

Let's go Offline

Die Welt der lokalen LLMs

Intro

Die Story

Persönlicher Einstieg

Ich wollte ChatGPT lokal und bin im Kaninchenbau gelandet"

- Datenschutz am eigenen Rechner
- Keine Cloud-Abhängigkeit
- Volle Kontrolle über meine Daten

Was ist ein Local LLM?

Definition

- **Large Language Model** auf eigener Hardware
- Keine Internet-Verbindung nötig
- Daten verlassen nie den eigenen Rechner

Beispiele

- LLaMA 3.1
- Mistral 7B
- Phi-3
- **gpt-oss** (aktueller Hype!)

Aktueller Hype: gpt-oss

Warum gpt-oss gerade viral geht:

- Open Source Alternative zu GPT-4
- Läuft komplett lokal
- Beeindruckende Benchmark-Ergebnisse
- Community wächst rasant

Aber ist der Hype berechtigt?

Garrett Mock

Game Developer

- [Lotum](#) 
- [programmier.bar](#) 
- Focus on Vue 



Warum Local LLM?

Datenschutz & Compliance

DSGVO-Konformität

- Daten verlassen nie das Unternehmen
- Keine Verarbeitung durch Dritte
- Vollständige Kontrolle über Datenfluss



Firmengeheimnisse

- Sensible Codebases bleiben intern
- Keine Übertragung von IP an Cloud-Anbieter
- Compliance-Anforderungen erfüllt

Beispiel: Deutsche Bank, Medizinische Forschung, Anwaltskanzleien

Kostenkontrolle

Cloud-Token Preise

GPT-4: ~\$20-60/Millionen Token

Claude: ~\$15-75/Millionen Token

Gemini: ~\$7-35/Millionen Token

Lokale Hardware

MacBook M3 Pro: ~€2500 (einmalig)

Gaming PC (RTX 4090): ~€2000 (einmalig)

Mac Studio M2 Ultra: ~€5000 (einmalig)

Bei 100k Token/Tag:

- GPT-4: ~\$600-1800/Monat
- Claude: ~\$450-2250/Monat

Break-even: Nach 2-4 Monaten bei hoher Nutzung

Offline-Fähigkeit & Kontrolle

Keine Internet-Abhängigkeit

- Arbeiten in abgeschotteten Netzwerken
- Flugzeug, Zug, schlechte Verbindung? Kein Problem!
- Keine Service-Ausfälle von OpenAI & Co.

Vollständige Kontrolle

- Modell-Parameter anpassen
- Temperatur, Top-p, etc. fein justieren
- Keine Rate-Limits oder API-Beschränkungen

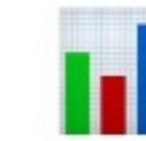
Versionskontrolle

- Modell-Snapshots für reproduzierbare Ergebnisse
- Keine unangekündigten Änderungen
- A/B-Testing verschiedener Modelle

Spezialisierung

🎯 Feintuning auf eigene Daten

- Trainingsdaten aus dem eigenen Domain
- Spezifische Terminologie und Stil
- Bessere Performance für Nischenbereiche



Beispiele

- **Code:** Firmenspezifische APIs und Patterns
- **Legal:** Juristische Texte und Verträge
- **Medical:** Medizinische Diagnosen und Berichte
- **Finance:** Finanzanalysen und Compliance

Resultat: Höhere Genauigkeit bei spezifischen Anwendungsfällen

State of the Art

Wo stehen wir?

Überblick: Die wichtigsten Modelle

LLaMA 3.1

- **Größen:** 7B, 70B, 405B
- **Highlight:** Sehr gute Code-Generation
- **Hardware:** 8GB - 200GB+ RAM

Mistral & Mixtral

- **Mistral 7B:** Kompakt, schnell
- **Mixtral 8x7B:** Mixture-of-Experts
- **Hardware:** 8GB - 48GB RAM

Phi-3

- **Microsoft Research**
- **Phi-3 Mini:** 3.8B Parameter
- **Highlight:** Sehr effizient, kleine Modelle

gpt-oss (DeepSeek-R1)

- **Aktueller Hype-Train**
- **Reasoning-fokussiert**
- **Ähnlich zu o1-Preview**

Benchmarks & ihre Aussagekraft

Modell	MMLU	HumanEval	RAM	Besonderheit
LLaMA 3.1 7B	69.4%	60.4%	8GB	Allrounder
Mistral 7B	64.1%	40.2%	8GB	Schnell
Phi-3 Mini	69.1%	61.0%	4GB	Effizient
gpt-oss	~78%*	~65%*	16GB	Reasoning

