PaddleOCR-VL: A 0.9B OCR Model That Runs on Any Potato

February 20, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** PaddleOCR-VL is a 0.9B vision-language model purpose-built for document OCR. Handles text, tables, formulas, charts, and seals across 109 languages. Beats Qwen2.5-VL-72B on OmniDocBench at 1/80th the size. Just merged into llama.cpp (b8110, Feb 19 2026). Q4_K_M quant is ~300MB for the language model. Runs on basically anything. Private document processing without a cloud API.

📚 **More on this topic:** Vision Models Locally · Local RAG Guide · Best Models Under 3B · Running AI Offline · Planning Tool

PaddleOCR-VL just got merged into llama.cpp as of build b8110 (February 19, 2026). It's a 0.9B parameter vision-language model from Baidu's PaddlePaddle team that does one thing: read documents. Text, tables, formulas, charts, seals — across 109 languages.

And it does it better than models 80 times its size.

What It Does

PaddleOCR-VL is not a general-purpose vision model. It's an OCR specialist. You give it an image of a document, and it extracts:

- **Text** — printed and handwritten, 109 languages including CJK, Arabic, Hindi, Cyrillic, Latin scripts
- **Tables** — structured extraction with cell boundaries and reading order
- **Formulas** — mathematical expressions converted to LaTeX
- **Charts** — bar charts, pie charts, line graphs parsed into structured data
- **Seals/stamps** — official document stamps recognized with SOTA accuracy (v1.5)

The v1.5 release (January 2026) added text spotting with bounding box localization, seal recognition, cross-page table merging, checkbox detection, and improved handling of skewed or photographed documents.

On OmniDocBench v1.5 — the standard benchmark for document understanding:

Model	Parameters	Score
PaddleOCR-VL v1.5	**0.9B**	**94.5%**
PaddleOCR-VL v1.0	0.9B	92.6%
Gemini 2.5 Pro	—	88.0%
Qwen2.5-VL-72B	72B	87.0%
GPT-4o	—	75.0%

A 0.9B model outperforming a 72B model and GPT-4o on document OCR. That's not a typo. Purpose-built beats general-purpose when the task is specific enough.

Why 0.9B Matters

Let's put this size in perspective.

Qwen2.5-VL-72B can also do document OCR. It needs 48GB+ VRAM at Q4. PaddleOCR-VL does it better in under 1GB.

The architecture is minimal: a NaViT-style dynamic resolution vision encoder, a 2-layer MLP connector, and ERNIE-4.5-0.3B as the language model. The vision encoder handles variable image sizes without distortion. The language model is 300 million parameters — smaller than most phone keyboards' prediction models.

At Q4_K_M quantization, the language model GGUF is roughly **300 MB**. The vision projector is larger (~880 MB at BF16, quantizable further). Total: you're looking at about 1-1.5 GB to run the complete model.

That means it runs on:

- An old laptop with 4GB RAM
- A $50 used ThinkPad from eBay
- A Raspberry Pi 4 with 4GB (slowly, but it fits)
- Literally any machine you have sitting in a closet

This is potato-tier hardware doing production-quality OCR.

How to Run It

llama.cpp (Recommended)

You need build b8110 or later (released February 20, 2026). GGUFs are available from the community.

```
# Download GGUF files (v1.5)
# From: huggingface.co/octopusmegalopod/some-paddleocr1.5-vl-ggufs

# Run the server
llama-server -m PaddleOCR-VL-1.5.gguf \
  --mmproj mmproj-PaddleOCR-VL-1.5.gguf \
  -c 20000 --fit off \
  --jinja --chat-template-file chat_template.jinja
```

Critical: You must include `--jinja` and `--chat-template-file chat_template.jinja` or the model will crash. The template file is included in the GGUF repo — download it alongside the model files. This handles the image placeholder tokens that PaddleOCR-VL expects.

Once the server is running, send images via the API:

```
curl http://localhost:8080/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "PaddleOCR-VL",
    "messages": [
      {"role": "user", "content": [
        {"type": "image_url", "image_url": {"url": "data:image/png;base64,..."} },
        {"type": "text", "text": "Extract all text from this document."}
      ]}
    ]
  }'
```

Ollama

A community upload exists:

```
ollama pull MedAIBase/PaddleOCR-VL:0.9b
```

Note: this is v1.0, not v1.5. Official Ollama library support for v1.5 is still pending. The community model works for basic text OCR.

vLLM

For GPU-accelerated batch processing:

```
vllm serve PaddlePaddle/PaddleOCR-VL-1.5
```

vLLM support has been available since November 2025. For bulk document processing on GPU hardware, this is the fastest path — the paper reports 1.2 pages/second on an A100 processing batches of 512 pages.

