CodeLlama vs DeepSeek Coder vs Qwen Coder: Best Local Coding Models Compared

February 11, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Qwen2.5-Coder wins this comparison at every VRAM tier. The 7B scores 88.4% on HumanEval, beating CodeLlama 70B (67.8%) on a fraction of the hardware. The 32B hits 92.7% and fits on a single 24GB GPU at Q4. Codestral 25.01 is the best option for pure autocomplete/FIM. CodeLlama is effectively obsolete for new setups.

📚 **More on this topic:** Best Coding Models · VRAM Requirements · Ollama vs LM Studio · Planning Tool

Four model families compete for best local coding LLM. Three of them are worth your time. One of them is still recommended in outdated guides and wastes your VRAM.

This article is a direct comparison. Same benchmarks, same quantizations, same hardware. By the end you'll know exactly which model to pull for your GPU and your workflow.

The Four Contenders

CodeLlama (Meta, August 2023) — The first serious open-source coding model. Comes in 7B, 13B, 34B, and 70B sizes with base, Python, and Instruct variants. It was the default recommendation for over a year. It's been lapped by everything else on this list.

DeepSeek Coder (DeepSeek, 2023-2024) — Two generations. V1 offered dense models from 1B to 33B. V2 switched to a Mixture-of-Experts architecture: the 236B model only activates 21B parameters per token, hitting 90.2% on HumanEval. The V2 Lite (16B/2.4B active) is the one most people can actually run.

Qwen2.5-Coder (Alibaba, October 2024) — The current champion. Available from 0.5B to 32B, Apache 2.0 licensed, 128K context window. The 7B beats models 3-5x its size. The 32B matches GPT-4o. This is what you should be running unless you have a specific reason not to.

Codestral (Mistral, May 2024) — Mistral's dedicated coding model. The original was 22B. The January 2025 update (Codestral 25.01) added a better tokenizer, 2x speed improvement, and

256K context. It hit #1 on the LMSys Copilot Arena for autocomplete and is the strongest FIM model you can run locally.

Benchmark Comparison

These are the two benchmarks that matter most for code generation. All scores are pass@1 (first attempt).

Model	Params	HumanEval	MBPP	Context
Qwen2.5-Coder 32B	32B	**92.7%**	—	128K
DeepSeek Coder V2	236B (21B active)	90.2%	76.2%	128K
Qwen2.5-Coder 14B	14B	~89%	—	128K
Qwen2.5-Coder 7B	7B	**88.4%**	—	128K
Codestral 25.01	22B	86.6%	80.2%	256K
Qwen2.5-Coder 3B	3B	84.1%	—	128K
DeepSeek Coder V2 Lite	16B (2.4B active)	81.1%	68.8%	128K
CodeLlama 70B Instruct	70B	67.8%	62.2%	16K
DeepSeek Coder V1 33B	33B	61.9%	70.1%	16K
CodeLlama 34B Python	34B	53.7%	56.2%	16K
CodeLlama 13B Python	13B	43.3%	49.0%	16K
CodeLlama 7B	7B	33.5%	41.4%	16K

The numbers tell a clear story. Qwen2.5-Coder 7B at 88.4% outperforms CodeLlama 70B Instruct at 67.8%. A model that fits on an 8GB GPU beats one that needs 40+ GB. That's not a marginal improvement; CodeLlama is a generation behind.

DeepSeek Coder V2 (the full 236B) hits 90.2%, but you need a multi-GPU rig to run it. The V2 Lite at 81.1% is more practical and still strong.

Codestral 25.01 at 86.6% sits between the Qwen 7B and 14B, but its real strength shows up in FIM benchmarks for autocomplete (85.9% average across languages), which the HumanEval numbers don't capture.

One caveat: HumanEval scores vary by evaluation pipeline. The numbers above use each model's official methodology. EvalPlus (HumanEval+) applies stricter test cases and typically scores 5-10% lower across the board. Relative rankings stay the same.

VRAM Requirements by Quantization

This is the table that actually determines which model you can run. File size is the model weight; runtime VRAM adds KV cache overhead (estimated at 8K context).

Model File Sizes (GGUF)

Model	Q4_K_M	Q5_K_M	Q8_0
CodeLlama 7B / Qwen-Coder 7B / DS-Coder 6.7B	~4.7 GB	~5.4 GB	~8.1 GB
CodeLlama 13B	~7.9 GB	~9.2 GB	~13.8 GB
Qwen2.5-Coder 14B	~8.7 GB	~10.2 GB	~15.2 GB
Codestral 22B	~13.3 GB	~15.7 GB	~23.6 GB
Qwen2.5-Coder 32B	~19.9 GB	~23.3 GB	~34.8 GB
CodeLlama 34B	~20 GB	~23 GB	~35 GB
CodeLlama 70B	~40 GB	~48 GB	~73 GB

Runtime VRAM (With KV Cache at 8K Context)

