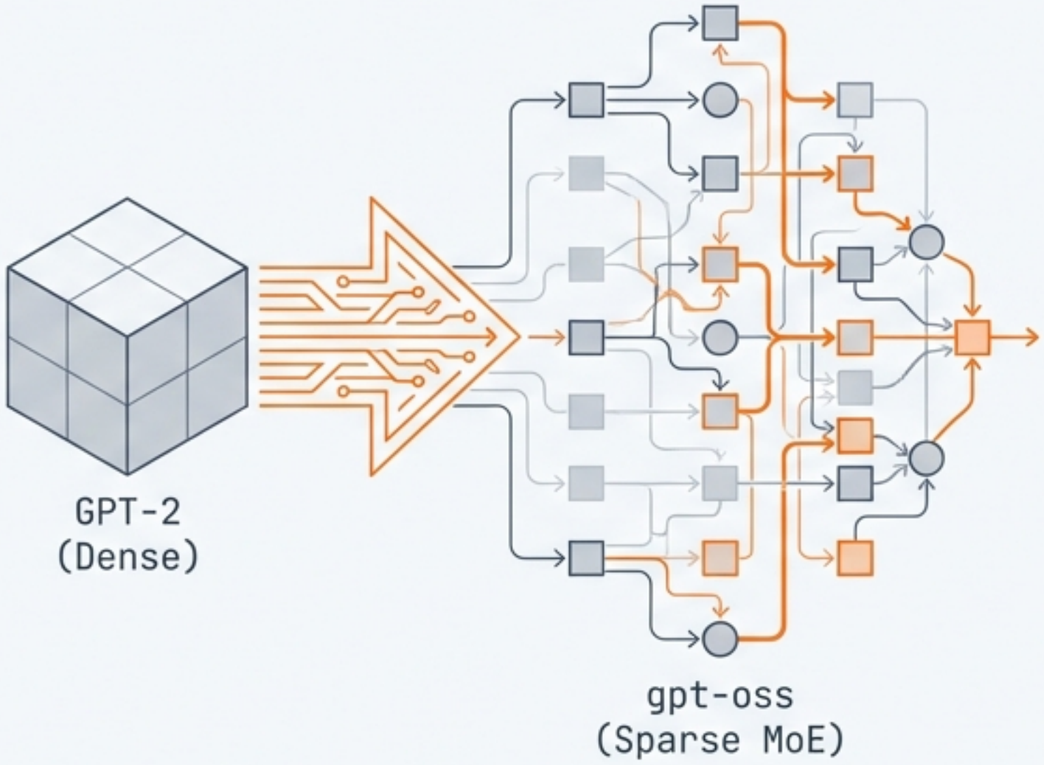
Technical Manuscript

Prompt

เจาะลึกสถาปัตยกรรม gpt-oss: จาก GPT-2 สู่ยุคใหม่ของ Open-Weights

Sarabun

วิเคราะห์ความก้าวหน้าทางเทคนิค เปรียบเทียบ Qwen3 และความท้าทายในภูมิทัศน์ AI ปี 2025



อ้างอิงจากบทวิเคราะห์โดย Sebastian Raschka, PhD (สิงหาคม 2025)

NotebookLM

การกลับมาของ Open Weights: gpt-oss-20b และ 120b

OpenAI ปล่อยโมเดล Open-Weight ครั้งแรกนับตั้งแต่ปี 2019 โดยมุ่งเน้นที่ความสามารถด้านเหตุผล (Reasoning) และการเขียนโค้ด

gpt-oss-20b

- **Size:** 20 Billion Parameters
- **Hardware:** Consumer GPU (RTX 50-series)
- **Focus:** Low-latency Coding / Instruction

gpt-oss-120b

- **Size:** 120 Billion Parameters
- **Hardware:** Enterprise (1x H100 80GB)
- **Focus:** Deep Reasoning / STEM

