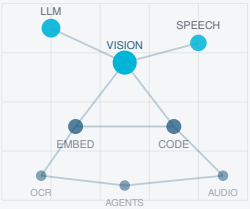
THE COMPLETE GUIDE

THE AI ATLAS

A Practitioner's Map of Every Model That Matters



NABIL EL MAHYAOUI

The AI Atlas

The AI Atlas

A Practitioner's Map of Every Model That Matters

Nabil El Mahyaoui

A must-read ebook for engineers, builders, and decision-makers

Copyright © 2026 Nabil El Mahyaoui.
All rights reserved.

Contents

1 How to Use This Atlas **1**

1.1 What This Book Is. 1

1.2 How to Navigate 2

1.3 The Taxonomy 3

1.4 A Note on Shelf Life 4

2 Large Language Models **5**

2.1 Frontier Proprietary Models 6

2.1.1 GPT-4o — OpenAI 6

2.1.2 GPT-4.5 — OpenAI 7

2.1.3 Claude Opus 4.6 — Anthropic 7

2.1.4 Claude Sonnet 4.5 — Anthropic 8

2.1.5 Claude Haiku 4.5 — Anthropic 9

2.1.6 Gemini 2.5 Pro — Google DeepMind 9

2.1.7 Gemini 2.0 Flash — Google DeepMind 10

2.1.8 Grok 3 — xAI 10

2.2 Open-Weight Models 11

2.2.1 Llama 4 — Meta 11

2.2.2 Llama 3.3 70B — Meta . 12

2.2.3 DeepSeek V3 — DeepSeek 12

2.2.4 Qwen 2.5 — Alibaba . . . 13

2.2.5 Mistral Large — Mistral AI 13

2.2.6 Command R+ — Cohere 14

2.2.7 DBRX — Databricks . . . 15

2.3 How to Choose 15

3 Reasoning Models 17

3.1 Why Reasoning Models Exist 18

3.2 The Landscape 18

3.2.1 01 — OpenAI 18

3.2.2 03 & 03-mini — OpenAI . 19

3.2.3 Claude with Extended Thinking — Anthropic . . 20

3.2.4 Gemini 2.5 Pro with Thinking — Google DeepMind 21

3.2.5 DeepSeek R1 — DeepSeek 21

3.2.6 QwQ-32B — Alibaba . . 22

3.3 When to Use Reasoning Models. 23

4 Small & Edge Models 25

4.1 The Leading Small Models 26

4.1.1 Phi-4 — Microsoft 26

4.1.2 Phi-4-mini — Microsoft . 26

4.1.3 Gemma 3 — Google DeepMind 27

4.1.4 Llama 3.2 1B & 3B — Meta 28

4.1.5 Qwen 2.5 0.5B–7B — Alibaba 28

4.1.6 SmolLM2 — Hugging Face 29

4.1.7 DeepSeek R1 Distilled — DeepSeek 29

4.2 Deployment Options. 30

4.3 Quantisation Quick Guide. 31

5 Embedding & Retrieval 33

5.1 Text Embedding Models 34

5.1.1 text-embedding-3-large — OpenAI 34

5.1.2 text-embedding-3-small — OpenAI 34

5.1.3 Cohere Embed v3 — Cohere 35

5.1.4 Voyage AI 3 / 3-lite — Voyage AI 35

5.1.5 Jina Embeddings v3 — Jina AI 36
5.1.6 BGE-M3 — BAAI 37
5.1.7 GTE-Qwen2 — Alibaba . 37
5.1.8 Nomic Embed Text v1.5 — Nomic AI 38
5.2 Multimodal Embedding 38
5.2.1 SigLIP / SigLIP 2 — Google 38
5.2.2 Jina CLIP v2 — Jina AI . 39
5.3 Reranking Models 39
5.4 RAG Architecture Patterns 40

