Best Vision Models You Can Run Locally: Every Model, Every GPU Tier (2026)

February 6, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Qwen3-VL 8B is the new default local vision model. It scores 85.8 on MathVista and handles OCR, charts, screenshots, and photos better than Qwen2.5-VL 7B while using the same VRAM. On 4GB, Gemma 3 4B at int4 (2.6GB) or SmolVLM2 2.2B (~2GB) are your options. On 16GB+, Gemma 3 27B QAT (14GB) leads general understanding. For documents specifically, PaddleOCR-VL 0.9B runs on nearly anything and scores 92.6 on OmniDocBench. Phi-4-reasoning-vision 15B is the new pick for math and science diagrams if you have 10GB+ VRAM. All major models except PaddleOCR-VL run in Ollama with a single command.

📚 **More on this topic:** VRAM Requirements · Qwen 3.5 Guide · PaddleOCR-VL Setup · Qwen2.5-VL in LM Studio · Best Local LLMs on Mac

You can point a local model at an image and ask “what’s in this?” — describe a photo, extract text from a screenshot, read a chart, convert handwritten notes, analyze a UI mockup. All of it runs on your GPU. Nothing leaves your machine.

A lot changed in early 2026. Qwen3-VL replaced Qwen2.5-VL as the vision model to beat. Phi-4-reasoning-vision can actually solve math problems from photographs now. PaddleOCR-VL made dedicated document OCR nearly free to run. And Qwen 3.5 baked vision directly into its base architecture, though Ollama support for that is still catching up.

This guide covers which model for which GPU, how to set each one up, what speed to expect, and where local still falls short of cloud APIs.

Which vision model should you run?

Short version, by GPU tier:

Your VRAM	Best Pick	Ollama Command	Why
4 GB	Gemma 3 4B (int4)	`ollama run gemma3:4b`	2.6 GB, real vision at minimum VRAM

Your VRAM	Best Pick	Ollama Command	Why
4 GB	SmolVLM2 2.2B	— (HuggingFace)	~2 GB, edge-grade but functional
8 GB	Qwen3-VL 8B (Q4)	`ollama run qwen3-vl:8b`	Best all-around, fits comfortably
8 GB	Qwen2.5-VL 7B (Q4)	`ollama run qwen2.5vl:7b`	Still excellent, proven track record
10-12 GB	Phi-4-reasoning-vision 15B (Q4)	— (llama.cpp)	Math and science diagrams, ~10 GB
16 GB	Gemma 3 27B (QAT int4)	`ollama run gemma3:27b`	Highest general visual understanding
24 GB	Qwen2.5-VL 32B (Q4)	`ollama run qwen2.5vl:32b`	Best all-around at 24 GB
48 GB+	Qwen2.5-VL 72B (Q4)	`ollama run qwen2.5vl:72b`	Maximum quality, needs serious hardware
Any (OCR only)	PaddleOCR-VL 0.9B	pip install	Runs on CPU, 92.6% doc accuracy

If you just want an answer: `ollama run qwen3-vl:8b` and start asking it about images.

What vision models do (and don't do)

Vision language models take an image plus a text prompt and return text. They understand images. They don't generate them.

What works:

- Describe photos, scenes, objects
- Read text from screenshots and scanned documents (OCR)
- Interpret charts, graphs, diagrams
- Analyze UI screenshots and identify elements
- Convert code screenshots to copyable text
- Answer questions about image content
- Solve math problems from photographed pages

What doesn't work:

- Generating images (that's Stable Diffusion or Flux)
- Real-time video analysis (frame-by-frame is possible but slow)
- Editing or modifying images
- Reliably counting large numbers of small objects

Every vision model worth running locally

Qwen3-VL 8B — the new default

Released late 2025, Qwen3-VL replaced Qwen2.5-VL at the top. The 8B model handles OCR, charts, screenshots, and math better than its predecessor at every benchmark.

Benchmark	Qwen3-VL 8B	Qwen2.5-VL 7B	Difference
MathVista	85.8	68.2	+17.6
MMMU	—	58.6	—
DocVQA	95+	95.7	Comparable
ChartQA	88+	87.3	Comparable

The MathVista jump matters most. Going from 68.2 to 85.8 means Qwen3-VL can actually reason about math problems in photographs, not just read the numbers.

