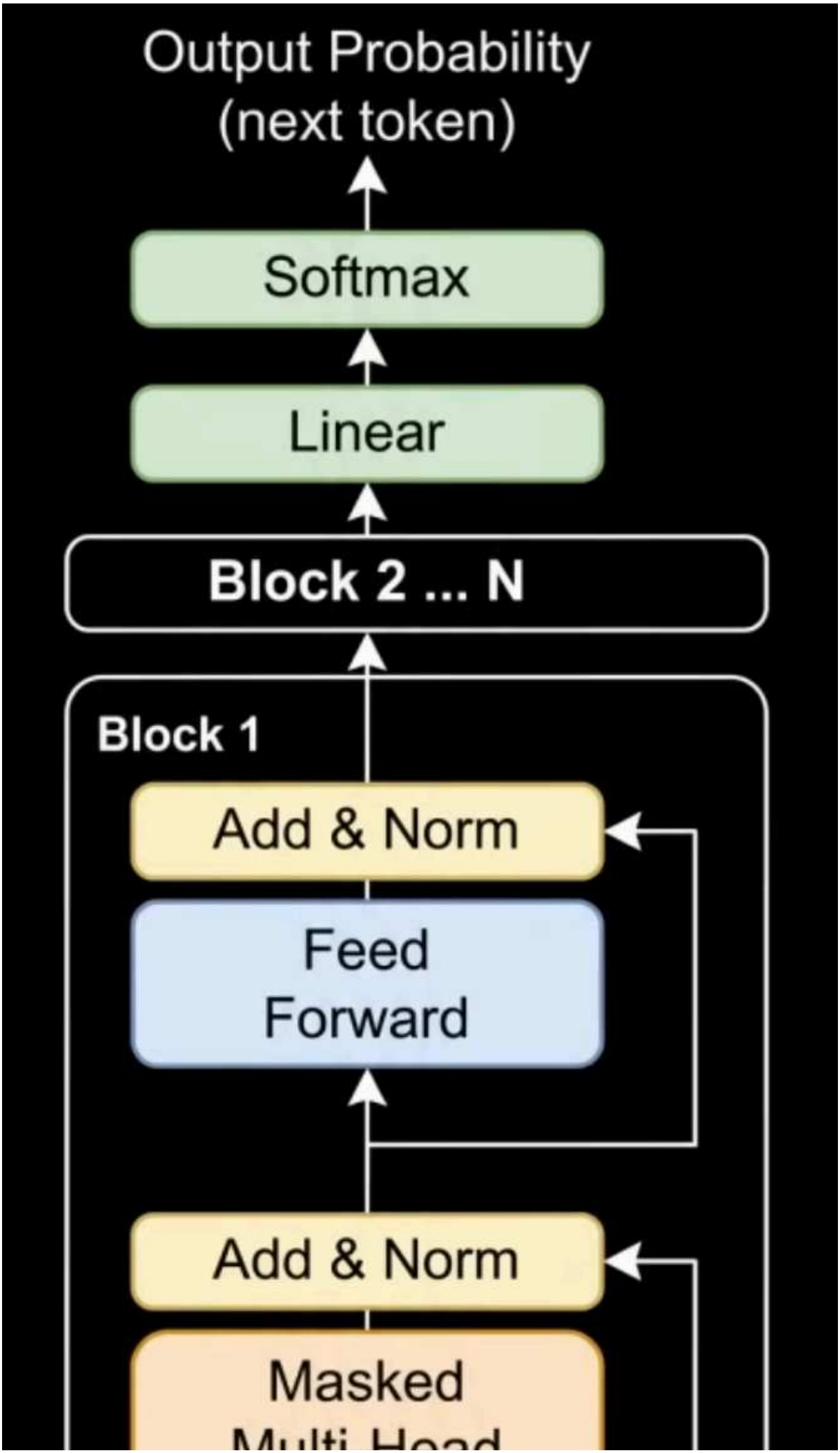
AI/Präsentation/Lokale KI

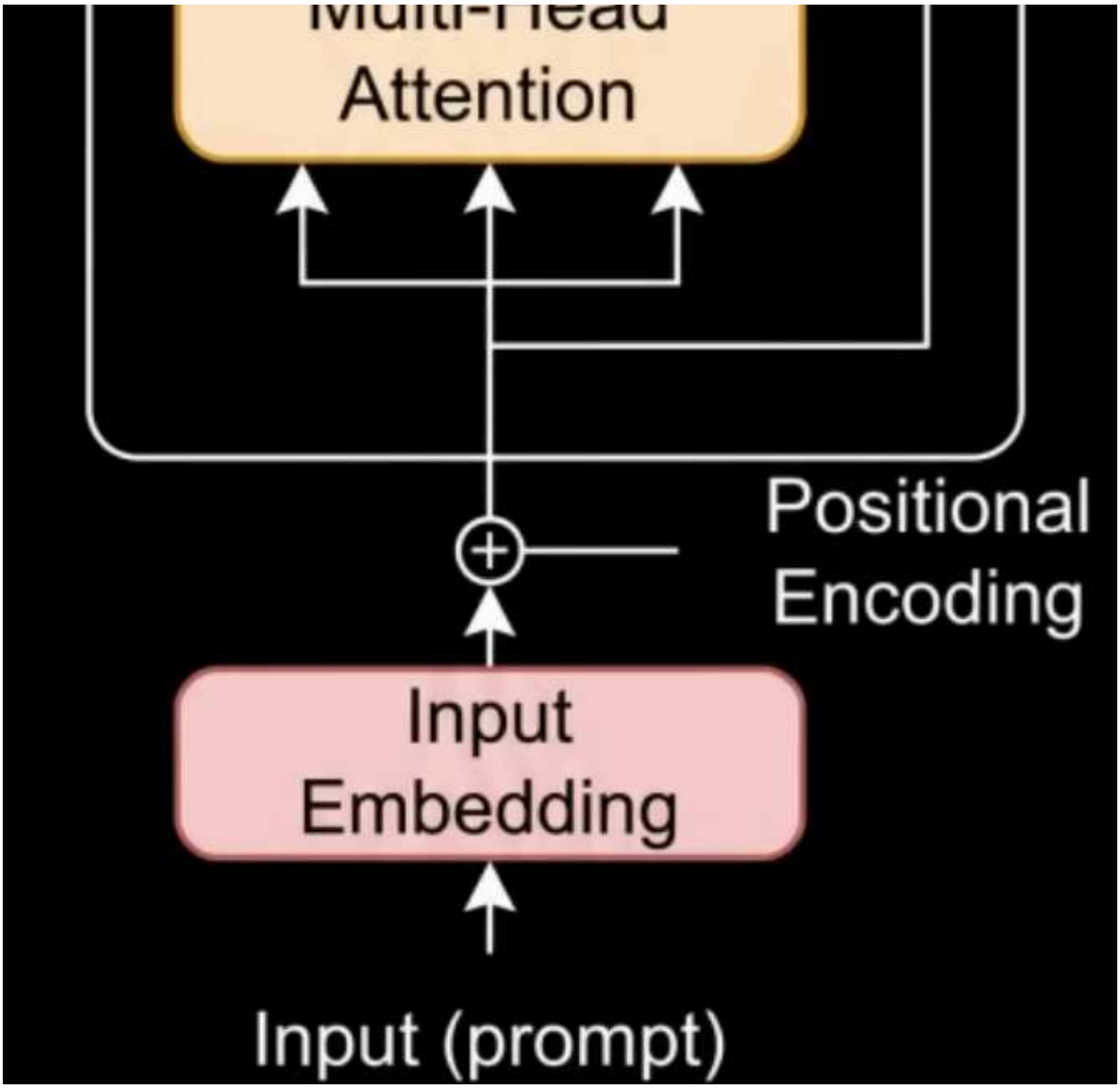
- ## Motivation und Anwendungsfälle
  - **Höchste Sicherheit und Privatsphäre**
    für die Verarbeitung von sensiblen Daten: Patientendaten Juristische Dokumente geheime Geschäftsdaten
  - **Externe Kosten einsparen**
  - **Offlineverfügbarkeit**
    Kein Internet notwendig für mobiles Arbeiten oder in stark gesicherten Netzwerken ohne Internet Zugang ("Air-Gapped Systems") Keine Abhängigkeit von KI-Anbietern => Das eigene Geschäftsmodell wird Unabhängig und berechenbarer/beeinflussbarer
  - **Kontrolle und Beständigkeit**
    Reproduzierbarkeit => Keine unangekündigten Updates durch einen KI-Anbieter, welche einem die App kaputt machen Anpassbarkeit => Voller Zughriff auf das Modell ermöglicht Feintuning (eigene Daten in das Modell bringen) und auswahl der gewünschten Parameter (Temperatur, System-Prompt)
  - **Ungefilterte Nutzung**
    große kommerzielle Modelle haben häufig strenge Leitplanken, die sie eine Antwort verweigern lassen, auch wenn es nicht nötig wäre (z.B. kreatives Schreiben über konfliktbehaftete Themen oder Analyse von Malware)
- ## Attention is all you need