Model	Q4_K_M	Q5_K_M	Minimum GPU
7B class (any family)	~6.2 GB	~7.3 GB	8 GB
CodeLlama 13B	~10.7 GB	~12 GB	12 GB
Qwen2.5-Coder 14B	~11 GB	~12.5 GB	12 GB
Codestral 22B	~15-16 GB	~18 GB	16 GB (Q4) / 24 GB (Q5)
Qwen2.5-Coder 32B	~22-24 GB	~25+ GB	24 GB (Q4 only)
CodeLlama 34B	~22-24 GB	~26 GB	24 GB (Q4 only)
CodeLlama 70B	~45-50 GB	~55+ GB	2x 24 GB or 48 GB+

Rule of thumb: Pick a GGUF quant whose file size is 1-2 GB smaller than your total GPU VRAM. That leaves headroom for the KV cache. Longer context eats more VRAM linearly, so if you're pushing 32K+ context, budget accordingly.

What Fits Your GPU

8GB VRAM (RTX 3060 8GB, RTX 4060)

You're limited to 7B models. That's fine because Qwen2.5-Coder 7B at Q4_K_M is better than most 34B models from 2023.

Model	Quant	VRAM	HumanEval	Verdict
Qwen2.5-Coder 7B	Q4_K_M	~6.2 GB	88.4%	Best choice. No contest.
DeepSeek Coder V2 Lite	Q4_K_M	~6 GB	81.1%	Good alternative, MoE architecture
CodeLlama 7B	Q4_K_M	~6 GB	33.5%	Skip. Nearly 3x worse than Qwen.

```
ollama pull qwen2.5-coder:7b
```

If you want both autocomplete and chat on 8GB, run a single Qwen2.5-Coder 7B for both. Splitting into separate models eats too much memory at this tier.

12GB VRAM (RTX 3060 12GB, RTX 4070)

The sweet spot opens up. Qwen2.5-Coder 14B fits comfortably and gives you a real step up from 7B. Codestral 22B is a tight squeeze at Q4 but doable with short context.

Model	Quant	VRAM	HumanEval	Verdict
Qwen2.5-Coder 14B	Q4_K_M	~11 GB	~89%	Best at this tier
Codestral 22B	Q3_K_M	~12 GB	86.6%	Tight fit, best FIM
CodeLlama 13B	Q4_K_M	~10.7 GB	43.3%	Outclassed. Skip.

```
ollama pull qwen2.5-coder:14b
```

The Qwen 14B at ~89% HumanEval on 11 GB VRAM is hard to argue with. You get near-Codestral performance with room to spare for context. If autocomplete speed is your priority, squeeze Codestral 22B at Q3 and accept the shorter context limit.

24GB VRAM (RTX 3090, RTX 4090)

This is where local coding matches cloud. The used RTX 3090 at $700-850 gets you here.

Model	Quant	VRAM	HumanEval	Verdict
Qwen2.5-Coder 32B	Q4_K_M	~22-24 GB	92.7%	Best overall. Matches GPT-4o.
Codestral 22B	Q5_K_M	~18 GB	86.6%	Best FIM. Comfortable fit.
CodeLlama 34B	Q4_K_M	~22-24 GB	53.7%	Same VRAM, half the score. Skip.

```
ollama pull qwen2.5-coder:32b
```

The ideal 24GB setup: Run Qwen2.5-Coder 32B for chat and reasoning, keep Codestral 22B available for autocomplete. You can't run both simultaneously on one card, but swap between them depending on the task. Or just use the Qwen 32B for everything if switching is annoying.

Autocomplete vs Chat vs Agents

Not every coding model does every job. This table breaks down what each family actually supports.

Feature	CodeLlama	DeepSeek Coder	Qwen2.5-Coder	Codestral
FIM (Autocomplete)	Yes	Yes	Yes	Yes (strongest)
Chat/Instruct	Yes (Instruct variant)	Yes (Instruct variant)	Yes (Instruct variant)	Yes
Agentic Coding	No	Limited	Moderate (function calling)	Limited
Context Window	16K	16K (V1) / 128K (V2)	128K	256K

Feature	CodeLlama	DeepSeek Coder	Qwen2.5-Coder	Codestral
Languages	~15	~80+	~92+	~80+

For autocomplete/tab-complete: You want FIM (fill-in-the-middle) support. All four families have it, but Codestral 25.01 leads the pack. It hit #1 on the LMSys Copilot Arena, which ranks models on real autocomplete quality rated by developers. If autocomplete is your primary use case, Codestral is the pick.

For chat ("explain this code," "write tests"): Qwen2.5-Coder Instruct variants are the strongest. The 32B produces coherent multi-file refactors and understands complex codebases through its 128K context window.

For agentic coding (autonomous task completion): None of these four families were designed for it. If you need a local model that can browse docs, run commands, and iterate on code autonomously, look at the newer models below.

FIM Template Tokens

If you're configuring a tool like Continue or Tabby manually, you'll need the right FIM tokens:

Model	Prefix	Suffix	Middle
CodeLlama	`<PRE>`	`<SUF>`	`<MID>`
DeepSeek Coder	`<|fim_begin|>`	`<|fim_hole|>`	`<|fim_end|>`
Qwen2.5-Coder	`<|fim_prefix|>`	`<|fim_suffix|>`	`<|fim_middle|>`
Codestral	`[PREFIX]`	`[SUFFIX]`	`[MIDDLE]`

Most tools handle this automatically if you select the right model. But if autocomplete isn't working, wrong FIM tokens are usually why.