6 Vision & Multimodal Understanding 43

6.1 Frontier Multimodal Models 44
6.1.1 GPT-4o Vision — OpenAI 44

6.1.2 Claude Vision — Anthropic 44

6.1.3 Gemini 2.5 Pro Vision — Google DeepMind 45

6.1.4 Grok Vision — xAI . . . 46

6.2 Open-Weight VLMs 46

6.2.1 Qwen2.5-VL — Alibaba . 46

6.2.2 InternVL 2.5 — Shanghai AI Lab 47

6.2.3 LLaVA-OneVision — UW / Microsoft 47

6.2.4 Gemma 3 with Vision — Google 48

6.2.5 Florence-2 — Microsoft . 48

6.3 Object Detection & Segmentation. 49

6.3.1 SAM 2 — Meta 49

6.3.2 YOLO v11 — Ultralytics . 49

6.3.3 Grounding DINO 1.5 — IDEA Research 50

7 Image Generation 51

7.1 Proprietary / Cloud 51

7.1.1 DALL-E 3 — OpenAI . . 52

7.1.2 Midjourney v6.1 — Midjourney 52

7.1.3 Google Imagen 3 — Google DeepMind 53

7.1.4 Adobe Firefly 3 — Adobe 53

7.1.5 Ideogram 2.0 — Ideogram 54

7.2 Open-Weight / Self-Hosted 54

7.2.1 Flux.1 — Black Forest Labs 55

7.2.2 Stable Diffusion 3.5 — Stability AI 55

7.2.3 SDXL — Stability AI . . . 56

7.3 Control & Editing Techniques 57

8 Video Generation 59

8.1 The Landscape 59

8.1.1 Sora — OpenAI 60

8.1.2 Veo 2 — Google DeepMind 60

8.1.3 Kling 1.6 — Kuaishou . . 61

8.1.4 Runway Gen-3 Alpha — Runway 61

8.1.5 Pika 2.0 — Pika Labs . . . 62

8.1.6 Wan2.1 — Alibaba 63

8.2 Current Limitations 63

8.3 Practical Workflow 64

9 Speech & Audio 66

9.1 Speech-to-Text (ASR) 66

9.1.1 Whisper large-v3 — OpenAI 67

9.1.2 Deepgram Nova-2 / Nova-3 — Deepgram . . . 67

9.1.3 AssemblyAI Universal-2 — AssemblyAI 68

9.1.4 NVIDIA Parakeet / Canary — NVIDIA 68

9.2 Text-to-Speech (TTS) 69

9.2.1 OpenAI TTS — OpenAI . 69

9.2.2 ElevenLabs — ElevenLabs 70

9.2.3 Cartesia Sonic — Cartesia 70

9.2.4 Fish Speech 1.5 — Fish Audio 71

9.2.5 Parler-TTS — Hugging Face 71

9.3 Music & Audio Generation 72

9.3.1 Suno v4 — Suno 72

9.3.2 Udio — Udio 72

9.3.3 MusicGen — Meta 73

9.3.4 Stable Audio 2.0 — Stability AI 73

10 Code Models **75**

10.1 Frontier Code Capabilities. 76

10.2 Open-Weight Code Models 77

10.2.1 DeepSeek Coder V2 / V3 — DeepSeek 77

10.2.2 Qwen2.5-Coder — Alibaba 78

10.2.3 Codestral — Mistral AI . 78

10.2.4 StarCoder 2 — BigCode / Hugging Face 79

10.2.5 CodeLlama — Meta . . . 79

10.3 Code Assistants & IDEs 80

10.3.1 GitHub Copilot — GitHub / Microsoft 80

10.3.2 Cursor — Cursor Inc. . . . 80

10.3.3 Claude Code — Anthropic 81

10.3.4 Windsurf — Codeium . . 81

10.4 Structured Output & Function Calling 82

11 OCR & Document AI **84**

11.1 Cloud Document AI Services. 84

11.1.1 Azure AI Document Intelligence — Microsoft . . 85

11.1.2 Google Document AI — Google Cloud 85

11.1.3 Amazon Textract — AWS 86

11.2 Open-Source OCR 86

11.2.1 Tesseract 5 — Google / Community 86

11.2.2 PaddleOCR — Baidu . . 87

11.2.3 EasyOCR — Jaided AI . . 87

11.2.4 Surya — VikParuchuri . . 88

11.2.5 DocTR — Mindee 88

11.2.6 GOT-OCR 2.0 — StepFun 89

11.3 PDF Processing 90

11.4 The VLM Approach 91

12 Agents & Frameworks **93**

12.1 What Makes an Agent 94

12.2 Agent SDKs 94

12.2.1 Claude Agent SDK — Anthropic 95

12.2.2 OpenAI Agents SDK — OpenAI 95

12.3 Orchestration Frameworks 96

12.3.1 LangChain / LangGraph — LangChain Inc. . 96

12.3.2 CrewAI — CrewAI 97

12.3.3 AutoGen — Microsoft . . 97

12.3.4 Semantic Kernel — Microsoft 98

12.3.5 Haystack — deepset . . . 98

12.3.6 Dify — Dify.AI 99

12.4 MCP: Model Context Protocol 100

12.5 How to Choose 101

References **103**

1

How to Use This Atlas

This book is a navigational tool, not a narrative. It maps the AI model landscape as it stands in early 2026—what exists, what works, and what to choose for which job.

1.1 What This Book Is

A concise, opinionated reference. Each chapter covers one category of AI model: language,

reasoning, vision, audio, code, embeddings, and more. For each model, you get the facts that matter: capabilities, parameters, context windows, licensing, and—crucially—what it is actually good at.

1.2 How to Navigate

The table of contents is your index. Each chapter is self-contained. Cross-references like (see p. 17) link to related chapters. Teal-coloured text is clickable in PDF readers.

If you are building a product, start with the chapter matching your modality. If you are evaluating the landscape, read front to back.

1.3 The Taxonomy

Chapter 2: Large Language Models — Frontier and open-weight text models (see p. 5)

Chapter 3: Reasoning Models — Chain-of-thought and deliberative systems (see p. 17)

Chapter 4: Small & Edge Models — Sub-7B models for on-device deployment (see p. 25)

Chapter 5: Embedding & Retrieval — Vector search, reranking, RAG (see p. 33)

Chapter 6: Vision & Multimodal — Image understanding and VLMs (see p. 43)

Chapter 7: Image Generation — Text-to-image systems (see p. 51)

Chapter 8: Video Generation — Text-to-video and image-to-video (see p. 59)

Chapter 9: Speech & Audio — ASR, TTS, and music generation (see p. 66)

Chapter 10: Code Models — Coding assistants and specialised code LLMs (see p. 75)

Chapter 11: OCR & Document AI — Text extraction and document parsing (see p. 84)

Chapter 12: Agents & Frameworks — Orchestration, tool use, and agentic systems (see p. 93)

1.4 A Note on Shelf Life

AI moves fast. Some models listed here will be superseded within months. The *categories*, however, are structural. The taxonomy will outlast the specifics. Use the models as snapshots; use the framework as a map.

2

Large Language Models

The foundation layer of modern AI. Large language models (LLMs) take text in and produce text out. Everything else—agents, code assistants, chatbots, summarisers—is downstream of this capability.

Two tiers define the market: **frontier proprietary** models (API access, closed weights) and **open-weight** models (downloadable, self-

hostable). The gap between them is narrowing.

2.1 Frontier Proprietary Models

2.1.1 GPT-4o — OpenAI

The workhorse of the OpenAI ecosystem. Multimodal (text, image, audio in/out). 128K context window. Strong across general reasoning, coding, and creative writing. Available via API and ChatGPT. Fast inference. The default choice for most general-purpose tasks.

Best for: General-purpose assistant, multimodal workflows, rapid prototyping.

2.1.2 **GPT-4.5** — OpenAI

Released February 2025. OpenAI's largest dense model. Improved "emotional intelligence," reduced hallucination rates, better at nuanced creative and analytical tasks. 128K context. Substantially more expensive than GPT-4o.

Best for: High-stakes creative writing, nuanced analysis, tasks requiring reduced hallucination.

2.1.3 **Claude Opus 4.6** — Anthropic

Anthropic's most capable model. 200K context window. Exceptional at long-document analysis, complex multi-step reasoning, code generation, and agentic workflows. Extended thinking mode for deliberative reasoning (see

p. 17). Strong instruction-following and safety alignment.

Best for: Complex analysis, agentic coding, long-context tasks, research.

2.1.4 **Claude Sonnet 4.5** — Anthropic

The balanced tier: near-Opus quality at significantly lower cost and faster latency. 200K context. Excellent coding, strong reasoning. The best price-performance ratio among frontier models for most production workloads.

Best for: Production APIs, coding assistants, cost-sensitive deployments needing frontier quality.

2.1.5 **Claude Haiku 4.5** — Anthropic

Ultra-fast, ultra-cheap. 200K context. Suitable for classification, extraction, routing, and high-volume tasks where latency and cost matter more than peak reasoning ability.

Best for: High-throughput pipelines, real-time classification, cost-optimised inference.

2.1.6 **Gemini 2.5 Pro** — Google DeepMind

Google's flagship. Natively multimodal (text, image, audio, video). Massive context window up to 1M tokens—the largest in production. Built-in "thinking" mode for complex reasoning. Strong at code, math, and multi-turn conversations.

Best for: Extremely long-context tasks, multimodal analysis, video understanding.

2.1.7 **Gemini 2.0 Flash** — Google DeepMind

High-speed tier with 1M context. Optimised for low-latency, high-throughput workloads. Multimodal input and output including native image generation and text-to-speech. Strong tool-use capabilities.

Best for: Agentic applications, real-time multimodal, high-volume API usage.

2.1.8 **Grok 3** — xAI

Released February 2025. Trained on one of the largest GPU clusters ever assembled (Colossus, 200K H100s). Competitive with GPT-4o and Claude on benchmarks. 128K context. Includes "Big Brain" mode for extended reasoning. Accessible through the X platform and API.

Best for: General-purpose tasks, users in the X/xAI ecosystem.

2.2 Open-Weight Models

2.2.1 **Llama 4** — Meta

Meta's latest generation. Mixture-of-Experts architecture. Maverick variant: 400B total parameters, 17B active per token—excellent efficiency. Scout variant: 109B active with a 10M token context window. Multilingual (12 languages). Licensed under Llama Community License.

Best for: Self-hosted production, multilingual tasks, long-context with Scout.

2.2.2 **Llama 3.3 70B** — Meta

The refined 70B dense model from December 2024. Matches or exceeds the older Llama 3.1 405B on many benchmarks at a fraction of the compute. 128K context. The standard choice for self-hosted deployments needing strong general capabilities.

Best for: Self-hosted general-purpose, fine-tuning base, on-premise deployment.

2.2.3 **DeepSeek V3** — DeepSeek

671B MoE model with 37B active parameters. 128K context. Trained for under $6M—a fraction of comparable Western models. Competitive with GPT-4o on code and math. Multi-head latent attention and FP8 mixed-precision training. MIT licence.

Best for: Cost-efficient self-hosting, code generation, math-heavy tasks.