Available at 2B, 8B, and 32B. The 30B-A3B MoE variant uses only 3B active parameters, so it fits in ~14-16 GB despite the total parameter count.

```
ollama run qwen3-vl:8b     # Best for 8GB GPUs
ollama run qwen3-vl:2b     # Tight VRAM
ollama run qwen3-vl:32b    # 24GB GPUs
```

It's the best local all-rounder right now. Strong OCR, strong math reasoning from images, dynamic resolution support. The main downside is that it's newer, so community tooling and tutorials still reference Qwen2.5-VL more often. The 32B variant needs 21+ GB.

Qwen2.5-VL 7B — still the proven workhorse

Qwen2.5-VL 7B held the crown for months and earned its reputation. Its 95.7 DocVQA score was the number that made people take local vision models seriously — a 7B model reading documents better than many 70B text-only models handle their own tasks.

Benchmark	Qwen2.5-VL 7B	Llama 3.2 Vision 11B
MMMU	58.6	50.7
DocVQA	95.7	88.4
ChartQA	87.3	—
MathVista	68.2	51.5
OCRBench	86.4	—

Still worth running if you're already using it, if you need the most battle-tested option, or if Qwen3-VL hasn't landed in your toolchain yet. Available at 3B, 7B, 32B, and 72B.

```
ollama run qwen2.5vl:7b     # Best for 8GB GPUs
ollama run qwen2.5vl:3b     # Testing and low VRAM
ollama run qwen2.5vl:32b    # 24GB GPUs
```

For LM Studio setup specifically, see our Qwen2.5-VL LM Studio guide.

Gemma 3 — Google's QAT advantage

Gemma 3 comes in three sizes with native vision: 4B, 12B, 27B. The 27B scores highest on MMMU (64.9) among mid-size consumer models, so if you care about general visual reasoning and have the VRAM, it's the pick.

The interesting part is Google's QAT (Quantization-Aware Trained) versions. These models were trained to maintain quality at int4, not just quantized after the fact. Gemma 3 27B QAT fits in 14 GB with near-BF16 quality.

Gemma 3 Size	BF16 VRAM	Int4/QAT VRAM	MMMU
4B	~8 GB	~2.6 GB	48.8
12B	~24 GB	~6.6 GB	59.6

Gemma 3 Size	BF16 VRAM	Int4/QAT VRAM	MMMU
27B	~54 GB	~14.1 GB	64.9

```
ollama run gemma3:4b    # Fits on 4GB VRAM
ollama run gemma3:12b   # Fits on 8GB VRAM
ollama run gemma3:27b   # Needs 16GB+ VRAM
```

Best general visual understanding at the 27B tier, and QAT preserves quality well. 128K context. The downside: DocVQA trails Qwen2.5-VL significantly (86.6 vs 95.7). If document reading is your main use, Qwen wins.

Phi-4-reasoning-vision 15B — the math specialist

Microsoft released this in March 2026. It pairs Phi-4-Reasoning with a SigLIP-2 vision encoder. The model decides when to engage deep reasoning and when a quick answer is enough, which Microsoft calls "adaptive thinking."

Benchmark	Phi-4-reasoning-vision 15B	Qwen2.5-VL 7B
MathVista	75.2	68.2
AI2D (science diagrams)	84.8	—
ChartQA	83.3	87.3
MMMU	54.3	58.6

The science diagram score (84.8 AI2D) is where this model stands out. If you're feeding it photographed textbook pages, chemistry diagrams, or physics problems, it outperforms the generalists.

~9-10 GB at Q4_K_M. Fits on a 12 GB GPU comfortably.

Gotcha: Not in Ollama's official library as of March 2026. Run it through llama.cpp with the mmproj file, or check HuggingFace for GGUF conversions. The separate mmproj architecture means Ollama support will lag behind.

Strong at math reasoning from images and science diagrams. Adaptive thinking saves tokens on simple tasks. But MMMU trails Qwen2.5-VL, it's not in Ollama, and it's brand new with limited community experience.

