KI-Coding-Modelle im Eigenbetrieb

Lokale Modelle auf MacBook Pro M5 Max und Windows-PC
mit Claude Haiku 4.5, Sonnet 4.6, Sonnet 5, Opus 4.8, Opus 5 und Fable 5 als Cloud-Referenz

7 Aufgaben aus dem .NET-/T-SQL-Alltag · objektiv verifiziert per Compile-, Unit- und Runtime-Tests

Kristian Költzsch · Juli 2026 - https://www.koeltzsch.com/blog/ki-modellvergleich-lokal-gegen-cloud

Ziel & Methodik

Wie gut schlägt sich ein selbst gehostetes Coding-Modell gegen die Cloud — und welche Hardware trägt es?

1

Sieben Praxisaufgaben

C#-Nebenläufigkeit, T-SQL, Code-Review, Blazor, EF Core, Regex, Legacy-Refactoring — identische Prompts für alle Modelle und Plattformen.

2

Zehn Modelle

Vier lokale (Qwen3-Coder-Next, gpt-oss-120b, Qwen 3.6 27B, Qwen2.5-Coder-32B) gegen Claude Haiku 4.5, Sonnet 4.6, Sonnet 5, Opus 4.8, Opus 5 und Fable 5.

3

Objektive Testbench

Nicht lesen, sondern messen: dotnet-Compile, xunit-Verhaltenstests, SQL-Runtime gegen Testdaten mit Fan-Out-Falle, EF-Übersetzbarkeit.

4

Zwei Umgebungen, drei Quants

MacBook Pro M5 Max (Unified Memory) und Windows-PC mit RTX 5070 Ti. Nachmessung mit bit-identischen Gewichten und je drei Läufen trennt Können von Sampling-Zufall.

Umgebung 1 · MacBook Pro M5 Max

Chip / RAM	Apple M5 Max · 128 GB Unified Memory (~546 GB/s)
OS / Runtime	macOS 26.6 · LM Studio 0.4.19+2 · MLX 1.10.1 · llama.cpp Metal 2.27.1
Modell	Qwen3-Coder-Next 80B · MLX 6-bit (65 GB) · GGUF Q6_K (66 GB) · Q8_0 (84 GB)
Konfiguration	GPU max · Kontext 65.536 · 4 Slots · Experten NICHT ausgelagert
Speicherbild	41,8 GB von 128 GB frei nach dem Laden · Endpoint Port 1234 (LAN)
Besonderheit	Alle drei Quants lauffähig — Q8_0 nur dank 128 GB Unified Memory

77

Token/s optimal konfiguriert

Ein Speicherpool für CPU und GPU: Die MoE-Experten liegen automatisch dort, wo sie gebraucht werden — ohne Transferkosten.

MLX ist Apples eigenes Tensor-Format und läuft nur auf Apple Silicon — deshalb war es die erste Wahl auf dem Mac. Für einen fairen Vergleich wurde GGUF Q6_K nachträglich auch hier gemessen, dazu Q8_0, das nur dank 128 GB Unified Memory überhaupt lädt.

Umgebung 2 · Windows-PC mit RTX 5070 Ti

Chip / RAM	RTX 5070 Ti 16 GB VRAM · 128 GB DDR4-2666 (~40 GB/s)
OS / Runtime	Windows 11 · LM Studio 0.4.19 · llama.cpp CUDA12 2.27.1
Modell	Qwen3-Coder-Next 80B · Unsloth GGUF Q6_K · 61,1 GB (bit-identisch zum Mac)
Konfiguration	GPU max · Kontext 65.536 · 1 Slot · Experten ins RAM ausgelagert
Speicherbild	7,1 GB VRAM + ~62 GB RAM (voller Offload)
Besonderheit	MTP-Speculative nicht verfügbar (GGUF ohne MTP-Head)

21

Token/s optimal konfiguriert

Die Experten passen nicht in 16 GB VRAM — sie wandern ins DDR4-RAM, dessen Bandbreite zum Flaschenhals wird.

Q6_K ist der gemeinsame Nenner beider Plattformen: Die Shards sind per SHA256 bit-identisch zu denen auf dem Mac. Identische Gewichte, identische Prompts, identischer Kontext — der einzige verbleibende Unterschied ist die Hardware und damit das Tempo.

