed to use hybrid fusion architectures, though their exact designs are not publicly documented.

4. Visual Grounding: How Images ProvideContext to LLMs

Visual grounding is the mechanism by which information from visual tokens influences the LLM's text generation. Because visual tokens participate in the self-attention computation across all layers (in early and hybrid fusion), every LLM layer "sees" the image. The visual tokens serve as persistent contextual anchors: they are not consumed or forgotten as generation proceeds but remain in the attention window throughout the entire generation process. Three distinct types of grounding operate in VLMs.

Spatial Grounding. The model attends to specific spatial regions of the image to answer location-dependent queries. This manifests as high attention weights on particular patch tokens corresponding to regions of interest. Spatial grounding enables bounding box reasoning ("Where is the defect in this X-ray?"), relative position understanding ("Is the signature above or below the date?"), and anomaly localization in medical imaging and satellite analysis. The fidelity of spatial grounding depends directly on the patch resolution — finer patches (smaller P) provide more precise localization at higher computational cost.

Semantic Grounding. At a higher level of abstraction, visual tokens bias the LLM's output distribution toward completions that are semantically consistent with the image content. When an image of a beach scene is present in context, the probability mass over tokens related to sand, waves, sun, and ocean

increases, while tokens related to snow, mountains, or urban environments are suppressed. This is not explicit reasoning but a distributional effect: the visual tokens shift the model's prior over the vocabulary toward visually-consistent language. Semantic grounding is essential for open-ended image description, where the model must generate coherent text that faithfully reflects the image's high-level content without being prompted about specific visual elements.

Temporal Grounding. In video processing, temporal grounding aligns text to specific temporal segments of the video via frame-level attention. The model can identify "the moment when the speaker raises their hand" by attending strongly to the frame tokens from the relevant time segment. Temporal grounding requires frame-level indexing in the token sequence and is most effective when frames are sampled at sufficient density to capture the events of interest. Models that process video as a sequence of frame embeddings can perform temporal reasoning by computing attention patterns that peak at specific frame positions — analogous to how text attention peaks at relevant words during question answering.

Prompting Strategies for Visual Reasoning

Empirical research has demonstrated that prompt ordering significantly affects VLM performance on analytic tasks. Specifically, placing the question before the image in the prompt sequence yields 5–10% higher accuracy compared to image-first ordering on tasks requiring detailed visual analysis. The mechanism is attentional priming: when the question appears first, the LLM processes the question tokens and establishes which concepts and relationships are relevant. When the visual tokens subsequently enter the attention window, the model's attention heads are already primed to attend to the relevant visual features, leading to more focused and accurate visual extraction.

Multi-turn conversations with VLMs face a practical limitation: high-resolution images consume substantial context window capacity. A single 1344 × 1344 image at P = 14 produces over 9,000 tokens. In multi-turn interactions where multiple images are referenced, the context fills rapidly, potentially pushing earlier text and visual tokens beyond the context window boundary. Strategies for managing this include dynamic resolution scaling (reducing image resolution for less critical images), visual token compression via Perceiver-style modules, and explicit context management that drops older images when the window is near capacity.

Context Injection Patterns

Four primary context injection patterns characterize how images interact with text in multimodal prompts. Image-as-evidence: the image provides supporting context for a text query ("Given this chart, what was Q3 revenue?"). The text query drives the reasoning; the image supplies the data. Image-as-query: the image is the primary query and text provides the task instruction ("Describe what you see" or "Extract all text from this document"). The image drives the output; text merely specifies the task format. Image-as-output: the model generates or references an image as part of its response, used in generation models like DALL-E and Stable Diffusion, or when an agent produces a chart or diagram. Interleaved modalities (Flamingo-style): images and text are interspersed throughout the prompt in

natural order, enabling few-shot learning with visual examples. This pattern is particularly powerful for teaching the model a new visual task by providing 2–4 worked examples before presenting the target image.