Common Specs: License Apache 2.0 | Format MXFP4 | Context 128k

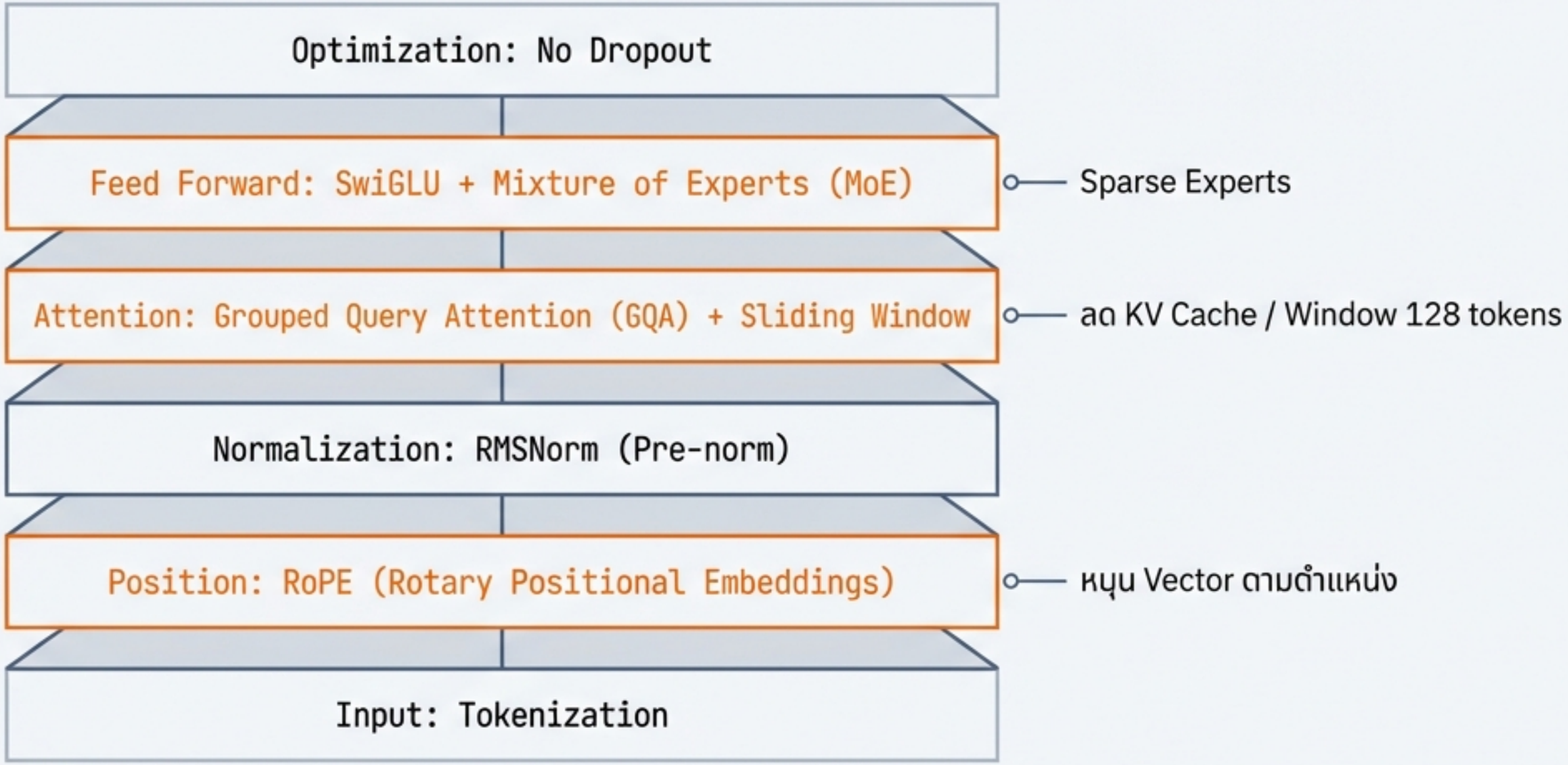
โมเดล 120B สามารถรันได้บน GPU H100 เพียงตัวเดียว (80GB VRAM) ในขณะที่รุ่น 20B ออกแบบมาสำหรับ RTX 50-series หรือการ์ดรุ่นเก่าที่ลง Patch แล้ว