Optimale Konfiguration · MacBook Pro

M5 Max · 128 GB Unified Memory · Qwen3-Coder-Next 80B

Konfiguration	Speicher frei	tok/s	Bewertung
Gespeicherte Konfiguration, unverändert	—	75,6	*bereits nahe am Optimum*
Experten ins RAM erzwungen (Windows-Optimum!)	3,3 GB	10,4	*unbrauchbar — Faktor 7 langsamer*
Experten nicht erzwungen, voller Offload	42,9 GB	75,4	*sehr gut, stabil*
Teil-Offload (gpu 0.75)	28,2 GB	59,3	*rund 21 % Verlust*
Voller Offload, Kontext 65.536, 4 Slots	**41,8 GB**	**76,8**	**OPTIMAL**
MLX 6-bit unter gleichen Bedingungen	—	77,1	*gleichauf, aber unruhiger*

Empfohlene Einstellung

lms load qwen3-coder-next --gpu max -c 65536 --parallel 4

Experten NICHT auf die CPU zwingen. Kontext 65.536 und vier Slots kosten nichts — also mitnehmen.

Die Windows-Empfehlung kehrt sich um

Modell und KV-Cache liegen ohnehin im selben Unified Memory. Eine zusätzliche CPU-Kopie der Experten gewinnt nichts, senkt den freien Speicher von 42,9 auf 3,3 GB und treibt das System in die Auslagerung — als einzige Variante nicht reproduzierbar (98,9 s gegen 139,3 s bei gleicher Tokenzahl).

Gemessen am 30.07.2026 · macOS 26.6 · LM Studio 0.4.19+2 · llama.cpp Metal 2.27.1 / MLX 1.10.1 · Netzbetrieb · identischer Prompt, max_tokens 1200, je zwei Läufe.

Optimale Konfiguration · Windows-PC

Fünf Varianten gemessen — der Hebel liegt nicht dort, wo man ihn vermutet

Konfiguration	VRAM	tok/s	Bewertung
Automatische Layer-Verteilung	—	4,2	*unbrauchbar*
Experten im RAM, Teil-Offload (gpu 0.75)	5,9 GB	13,5–14,4	*ein Drittel verschenkt*
Experten im RAM, voller Offload (gpu max)	**7,1 GB**	**20,1–22,0**	**OPTIMAL**
Zusätzlich 15 % Experten auf die GPU	14,6 GB	21,1–22,6	*kein Gewinn, OOM-Risiko*
Voller Offload mit 4 parallelen Slots	7,1 GB	19,3	*leicht langsamer*
Engine 2.25.2 statt 2.27.1	7,1 GB	20,1–22,0	*kein Unterschied*

Empfohlene Einstellung

Experten ins RAM
(numCpuExpertLayersRatio 1), alles Übrige vollständig auf die GPU (gpu max), Kontext 65.536, ein Slot.