5. Vector-Based Retrieval Across Modalities(Multimodal RAG)

5.1 From Text RAG to Multimodal RAG

The standard Retrieval-Augmented Generation (RAG) pipeline operates as follows: a user query is embedded into a dense vector via a text embedding model, approximate nearest neighbor (ANN) search retrieves the top-k text chunks from a vector index, and these chunks are injected into the LLM's context window as grounding evidence for generation. This pipeline works well when the knowledge base is purely textual. However, it fails fundamentally when the corpus contains documents with embedded images, charts, diagrams, and tables — the visual content is either discarded entirely during text extraction or reduced to meaningless OCR artifacts that lose the spatial and visual semantics that make the content informative.

Consider a concrete failure case: an SEC 10-K filing contains a bar chart showing quarterly revenue trends. Standard text extraction produces nothing from this chart — it is a raster image embedded in the PDF. Even if OCR extracts axis labels and data values, the spatial relationships (which bar is taller, the trend direction, the relative magnitude of changes) are lost. A text RAG system queried about "revenue growth trajectory" would retrieve narrative text passages but miss the most informative evidence in the document. Multimodal RAG addresses this by either converting visual content to text descriptions or by natively embedding and retrieving visual content.

5.2 Two Technical Paths

Path A — Modality Conversion. The first approach converts all non-text content to text before indexing. OCR extracts text from images and scanned pages. VLM captioning generates textual descriptions of charts and diagrams. Layout parsers reconstruct table structure as markdown or HTML. The resulting text is then processed through the standard text RAG pipeline. This approach has the advantage of compatibility with existing text RAG infrastructure — no changes to the embedding model, vector index, or retrieval logic are required. However, information loss is significant: chart captions cannot fully capture the data relationships visible in the visualization; table OCR frequently misaligns columns; and dense visual content like heat maps, scatter plots, and technical diagrams resist accurate textual description.

Path B — Native Multimodal Embedding. The second approach embeds images directly as vectors in a shared embedding space alongside text. Models like CLIP, ColPali, and Cohere Embed v4 produce

embeddings where text and images coexist in the same vector space — a query text can retrieve relevant images, and an image query can retrieve relevant text. These embeddings are stored in a multimodal-capable vector database. Retrieval returns mixed text-and-image results, which are then passed to a VLM for grounded generation. This approach preserves full visual semantics — the VLM sees the original chart, diagram, or table page rather than a lossy textual approximation. The trade-off is that it requires a multimodal embedding model and a tensor-capable index (for late-interaction models like ColPali), adding infrastructure complexity.

Dimension	Path A (Modality Conversion)	Path B (Native Multimodal)
Infrastructure	Standard text RAG stack; no changes needed	Multimodal embedding model + tensor-capable index
Information fidelity	Lossy: OCR errors, layout loss, chart simplification	Lossless: original visual content preserved
Query types served	Keyword/semantic text queries	Text queries + visual similarity queries
Latency	Low (text embedding only)	Moderate (image encoding at ingest; fast at query)
Best for	Text-heavy documents with occasional images	Image-rich documents: charts, diagrams, scans
Example tools	Tesseract OCR, LLM captioning, Unstructured.io	ColPali, CLIP, Cohere Embed v4, Qdrant

5.3 Key Embedding Models

The following table summarizes the principal embedding models used for multimodal retrieval as of 2025–2026.

Model	Modalities	Dim.	Key Characteristics
CLIP (OpenAI)	Image, Text	512–768 d	Dual encoder, contrastive training on 400M pairs, 77-token text limit
OpenCLIP	Image, Text	512–1024d	Open-source CLIP variants, trained on LAION-2B, multiple ViT sizes
ColPali	Document pages, Text	128d/token	Late interaction (ColBERT-style), per-patch similarity, PDF-page-level retrieval

Cohere Embed v4	Image, Text, Mixed	1024d	Production API, natively handles interleaved text+image documents
ImageBind (Meta)	6 modalities	1024d	Unified space for text, image, audio, video, IMU, depth
Nomic Embed Vision	Image, Text	768d	On-prem via Ollama, Nomic Atlas visualization compatible