2.2.4 **Qwen 2.5** — Alibaba

Available from 0.5B to 72B parameters. 128K context (32K for smaller sizes). Apache 2.0 licence. Particularly strong in Chinese and multilingual tasks. Specialised variants for code (see p. 78) and math. Fine-tuning ecosystem via Qwen-Agent.

Best for: Multilingual (especially CJK), fine-tuning, wide range of deployment sizes.

2.2.5 **Mistral Large** — Mistral AI

123B parameters. 128K context. Strong reasoning and multilingual capabilities (natively fluent in English, French, German, Spanish, Ital-

ian). Function calling support. Available via API (Le Platforme) and downloadable.

Best for: European multilingual deployments, function calling, enterprise use.

2.2.6 **Command R+** — Cohere

Optimised specifically for RAG (retrieval-augmented generation) and enterprise search. 128K context. Built-in citation generation—outputs include source attributions. Strong tool use. Available via Cohere API.

Best for: Enterprise RAG pipelines, search-grounded generation, citation-critical workflows.

2.2.7 **DBRX** — Databricks

132B MoE (36B active). 32K context. Fine-grained MoE with 16 experts, 4 active. Strong at code and general language tasks. Open under Databricks Open Model Licence.

Best for: Databricks ecosystem users, MoE experimentation.

2.3 How to Choose

- **Maximum quality, no budget constraint:** Claude Opus 4.6 or Gemini 2.5 Pro
- **Best general-purpose API:** GPT-4o or Claude Sonnet 4.5
- **Longest context:** Gemini 2.5 Pro (1M tokens) or Llama 4 Scout (10M)
- **Self-hosted, best quality:** Llama 4 Maverick, DeepSeek V3, or Llama 3.3 70B

- **Cost-optimised high volume:** Claude Haiku 4.5 or Gemini 2.0 Flash
- **RAG/search-grounded:** Command R+ (see p. 33)
- **On-premise/regulated:** Llama 3.3 70B or Mistral Large

For reasoning-intensive tasks requiring step-by-step deliberation, see Chapter 3. For models under 7B parameters for edge deployment, see Chapter 4.

3

Reasoning Models

Standard LLMs generate tokens left to right, one at a time, with no pause for reflection. Reasoning models introduce a "thinking" phase: before producing the final answer, the model generates an internal chain of thought—breaking the problem into steps, considering alternatives, checking its own logic.

This sounds minor. It changes everything.

3.1 Why Reasoning Models Exist

Traditional LLMs fail predictably on tasks that require multi-step logic: mathematical proofs, complex planning, code debugging with subtle edge cases, scientific reasoning. The failure mode is always the same: the model commits to an answer too early, before it has thought the problem through.

Reasoning models fix this by trading latency for accuracy. They take longer. They think first. They get it right more often.

3.2 The Landscape

3.2.1 o1 — OpenAI

The model that launched the reasoning category (September 2024). Uses a trained chain-

of-thought process that is hidden from the user. 200K context input, 100K output. Excels at math, science, and complex coding. Significantly slower than GPT-4o; significantly more accurate on hard problems.

Best for: Graduate-level math, competitive programming, scientific analysis.

3.2.2 o3 & o3-mini — OpenAI

The successor to o1, released early 2025. Higher ceiling on reasoning tasks. o3-mini offers adjustable "reasoning effort" (low/medium/high)—a practical knob for trading speed against accuracy. o3-mini-high matches full o1 on many benchmarks at lower cost.

Best for: Adjustable-effort reasoning, STEM tasks, situations requiring a speed–accuracy dial.

3.2.3 Claude with Extended Thinking — Anthropic

Available on Claude Sonnet 4.5 and Claude Opus 4.6. The model's chain of thought is *visible* to the developer (unlike OpenAI's hidden reasoning). Configurable thinking budget. Excels at complex agentic workflows, multi-file code analysis, and long-horizon planning.

Best for: Transparent reasoning, agentic systems, complex code generation.

3.2.4 **Gemini 2.5 Pro with Thinking** — Google DeepMind

Gemini 2.5 Pro includes a built-in thinking mode. Combines reasoning with Gemini's 1M context window and multimodal capabilities. Strong on math and code benchmarks. The combination of massive context + reasoning is unique in the market.

Best for: Long-context reasoning, multimodal analysis requiring deliberation.

3.2.5 **DeepSeek R1** — DeepSeek

Released January 2025. Open-weight reasoning model (671B MoE, 37B active). Trained with reinforcement learning (GRPO) rather than supervised fine-tuning on reasoning traces. MIT licence. Performance matches

o1 on many math and code benchmarks. Distilled variants available at 1.5B, 7B, 8B, 14B, 32B, and 70B.

Best for: Self-hosted reasoning, research, budget-conscious reasoning at scale.

3.2.6 **QwQ-32B** — Alibaba

32B parameter reasoning model from the Qwen team. 128K context. Apache 2.0 licence. Competitive with much larger reasoning models. Efficient enough to run on a single high-end GPU.

Best for: Self-hosted reasoning on limited hardware, Asian language reasoning.

3.3 When to Use Reasoning Models

Use reasoning when:

- The task involves multi-step logic (math, planning, debugging)
- Accuracy matters more than speed
- The problem has a verifiable correct answer
- Standard LLMs consistently fail on the task

Use standard LLMs when:

- Speed and cost are the priority
- The task is creative, conversational, or open-ended
- The problem is well within the model's training distribution

Reasoning models cost 3–10x more per token and run 2–20x slower. Use them surgically, not by default.

For smaller reasoning-capable models deployable on edge devices, see Chapter 4. For coding-specific reasoning, see Chapter 10.

4

Small & Edge Models

Not every problem needs a 400B-parameter model running on a GPU cluster. A growing class of models under 7 billion parameters can run on laptops, phones, and edge devices—delivering useful intelligence at zero API cost and zero network latency.

The economics are compelling: a 3B model running locally costs nothing per query. A

frontier API costs $3–30 per million tokens. At scale, the difference is existential.

4.1 The Leading Small Models

4.1.1 **Phi-4** — Microsoft

14B parameters. Microsoft's flagship small model. Trained on a curated mix of synthetic and web data emphasising reasoning quality. Remarkably strong on math, science, and code for its size. 16K context. MIT licence.

Best for: On-device reasoning, STEM tasks, when quality/size ratio is paramount.

4.1.2 **Phi-4-mini** — Microsoft

3.8B parameters. Distilled from Phi-4. Multimodal (text + vision). Runs comfortably on

modern smartphones. 128K context support. Strong at structured output and function calling.

Best for: Mobile deployment, lightweight multimodal, edge inference.

4.1.3 **Gemma 3** — Google DeepMind

Available at 1B, 4B, 12B, and 27B. The 27B variant runs on a single GPU and approaches much larger models in quality. Multimodal (image + text) from 4B up. 128K context. Supports 140+ languages. Open under Gemma licence (permissive, commercial use allowed).

Best for: Multilingual on-device, multimodal mobile, single-GPU deployment.

4.1.4 Llama 3.2 1B & 3B — Meta

Purpose-built for edge. 128K context. Optimised for summarisation, instruction following, and tool use. The 3B variant punches well above its weight on general tasks. Llama Community Licence.

Best for: On-device assistants, edge summarisation, mobile tool use.

4.1.5 Qwen 2.5 0.5B–7B — Alibaba

The widest range of small sizes: 0.5B, 1.5B, 3B, 7B. Apache 2.0 licence. The 7B variant is among the strongest open models at its size. Specialised Qwen2.5-Coder variants available (see p. 78). 32K–128K context depending on size.