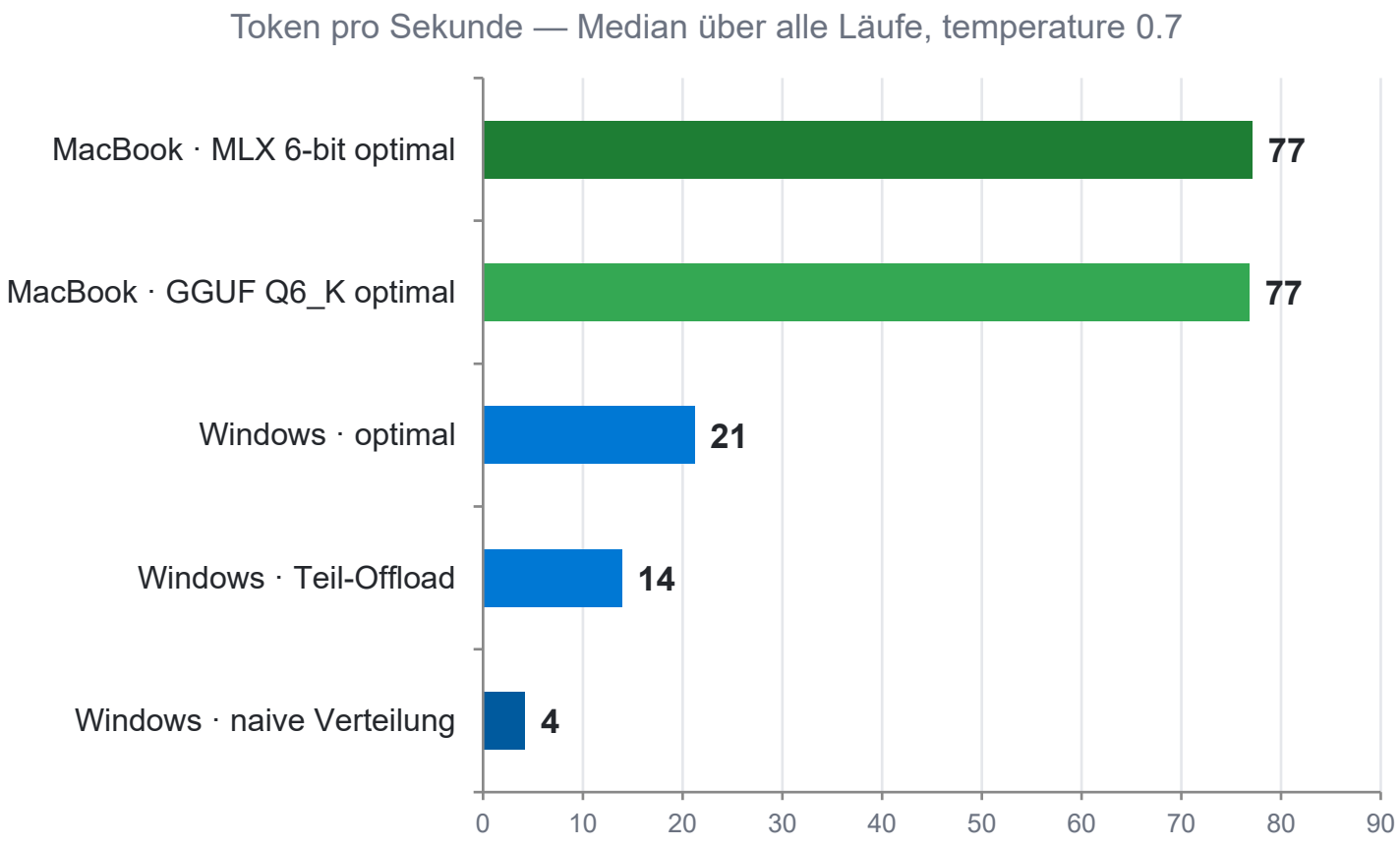
LM Studios Automatik lässt hier ein Drittel Leistung liegen.

Mehr Grafikspeicher hilft nicht

Doppelte VRAM-Belegung brachte 4 % — innerhalb der Streuung. Die Expertenauswahl wechselt je Token, 85 % müssen weiter aus dem RAM kommen. Von 16 GB gegen ~55 GB Expertengewichte ist kein Hebel zu erwarten.

Alle Werte am 28.07.2026 auf derselben Maschine gemessen: identischer Prompt, max_tokens fest auf 1200, je zwei Läufe.

Performance: Speicherbandbreite entscheidet



3,6×

schneller auf dem M5 Max

546 GB/s Unified Memory gegen **~40 GB/s** DDR4. Beide Erstmessungen waren zu niedrig, weil suboptimal konfiguriert: Windows lief mit unvollständigem Offload (13,9 statt 21,2), der Mac kam mit 37,2 statt 76,8 nur auf die Hälfte. Erst nach getrennter Optimierung beider Seiten ist der Faktor belastbar.