PaddleOCR-VL 0.9B — the document specialist

This is not a general-purpose vision model. It's an OCR engine that uses a vision-language architecture. If you need to extract text, tables, formulas, and charts from documents, PaddleOCR-VL is absurdly good for its size.

0.9 billion parameters. Runs on CPU. Scores 92.6 on OmniDocBench. Supports 109 languages.

Benchmark	PaddleOCR-VL 0.9B	Qwen2.5-VL 7B
OmniDocBench	92.6	—
Text Edit Distance	0.035	—
Formula CDM	91.4	—
DocVQA	—	95.7

The trade-off is clear: PaddleOCR-VL extracts document content with higher fidelity (tables, formulas, layouts) than a general VLM, but it can't describe a photo, read a chart contextually, or answer open-ended questions about images.

```
pip install "paddleocr[doc-parser]"
# Launch as a server:
paddleocr genai_server --model_name PaddleOCR-VL-0.9B --backend vllm --port 8118
```

Gotchas: Requires PaddlePaddle, not PyTorch. Not available in Ollama or LM Studio. vLLM backend supported. Think of it as specialized infrastructure, not a chat model.

For the full setup walkthrough, see our PaddleOCR-VL guide.

SmolVLM2 — vision on nothing

HuggingFace's SmolVLM2 comes in 256M, 500M, and 2.2B sizes. The 2.2B version uses about 2 GB of VRAM and produces usable results for captioning, basic VQA, and simple document reading.

This is the model for Raspberry Pis, old phones, and situations where you genuinely have no VRAM budget. It won't compete with Qwen3-VL on quality, but it runs where nothing else fits.

It runs on almost anything and even handles basic video understanding. Accuracy drops hard compared to 7B+ models, though. Not in Ollama, so you'll need HuggingFace transformers to run it.

Llama 3.2 Vision 11B — falling behind

Meta's entry still works and has strong community support, but benchmark-for-benchmark it loses to Qwen2.5-VL 7B on nearly every task despite being larger. With Qwen3-VL now available, Llama 3.2 Vision is two generations behind on performance.

```
ollama run llama3.2-vision        # 11B default
ollama run llama3.2-vision:90b    # Multi-GPU only
```

Use this if: You're locked into the Llama ecosystem or following tutorials that reference it.
Otherwise: Qwen3-VL 8B or Qwen2.5-VL 7B.

Qwen 3.5 — native multimodal (with a catch)

Qwen 3.5 is different from Qwen3-VL. It's not a vision model bolted onto a text model — vision is baked into the architecture from the start using "early fusion" training. Every Qwen 3.5 model, from the 0.8B to the 397B-A17B MoE flagship, handles images natively.

The catch: as of March 2026, Qwen 3.5 GGUF does not work for vision tasks in Ollama. The separate mmproj (multimodal projector) files aren't handled correctly. You need llama.cpp directly or wait for Ollama to catch up.

For text-only Qwen 3.5, see our Qwen 3.5 guide. For vision right now, stick with Qwen3-VL or Qwen2.5-VL through Ollama.

LLaVA — the pioneer, now legacy

LLaVA proved open-source vision models could work in 2023. LLaVA-OneVision (2024) improved things with multi-image support and higher resolutions. But the field moved past it. Qwen2.5-VL, Qwen3-VL, Gemma 3, and Phi-4 all outperform LLaVA variants at comparable sizes.

LLaVA-OneVision isn't even in Ollama's official library — there's an open GitHub issue (#6255) for it. The older LLaVA 1.6 is still available:

```
ollama run llava           # LLaVA 1.6 7B
ollama run llava:34b       # Needs 24GB+ VRAM
```

Use LLaVA if: You're following older tutorials that reference it. **Otherwise:** Start with Qwen3-VL 8B.