Best for: Fine-tuning, Chinese/multilingual edge, wide deployment flexibility.

4.1.6 **SmolLM2** — Hugging Face

135M, 360M, and 1.7B variants. Designed for extreme edge: IoT, embedded systems, wearables. Trained on high-quality curated data (FineWeb-Edu, Cosmopedia). Apache 2.0.

Best for: Extreme edge (IoT, wearables), ultra-low resource environments.

4.1.7 **DeepSeek R1 Distilled** — DeepSeek

Reasoning capabilities distilled into small form factors: 1.5B (from Qwen), 7B and 8B (from Qwen/Llama). These models inherit genuine chain-of-thought reasoning ability from the full R1 model (see p. 21). MIT licence.

Best for: On-device reasoning, budget reasoning pipelines.

4.2 Deployment Options

Ollama — Pull-and-run local inference. One-command setup. Supports GGUF quantised models. The easiest on-ramp for local LLMs.

llama.cpp — C/C++ inference engine. GGUF format. CPU and GPU. The performance baseline for local deployment.

vLLM — High-throughput GPU serving with PagedAttention. The standard for self-hosted API serving.

MLX (Apple) — Native Apple Silicon inference. Optimised for M-series chips. Best option for Mac deployment.

MLC-LLM — Cross-platform: Android, iOS, web, GPU. Universal local deployment.

ONNX Runtime — Microsoft's cross-platform inference. Good for Windows/Azure edge scenarios.

4.3 Quantisation Quick Guide

Smaller models get smaller still through quantisation:

FP16/BF16 — Full precision. Best quality, 2 bytes per parameter.

INT8 — 8-bit. Minimal quality loss, half the memory. Safe default.

INT4 / GPTQ / AWQ — 4-bit. Noticeable quality loss on edge cases, quarter the memory. Good for constrained hardware.

GGUF Q4_K_M — llama.cpp's recommended 4-bit. Best quality/size balance for CPU inference.

Rule of thumb: A 7B model at Q4 fits in 4 GB RAM. A 3B at Q4 fits in 2 GB. Plan accordingly.

For full-size models, see Chapter 2. For code-specific small models, see Chapter 10.

5

Embedding & Retrieval

An embedding model converts text (or images, or audio) into a dense vector—a list of numbers that captures meaning. Similar inputs land near each other in vector space. This is the foundation of semantic search, RAG, clustering, classification, and recommendation.

If LLMs are the brain (see p. 5), embeddings are the memory.

5.1 Text Embedding Models

5.1.1 **text-embedding-3-large** — OpenAI

3072 dimensions (adjustable via Matryoshka). 8191 token input. Strong MTEB performance. Supports dimension truncation: use 256–1536 dimensions for faster search with modest quality loss. The default choice for most API-based RAG pipelines.

Best for: General-purpose semantic search, RAG, plug-and-play via API.

5.1.2 **text-embedding-3-small** — OpenAI

1536 dimensions. Same 8191 token input. 5x cheaper than the large variant. Quality is surprisingly close for many tasks. Use this unless your retrieval demands the absolute best.

5.1.3 **Cohere Embed v3** — Cohere

1024 dimensions. 512 token input. Available in `english` and `multilingual` variants. Native support for different input types (search_document, search_query, classification, clustering)—type-aware embeddings improve retrieval quality. Compression to int8/binary supported.

Best for: Multilingual RAG, type-aware embedding, enterprise search with Cohere Rerank.

5.1.4 **Voyage AI 3 / 3-lite** — Voyage AI

1024 dimensions. 32K token context—the longest input window of any embedding model. Variants for code (voyage-code-3), law

(voyage-law-2), finance (voyage-finance-2). Top MTEB scores in domain-specific retrieval.

Best for: Long-document embedding, domain-specific retrieval (code, legal, finance).

5.1.5 **Jina Embeddings v3** — Jina AI

1024 dimensions, Matryoshka support. 8192 token context. Multilingual (90+ languages). Task-specific LoRA adapters for retrieval, classification, separation, and text matching. Open-weight (Apache 2.0).

Best for: Multilingual search, flexible task adaptation, self-hosted embedding.

5.1.6 **BGE-M3** — BAAI

Multi-Functionality, Multi-Linguality, Multi-Granularity. Supports dense, sparse, and ColBERT retrieval from a single model. 8192 tokens. 100+ languages. Open-weight (MIT). The most versatile open embedding model.

Best for: Hybrid retrieval (dense + sparse), multilingual, self-hosted.

5.1.7 **GTE-Qwen2** — Alibaba

Based on Qwen2 backbone. Available at 1.5B and 7B sizes. 128K context—suitable for embedding entire documents without chunking. Strong MTEB performance. Apache 2.0.

Best for: Whole-document embedding, eliminating chunking complexity.

5.1.8 **Nomic Embed Text v1.5** — Nomic AI

768 dimensions. Matryoshka support. 8192 tokens. Fully open-source (weights, code, training data). The transparency champion: you can audit exactly what it learned.

Best for: Open-source-first teams, reproducibility requirements.

5.2 Multimodal Embedding

5.2.1 **SigLIP / SigLIP 2** — Google

Successor to CLIP. Pairwise sigmoid loss instead of contrastive loss—better calibration, works with smaller batches. Multiple resolutions. The backbone of many VLMs (see p. 43).

5.2.2 Jina CLIP v2 — Jina AI

Joint text-image embeddings. 1024 dimensions. 8192-token text input. Multilingual. Can mix text and image queries in the same vector space.

Best for: Cross-modal search (find images with text queries, or vice versa).

5.3 Reranking Models

Embedding retrieval is fast but imprecise. Rerankers are slow but precise. The standard pattern: retrieve 50–100 candidates with embeddings, then rerank the top results with a cross-encoder.

Cohere Rerank 3.5 — The API standard. Multilingual. Handles 4096 token documents. Strong zero-shot relevance scoring.

Jina Reranker v2 — Open-weight (Apache 2.0). 8192 tokens. Multilingual. Best open alternative to Cohere.

BGE Reranker v2-M3 — Open-weight (MIT). Designed to pair with BGE-M3 embeddings. Multi-lingual.

RankLLM / RankGPT — Uses LLMs as rerankers via listwise prompting. Expensive but effective for high-stakes retrieval.

5.4 RAG Architecture Patterns

Naive RAG — Chunk, embed, retrieve top-K, stuff into prompt. Simple. Fragile. Start here.

Hybrid Search — Combine dense (semantic) and sparse (BM25/keyword) retrieval. Covers both conceptual and exact matches.

ColBERT / Late Interaction — Token-level matching between query and document. Higher precision than single-vector embedding. BGE-M3 supports this natively.

Contextual Retrieval — Prepend chunk context (from Anthropic's technique): add a sentence explaining *where* the chunk comes from before embedding. Reduces retrieval failure by 49%.

Agentic RAG — The LLM decides *what* to retrieve, *when*, and *how many times*. Uses tool calling (see p. 93) to query vector stores iteratively.

For using retrieved context in generation, see Chapter 2 (Command R+ (see p. 14) is opti-

mised for this). For document parsing before embedding, see Chapter 11.

6

Vision & Multimodal Understanding

Vision-language models (VLMs) take images as input alongside text and reason about them. This is not image classification—it is open-ended visual understanding: describing scenes, reading charts, extracting data from screenshots, answering questions about photographs.

Every frontier LLM (see p. 5) now supports vision. The differentiation is in *how well*.