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

[12]

- **Was ist eigentlich die aktuelle KI?**
- Der Attention-Transformer


Output Probability
(next token)
Softmax
Linear
Block 2 ... N
Block 1
Add & Norm
Feed
Forward
Add & Norm
Masked



{:height 600, :width 600} [13]

- **Speicherverbrauch**
  Diese Formeln gelten für den Standard Attention Transformer $$\text{Bytes pro Token} = 2 \times n_{\text{layers}} \times n_{\text{kv_heads}} \times d_{\text{head}} \times b$$ $$\text{VRAM}*{\text{KV-Cache}} = 2 \times L \times B \times n*{\text{layers}} \times n_{\text{kv_heads}} \times d_{\text{head}} \times b$$ Parameter 2 = Fest, repräsentiert K + V (Key- + Value-Matrix) L = Context Length => Gewünschte Context Window größe B = Batch Size => Anzahl paralleler Anfragen, bei lokaler Nutzung meist 1 n_{layers} = language_model.model.layers = Anzahl der Transformer-Schichten $n_{\text{kv_heads}}$ = language_model.model.layers.0.self_attn.b_proj.weight [**kv_heads**, hidden_size] => Anzahl der Attention Heads für K und V d_{head} = language_model.model.layers.0.self_attn.A_log => Dimension eines einzelnen Attention-Heads (meistens 128) b = Bytes pro Zahl
  - FP32 = 4
  - FP16/BF16 = 2
  - FP8/INT8 = 1
  - INT4 = 0,5

- # Hardware
  - ## VRAM (Video RAM)
    LLMs generieren Text Token für Token. Bei jedem Durchgang muss ein mal der gesamte aktive Teil des Modells aus dem Speicher gelesen und verarbeitet werden. Entscheidend für die Geschwindigkeit ist also hauptsächlich die Speicherbandbreite. Vergleich Bandbreite (Bandbreite in MB/s = Übertragungsrate in MT/s * 8 Byte)
    - **Vergleich Geschwindigkeiten**

Bezeichnung	**Geschwindigkeit**
Normaler System RAM	
DDR4 Dualchannel (DDR4-3200)	ca. 50 GB/s
DDR5 Dualchannel (DDR5-6000)	ca. 100 GB/s
_	_
Apple M5 Max	614 GB/s
_	_
VRAM	
GDDR6 (Geforce RTX 3090 mit 384-Bit Busbreite)	1000 GB/s
GDDR7 (Geforce RTX 5090 mit 512-Bit Busbreite)	1800 GB/s
_	_
HBM (High Bandwith Memory, wie er in modernen KI-Servern eingesetzt wird)	
Nvidia Vera-Rubin NVL72 mit HBM4	20.000 GB/s [1]
AMD MI455X mit HBM4	23.000 GB/s [2]
Skalierung im Server-Rack (Helios)	1.700.000 GB/s
  - ## Apple Silicon, DGX Spark und andere ARM-Varianten
    Bandbreite ist nicht alles: Große KI-Modelle benötigen mehr Speicher. Die ARM-Prozessoren von Apple (M4/M5) oder Nvidia (GB10) ermöglichen es, dass die Grafikkarte und die CPU den gleichen Arbeitsspeicher Nutzen können. Während VRAM normal sehr teuer ist, kann man mit relativ kleinem Budget sehr große Modelle auf der Grafikkarte laufen lassen.
    - **Vergleich Speichergrößen**