⚠️ Vorsicht bei Benchmarks:

- Synthetic Tasks ≠ Real-World Performance
- Benchmark-Hacking möglich
- Domain-spezifische Performance variiert stark

Hardware-Anforderungen

MacBook M-Serie

- **M1/M2/M3:** 16GB RAM → 7B Modelle
- **M1/M2/M3 Pro:** 32GB RAM → 13B Modelle
- **M1/M2 Ultra:** 64-128GB → 70B+ Modelle

Vorteil: Unified Memory, sehr effizient

Gaming PC

- **RTX 3080:** 10GB VRAM → 7B
- **RTX 4080:** 16GB VRAM → 13B
- **RTX 4090:** 24GB VRAM → 70B (quantized)

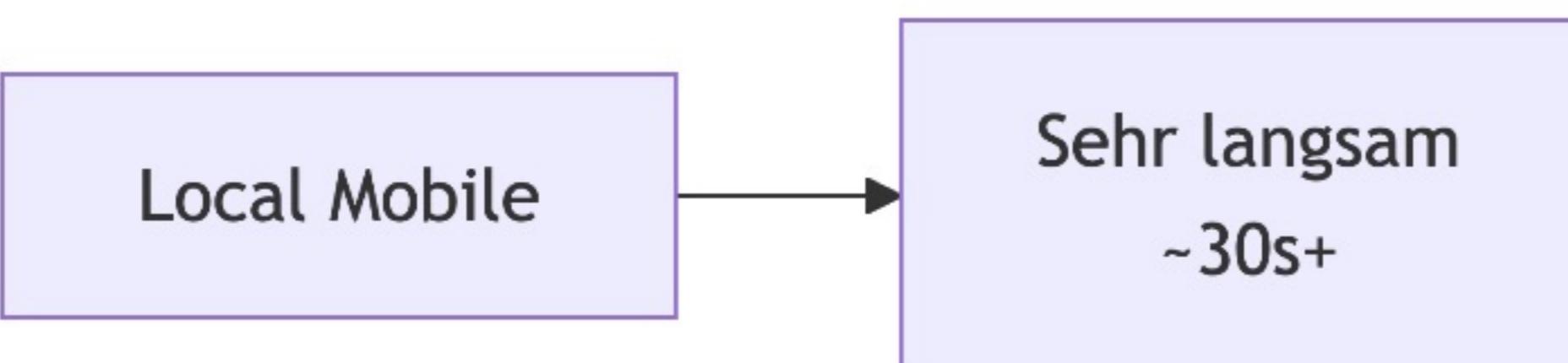
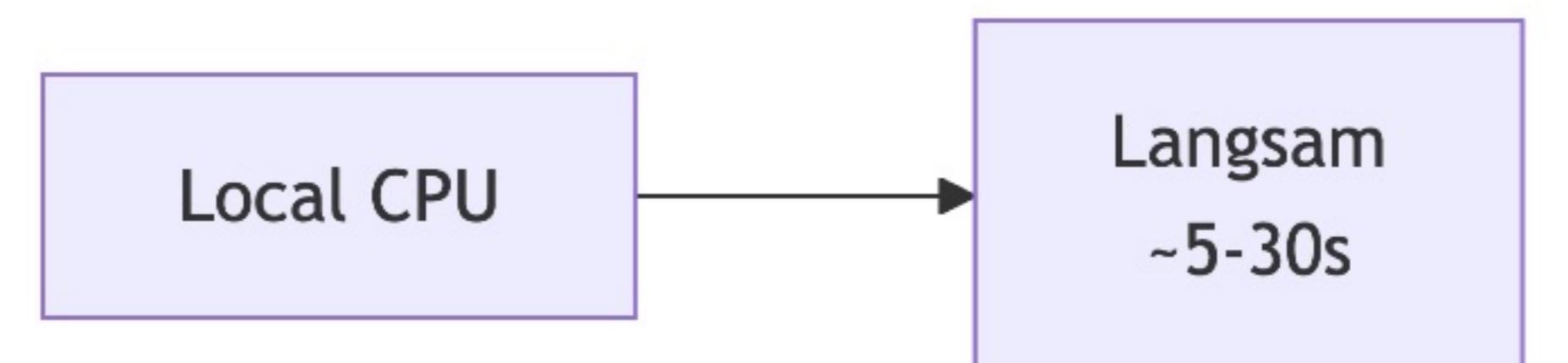
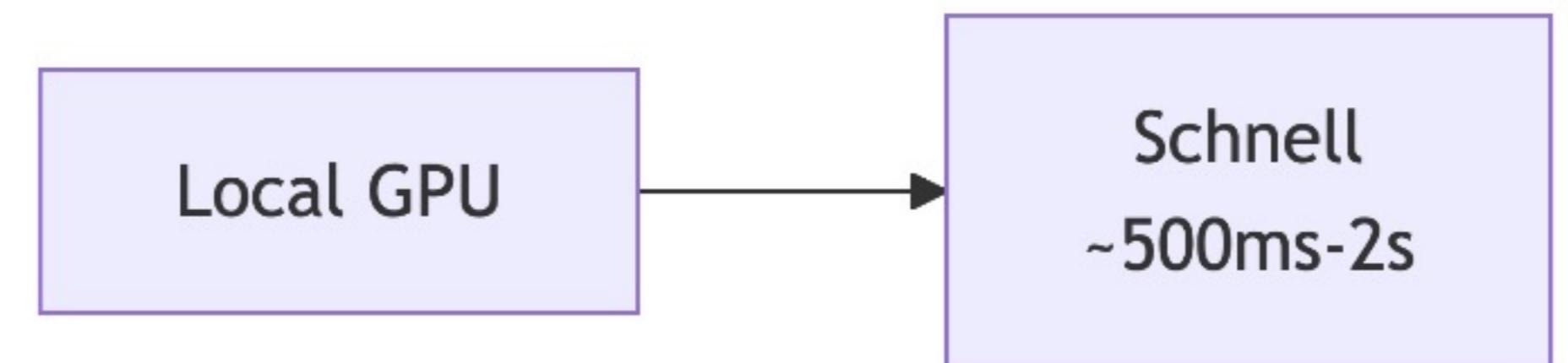
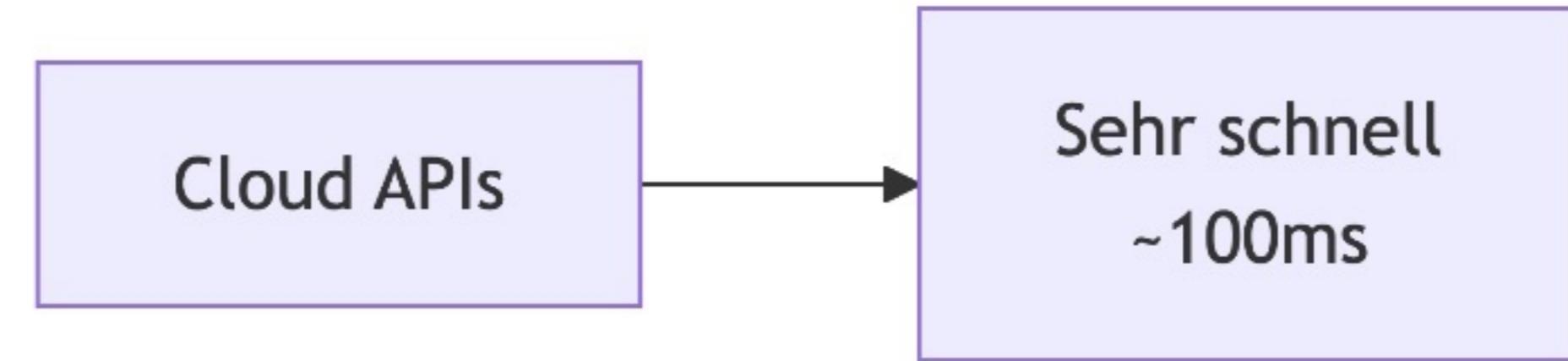
Vorteil: Sehr schnelle Inferenz

Kleine Server

- **128GB RAM:** Mehrere 70B Modelle
- **AMD Threadripper:** Viele CPU-Kerne
- **Server-GPUs:** A100, H100 für Enterprise

Vorteil: Multi-User, hoher Durchsatz

Performance-Vergleich



Hands-on: Lokale Tools

Ollama - Der einfache Einstieg

🚀 Was ist Ollama?