ColPali deserves particular attention for its novel retrieval mechanism. Unlike standard embedding models that produce a single vector per document or image, ColPali uses a late interaction architecture inspired by ColBERT. Each patch token in the vision encoder's output receives its own embedding vector of dimensionality 128. At query time, each query token computes its maximum similarity against all patch tokens in a candidate document, and these per-token maximum similarities are summed to produce the final relevance score:

$$\text{score}(q, d) = \Sigma_i \max_j \text{sim}(q_i, d_j)$$

This fine-grained, per-patch similarity computation enables retrieval based on specific visual regions — a query about "revenue table" can match strongly with the patch tokens corresponding to the table area of a PDF page, even if the rest of the page contains unrelated content. ColPali operates at the full PDF page level, eliminating the need for text extraction or chunking entirely. This represents a paradigm shift: instead of extracting text from documents and searching text, you render documents as images and search the visual representation directly.

5.4 Hybrid Indexing Architecture

Production-grade multimodal RAG systems employ a hybrid indexing architecture that combines multiple retrieval strategies to maximize recall across different content types and query patterns. The architecture consists of three parallel indices.

Index 1: BM25[7] / Full-Text Index. A traditional inverted index providing keyword-based retrieval with TF-IDF weighting. This captures exact term matches, acronyms, entity names, and numerical values that dense embeddings may miss. Particularly important for domain-specific terminology and rare tokens that fall outside the embedding model's training distribution.

Index 2: Dense Vector Index. A vector index (HNSW or IVF-PQ) storing dense embeddings from a text or multimodal embedding model. This captures semantic similarity — queries that are conceptually

[7] Okapi BM25 is a TF-IDF-style probabilistic retrieval function derived from the Probabilistic Relevance Framework, incorporating term-frequency saturation and document-length normalization. See S. Robertson and H. Zaragoza, *The Probabilistic Relevance Framework: BM25 and Beyond*, *Foundations and Trends in Information Retrieval*, 2009. https://doi.org/10.1561/1500000019

related to documents even when they share no exact terms. Handles paraphrase, synonym, and concept-level matching. HNSW (Hierarchical Navigable Small World) graphs provide O(log N) approximate nearest neighbor search with recall rates above 95% at practical scale.

Index 3: Tensor Index. For late-interaction models like ColPali, a tensor index stores per-token embeddings for each document. This enables the fine-grained patch-level matching described above. The tensor index is particularly valuable for visual document retrieval, where relevance depends on matching specific visual regions rather than holistic document similarity.

Retrieval results from all three indices are combined via Reciprocal Rank Fusion (RRF). Given ranked lists R_1, R_2, ..., R_k from k retrieval systems, the fused score for document d is:

$$RRF(d) = \Sigma_{i=1..k} \, 1 / (c + rank_i(d))$$

where c is a constant (typically 60) that dampens the influence of low-ranked documents. RRF is robust, parameter-free (beyond c), and does not require score normalization across different retrieval systems. Optionally, a VLM reranker can be applied over the top-k fused results to perform cross-modal reasoning before final selection — for example, asking the VLM "Does this page contain information relevant to the query?" and using the answer to reorder results.

Technology options for implementing this architecture include Qdrant (native vector and sparse vector support, Rust-based, high performance), pgvector (PostgreSQL extension for dense vectors, good for integrated stacks), and Vespa (hybrid search with native tensor support and BM25). For the course cluster, the recommended stack is Qdrant + OpenCLIP for retrieval, running on the existing GPU nodes alongside Ollama for VLM inference.

6. Multimodal Agentic AI Patterns

Multimodal perception integrates directly into the agentic AI architecture — the perception-planning-memory-execution cycle that forms the backbone of autonomous agent design. A multimodal agent does not merely process text instructions and generate text outputs; it perceives visual, auditory, and document-based inputs, reasons over them, and takes actions that may produce multimodal outputs. The following four patterns characterize the principal ways multimodal capabilities enhance agentic behavior.

Pattern 1: Visual Perception → Planning

The agent receives visual input — a screenshot, camera feed, scanned document, or satellite image — and the VLM component generates a structured description, extracts entities, or identifies anomalies. This structured output feeds into the planner, which selects the next action based on the perceived state. The visual perception step converts unstructured pixel data into the structured symbolic representations that planners require.