Speed on Real Hardware

Benchmarks don't matter if the model takes 5 seconds to start generating. These are real-world speeds on common GPUs.

Generation Speed (tokens/sec, Q4_K_M)

Model	RTX 3090 (24 GB)	RTX 4090 (24 GB)	M2/M3 Max (64 GB)
7B class	~112 t/s	~128 t/s	~20 t/s
Qwen2.5-Coder 14B	~55-65 t/s	~70-80 t/s	~15 t/s
Codestral 22B	~25-30 t/s	~30-40 t/s	~12 t/s
Qwen2.5-Coder 32B	~32 t/s	~20-25 t/s*	~22 t/s (MLX)
CodeLlama 34B	~18-22 t/s	~18-22 t/s*	—

*32B+ models on the 4090 may need partial CPU offload depending on context length, which drops speed.

For autocomplete, you want responses in under 200ms. That means you need ~50+ tokens/sec from your FIM model. On the RTX 3090 and 4090, any 7B model clears this easily. Codestral 22B at ~30 t/s is borderline — still usable but you'll notice a slight delay compared to a 7B.

For chat, speed matters less. Even 20 t/s is comfortable for reading code explanations.

The 3090 vs 4090 difference is smaller than you'd expect for LLM inference. The 4090 has more CUDA cores and faster memory bandwidth, but the bottleneck for large models is VRAM capacity, not raw compute. Both have 24GB. For coding models specifically, save the $800 and grab a used 3090.

Which Model for Which Job

Use Case	Best Model	Why
Autocomplete on 8 GB	Qwen2.5-Coder 7B	88.4% HumanEval, fast FIM, fits easily
Autocomplete on 12-24 GB	Codestral 25.01	#1 on Copilot Arena, 85.9% FIM average
Chat assistant on 24 GB	Qwen2.5-Coder 32B	92.7% HumanEval, 128K context, matches GPT-4o
Tight VRAM budget	Qwen2.5-Coder 3B	84.1% HumanEval at ~3 GB VRAM
Max quality (multi-GPU)	DeepSeek Coder V2 236B	90.2% HumanEval, but needs serious hardware

Use Case	Best Model	Why
Legacy compatibility	CodeLlama	Only if a tool explicitly requires it

The one to skip: CodeLlama. At every size, a newer model scores higher on less VRAM. CodeLlama 70B Instruct (67.8%) needs 40+ GB and gets beaten by Qwen 7B (88.4%) on 6 GB. There's no scenario where CodeLlama is the right choice for a new setup in 2026.

What About Newer Models?

A few models released after the four contenders above are worth knowing about:

Qwen3-Coder-Next (February 2026) — An 80B MoE model that only activates 3B parameters per token. It scores 70.6% on SWE-Bench Verified (a much harder benchmark than HumanEval that tests real-world repo-level coding). The sparse architecture means it runs at speeds comparable to a 3-7B dense model while producing much better output. Apache 2.0 license. This is the one to watch for agentic coding.

Devstral Small 2 (December 2025) — Mistral's 24B model designed specifically for agentic coding. 68% on SWE-Bench, Apache 2.0, 256K context. If you want a model that can work autonomously with tools, this is the current open-weight leader at its size.

DeepSeek V3.2 (December 2025) — The 671B MoE general-purpose model that scores 70.2% on SWE-Bench. Not a dedicated coding model, but strong enough at code to compete. Needs multi-GPU setups.

These newer models compete on agentic benchmarks (SWE-Bench) rather than simple code generation (HumanEval). If you're running an AI coding agent that needs to navigate repos, run tests, and iterate, they're the next tier up. For straightforward autocomplete and chat coding, Qwen2.5-Coder and Codestral remain the picks.

The Bottom Line

CodeLlama had its moment. That moment passed in late 2024. For any new local coding setup in 2026:

- **8 GB VRAM:** `ollama pull qwen2.5-coder:7b` and you're done

- **12 GB VRAM:** `ollama pull qwen2.5-coder:14b` for chat, or squeeze Codestral 22B at Q3 for autocomplete
- **24 GB VRAM:** `ollama pull qwen2.5-coder:32b` for chat, Codestral 22B at Q5 for autocomplete

Pair with Continue in VS Code for a free, private, offline Copilot replacement that outscores the original on benchmarks.

Related Guides

- Best Models for Coding Locally in 2026
- What Can You Run on 8GB VRAM?
- What Can You Run on 24GB VRAM?
- DeepSeek Models Guide
- Qwen Models Guide
- Local AI Planning Tool — VRAM Calculator

Sources: Qwen2.5-Coder Technical Report, Code Llama Paper, DeepSeek-Coder-V2, Codestral 25.01, Qwen3-Coder-Next, EvalPlus Leaderboard

---