VRAM requirements table

What you can actually run on your GPU, with real numbers:

Model	Params	Q4 VRAM	Q8 VRAM	FP16 VRAM	Ollama?
SmolVLM2 2.2B	2.2B	~2 GB	~3 GB	~4.5 GB	No
PaddleOCR-VL 0.9B	0.9B	~1-2 GB	~2 GB	~2 GB	No
Gemma 3 4B	4B	~2.6 GB	~5 GB	~8 GB	Yes
Qwen3-VL 2B	2B	~2 GB	~3 GB	~4 GB	Yes
Phi-4-multimodal	5.6B	~4 GB	~7 GB	~11 GB	No
Qwen2.5-VL 7B	7B	~6 GB	~8 GB	~14 GB	Yes
Qwen3-VL 8B	8B	~6 GB	~9 GB	~16 GB	Yes
Gemma 3 12B	12B	~6.6 GB	~13 GB	~24 GB	Yes
Llama 3.2 Vision 11B	10.7B	~8 GB	~12 GB	~22 GB	Yes
Phi-4-reasoning-vision	15B	~10 GB	~16 GB	~30 GB	No
Gemma 3 27B (QAT)	27B	~14 GB	~28 GB	~54 GB	Yes
Qwen3-VL 32B	32B	~21 GB	~34 GB	~64 GB	Yes
Qwen2.5-VL 72B	72B	~40 GB	~71 GB	~144 GB	Yes

Key points:

- Vision models use more VRAM than text-only models of the same size because the vision encoder adds overhead
- High-resolution images spike VRAM temporarily — leave 1-2 GB headroom

- The “Q4 VRAM” column is what matters for consumer GPUs — that’s the sweet spot between quality and fit

For text model VRAM numbers, see our VRAM requirements guide. Use our planning tool to check exact fit for your setup.

Speed expectations

Vision tasks are slower than pure text because images consume hundreds to thousands of tokens. A single image might cost 729 to 3,600 tokens depending on the model and resolution. That means higher first-token latency and more compute per turn.

Approximate generation speed (tokens/second, Q4 quantization):

Model	RTX 4090 (24 GB)	RTX 3090 (24 GB)	M4 Pro (24 GB)	CPU only
Qwen3-VL 8B	100-140 t/s	80-120 t/s	40-60 t/s	8-15 t/s
Gemma 3 12B	60-80 t/s	50-65 t/s	30-45 t/s	5-10 t/s
Phi-4-vision 15B	40-60 t/s	35-50 t/s	25-35 t/s	4-8 t/s
Gemma 3 27B	30-40 t/s	25-35 t/s	15-25 t/s	2-5 t/s

These are output token speeds. The first token takes noticeably longer with vision because the model needs to process the image through the vision encoder before generating. Expect 2-5 seconds for first token on a 7-8B model with a standard image, longer for high-resolution or multi-image inputs.

How to run vision models

Ollama (easiest)

Pass images via the CLI, Python SDK, or REST API.

CLI:

```
# Pull the model
ollama pull qwen3-vl:8b
```

```
# Describe an image
ollama run qwen3-vl:8b "What's in this image?" --image ./photo.jpg

# Multiple images
ollama run qwen3-vl:8b "Compare these two screenshots" --image ./before.png --image ./after.png
```

Python SDK:

```
import ollama

response = ollama.chat(
    model='qwen3-vl:8b',
    messages=[{
        'role': 'user',
        'content': 'What text is in this screenshot?',
        'images': ['screenshot.png']
    }]
)
print(response['message']['content'])
```

REST API:

```
curl http://localhost:11434/api/chat -d '{
  "model": "qwen3-vl:8b",
  "messages": [{
    "role": "user",
    "content": "Describe this image",
    "images": ["BASE64_STRING_HERE"]
  }]
}'
```

Gotchas:

- Don't include `` in API calls — Ollama expects raw base64 only
- Unsupported image formats can hang the server instead of returning an error. Stick to JPEG, PNG, and WebP
- Vision models load slower than text-only on first inference due to vision encoder initialization
- Qwen 3.5 vision does NOT work in Ollama as of March 2026 — use Qwen3-VL instead

For detailed Ollama setup, see our beginner tutorial.

LM Studio

LM Studio supports vision models with drag-and-drop image input. Supports Gemma 3, Qwen2.5-VL, LLaVA, and other GGUF vision models. Drag images into the chat window, or drop PDFs for document analysis. On Mac, the MLX engine provides native Apple Silicon acceleration.