6.1 Frontier Multimodal Models

6.1.1 GPT-4o Vision — OpenAI

Native multimodal: images are first-class inputs, not bolted on. Strong at chart reading, document understanding, UI analysis, and spatial reasoning. Supports multiple images per turn. The most widely used vision API.

Best for: General visual QA, chart/graph analysis, UI understanding.

6.1.2 Claude Vision — Anthropic

Available across Opus, Sonnet, and Haiku tiers. Excels at dense document analysis—can

read and reason about multi-page PDFs, complex tables, and fine print. Strong at following visual instructions precisely. Supports up to 20 images per request.

Best for: Document analysis, detailed image reasoning, screenshot-to-code workflows.

6.1.3 Gemini 2.5 Pro Vision — Google DeepMind

Natively multimodal with the added ability to process **video**—up to 1 hour of footage can be analysed within the 1M context window. Also handles interleaved images, audio, and text. The only frontier model with native long-video understanding.

Best for: Video analysis, multimodal interleaved documents, long visual sequences.

6.1.4 **Grok Vision** — xAI

Multimodal input via the Grok 3 model. Available through X platform and API. Competitive on standard visual benchmarks.

6.2 Open-Weight VLMs

6.2.1 **Qwen2.5-VL** — Alibaba

Available from 3B to 72B. Dynamic resolution—processes images at their native aspect ratio without distortion. The 72B variant rivals frontier models on many vision benchmarks. Video understanding supported. Apache 2.0.

Best for: Self-hosted vision, dynamic resolution needs, multilingual visual QA.

6.2.2 **InternVL 2.5** — Shanghai AI Lab

Available from 1B to 78B. Tops multiple vision benchmarks. Supports multi-image input, video, and documents. 8K–32K text context. Open-weight.

Best for: Research, benchmark-optimised vision tasks, multi-image reasoning.

6.2.3 **LLaVA-OneVision** — UW / Microsoft

Open-source VLM with strong single-image, multi-image, and video understanding. Available at 7B and 72B (based on Qwen2). The standard open research VLM.

Best for: Academic research, fine-tuning on custom visual tasks.

6.2.4 **Gemma 3 with Vision** — Google

Available from 4B up. Multimodal from 4B—making it one of the smallest VLMs suitable for on-device image understanding. 128K context. Gemma licence.

Best for: On-device vision, mobile image understanding.

6.2.5 **Florence-2** — Microsoft

230M and 770M. Lightweight but capable of captioning, object detection, grounding, OCR, and segmentation from a single model. Sequence-to-sequence architecture with spatial tokens. MIT licence.

Best for: Efficient multi-task vision, when you need detection + captioning + OCR in one small model.

6.3 Object Detection & Segmentation

6.3.1 **SAM 2** — Meta

Segment Anything Model 2. Zero-shot segmentation of any object in images *and video*. Point, box, or text prompts. Real-time on GPU. The foundation model for segmentation.

Best for: Interactive segmentation, video object tracking, annotation pipelines.

6.3.2 **YOLO v11** — Ultralytics

The latest YOLO. Real-time object detection, segmentation, pose estimation, and classification. Sizes from nano (2.6M params) to extra-large (57M). Runs at 200+ FPS on GPU. AGPL licence (commercial licence available).

Best for: Real-time detection, production edge deployment, surveillance, robotics.

6.3.3 Grounding DINO 1.5 — IDEA Research

Open-set detection: describe an object in text, and the model finds it in the image. No predefined categories needed. Combines with SAM 2 for "describe and segment" workflows.

Best for: Open-vocabulary detection, zero-shot object localisation.

For image generation (producing images rather than understanding them), see Chapter 7. For OCR and document parsing, see Chapter 11.

7

Image Generation

Text-to-image has gone from parlour trick to production tool. The market has bifurcated: **API/cloud models** for integration into products, and **open-weight diffusion models** for customisation and self-hosting.

7.1 Proprietary / Cloud

7.1.1 **DALL-E 3** — OpenAI

Integrated into ChatGPT and available via API. Strongest advantage: prompt adherence. Uses GPT-4 to rewrite user prompts before generation, improving composition and reducing ambiguity. 1024×1024 and 1792×1024. Built-in content policy filtering.

Best for: Product integration via API, reliable prompt following, editorial illustration.

7.1.2 **Midjourney v6.1** — Midjourney

The aesthetic benchmark. Produces the most consistently beautiful images in the market. Excels at photorealism, painterly styles, and coherent scenes. Discord-based workflow (web editor in development). No API (third-party wrappers exist). Subscription model.

Best for: Art direction, concept art, marketing visuals, highest aesthetic quality.

7.1.3 **Google Imagen 3** — Google DeepMind

Available via Vertex AI and Gemini. Strong photorealism and text rendering within images. SynthID watermarking built in. Good at following detailed compositional prompts.

Best for: Google Cloud users, photorealistic marketing content, text-in-image.

7.1.4 **Adobe Firefly 3** — Adobe

Trained exclusively on licensed content (Adobe Stock, public domain). Commercially safe—no copyright infringement risk from training data. Integrated into Photoshop, Il-

lustrator, and Express. Generative Fill and Expand.

Best for: Commercial use where IP safety matters, Adobe workflow integration.

7.1.5 **Ideogram 2.0** — Ideogram

Best-in-class text rendering in images. If your use case requires legible text embedded in generated images (posters, logos, signage), Ideogram leads. Strong compositional control.

Best for: Text-heavy images, posters, logos, signage, typography design.

7.2 Open-Weight / Self-Hosted

7.2.1 **Flux.1** — Black Forest Labs

From the original creators of Stable Diffusion (who left Stability AI). Three tiers: Pro (API), Dev (open, non-commercial), and Schnell (open, Apache 2.0, fastest). Rectified flow architecture. 12B parameters. Excellent prompt adherence and image quality. The new open standard.

Best for: Self-hosted generation, fine-tuning with LoRA, open model ecosystem.

7.2.2 **Stable Diffusion 3.5** — Stability AI

Multimodal Diffusion Transformer (MMDiT) architecture. Available at Large (8B), Medium (2.5B), and Turbo variants. Improved text rendering and composition over SDXL. Open un-

der Stability Community Licence (commercial use with restrictions).

Best for: Legacy SD ecosystem, ControlNet workflows, community fine-tunes.

7.2.3 **SDXL** — Stability AI

The previous generation, but still the ecosystem champion. Massive library of LoRA fine-tunes, ControlNet models, and community tools. 1024×1024 native. If you need a specific style and someone has already trained a LoRA for it, SDXL is likely your model.

Best for: Community fine-tunes, established workflows, maximum ecosystem support.

7.3 Control & Editing Techniques

ControlNet — Condition generation on edges, depth maps, pose skeletons, or segmentation masks. Available for SD 1.5, SDXL, and SD 3.5.

IP-Adapter — Reference-image conditioning. "Generate something in the style of this image." Works with most diffusion models.

Inpainting — Mask a region, regenerate only that region. All major models support this.

Img2Img — Start from an existing image and modify it via prompt. Control the strength of modification.

LoRA fine-tuning — Train a small adapter (4–128 MB) on custom images. 20–50 images can teach a specific style, face, or object. Training takes 15–60 minutes on a single GPU.

For understanding and analysing existing images, see Chapter 6. For generating video, see Chapter 8.

8

Video Generation

Video generation is the youngest category in this atlas. The technology moved from "impressive demo" to "usable tool" in 2024–2025, but limitations remain stark: short duration, high cost, limited controllability.