- Docker für LLMs
- CLI + REST API
- Automatisches Model Management
- Cross-Platform (Mac, Linux, Windows)

⚡ Quick Start

```
# Installation
curl -fsSL https://ollama.ai/install.sh | sh

# Model laden
ollama pull llama3.1

# Chat starten
ollama run llama3.1
```

💡 **Perfect für:** Entwickler, CLI-Fans, API-Integration

LM Studio - UI First Ansatz

■ Features

- Grafische Benutzeroberfläche
- Model Browser mit Ratings
- Einfaches Download Management
- Chat Interface
- Local Server Modus

■ Vorteile

- **Anfängerfreundlich:** Drag & Drop Interface
- **Transparenz:** Hardware-Usage in Echtzeit
- **Flexibilität:** Verschiedene Quantisierungen
- **Testing:** A/B-Vergleiche zwischen Modellen

 **Perfect für:** Einsteiger, Experimente, schnelle Prototypen

OpenRouter - Model Playground

Konzept

- Zugriff auf 200+ Modelle
- Einheitliche API
- Local + Cloud Models
- Kosten-Transparenz

Pricing Vergleich

GPT-4o:	\$5.00/1M tokens
Claude 3.5:	\$3.00/1M tokens
LLaMA 70B:	\$0.80/1M tokens
Mistral 7B:	\$0.20/1M tokens

 **Perfect für:** Model-Vergleiche, Testing, Hybrid-Ansatz

Model Downloads - Die Quellen

🤗 Hugging Face

- Größte Sammlung
- GGUF, Safetensors
- Community-driven
- Quantisierte Versionen

Empfehlung:

- TheBloke's Quantizations
- Microsoft/Meta official

🚀 GPT4All

- Kuratierte Auswahl
- Qualitäts-getestet
- Einfache Downloads
- Desktop App

Vorteile:

- Weniger Auswahl = weniger Verwirrung
- Getestet auf Consumer Hardware

📦 Ollama Library

- Automatisches Management
- Optimierte Quantisierung
- Dependency Handling
- Version Control

Commands:

```
ollama list  
ollama pull model:tag  
ollama rm model
```

Tipps für Model-Selection

⌚ Nach Anwendungsfall wählen:

- **Code:** CodeLlama, Phi-3, DeepSeek-Coder
- **Chat:** LLaMA 3.1, Mistral, Qwen
- **Reasoning:** gpt-oss, o1-mini alternatives
- **Multilingual:** Qwen, Command-R

📏 Größe vs. Performance:

- **7B:** Schnell, MacBook M1 16GB
- **13B:** Gute Balance, MacBook M2 Pro
- **70B:** Beste Qualität, braucht 48GB+ RAM

12 34 **Quantisierung verstehen:**

- **Q4_K_M:** Standard, gute Balance
- **Q8_0:** Bessere Qualität, mehr RAM
- **Q2_K:** Sehr kompakt, Qualitätsverlust

Live-Demo

Coding-Integration

Continue VSCode Plugin

🔌 Was ist Continue?

- GitHub Copilot Alternative
- Lokale + Cloud Models
- Autocompletion + Chat
- Open Source

⚙️ Setup

1. VSCode Extension installieren
2. Config anpassen (`~/.continue/config.json`)
3. Ollama Model wählen
4. Fertig! 🎉

```
{  
  "models": [  
    {  
      "title": "LLaMA 3.1",  
      "provider": "ollama",  
      "model": "llama3.1:7b"  
    }  
  ]  
}
```

OpenCode - Chat + Code

💬 Features

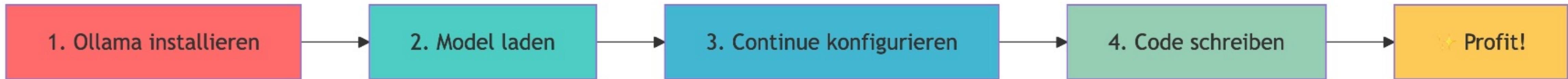
- Code-bewusster Chat
- Datei-Kontext einbeziehen
- Refactoring-Vorschläge
- Multi-Language Support

🚀 Installation

```
npm install -g @opencode/cli  
# oder  
pip install opencode  
  
# Integration mit lokalem Modell  
opencode --model ollama/llama3.1
```

🎯 **Use Cases:** Code Review, Bug-Fixing, Refactoring, Dokumentation

Demo-Szenario: "Von 0 zu LLM-Coding in 3 Minuten"



🎬 Live Demo Steps:

1. Ollama Setup (30 Sekunden)

```
curl -fsSL https://ollama.ai/install.sh | sh
ollama pull phi3:mini
```

2. VSCode Extension (30 Sekunden)

- Continue installieren
- Konfiguration anpassen

3. Code Generation (2 Minuten)

- React Component erstellen
- Refactoring mit LLM
- Bug-Fix mit Kontext

Praktische Tipps

⚡ Performance-Optimierung

- **Kleines Modell** für Autocompletion (Phi-3 Mini)
- **Größeres Modell** für komplexe Tasks (LLaMA 7B)
- **GPU-Acceleration** aktivieren
- **Batch-Size** anpassen

🧠 Prompting-Strategien

- **Kontext bereitstellen:** Dateien, Fehler, Ziele
- **Spezifisch sein:** "Refactor this function to use async/await"
- **Iterative Verbesserung:** Nachfragen, verfeinern
- **Code-Style definieren:** ESLint, Prettier

🎯 Realistische Erwartungen:

- Boilerplate Code, einfache Funktionen
- Code-Erklärungen, Dokumentation
- ! Komplexe Algorithmen, Business-Logik
- Perfekter Code ohne Review

Alternative Tools



- Terminal-basiert
- Git-Integration
- Multi-File Edits

```
pip install aider-chat  
aider --model ollama/llama3.1
```



LocalGPT

- Web Interface
- Document Chat
- RAG-Integration

```
git clone localGPT  
python ingest.py  
python run_localGPT.py
```



Tabby

- Self-hosted Copilot
- Team-Deployment
- Fine-tuning Support

```
docker run -it tabbyml/tabby
```

Blick nach vorn

Multi-Modal wird lokal

👀 Vision Models

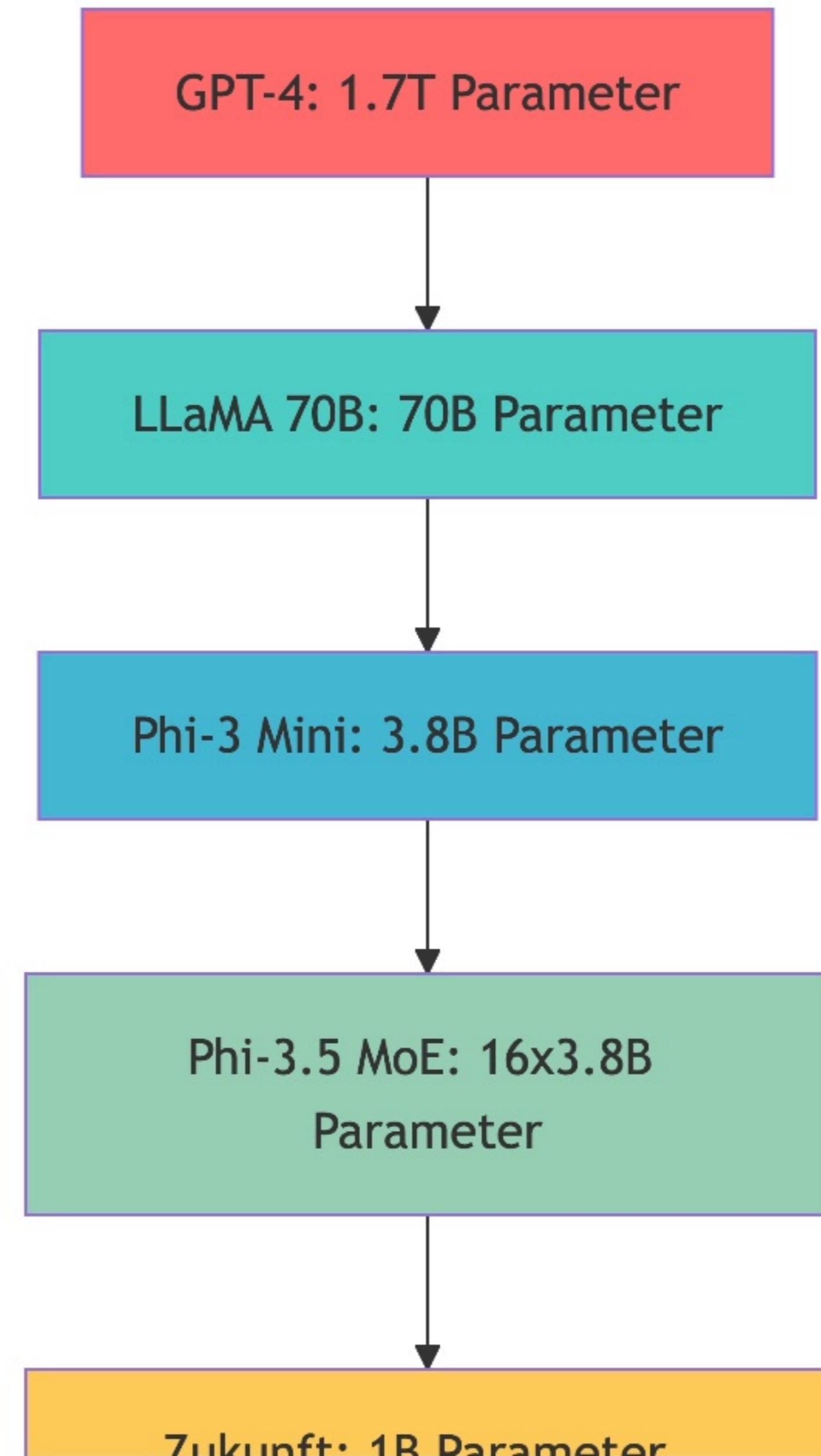
- **LLaVA:** Bilder verstehen und beschreiben
- **MiniCPM-V:** Kompakt, für Consumer Hardware
- **Qwen-VL:** Multilinguale Vision