A concrete example: an SEC filing agent ingests each page of a 10-K annual report as a rendered image (rather than extracting text, which would lose table formatting). The VLM identifies and classifies regions of each page — narrative text, financial tables, risk factor lists, performance charts. Extracted entities and relationships are used to populate a knowledge graph in Neo4j, creating a structured, queryable representation of the filing's content. The planner then routes analyst queries to the appropriate graph traversal or table lookup, with the original page images available as visual evidence for verification. This pattern demonstrates a key principle: visual perception feeds the symbolic reasoning layer, which in turn drives action selection.

Pattern 2: Multimodal Tool Use

In this pattern, the agent calls an external tool whose output is visual rather than textual. The tool might generate an analytics chart, render a GIS satellite tile, produce a PDF page, or capture a screenshot of a web application. The VLM interprets the visual output, extracting data points, identifying trends, or detecting anomalies, and passes a structured summary to the next stage of the agent pipeline. Critically, both the tool input and output can be visual: the agent might submit an image as a query to a visual search API and receive matching images as results, with the VLM reasoning over the visual similarity.

This pattern is increasingly relevant as AI agents interact with graphical user interfaces. Browser agents (e.g., those built on Playwright) capture screenshots of web pages, pass them to a VLM for understanding, and generate click/type actions based on the visual interpretation. Similarly, desktop automation agents capture application screenshots, identify UI elements, and execute interactions. The visual tool use pattern transforms any GUI application into a tool that an agent can operate, without requiring an API — the visual interface itself becomes the API.

Pattern 3: Cross-Modal Memory and Retrieval

The agent maintains a multimodal vector memory store that contains text notes, associated images, figures, charts, and other visual artifacts. When a new query arrives, the retrieval system searches across both text and visual embeddings, returning heterogeneous evidence — a mix of text passages, chart images, and table screenshots. The VLM synthesizes a grounded, evidence-backed answer over this mixed evidence, citing specific visual and textual sources. This pattern is the multimodal extension of RAG and connects directly to the DAIS project infrastructure used in this course (Qdrant for vector storage, Ollama for local model inference, Qwen2.5-VL for visual understanding).

A practical implementation of this pattern might structure the memory store as follows: each document page is stored as both an image embedding (via OpenCLIP or ColPali) and a text embedding (from extracted text, if available). The agent's query hits both embedding spaces in parallel, and RRF fuses the results. The top-k results — which may include a mix of text chunks and page images — are assembled into a multimodal prompt that the VLM processes to generate a final answer. The key design decision is the ratio of text to visual evidence in the prompt: too many images exhaust the context window; too few images may miss critical visual evidence.

Pattern 4: Multimodal Evaluation and Verification

The agent generates a visual output — a chart, dashboard, presentation slide, or data visualization — and a VLM judge evaluates the output for visual quality, factual accuracy, and compliance with design requirements. This closes the generation-evaluation loop: the agent can iteratively refine its visual output based on the VLM judge's feedback. For example, an agent generating a quarterly earnings dashboard can submit each chart to a VLM evaluator that checks whether axis labels are correct, data points match the source data, and the color scheme meets accessibility standards. Discrepancies trigger re-generation with corrective instructions.

This pattern extends to code generation and testing: an agent writes code to produce a visualization, renders it, captures the output as an image, and asks a VLM judge to verify that the visualization matches the specification. If the judge identifies issues (missing legend, wrong axis scale, overlapping labels), the agent receives structured feedback and modifies the code. This generate-render-evaluate loop is a powerful self-correction mechanism that leverages multimodal understanding for quality assurance in automated pipelines.

7. Current Model Landscape (2025–2026)

The multimodal LLM ecosystem has matured rapidly, with both proprietary and open-weight models achieving strong cross-modal reasoning capabilities. The following profiles summarize the key models relevant to research and deployment as of early 2026.