8.1 The Landscape

8.1.1 **Sora** — OpenAI

Launched December 2024 as part of ChatGPT Pro ($200/month). Generates up to 20 seconds of 1080p video. Diffusion Transformer (DiT) architecture with spacetime patches. Good at cinematic composition, consistent subjects, and realistic physics. Storyboard feature for multi-scene editing. Slow generation (minutes per clip).

Best for: High-quality short cinematic clips, storyboard-driven content.

8.1.2 **Veo 2** — Google DeepMind

Available via VideoFX and Vertex AI. Generates 4K-capable video with strong understanding of real-world physics, lighting, and camera

movement. 8+ seconds at high quality. Supports various cinematic styles and lens effects.

Best for: Realistic video, physics-aware generation, Google Cloud integration.

8.1.3 **Kling 1.6** — Kuaishou

Chinese-developed. Up to 2 minutes of 1080p video—the longest of any model. Strong motion quality. Available via API. Motion Brush feature for directing movement in specific regions.

Best for: Longer-form video, motion-heavy content, budget-friendly generation.

8.1.4 **Runway Gen-3 Alpha** — Runway

The pioneer of creative AI video. 10 seconds at 768p. Strong artistic control. Motion Brush,

camera controls, and image-to-video. Most mature creative tooling (web editor with timeline). Used heavily in advertising and film previsualisation.

Best for: Advertising, creative workflows, iterative editing, film previz.

8.1.5 **Pika 2.0** — Pika Labs

Consumer-focused. "Pikaffects" system for adding specific effects (inflate, melt, explode, crush, etc.) to images and videos. Lip-sync feature. Accessible interface for non-technical users.

Best for: Social media content, effects-driven clips, accessibility for non-experts.

8.1.6 Wan2.1 — Alibaba

Open-source video generation model. Available at 1.3B and 14B. Text-to-video and image-to-video. 480p–720p, up to 5 seconds at 14B. Apache 2.0. The leading open-weight option.

Best for: Self-hosted video generation, research, open-source workflows.

8.2 Current Limitations

- **Duration:** Most models cap at 5–20 seconds. Kling reaches 2 minutes but at lower consistency.
- **Consistency:** Characters and objects can morph between frames. Multi-shot consistency (same character across scenes) remains difficult.

- **Text and fine detail:** Legible text in video is still unreliable.
- **Physics:** Improving rapidly (Veo 2, Sora) but still fails on complex interactions.
- **Cost and speed:** Generation is slow (30 seconds to several minutes per clip) and expensive.
- **Audio:** None of these models generate synchronised audio. Audio must be added separately (see p. 66).

8.3 Practical Workflow

Most production video-AI workflows combine multiple tools:

1. **Script and storyboard** with an LLM (see p. 5)

2. **Generate key frames** with an image model (see p. 51)
3. **Animate with image-to-video** (Runway, Kling, or Wan)
4. **Add voiceover** with TTS (see p. 66)
5. **Add music** with an audio model (see p. 66)
6. **Edit and composite** in traditional tools (Premiere, DaVinci Resolve)

Pure end-to-end AI video production remains aspirational. The assembly pipeline above is how practitioners actually ship today.

9

Speech & Audio

Speech AI splits into three tasks: **speech-to-text** (ASR), **text-to-speech** (TTS), and **music/audio generation**. Each has its own models, trade-offs, and deployment patterns.

9.1 Speech-to-Text (ASR)

9.1.1 **Whisper large-v3** — OpenAI

The open-source ASR standard. 1.55B parameters. Supports 100+ languages. Robust to background noise, accents, and domain-specific vocabulary. Available via OpenAI API or self-hosted. large-v3-turbo variant is 4x faster with minimal quality loss.

Self-hosting: Run with faster-whisper (CTranslate2 backend) for 4–8x speedup over the original PyTorch implementation. Insanely-fast-whisper adds batched inference.

Best for: General-purpose transcription, multilingual, the default starting point.

9.1.2 **Deepgram Nova-2 / Nova-3** — Deepgram

API-only. Nova-2 set the accuracy benchmark in 2024. Nova-3 adds improved multilingual

coverage and diarisation (speaker identification). Real-time streaming with sub-300ms latency. Enterprise pricing.

Best for: Real-time transcription, call centres, production streaming ASR.

9.1.3 **AssemblyAI Universal-2** — AssemblyAI

API-only. Best-in-class accuracy for English. Built-in speaker diarisation, sentiment analysis, entity detection, chapter summarisation. Near-human accuracy on clean audio.

Best for: English-centric pipelines needing transcription + analysis in one call.

9.1.4 **NVIDIA Parakeet / Canary** — NVIDIA

Open-weight via NVIDIA NeMo. Parakeet for English ASR (CTC + RNNT architec-

ture). Canary for multilingual ASR + translation. Strong for on-premise GPU deployment. Apache 2.0.

Best for: On-premise GPU deployment, NVIDIA ecosystem, multilingual ASR.

9.2 Text-to-Speech (TTS)

9.2.1 OpenAI TTS — OpenAI

API-only. Six preset voices. Two quality tiers: `tts-1` (fast, lower quality) and `tts-1-hd` (slower, studio quality). Natural prosody. Simple integration. No voice cloning.

Best for: Quick integration, consistent voice quality, prototype-to-production.

9.2.2 **ElevenLabs** — ElevenLabs

The quality leader. Voice cloning from 30 seconds of audio. 32 languages. Voice design from text description. Real-time streaming. Professional dubbing and lip-sync. The industry standard for content creators.

Best for: Voice cloning, audiobook narration, content creation, dubbing.

9.2.3 **Cartesia Sonic** — Cartesia

State-space model (SSM) architecture. Ultra-low latency (under 90ms). Real-time streaming. Voice cloning. Designed for conversational AI—when the voice needs to respond *instantly*.

Best for: Real-time voice assistants, conversational AI, latency-critical applications.

9.2.4 **Fish Speech 1.5** — Fish Audio

Open-source (Apache 2.0). Zero-shot voice cloning from 10–30 seconds of audio. VQGAN + Transformer architecture. Supports Chinese, English, Japanese, Korean. Runs on consumer GPUs.

Best for: Self-hosted TTS, open-source voice cloning, Asian languages.

9.2.5 **Parler-TTS** — Hugging Face

Describe the voice in natural language ("a young woman speaking quickly with excitement"). No reference audio needed. Open-source. Good for generating diverse synthetic voices without cloning.

Best for: Synthetic voice diversity, no-sample voice creation, research.

9.3 Music & Audio Generation

9.3.1 **Suno v4** — Suno

Full-song generation from text prompts. Lyrics, vocals, instrumentation, mixing. Generates 2–4 minute songs with verse-chorus structure, multiple instruments, and coherent vocals. The consumer standard.

Best for: Full song production, prototyping, content soundtracks.

9.3.2 **Udio** — Udio

Suno's primary competitor. Better tonal fidelity in some genres (classical, jazz). Stem separation. Inpainting (regenerate specific sections). More fine-grained control over arrangement.

Best for: Genre-specific quality, arrangement control, musical experimentation.

9.3.3 **MusicGen** — Meta

Open-weight (MIT). Text-to-music and melody-conditioned generation. 300M to 3.3B parameters. 30-second clips. No vocals. The open-source baseline for instrumental music generation.

Best for: Self-hosted background music, open-source audio pipelines.

9.3.4 **Stable Audio 2.0** — Stability AI

Up to 3 minutes of 44.1 kHz stereo. Audio-to-audio transformation. Latent diffusion on audio spectrograms. Good at sound effects and ambient audio in addition to music.