A AsyncLazyCache — C#-Nebenläufigkeit

Thread-sicherer Async-Cache mit TTL und Stampede-Schutz: Bei 25 parallelen Anfragen auf denselben Key darf die Factory genau einmal laufen. Verifiziert per xunit-Verhaltenstest.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

gescheitert

Compilefehler in allen drei Läufen

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

teilweise

1 von 6 Läufen bestanden; Compilefehler und ein Deadlock

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

gescheitert

Kompiliert, aber die Factory lief 25/25-mal — Stampede-Schutz wirkungslos

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: Deadlock · Sonnet 4.6: 2/2 · Sonnet 5: 2/2 · Opus 4.8: 2/2 · Opus 5: 2/2 · Fable 5: 2/2

B sp_GetGroupBalances — T-SQL-Aggregation

Stored Procedure für Salden je Gruppenmitglied. Testdaten mit konstruierter Fan-Out-Falle (2 Zahlungen × 2 Anteile) — einfache Fälle decken den Fehler nicht auf. Verifiziert gegen SQL Server mit Soll-Salden.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

teilweise

Nur 1 von 3 Läufen korrekt; zwei mit Fan-Out: Anna 300 statt 150 EUR

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

teilweise

2 von 6; sonst Fan-Out (Anna 300 statt 150) oder Syntaxfehler

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

gescheitert

Fan-Out und falsche Spalte: Anna 130 | 130 | 0 statt 150 | 65 | 85

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: FAIL (Fan-Out) · Sonnet 4.6: PASS · Sonnet 5: PASS · Opus 4.8: PASS · Opus 5: PASS · Fable 5: PASS

C Code-Review — Analysequalität

C#-Snippet mit vier platzierten Bugs (async void, HttpClient in Schleife, Mehrfach-Enumeration, kulturabhängiger Dateiname). Gemessen als Trefferquote gegen die bekannte Bug-Liste.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

bestanden

Alle 3 Läufe mit 4/4 erkannten Bugs, dazu Zusatzfunde

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

bestanden

6 von 6 Läufen mit 4/4 erkannten Bugs

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

bestanden

4/4 Bugs gefunden, dazu Zusatzfunde

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: 4/4 · Sonnet 4.6: 4/4 · Sonnet 5: 4/4 · Opus 4.8: 4/4 · Opus 5: 4/4 · Fable 5: 4/4

D EditSettingDialog — Blazor Server

Modal-Dialog mit EditForm, Validierung, EventCallbacks, Fehlerbehandlung und Spinner. Verifiziert per Compile-Check in einer Razor Class Library mit Standard-Imports.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

gescheitert

Alle 3 Läufe mit Razor- und C#-Fehlern

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

gescheitert

0 von 6; fehlerhaftes bind-value, fehlende using-Direktiven

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

gescheitert

Halluzinierte APIs Modal und FormContext — CS0246

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: OK · Sonnet 4.6: OK · Sonnet 5: OK · Opus 4.8: OK · Opus 5: OK · Fable 5: OK

E GetTopGroupsAsync — EF Core 10

Top-N-Gruppen nach Ausgabensumme in exakt einem SQL-Roundtrip, vollständig serverseitig übersetzbar. Verifiziert zur Laufzeit gegen SQL Server — die Aufgabe, die auch die Cloud-Spitze gerissen hat.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

gescheitert

Alle 3 Läufe: Übersetzungsfehler, decimal?-Mismatch

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

teilweise

2 von 6 Läufen bestanden; sonst Übersetzungs- und Compilefehler

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

gescheitert

GroupBy-Umweg und Konstruktor-Projektion — nicht übersetzbar

Die auffälligste Einzelabweichung: Mac 0 von 3, Windows 2 von 3 — bei bit-identischen Gewichten. Bei nur drei Läufen je Seite ist das Zufall, keine Plattformeigenschaft. Erst die Summe über alle 21 Läufe trägt eine Aussage: 5 von 21 gegen 7 von 21.