NotebookLM

Prompt: ภาพรวมสถาปัตยกรรม: มาตรฐานใหม่ของปี 2025

แม้พื้นฐานยังเป็น Transformer แต่ส่วนประกอบย่อยถูกอัปเกรดเพื่อประสิทธิภาพและความแม่นยำสูงสุดเมื่อเทียบกับยุค GPT-2

Tech Stack



NotebookLM

Technical Manuscript

ตัดส่วนเกิน: วิวัฒนาการจาก GPT-2 (Dropout & Norms)

1. No Dropout (ไม่มีการใช้ Dropout)

Dropout ถูกออกแบบมาป้องกัน Overfitting ในยุคที่เทรนหลาย Epoch แต่สำหรับ LLM ยุคใหม่ที่เทรนแบบ Single-epoch บนข้อมูลมหาศาล Dropout กลายเป็นสิ่งไม่จำเป็นและลดประสิทธิภาพ

2. RMSNorm Replaces LayerNorm

เปลี่ยนจาก LayerNorm มาใช้ RMSNorm เพื่อความรวดเร็วในการคำนวณ โดยตัดการคำนวณ Mean และลดขั้นตอน Cross-feature reductions

Normalization Math

LayerNorm (GPT-2)

$$y = \frac{x - mean}{std} * gamma + beta$$

Simpler Computation

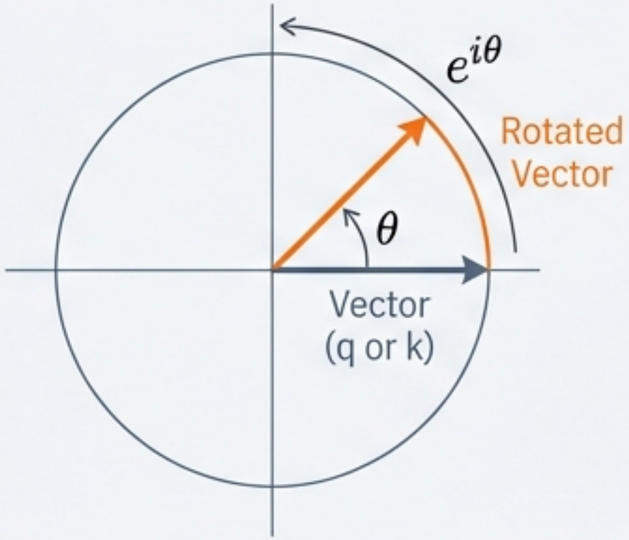
RMSNorm (gpt-oss)

$$y = \frac{x}{rms(x)} * gamma$$

NotebookLM

การอัปเกรดคณิตศาสตร์: RoPE และ SwiGLU

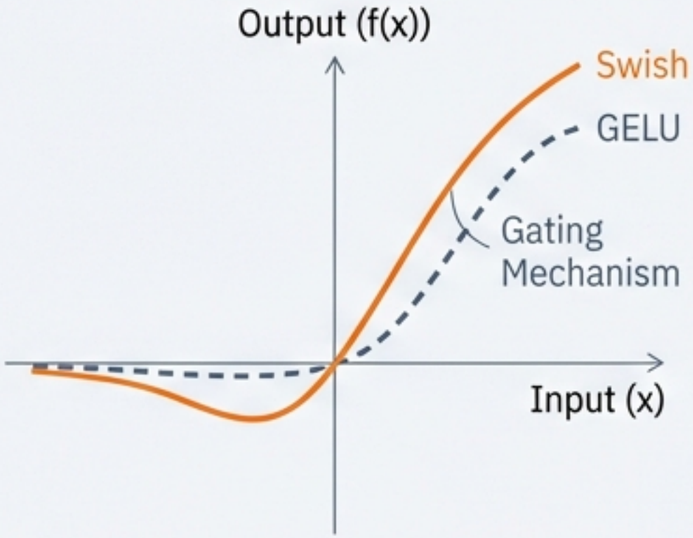
RoPE (Rotary Positional Embeddings)