Best for: Sound design, ambient audio, longer instrumental generation.

For voice-driven agent interactions, see Chapter 12. For audio transcription as part of document processing, see Chapter 11.

10

Code Models

Every frontier LLM can write code. But there is a meaningful difference between "can write code" and "is optimised for code." This chapter covers models and tools specifically built for software engineering.

10.1 Frontier Code Capabilities

The best code models in early 2026, ranked by practitioner consensus:

1. **Claude Opus 4.6 / Sonnet 4.5** (see p. 7) — Best agentic coding. Excels at multi-file edits, complex refactoring, and understanding large codebases. Extended thinking mode (see p. 20) enables multi-step debugging. Powers Claude Code (CLI agent).

2. **Gemini 2.5 Pro** (see p. 9) — 1M context allows feeding entire repositories. Strong at code review, migration, and cross-file reasoning.

3. **GPT-4o** (see p. 6) — Fast, reliable, good at generating boilerplate and standard patterns. Powers GitHub Copilot.

4. **o3 / o3-mini** (see p. 19) — Best for competitive programming and algorithmic challenges. Reasoning mode catches edge cases that standard models miss.

10.2 Open-Weight Code Models

10.2.1 DeepSeek Coder V2 / V3 — DeepSeek

DeepSeek V3 (see p. 12) is the strongest open code model, competitive with GPT-4o on HumanEval and SWE-bench. 128K context. MoE architecture (37B active). The earlier Coder V2 (16B/236B) remains popular for its efficiency.

Best for: Self-hosted coding agent, competitive programming, cost-efficient code generation.

10.2.2 **Qwen2.5-Coder** — Alibaba

Available at 1.5B, 7B, 14B, and 32B. The 32B variant is the strongest open code model at single-GPU scale. 128K context. Excels at code completion, repair, and reasoning. Apache 2.0.

Best for: Single-GPU self-hosted coding, fine-tuning on proprietary codebases.

10.2.3 **Codestral** — Mistral AI

22B. Purpose-built for code. 32K context. Trained on 80+ programming languages. Fill-in-the-middle (FIM) support for inline completion. Available via API and downloadable. Non-production licence for the open version.

Best for: IDE completion (FIM), multilingual codebases, Mistral ecosystem.

10.2.4 **StarCoder 2** — BigCode / Hugging Face

3B, 7B, and 15B. Trained on The Stack v2 (600+ languages). 16K context. Fully open (BigCode OpenRAIL-M licence). Strong at code completion, particularly for less common languages.

Best for: Open-source compliance, niche language support, completion tasks.

10.2.5 **CodeLlama** — Meta

7B, 13B, 34B, and 70B. Based on Llama 2. FIM support. Python-specialised variant. 100K context at 7B/13B. Being superseded by newer models but still widely deployed.

Best for: Legacy deployments, Python-specific tasks, Llama ecosystem.

10.3 Code Assistants & IDEs

10.3.1 **GitHub Copilot** — GitHub / Microsoft

The market leader. Inline completion, chat, multi-file edit. Now supports model selection (GPT-4o, Claude Sonnet, Gemini). Copilot Workspace for issue-to-PR workflows. $10/month individual, $19/month business.

10.3.2 **Cursor** — Cursor Inc.

VS Code fork with AI deeply integrated. Tab completion, inline edits, multi-file "Composer" agent. Supports Claude Sonnet/Opus, GPT-4o, and custom models. Agent mode for autonomous task completion. $20/month Pro.

10.3.3 **Claude Code** — Anthropic

Terminal-based agentic coding assistant. Reads, edits, and navigates codebases autonomously. Runs shell commands, creates commits, manages PRs. Extended thinking for complex tasks. Works directly in the developer's environment—no IDE lock-in.

Best for: Terminal-first developers, complex multi-file refactoring, agentic workflows.

10.3.4 **Windsurf** — Codeium

VS Code fork focused on "agentic IDE." Cascade system for multi-step code generation. Context-aware across the workspace. Free tier available.

10.4 Structured Output & Function Calling

Most production code involves LLMs returning structured data (JSON, function calls), not free text. Support varies:

OpenAI — JSON mode, structured outputs with JSON Schema enforcement, function calling with parallel tool use. The most mature implementation.

Anthropic — Tool use with JSON Schema. Forced tool use mode. Strong schema compliance.

Google — Function calling, JSON mode. Gemini 2.0 supports code execution in a sandbox.

Open models — Outlines, llama.cpp grammars, vLLM guided decoding. Enforce structure at the inference engine level.

For building full agentic systems around code models, see Chapter 12. For embedding code for retrieval, see Voyage-Code (see p. 35).

11

OCR & Document AI

Extracting text from images, PDFs, and scanned documents. This unglamorous category is the plumbing of most enterprise AI—because real-world data lives in documents, not databases.

11.1 Cloud Document AI Services

11.1.1 **Azure AI Document Intelligence** — Microsoft

Formerly Form Recognizer. Prebuilt extractors for invoices, receipts, ID cards, tax forms, health insurance cards. Custom model training. Table extraction, key-value pair extraction, layout analysis. The enterprise leader.

Best for: Enterprise document processing, structured extraction, Microsoft ecosystem.

11.1.2 **Google Document AI** — Google Cloud

Specialised processors for invoices, contracts, lending documents. Custom document extraction with Workbench. OCR via Cloud Vision API. Strong at handwriting recognition.

Best for: Google Cloud users, handwriting-heavy documents, custom extractors.

11.1.3 **Amazon Textract** — AWS

Table and form extraction. Queries feature: ask natural language questions about document content. AnalyzeExpense for receipts and invoices. AWS ecosystem integration.

Best for: AWS users, query-based document analysis, receipt processing.

11.2 Open-Source OCR

11.2.1 **Tesseract 5** — Google / Community

The venerable open-source OCR engine. LSTM-based. 100+ languages. Free. Quality is moderate: good on clean, high-resolution text; struggles with complex layouts, tables, and noisy scans. The baseline everyone benchmarks against.

Best for: Simple OCR tasks, budget-zero deployments, 100+ language coverage.

11.2.2 **PaddleOCR** — Baidu

Comprehensive OCR toolkit: detection, recognition, layout analysis, table recognition, key information extraction. PP-OCRv4 engine. Excellent for Chinese, Japanese, Korean. 80+ languages. Apache 2.0.

Best for: CJK documents, full-pipeline OCR, layout analysis, production deployment.

11.2.3 **EasyOCR** — Jaided AI

Simple Python API. 80+ languages. Uses CRAFT for detection and CRNN for recognition. Easy to set up—pip install and three lines of code. Moderate accuracy. Apache 2.0.

Best for: Quick prototyping, simple OCR needs, when setup speed matters most.

11.2.4 **Surya** — VikParuchuri

Modern OCR with layout detection, text recognition, reading order detection, and table recognition. 90+ languages. Transformer-based. Outperforms Tesseract on most benchmarks. GPL-3.0 (commercial licence available).

Best for: High-quality open-source OCR, complex layouts, research.

11.2.5 **DocTR** — Mindee

Two-stage OCR (detection + recognition) with TensorFlow and PyTorch backends.

Word-level and line-level recognition. Good table handling. Apache 2.0.

Best for: Modular OCR pipeline, when you need to swap detection/recognition components.

11.2.6 GOT-OCR 2.0 — StepFun

"General OCR Theory." End-to-end vision-language model for OCR. Handles plain text, formatted text, tables, mathematical formulas, molecular structures, sheet music, and geometric figures from a single model. Open-weight.