Bezeichnung	**Größe**
Gängiger GPU VRAM	2-24GB
Gängiger CPU RAM	8-128 GB
M5 Max unified Memory	128 GB
KI-Server mit AMD MI455X mit HBM4	432 GB
Skalierung im Server-Rack (Helios)	31.000 GB
  - ## NPU (Neuromorpher Prozessor)
    Neuere CPUs enthalten eine Komponente, die speziell für die KI-Aufgaben gemacht ist: NPU NPU übernehmen kleinere KI-Rechenaufgaben sehr effizient gegenüber einer

CPU und GPU Ermöglicht hohe Speicherbandbreiten mit dem System RAM, wenn in CPUs integriert

- # Die Modelle & deren Herkunft
  - **Huggingface**
    DIE zentrale Open-Source-Plattform für KI Sammlung an fertigen Modellen Datensätze und Bibliotheken zum trainieren von KI Huggingface Spaces: ein Ort um KI-demos direkt im Web aus zu probieren ohne selbst code ausführen zu müssen
  - **Modelle**
    Demo, wie man passende Modelle findet und wo drauf man achten muss
- # Die Engine
  KI Modelle müssen irgendwo laufen und gestartet werden. Je nach Anwendungsfall sollte man dafür das passende Backend auswählen. Das Backend stellt eine REST-API bereit, über das weitere Tools angebunden werden können.
  - **Wie alles anfing**
    - **Die Prä-Transformer-Ära (bis 2017)**
      torch und tensorflow wurden verwendet um sogenannte RNN und LSTM zu trainieren. Nicht sehr Effekltiv, aber sehr flexibel.
    - **Der Durchbruch des Transformers (2017–2022)**
      Google veröffentlicht das Papier "*Attention Is All You Need*" [4] was zu der Entwicklung des pytorch-transformers führt. Schnell wurden die Modelle, welche von OpenAI, Nvidia und Microsoft veröffentlicht wurden zu groß für lokale Hardware. Die Community entwickelt den bitsandbytes-transformer [5], der es ermöglicht mit Quantisierung den VRAM-Bedarf zu senken.
    - **LLAMA (2023)**
      Meta (ehem. Facebook) veröffentlicht das Large Language Model Meta AI (LLAMA). [6] Der Entwickler Georgi Gerganov schreibt ein komplett neues Framework in C/C++ und führt das GGML und später das GGUF Format ein. [7]
  - **llama.cpp**
    llamacpp hat sich als quasi-Standard für den Normalnutzer herauskristallisiert. Man kann llama.cpp entweder direkt laufen lassen um die volle Kontrolle zu haben oder ein darauf aufbauendes Tool, wie Ollama [8] oder LM Studio [9] verwenden. Die Tools versuchen in der Standardeinstellung den Speicherverbrauch für Einzelplatz Hardware (Den eigenen PC) zu optimieren. Es ist möglich verschiedene Hardwarekombinationen mit verschiedenen Optimierungen zusammen laufen zu lassen. CUDA für Nvidia ROCm für AMD Vulkan für Nvidia + AMD
  - **vLLM**
    Python/CUDA basiertes Framework, welches sich als quasi-Standard für Enterprise-Rechenzentren herauskristallisiert hat. Effizeientes VRAM-Management für viele gleichzeitige Nutzer
  - **Ausblick: tinygrad**

hoch effizientes Python-Framework, welches von Grund auf in Assembler geschrieben wurde und schneller und effizienter arbeitet, als die Implementierungen von Nvidia und AMD ohne CUDA/ROCm zu benötigen und läuft auf nahezu jedem System. [10] Die Entwicklung ist noch nicht abgeschlossen und für die meisten Privatanwender noch nicht effektiv nutzbar. Falls sich jemand einbringen möchte: Der Entwickler (bekannt als Geohot [George Hotz]) bietet cash bounties für gute pull requests (KI explizit ausgeschlossen), die Liste ist im Github Repo verlinkt. [11]