แทนที่ Absolute Embeddings แบบเดิมด้วยการหมุน Vector ของ Query และ Key ตามตำแหน่ง

Insight: จัดการความสัมพันธ์ของลำดับข้อมูล (Sequence) ได้ดีกว่า โดยเฉพาะใน Context ยาว

SwiGLU vs GELU



เปลี่ยน Activation Function จาก GELU เป็น Swish ร่วมกับ Gated Linear Unit (GLU)

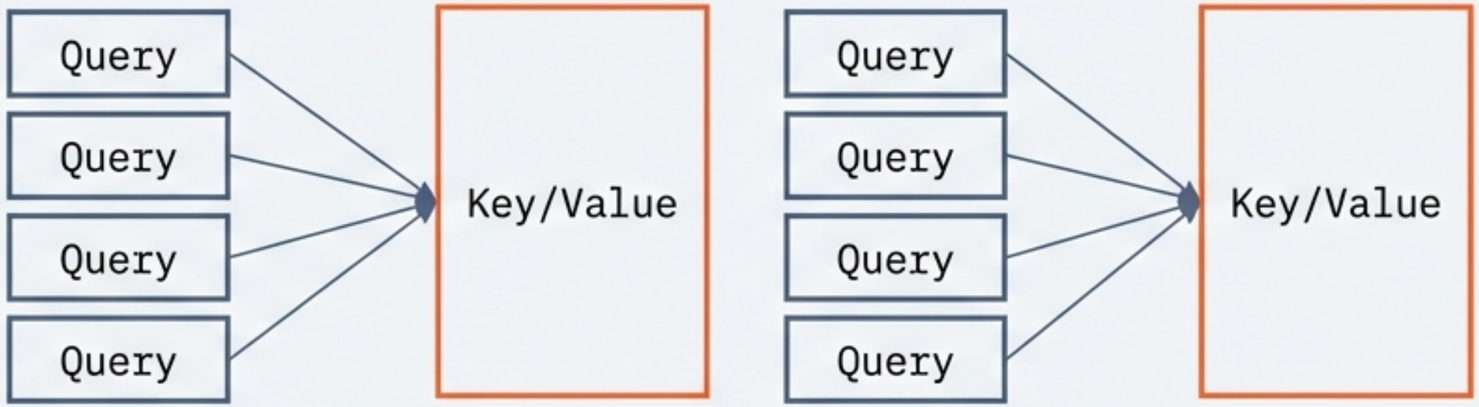
Insight: ในการใช้งานจริงจะลดขนาด Layer ลงครึ่งหนึ่ง และเพิ่มความสามารถในการแสดงออก (Expressivity) ผ่าน Multiplicative Interaction

NotebookLM

ประสิทธิภาพของ Attention: GQA และ Sliding Window

Grouped Query Attention (GQA)