🎵 Audio Integration

- **Whisper:** Speech-to-Text lokal
- **Bark:** Text-to-Speech Generation
- **MusicGen:** Musik aus Text

🚀 **Vision 2025:** Ein lokales System für Text, Bild, Audio und Video

Kleiner werdende Modelle



Private RAG & Orchestrierung

Lokale RAG-Systeme

- **Embeddings:** all-MiniLM, BGE-M3
- **Vector DBs:** ChromaDB, Qdrant, Weaviate
- **Documents:** PDFs, Wikis, Code-Repos
- **Privacy:** Alles bleibt lokal

LangChain Lokal

- **Agents:** Planning, Tools, Memory
- **Chains:** Multi-Step Workflows
- **Tools:** Web Search, Calculators, APIs
- **No Cloud:** Komplett lokale Orchestrierung

Beispiel Use Case:

```
Analysiere meine E-Mails → Finde relevante Dokumente →  
Erstelle Zusammenfassung → Generiere Antwort-Entwurf
```

Hybrid-Ansatz: Das Beste aus beiden Welten

Lokal für:

- **Sensitive Daten:** Passwörter, Verträge, Medizin
- **Batch Processing:** Code-Generierung über Nacht
- **Development:** Coding-Assistenz, Refactoring
- **Offline:** Flugzeug, schlechte Verbindung

Cloud für:

- **Complex Reasoning:** Mathematik, komplexe Logik
- **Latest Knowledge:** Aktuelle Events, neue APIs
- **High Throughput:** Viele parallele Anfragen
- **Latest Models:** GPT-o1, Claude 3.5 Opus

Smart Routing:

- Einfache Fragen → Lokales 7B Modell
- Code-Review → Lokales Code-Model
- Komplexe Analyse → Cloud GPT-4
- Sensitive Daten → Immer lokal

Fazit: Die Zukunft ist hybrid

2025 Prediction:

- Lokal: 80% aller Developer-Tasks
- Cloud: 20% für komplexeste Probleme

Hardware-Trends:

- Apple M4 mit 32GB unified memory als Standard
- NVIDIA RTX 5090 mit 32GB VRAM
- Specialized AI-Chips (Google TPU, Apple Neural Engine)

Ecosystem:

- Bessere Tools, einfachere Installation
- Enterprise-grade lokale LLM-Stacks
- Regulation treibt lokale Lösungen

 **Bottom Line:** Local LLMs sind nicht mehr "Nice to have" – sie werden zum Standard für datensensitive Anwendungen

Resources & Weiterführende Links

🔗 Tools

- Ollama - CLI für lokale LLMs
- LM Studio - GUI für Experimente
- Continue - VSCode Integration
- Aider - Terminal Coding

📚 Lernen

- Hugging Face Course - Deep Learning Basics
- r/LocalLLaMA - Community
- Papers with Code - Latest Research
- TheBloke - Quantized Models

🤝 Q&A Zeit!

Fragen, Diskussion, Erfahrungsaustausch