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: PASS · Sonnet 4.6: FAIL · Sonnet 5: PASS nach Korrektur · Opus 4.8: PASS · Opus 5: PASS · Fable 5: FAIL

F GermanAmountParser — Regex

Deutsche Geldbeträge aus Freitext extrahieren, inkl. Negativfällen. [GeneratedRegex] mit ReDoS-Timeout. Verifiziert per xunit mit 7 Parse-Fällen.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

gescheitert

Alle 3 Läufe mit Compilefehlern

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

gescheitert

0 von 6; durchgehend SYSLIB1040 — Attribut nicht wohlgeformt

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

gescheitert

Erfundener Parameter matchTimeout — dieselbe Halluzination

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: 4/7 · Sonnet 4.6: 6/7 · Sonnet 5: 7/7 nach Korrektur · Opus 4.8: 7/7, aber ohne den geforderten ReDoS-Timeout → Note 7 · Opus 5: 7/7 · Fable 5: 7/7

G BewerbungManager — Legacy-Refactoring

Methode mit SQL-Injection, Reader/Update-Verschränkung (MARS-Falle), Ressourcenlecks und vermischten Verantwortlichkeiten sanieren. Verifiziert per Compile-Check mit SqlClient.

Mac · GGUF Q6_K

3 Läufe · identische Gewichte

teilweise

Nur 1 von 3 Läufen korrekt; Tippfehler im DTO-Namen, Interface-Mismatch

Windows · GGUF Q6_K

6 Läufe · identische Gewichte

teilweise

1 von 6; erfundene APIs, fehlende using-Direktiven

Mac · MLX 6-bit

Einzellauf · nur auf Apple Silicon

gescheitert

IDbConnection.Clone() existiert nicht; Interface referenziert, nie definiert

Grün = Mac, Blau = Windows (nur Plattform-Kennung). Ergebnis-Marke: dunkel = bestanden, grau = teilweise, rot = gescheitert. Mac 3 Läufe, Windows 6 (zwei Engine-Versionen), MLX Einzellauf.

Cloud-Referenz: Haiku 4.5: OK · Sonnet 4.6: OK (6 Nachlieferungen) · Sonnet 5: OK · Opus 4.8: OK · Opus 5: OK · Fable 5: OK

Die drei Offline-Varianten

Dasselbe Modell in drei Auslieferungsformen — alle Tempowerte auf dem MacBook gemessen, sonst wären sie nicht vergleichbar

MLX 6-bit

Apples eigenes Tensor-Format

65 GB · Mac 77,1 tok/s
läuft nur auf Apple Silicon

Die zuerst eingesetzte Variante. Apples eigenes Tensor-Format — auf Windows schlicht nicht lauffähig. Genau deshalb war der Erstvergleich verzerrt: Mac und PC liefen mit verschiedenen Quantisierungen.

GGUF Q6_K

der gemeinsame Nenner

66 GB · Mac 76,8 · Win 21,2 tok/s
läuft auf Mac und Windows

Der gemeinsame Nenner beider Plattformen. Die Shards auf dem Mac sind per SHA256 bit-identisch zum Windows-Anker — damit ist jede Qualitätsdifferenz zwischen den Plattformen ausgeschlossen.

GGUF Q8_0

höchstes getestetes Quant

84 GB · Mac 34,6* tok/s
läuft nur auf dem Mac (128 GB)

Nur dank 128 GB Unified Memory überhaupt lauffähig — auf 16 GB VRAM undenkbar. Sollte die These „mehr Bits = mehr Qualität" belegen. Tat es nicht.

MLX und GGUF liegen gleichauf und liefern dieselbe Qualität.

Dasselbe Q6_K erreicht auf dem Windows-PC 21,2 tok/s — der Unterschied kommt von der Hardware, nicht vom Format. Die MoE-Architektur aktiviert nur ~2,5 Mrd. Parameter je Token, deshalb ändert auch die Bit-Tiefe wenig: Die Fehler sind in allen drei Varianten dieselben. *Q8_0 wurde nur in der Erstmessung erfasst, nicht unter optimaler Konfiguration.

Nachmessung: 3 Läufe statt Einzelschuss

84 Läufe insgesamt — auf Windows zusätzlich mit zwei Engine-Versionen gegengeprüft

Aufgabe	Typ	Mac Q6_K	Mac Q8_0	Win Q6_K	Befund
A · AsyncLazyCache	Code-Gen	0/3	0/3	*1/6*	Compilefehler, einmal Deadlock
B · SQL Fan-Out-SP	Code-Gen	1/3	0/3	*2/6*	Fan-Out: Anna 300 statt 150 EUR
C · Code-Review	8,5	8,5	8,5	*9*	9
D · Blazor-Dialog	3,5	2	7	*9*	9
E · EF-Core-Query	2	2	9	*5*	7
F · Regex-Parser	3,5	3	2,5	*6,5*	7
G · Refactoring	6	5	7	*8,5*	9
GESAMT		**5/21**	**4/21**	**12/42**	**statistisch gleichauf — Rauschen**

Zwei Antworten

Plattform?
Kein Qualitätsunterschied. Bit-identische Gewichte, nur das Tempo trennt Mac und PC.