Best for: Multi-format document understanding, scientific documents, sheet music.

11.3 PDF Processing

PDFs are not OCR—they contain embedded text (usually). But extracting structured content from PDFs is its own category of pain.

Marker — Converts PDFs to Markdown. Uses Surya for OCR, layout detection, and table extraction. Handles multi-column layouts, headers, footers, and equations. The best open-source PDF-to-Markdown tool.

PyMuPDF / fitz — Fast PDF text extraction. Handles text-based PDFs well. No OCR. Lightweight.

Unstructured.io — Ingestion pipeline for all document types: PDF, DOCX, PPTX, HTML, images. Combines multiple tools (Tesseract, PaddleOCR, layout models). Outputs structured elements.

LlamaParse — LlamaIndex's document parser. Cloud service. Uses VLMs to understand complex document layouts. Handles tables, charts, and images within documents.

11.4 The VLM Approach

Increasingly, practitioners skip traditional OCR entirely and feed document images directly to vision-language models (see p. 43). Claude Vision (see p. 44) and Gemini 2.5 Pro (see p. 45) can read multi-page documents with high accuracy, extracting not just text but semantic structure, relationships, and meaning.

This approach is more expensive per page but dramatically simpler to implement and more robust to layout complexity.

When to use VLMs: Complex layouts, mixed content (text + charts + tables), when you need *understanding* not just extraction.

When to use traditional OCR: High volume (millions of pages), simple layouts, cost-sensitive batch processing, offline/on-premise requirements.

For embedding extracted document text for search, see Chapter 5.

12

Agents & Frameworks

An AI agent is an LLM that can *act*: read files, call APIs, query databases, browse the web, execute code, and iterate on its own output until the task is done. The shift from "chat" to "agent" is the defining transition of 2025–2026.

12.1 What Makes an Agent

An agent has three capabilities beyond a vanilla LLM:

Tool use — The model calls external functions (APIs, databases, file systems) by emitting structured tool-call requests. The framework executes the call and feeds the result back.

Planning — The model decomposes complex tasks into steps, executes them in sequence or parallel, and adjusts the plan based on intermediate results.

Memory — The model retains context across turns and interactions: conversation history, working state, and long-term knowledge.

12.2 Agent SDKs

12.2.1 **Claude Agent SDK** — Anthropic

Official Python/TypeScript SDK for building agents with Claude. Tool use, extended thinking (see p. 20), multi-turn orchestration. Tight integration with Claude Code (see p. 81). Guardrails and safety built in. The simplest path from prototype to production for Claude-based agents.

Best for: Production Claude agents, safety-critical deployments.

12.2.2 **OpenAI Agents SDK** — OpenAI

Released March 2025. Successor to the experimental Swarm framework. Agents, handoffs (routing between specialised agents), guardrails, tracing. Built-in tool types: web

search, file search, code interpreter, computer use. Production-ready.

Best for: Multi-agent workflows, OpenAI ecosystem, tool-heavy agents.

12.3 Orchestration Frameworks

12.3.1 LangChain / LangGraph — LangChain Inc.

The ecosystem giant. LangChain provides chains, prompts, retrievers, and tool integrations. LangGraph adds stateful, cyclic agent graphs—agents that loop, branch, and checkpoint. LangSmith for tracing and evaluation. Supports every major LLM provider.

Strengths: Largest ecosystem, most integrations, graph-based agent design.
Weakness: Abstraction overhead. Simple tasks become complex. Learning curve.

Best for: Complex multi-step agents, teams that need observability, production RAG.

12.3.2 **CrewAI** — CrewAI

Multi-agent role-playing framework. Define agents with roles ("researcher," "writer," "reviewer"), assign tasks, and let them collaborate. Sequential or hierarchical execution. Simple API.

Best for: Multi-agent collaboration patterns, role-based task decomposition.

12.3.3 **AutoGen** — Microsoft

Multi-agent conversation framework. Agents communicate via messages, can include humans in the loop, and support code execu-

tion. GroupChat for multi-agent discussions. Strong research backing.

Best for: Research, human-in-the-loop agents, multi-agent discussion patterns.

12.3.4 **Semantic Kernel** — Microsoft

Enterprise SDK for .NET, Python, and Java. Plugins, planners, and memory. Azure-first but supports OpenAI and other providers. The choice for enterprise .NET shops.

Best for: Enterprise .NET environments, Azure integration.

12.3.5 **Haystack** — deepset

Pipeline-based framework. Components (retrievers, readers, generators) connect into DAGs. Strong at RAG (see p. 33). Clean architec-

ture. Smaller community but more principled design.

Best for: RAG-centric agents, clean pipeline architecture.

12.3.6 **Dify** — Dify.AI

Visual workflow builder for AI applications. No-code/low-code. RAG, agents, and workflow orchestration via drag-and-drop. Self-hostable. Good for teams with mixed technical backgrounds.

Best for: Visual agent design, non-developer teams, rapid prototyping.

12.4 MCP: Model Context Protocol

Anthropic's open standard for connecting AI agents to external tools and data sources. Instead of building custom tool integrations per agent, MCP provides a universal protocol:

- **MCP Servers** expose tools and data (filesystem, databases, APIs, SaaS apps)
- **MCP Clients** (the AI agent) discover and call these tools via a standard JSON-RPC protocol
- Growing ecosystem: official servers for GitHub, Slack, PostgreSQL, filesystem, Google Drive, and many more

Why it matters: MCP solves the N×M integration problem. Instead of every agent framework integrating with every tool in-

dependently, each tool publishes one MCP server and every agent can use it.

12.5 How to Choose

- **Simple single-agent:** Use the provider's SDK directly (Claude Agent SDK, OpenAI Agents SDK). No framework overhead needed.
- **Complex multi-agent:** LangGraph or CrewAI.
- **Enterprise .NET:** Semantic Kernel.
- **RAG-focused:** Haystack or LangChain.
- **Non-technical team:** Dify.
- **Research:** AutoGen.

The most common mistake is reaching for a framework too early. Start with raw API calls. Add a framework when you need state man-

agement, multi-agent coordination, or production observability—not before.

For the underlying models that power agents, see Chapter 2 and Chapter 3. For code-specific agents, see Chapter 10.

References & Further Reading

Vaswani, A. et al. (2017). "Attention Is All You Need." *NeurIPS*.
The Transformer architecture that started it all.

Kaplan, J. et al. (2020). "Scaling Laws for Neural Language Models." *arXiv:2001.08361*.
Why making models bigger keeps working.

Muennighoff, N. et al. (2023). "MTEB: Massive Text Embedding Benchmark." *EACL*.
The standard benchmark for comparing embedding models.

Rombach, R. et al. (2022). "High-Resolution Image Synthesis with Latent Diffusion Models." *CVPR*.
The paper behind Stable Diffusion.

Radford, A. et al. (2023). "Robust Speech Recognition via Large-Scale Weak Supervision." *ICML*.
The Whisper paper.

DeepSeek-AI (2025). "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning." *arXiv:2501.12948*.
How reinforcement learning produces reasoning without supervised traces.

Anthropic (2025). "Model Context Protocol Specification." *modelcontextprotocol.io*.
The open standard for agent-tool communication.

Grattafiori, A. et al. (2025). "The Llama 4 Herd of Models." *Meta AI*.

Mixture-of-Experts at scale with 10M token context.

About the Author

Nabil El Mahyaoui

https://nabil-el-mahyaoui.com/

https://www.linkedin.com/in/elmahyaoui