- # Benutzeroberflächen
  - **All-in-one Lösungen**
    Für die meisten Nutzer passen die folgenden tools, wobei beachtet werden muss, dass teilweise auch Cloud-Modelle mit genutzt werden können. LM Studio Ollama AnythingLLM Unsloth Studio (+KI training)
  - **Web-Interfaces**
    Für Unternehmen oder technisch versiertere Benutzer Open-WebUI LibreChat
  - **Kommandozeilen Tools**
    Claude-CLI pi.dev OpenAI Codex
- # AI-Harness
  - Ein AI-Harness ist wie das Geschirr für einen Schlittenhund oder Hufeisen und Sattel für das Pferd: Es ermöglicht der KI mehr zu machen und schränkt es indem, was das KI-Modell machen kann ein. Viele Benutzeroberflächen bieten entsprechende Einstellungen und Schnittstellen
  - **RAG (Retrieval-Augmented Generation)**
    - **Suche (Retrieval):** Das Rag-System durchsucht angeschlossene Wissensdatenbanken nach Informationen, welche zu den vom User eingegebenen Text passen. Eigene Dokumente Vektordatenbanken Spezialisierte KI-Datenbanken Suchmaschinen (Google, Brave)
    - **Erweiterung (Augmentation):** Die gefundenen Informationen werden zusammen mit der Originalfrage an das LLM geschickt.
  - **Function Calling / Tool Use**
    Innerhalb des Harness können dem LLM grundlegende Funktionen bereit gestellt werden, um z.B. Dateien zu lesen/schreiben oder die Kommandozeile des Systems zu benutzen.
  - **MCP (Model Context Protocol)**
    Um den Funktionsumfang des LLM dynamisch zu erweitern, können dem LLM damit weitere Datenquellen, Tools und Programme bereitgestellt werden. Es ist quasi der USB-Anschluss für KI an andere Systeme.
  - **Agents**
    KI, die über Kommandozeile oder ein Chatprogramm gesteuert wird und semi-autonom (fragt nach Bestätigung für Aktionen) bis voll-autonom Aufgaben erledigt. Je nach autonomie level kann es hier große Gefahren geben, wenn die KI

selbstständig Aktionen ausführt. Claude-CLI oder OpenAI Codex mit Cronjob Hermes-Agent OpenClaw

- # Grenzen und Zukunftsaussichten

  Lokale KI sollte immer möglichst klein und Spezifisch für die Aufgabe sein. Kontextfenster Lokaler Modelle können Probleme machen, weil die KI "vergisst", was man von ihr will und was das Thema ist. Logik-Lücken: Lokale Modelle eignen sich für eher einfache Aufgaben. Für Hochkomplexe Aufgaben gibt es derzeit keine sinnvolle Alternative zu den großen Modellen der KI-Anbieter. Das Thema KI ist noch stark in der Entwicklung. In den letzten 3 Jahren gab es jedes Jahr eine ver-10-fachung des Könnens und der Fähigkeiten von KI. Die Rechenleistung steigt noch immer um das doppelte pro Jahr und hat in den letzten Jahren sogar an Geschwindigkeit zugenommen. Auch kleine KI-Modelle werden immer schlauer. KI wird bleiben und teil des Alltags. Daten, die wir erzeugen und den Erstellern von KI-Modellen zur Verfügung stellen werden genutzt werden um die Modelle zu trainieren.
- # Quellen:
  - https://www.computerbase.de/news/wirtschaft/hbm4-fuer-vera-rubin-zurueck-von-22-auf-20-tb-s-fuer-mehr-passende-chips.96376/ logseq.order-list-type:: number
  - https://www.pcgameshardware.de/Grafikkarten-Grafikkarte-97980/Specials/AMD-Instinct-MI455X-CDNA-5-432-GiB-HBM4-Technik-Check-1549242/ logseq.order-list-type:: number
  - https://huggingface.co/ logseq.order-list-type:: number
  - https://arxiv.org/html/1706.03762v7 logseq.order-list-type:: number
  - https://github.com/bitsandbytes-foundation/bitsandbytes logseq.order-list-type:: number
  - https://github.com/meta-llama/llama logseq.order-list-type:: number
  - https://github.com/ggml-org/llama.cpp logseq.order-list-type:: number
  - https://ollama.com/ logseq.order-list-type:: number
  - https://lmstudio.ai/ logseq.order-list-type:: number
  - https://www.youtube.com/watch?v=QyrQeeDZUG4 logseq.order-list-type:: number
  - https://github.com/tinygrad/tinygrad logseq.order-list-type:: number
  - https://www.ee.cityu.edu.hk/~lmpo/ee4016/pdf/2026_AI_L06A_Transformer.pdf logseq.order-list-type:: number
  - https://www.youtube.com/watch?v=U2hZFMVNSE0 logseq.order-list-type:: number