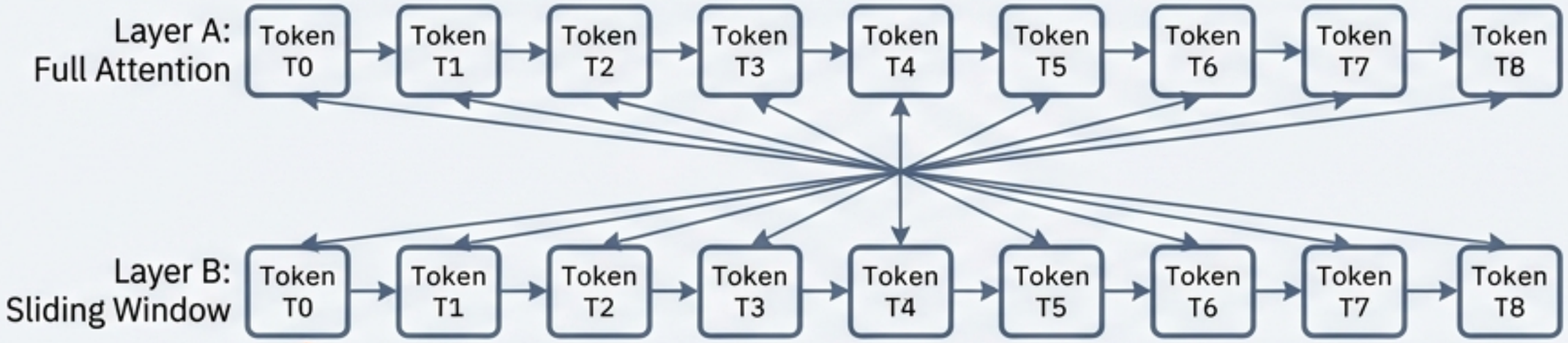
ลดการใช้หน่วยความจำโดยการจัดกลุ่มให้หลาย Query heads แชร์ Key/Value เดียวกัน



Shared KV Heads reduce memory bandwidth

Sliding Window Attention

gpt-oss ใช้เทคนิคสลับ Layer: Layer A (Full Attention) สลับกับ Layer B (Window 128 tokens)