Mehr Bits?
Kein Gewinn. Q8_0 mit 84 GB liegt gleichauf mit Q6_K — der größere Footprint lohnt nicht.

Zum Vergleich die Cloud-Modelle: Haiku 4.5 Ø 5,5 · Sonnet 4.6 Ø 8,0 · Sonnet 5 Ø 8,4 · Opus 4.8 Ø 9,0 · Opus 5 Ø 9,0 · Fable 5 Ø 8,4

Das Muster ist reproduzierbar: Code-Review besteht ausnahmslos jeder Lauf, Generierung selten mehr als ein Drittel. Als Review-Werkzeug lokal brauchbar — als Code-Generator nicht produktionsreif.

Ein zweites lokales Modell: gpt-oss-120b

Nicht die Größe limitiert, sondern die aktiven Parameter je Token

Aufgabe	Qwen3-Coder-Next	gpt-oss-120b	Befund
A · Cache/Concurrency	4	9	*2*
B · T-SQL Fan-Out	1 von 3	3 von 3	*Falle zuverlässig erkannt*
C · Code-Review	8,5	8,5	*8,5*
D · Blazor	0 von 3	0 von 3	*Syntaxfehler, unvollständiger Code*
E · EF Core	0 von 3	0 von 3	*Query nicht übersetzbar*
F · Regex	0 von 3	0 von 3	*Attribut nicht wohlgeformt*
G · Refactoring	6	5	*7*
GESAMT	**5 von 21**	**10 von 21**	**Trefferquote verdoppelt**

Der Hebel sind die aktiven Parameter

Qwen aktiviert 2,5 von 80 Mrd. Parametern je Token, gpt-oss rund 5 von 117 Mrd. — doppelte Rechenleistung pro Token bei kleinerem Speicherbedarf (63,4 statt 65,6 GB).

Was auch das zweite Modell nicht löst

D, E und F bleiben bei 0 von 3 — dieselben Aufgaben, an denen auch Cloud-Modelle straucheln: Die EF-Falle riss Sonnet 4.6, Fable 5 und Sonnet 5 im Erstentwurf.

Gesamtmatrix

Bewertung 0–10, korrigiert durch die objektive Testbench

Aufgabe	Qwen lokal	gpt-oss	Haiku 4.5	Sonnet 4.6	Sonnet 5	Opus 4.8	Opus 5	Fable 5
A · Cache/Concurrency	4	9	2	9	9	9	9	9
B · T-SQL-Salden	2	9	2,5	9	9	9	9	9
C · Code-Review	8,5	8,5	8,5	9	9	9	9	9
D · Blazor-Dialog	3,5	2	7	9	9	9	9	9
E · EF-Core-Query	2	2	9	5	7	9	9	5
F · Regex-Parser	3,5	3	2,5	6,5	7	7	9	9
G · Refactoring	6	5	7	8,5	9	9	9	9
Ø	**4,2**	**5,5**	**5,5**	**8,0**	**8,4**	**8,7**	**9,0**	**8,4**

Das lokale Modell steht bewusst in EINER Spalte: Die Nachmessung mit bit-identischen Gewichten (SHA256) belegt, dass die Plattform nur das Tempo beeinflusst, nicht die Qualität. Amber: Opus 5 als jüngste Referenzspitze.

Zwei Ligen

Opus 5 (9,0)
löst als einziges Modell alle sieben Aufgaben vollständig.

Das lokale Modell
erreicht mit gpt-oss-120b erstmals Haiku-Niveau (5,5) — bei voller Datenhoheit, ohne API-Kosten.