For Qwen2.5-VL specifically in LM Studio, see our dedicated setup guide.

llama.cpp (advanced)

llama.cpp handles vision models through its `libmtmd` library. You need two files: the main model GGUF and a separate `mmproj` (multimodal projector) GGUF.

```
# Auto-download model + mmproj from HuggingFace
llama-server -hf ggml-org/gemma-3-4b-it-GGUF

# Or specify files manually
llama-server --model gemma-3-4b-Q4_K_M.gguf --mmproj mmproj-model-f16.gguf

# Interactive CLI with vision
llama-mtmd-cli -hf ggml-org/gemma-3-4b-it-GGUF
# Then type /image path-to-image.jpg in the chat
```

This is how you run Phi-4-reasoning-vision and other models not yet in Ollama. The `-hf` flag auto-downloads both files. See our llama.cpp vs Ollama vs vLLM comparison for when to use each.

Use cases with real examples

Document OCR and text extraction

This is where local vision models are genuinely competitive with cloud APIs. Feed the model receipts, invoices, handwritten notes, scanned pages, or PDFs.

Best models: PaddleOCR-VL 0.9B (specialized, 109 languages), Qwen2.5-VL 7B / Qwen3-VL 8B (general purpose)

```
Prompt: "Extract all text from this document. Preserve the table structure."
```

PaddleOCR-VL is better when you need exact layout preservation — tables, formulas, column structures. Qwen models are better when you need the model to understand the document and answer questions about its content.

For privacy-sensitive documents (medical records, legal filings, financial statements), local vision models are the only option that guarantees nothing leaves your network.

Screenshot and UI analysis

Point a model at a screenshot and ask what's on screen. Identify errors, describe UI layouts, extract data from application windows.

Best models: Qwen3-VL 8B, Phi-4-reasoning-vision 15B (strongest at GUI grounding)

```
Prompt: "What error is shown in this dialog? What should I click to fix it?"
```

Code screenshot to copyable text

Someone posts code as a screenshot on Twitter or in a presentation. Feed it to a vision model and get actual text back.

```
Prompt: "Convert the code in this screenshot to text. Preserve indentation."
```

This works well for clean screenshots. Messy or low-resolution images drop accuracy. Qwen2.5-VL and Qwen3-VL are the most reliable here thanks to their OCR training.

Chart and diagram reading

Feed a bar chart, line graph, or architecture diagram and ask the model to interpret it.

Best models: Qwen3-VL 8B (ChartQA 88+), Qwen2.5-VL 7B (ChartQA 87.3)

```
Prompt: "What trend does this chart show? What are the highest and lowest values?"
```

Math problems from photographs

Photograph a textbook page, a whiteboard, or a problem set. The model reads the problem and works through it.

Best models: Qwen3-VL 8B (MathVista 85.8), Phi-4-reasoning-vision 15B (MathVista 75.2, better on science diagrams)

```
Prompt: "Solve the math problem shown in this image. Show your work."
```

This is where the generational improvement matters. Qwen2.5-VL scored 68.2 on MathVista. Qwen3-VL scores 85.8. That's the difference between "sometimes gets it right" and "usually gets it right."

Design-to-code

Convert a mockup, wireframe, or screenshot into HTML/CSS/React components.

```
Prompt: "Convert this UI screenshot to a React component with Tailwind CSS."
```

This works for simple layouts. Complex designs with animations, gradients, and precise spacing still need human refinement. The model gets the structure right maybe 70-80% of the time, which saves real typing but isn't something you'd ship directly.

How local vision compares to cloud APIs

How local stacks up against GPT-4V, Claude Vision, and Gemini:

Capability	Local (Qwen3-VL 8B)	Cloud (GPT-4V / Claude)
Photo description	Good	Better — more nuance, fewer errors
Document OCR	Excellent (95+ DocVQA)	Excellent — roughly equivalent
Chart reading	Good (88+ ChartQA)	Better — handles complex multi-chart layouts
Math from images	Good (85.8 MathVista)	Better — GPT-4V scores 90+
Multi-image reasoning	Basic	Much better — handles 10+ images with context
Speed (first token)	2-5 seconds	1-3 seconds
Speed (generation)	80-140 t/s on good GPU	Varies, often faster
Privacy	Complete — nothing leaves your machine	Data goes to cloud provider

Capability	Local (Qwen3-VL 8B)	Cloud (GPT-4V / Claude)
Cost	Free after hardware	$0.01-0.04 per image
Availability	Always, no internet needed	Requires API access and connectivity

Local matches cloud on document OCR and text extraction. A good local model reads receipts and invoices about as well as GPT-4V. If documents are your main use case, local wins on privacy and cost.