Window size 128 tokens (เล็กมาก) ช่วยลด Compute cost ได้มหาศาล

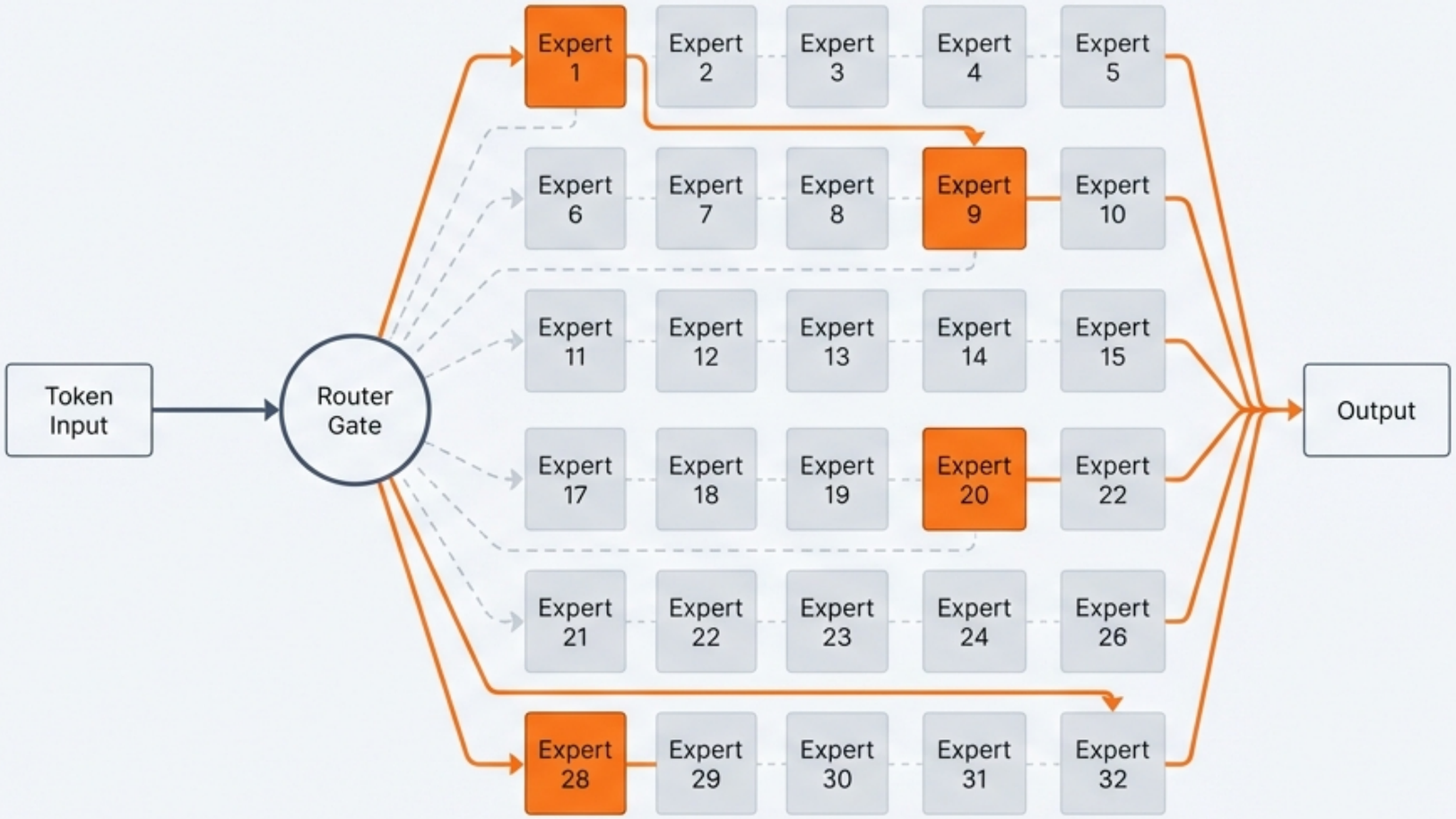
NotebookLM

พลังของความเบาบาง: Mixture of Experts (MoE)

Concept: Sparsity

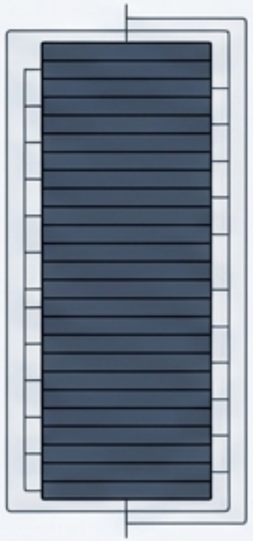
แทนที่ Feed Forward Layer เดี่ยวขนาดใหญ่ ด้วย Experts ย่อยหลายตัว และ ‘Activate’ เฉพาะ Expert ที่เกี่ยวข้องกับ Token นั้นๆ

```
Total Experts: 32
Active per Token: 4
```