Vier Erkenntnisse

Analyse ist demokratisiert

Beim Code-Review sind alle Modelle gleichauf. Reviews, Erklärungen und Bug-Suche laufen bedenkenlos lokal.

Halluzinationen sind quant-stabil

<Modal> und die erfundene GeneratedRegex-Signatur erscheinen in MLX und GGUF — Modellwissen, kein Quantisierungsartefakt.

Kommentare sind nicht Verhalten

Opus 4.8 kommentiert einen ReDoS-Schutz, der im Code fehlt — und bestand trotzdem alle sieben Tests. Eine Testbench prüft nur, wonach sie fragt.

Unified Memory gewinnt bei MoE

Faktor 3,6 im Tempo und drei lauffähige Quants statt einem: 546 GB/s Unified Memory schlagen PCIe + DDR4 deutlich.

Was der Tempovorteil im Alltag bedeutet

Wartezeit bis zur fertigen Antwort — gerechnet mit den Optimalwerten: 76,8 (Mac) · 21,2 (Windows) · 70 tok/s (Cloud)

Typische Anfrage	MacBook	Windows-PC	Claude (Cloud)
Kurze Code-Antwort (~1.500 Token)	**20 s**	**1:11 min**	**21 s**
Code-Review einer Klasse (~2.500 Token)	**33 s**	**1:58 min**	**36 s**
Fünf agentische Runden hintereinander	**1:38 min**	**5:54 min**	**1:47 min**
Kompletter Testlauf (21 Anfragen)	**7 min**	**25 min**	**8 min**

Der Unterschied wächst mit jeder Iteration. Cloud-Spalte: Sonnet 4.6 mit ~70 tok/s aus dem API-Lauf; für Opus 4.8, Sonnet 5 und Fable 5 wurde kein Tempo gemessen, da diese Läufe im Chat stattfanden.

Läuft ohne Tuning

Auf dem PC brachte erst die manuelle MoE-Verteilung den Faktor 3 (4,2 → 13,9 tok/s). Der Mac lädt das Modell und liefert sofort volle Geschwindigkeit.

Ein Gerät, überall dabei

Kein zweiter Rechner muss laufen. Das Modell arbeitet im Akkubetrieb und ohne Netz — Datenhoheit auch unterwegs.

Spielraum nach oben

128 GB tragen drei Quantisierungen bis 84 GB. Auf 16 GB VRAM ist bei Q6_K Schluss — Experimente fallen aus.

Beim Tempo ist der Mac vorn — bei der Qualität die Cloud. Optimal konfiguriert liefert der M5 Max 76,8 tok/s und überholt damit die gemessene Cloud-Referenz. Was bleibt, ist der Qualitätsabstand: 12 von 42 gegen fehlerfreie Durchläufe. Lokal für Reviews und sensible Daten, Cloud für alles, was auf Anhieb stimmen muss.

Fazit & Systembewertung

MacBook Pro M5 Max

voll praxistauglich

Empfehlung für den produktiven lokalen Einsatz

- 76,8 tok/s — die Ausgabe liest sich flüssig mit
- 128 GB Unified Memory tragen das 61-GB-Modell mühelos
- Einrichtung unkompliziert (MLX nativ)

Windows-PC RTX 5070 Ti

eingeschränkt tauglich

Geeignet für Reviews und kurze Snippets

- 21 tok/s bei vollem GPU-Offload (Automatik ließ ein Drittel liegen)
- DDR4-Bandbreite ist der Flaschenhals, nicht die GPU
- Besser: kleineres Quant im VRAM oder DDR5-Plattform

Cloud-Referenz (Opus 5)

Spitzenreiter

Qualitätsmaßstab, keine lokale Hardware

- Einziges Modell mit 7/7 vollständig erfüllt (Ø 9,0)
- Löst die EF-Falle, an der Sonnet 4.6 & Fable scheitern
- Kosten & Datenabfluss statt lokaler Hoheit

Lokal für Reviews und sensible Daten — mit einem MoE-Modell wie gpt-oss-120b inzwischen auch für Nebenläufigkeit und SQL-Aggregation. Cloud bleibt nötig für Framework-Randbereiche. Generierten Code immer durch Compile und Test schicken.