Local falls short on complex reasoning across multiple images, subtle visual humor, cluttered or low-quality images, and edge cases. Cloud models have been trained on vastly more visual data.

The practical split: use local for document processing, screenshot analysis, and anything involving sensitive images. Use cloud for one-off complex queries where you need maximum accuracy and don't mind the cost.

Benchmarks compared

Full comparison across all major local vision models:

Model	MMMU (General)	DocVQA (Documents)	ChartQA (Charts)	MathVista (Math)	Size
Qwen3-VL 8B	—	95+	88+	85.8	8B
Qwen2.5-VL 72B	70.2	96.4	—	74.8	72B
Gemma 3 27B	64.9	86.6	78.0	—	27B
Llama 3.2 90B	60.3	90.1	85.5	—	90B
Gemma 3 12B	59.6	87.1	75.7	—	12B
Qwen2.5-VL 7B	58.6	95.7	87.3	68.2	7B
Phi-4-reasoning-vision	54.3	—	83.3	75.2	15B
Pixtral 12B	52.5	90.7	81.8	58.0	12B
Llama 3.2 11B	50.7	88.4	—	51.5	11B
Gemma 3 4B	48.8	75.8	68.8	—	4B

What jumps out:

- Qwen3-VL 8B's MathVista score (85.8) beats everything on this table except cloud models. An 8B model doing math from photos better than a 72B model.
- Qwen2.5-VL 7B's document score (95.7) is still higher than Llama 3.2 90B (90.1). A 7B model reading documents better than a 90B model.
- Phi-4 occupies a weird niche: strong on math (75.2) and science diagrams (84.8 AI2D) but below average on general understanding (54.3 MMMU).
- For OCR specifically, Qwen wins at every tier.

The bottom line

Most people (8 GB+ VRAM): `ollama run qwen3-vl:8b`. Best all-around. If Qwen3-VL isn't available in your setup yet, `ollama run qwen2.5vl:7b` is still excellent.

Tight VRAM (4-6 GB): Gemma 3 4B at int4 (2.6 GB) gives you real vision capability. SmolVLM2 2.2B (~2 GB) if you're on a Raspberry Pi or old phone.

Documents and OCR: PaddleOCR-VL 0.9B runs on CPU and scores 92.6 on document benchmarks. Use it alongside a general VLM — PaddleOCR for extraction, Qwen for understanding.

Math and science diagrams: Phi-4-reasoning-vision 15B if you have 10+ GB VRAM. Qwen3-VL 8B if you don't.

Maximum quality (16 GB+): Gemma 3 27B QAT at 14 GB for general understanding. Qwen2.5-VL 32B at 21 GB if you have 24 GB VRAM and want the best document accuracy.

A year ago, LLaVA was the only option. Now there are five competitive model families in Ollama, specialized OCR models that run on CPU, and math reasoning from photos that actually works. If you're already running text models locally and haven't tried vision yet, `ollama run qwen3-vl:8b` followed by `--image photo.jpg` is all it takes.

Related guides

- VRAM Requirements for Local LLMs — full VRAM tables for every model
- Qwen 3.5 Local Guide — the broader Qwen 3.5 ecosystem

- PaddleOCR-VL Setup — dedicated document OCR guide
- Qwen2.5-VL in LM Studio — GUI setup for vision models
- Running LLMs on Mac — Apple Silicon specifics
- llama.cpp vs Ollama vs vLLM — which inference engine to use
- Quantization Explained — understanding Q4, Q8, FP16
- Stable Diffusion Locally — if you want to generate images instead

Get notified when we publish new guides.

Subscribe — free, no spam