NotebookLM

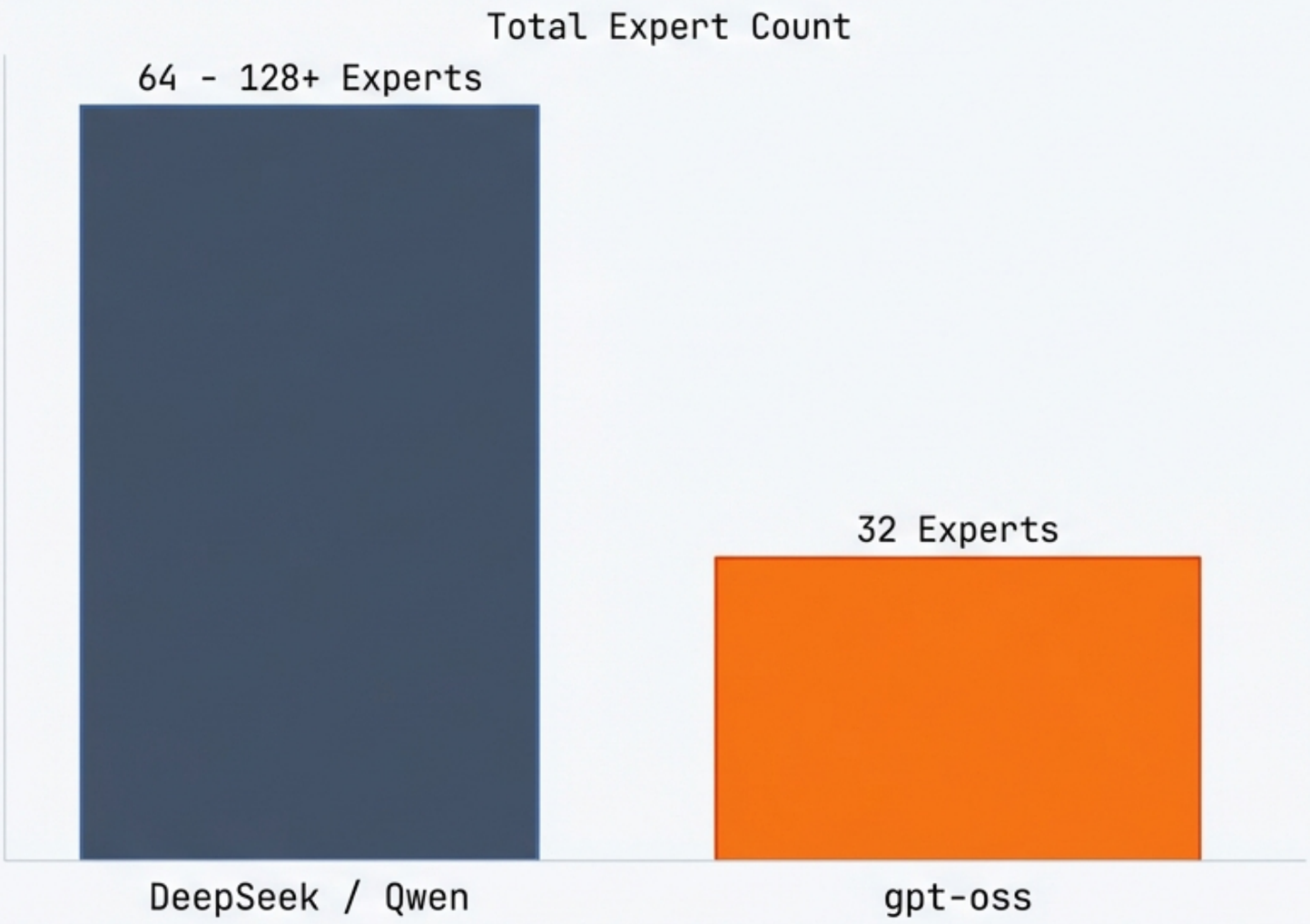
การปะทะเชิงโครงสร้าง: gpt-oss vs. Qwen3

gpt-oss (Wide & Shallow)	Qwen3 (Narrow & Deep)
	
Embedding Dim: 2880 Layers: 24 Blocks Advantage: Parallelization (Inference เร็วกว่า)	Embedding Dim: 2048 Layers: 48 Blocks Advantage: Reasoning Steps (Complexity สูงกว่า)

gpt-oss เลือกความกว้างเพื่อ Speed ในขณะที่ Qwen3 เลือกความลึกเพื่อ Complexity

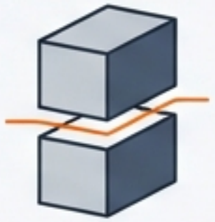
NotebookLM

กลยุทธ์ของ Expert: ขนาดใหญ่จำนวนน้อย vs ขนาดเล็กจำนวนมาก



The Analyst's Hypothesis

โครงสร้างของ gpt-oss-20b อาจเกิดจากการ 'ตัดทอน' (Chopped) มาจากโมเดล 120B เป็นการลดจำนวน Layer และ Expert ลงเพื่อสร้างรุ่นเล็กแทนที่จะออกแบบใหม่ตั้งแต่ต้น