Model	Type	Context	Key Strengths
GPT-4o (OpenAI)	Proprietary	128K	Native audio I/O, best cross-modal reasoning, unified image+text+audio processing
Gemini 2.5 Pro (Google)	Proprietary	1M	Largest context, strongest video understanding, natively interleaved modalities
Claude 3.5 Sonnet (Anthropic)	Proprietary	200K	Strong document/PDF analysis, precise text extraction, reliable instruction following
LLaVA-1.6	Open	Variable	LLaMA + CLIP ViT-L/14, strong visual instruction following, fully self-hostable
Qwen2.5-VL 72B (Alibaba)	Open	128K	Matches GPT-4o on benchmarks, LoRA fine-tunable, Ollama — course cluster recommended

InternVL3 38B (Shanghai AI Lab)	Open	64K	Strong OCR + chart understanding, layout-aware, top document benchmarks
GLM-4.6V (Zhipu AI)	Open	128K	Native multimodal tool calling — images as tool parameters, agentic workflows
ImageBind (Meta)	Open	N/A	6-modality unified embedding: text, image, audio, video, IMU, depth

On-premises deployment recommendation. For the course cluster, two configurations are recommended based on task requirements. For fast inference in interactive applications (e.g., agentic loops with real-time visual perception), Qwen2.5-VL 7B provides strong performance at manageable GPU memory requirements (~16 GB VRAM). For maximum quality on complex visual reasoning tasks (e.g., detailed chart analysis, multi-page document understanding), Qwen2.5-VL 72B delivers near-GPT-4o performance but requires multi-GPU deployment (~140 GB VRAM across nodes). For tasks where document OCR and chart understanding are the primary requirements, InternVL3 38B offers state-of-the-art layout-aware extraction. For multimodal retrieval, OpenCLIP combined with Qdrant provides a production-ready vector search pipeline that runs entirely on the existing cluster GPU nodes.

Benchmarks and Evaluation

The principal benchmarks for evaluating multimodal LLMs span several capability axes.
MMBench and MMMU test general multimodal reasoning across diverse visual domains (charts, natural images, diagrams, UI screenshots). DocVQA and ChartQA evaluate document and chart understanding specifically — the ability to extract precise data values from complex visual layouts. MathVista tests mathematical reasoning over visual inputs (geometric proofs, function plots). RealWorldQA evaluates practical visual understanding in real-world contexts (street scenes, product images, satellite imagery). Performance on these benchmarks varies significantly across models: Qwen2.5-VL 72B achieves 83.0% on MMMU (vs. GPT-4o's 69.1%), while Gemini 2.5 Pro leads on video understanding tasks due to its 1M-token context enabling full-video ingestion.

8. Recommended Reading

The following references provide foundational and advanced coverage of the topics discussed in these lecture notes.

1. Radford, A., Kim, J. W., Hallacy, C., et al. (2021). "Learning Transferable Visual ModelsFrom Natural Language Supervision." OpenAI. https://arxiv.org/abs/2103.00020
2. Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). "Visual Instruction Tuning" (LLaVA). NeurIPS2023. https://arxiv.org/abs/2304.08485
3. Faysse, M., Music, H., Hudelot, C., Clinchant, S., & Piwowarski, B. (2024). "ColPali: Efficient Document Retrieval with Vision Language Models." https://arxiv.org/abs/2407.01449

4. Girdhar, R., El-Nouby, A., Liu, Z., et al. (2023). "ImageBind: One Embedding Space ToBind Them All." CVPR 2023. https://arxiv.org/abs/2305.05665
5. RAGFlow Team (2025). "From RAG to Context: 2025 Year-End Review of RAG."https://ragflow.io/blog/rag-review-2025-from-rag-to-context
6. Beyer, L., Steiner, A., Pinto, A. S., et al. (2024). "PaliGemma: A Versatile 3B VLM forTransfer." Google DeepMind. https://arxiv.org/abs/2407.07726
7. BentoML (2026). "Multimodal AI: Best Open-Source Vision Language Models." https://www.bentoml.com/blog/multimodal-ai-a-guide-to-open-source-vision-language-models