Real Use Cases

Local RAG Pipeline

This fills the biggest gap in most local RAG setups. Scanned PDFs, photographed documents, image-based forms — these are opaque to text-based RAG. PaddleOCR-VL extracts the text so your LLM can search and answer questions about it.

Pipeline: scan/photograph document → PaddleOCR-VL extracts text → chunk and embed → store in vector DB → query with your local LLM.

Privacy-First Document Processing

Lawyers, accountants, medical offices, small businesses — anyone handling sensitive paperwork. Client contracts, tax documents, patient records, financial statements. All processed locally. Nothing leaves your machine. No cloud API key. No per-page pricing. Zero ongoing cost.

Compare: Google Cloud Vision charges $1.50 per 1,000 pages. AWS Textract charges $1.50-$15 per 1,000 pages depending on features. PaddleOCR-VL charges nothing, forever.

Multilingual Documents

109 languages (111 in v1.5). If you handle documents in Chinese, Japanese, Korean, Arabic, Hindi, Russian, or any combination — this is one of the few models that handles mixed-language documents well at any size. The v1.5 seal recognition is specifically designed for East Asian official documents.

Digitizing Handwritten Notes

The v1.5 model scores 0.895 on handwritten Chinese and 0.916 on handwritten English in text spotting benchmarks. Not perfect, but significantly better than Tesseract (which barely handles handwriting at all).

What It Can't Do (Honestly)

Complex Multi-Element Pages Need the Layout Model

PaddleOCR-VL has a two-stage pipeline in its full form. Stage 1 is **PP-DocLayoutV2** — a separate 15.6M parameter layout analysis model that detects 23 types of document elements and predicts reading order. Stage 2 is PaddleOCR-VL-0.9B doing the actual recognition.

The llama.cpp integration is Stage 2 only. For simple documents — a page of text, a single table, an invoice — the VLM alone works great. For complex pages with mixed tables, formulas, multi-column text, and images interleaved, you get better results with the full pipeline.

The catch: PP-DocLayoutV2 uses the PaddlePaddle framework (not PyTorch, not GGUF). Running the full pipeline still requires the PaddlePaddle Python library. If you only need the llama.cpp/GGUF path, you're limited to Stage 2.

For most practical OCR tasks — receipts, letters, forms, simple tables, scanned text pages — Stage 2 alone is more than enough.

Not a General Vision Model

Don't ask it to describe what's in a photo or answer questions about a scene. It's trained for documents. Give it a selfie and you'll get garbage. Use Qwen2.5-VL or LLaVA for general vision tasks.

Fresh in llama.cpp — Expect Edges

The merge is two days old. The chat template crash without `--jinja` is the first rough edge. There will be others. If you hit issues, check the llama.cpp issue tracker and update to the latest build.

PaddleOCR-VL vs the Alternatives

Tool	Type	Languages	Tables	Formulas	Cost	Hardware
PaddleOCR-VL	VLM (0.9B)	109	Yes	Yes	Free	~1.5 GB RAM
Tesseract	Traditional OCR	100+	No	No	Free	Minimal
EasyOCR	DL-based OCR	80+	No	No	Free	GPU helps
Qwen2.5-VL-72B	General VLM	Many	Yes	Yes	Free	48GB+ VRAM
Google Cloud Vision	Cloud API	200+	Yes	No	$1.50/1K pages	None (cloud)
AWS Textract	Cloud API	English+	Yes	No	$1.50-15/1K pages	None (cloud)

Tesseract is the old standard — it works for clean printed text but falls apart on tables, formulas, handwriting, or anything beyond basic text extraction. EasyOCR is better but still limited to text recognition.

PaddleOCR-VL is a generational leap: it understands document structure, not just individual characters. It reads tables as tables, formulas as formulas, and maintains reading order across columns. Tesseract can't do any of that.

The only tools that match PaddleOCR-VL's capabilities are 72B+ VLMs (which need 50x the hardware) or paid cloud APIs (which need your documents to leave your network).

Bottom Line

A 0.9B model doing SOTA document OCR on hardware that costs less than a month of cloud API fees. 109 languages. Text, tables, formulas, charts, seals. Private, local, free.

If you process documents for work — or if you want to add document understanding to your local RAG pipeline — PaddleOCR-VL is the most capable OCR model you can run on budget hardware. The llama.cpp integration means you don't need the PaddlePaddle framework for basic use. Download the GGUF, point llama-server at it, and start scanning.

The model that proves you don't need a datacenter for document AI. You barely need a computer.

Source: https://insiderllm.com/guides/paddleocr-vl-local-document-ocr/

Free guides for running AI locally