NotebookLM

จุดเด่นทางสถาปัตยกรรมที่ไม่เหมือนใคร (Quirks)

```
1 class Attention(nn.Module):
2     def __init__(self, ...):
3         super().__init__()
4         self.c_attn = nn.Linear(..., bias=True) # <--- Unusual!
5         self.sink_logits = nn.Parameter(...)    # Learned Sinks
```

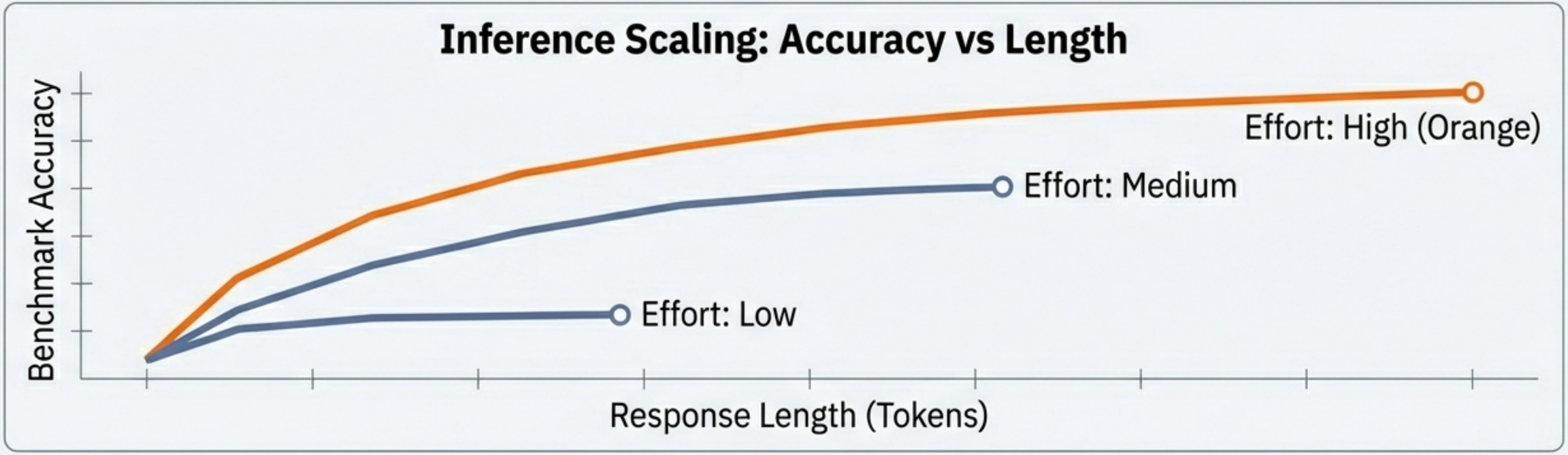
Reintroduction of Attention Bias: นำ Bias units กลับมาใช้ใน Attention layers ซึ่งหายไปตั้งแต่ยุค GPT-2

Attention Sinks Implementation: ใช้ 'Learned Bias Logits' แทน input tokens จริง เพื่อ stabilize context ยาวโดยไม่ต้องแก้ Input sequence

NotebookLM

การเทรนเพื่อ ‘การใช้เหตุผล’ (Reasoning)

Training: 2.1 Million H100 hours	**Focus:** STEM, Coding, General Knowledge

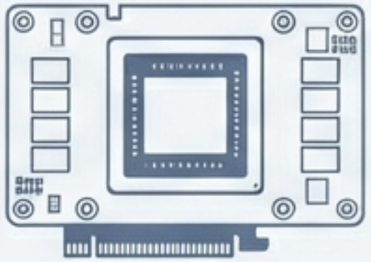


ผู้ใช้สามารถกำหนด `Reasoning effort` ใน System Prompt เพื่อควบคุมความยาวของคำตอบและความลึกของการคิด (Thinking steps)

NotebookLM

การใช้งานจริงบน Hardware: พลังของ MXFP4

การทำ Quantization แบบ MXFP4 ทำให้โมเดลขนาดใหญ่รันบน Hardware ที่เล็กลงได้

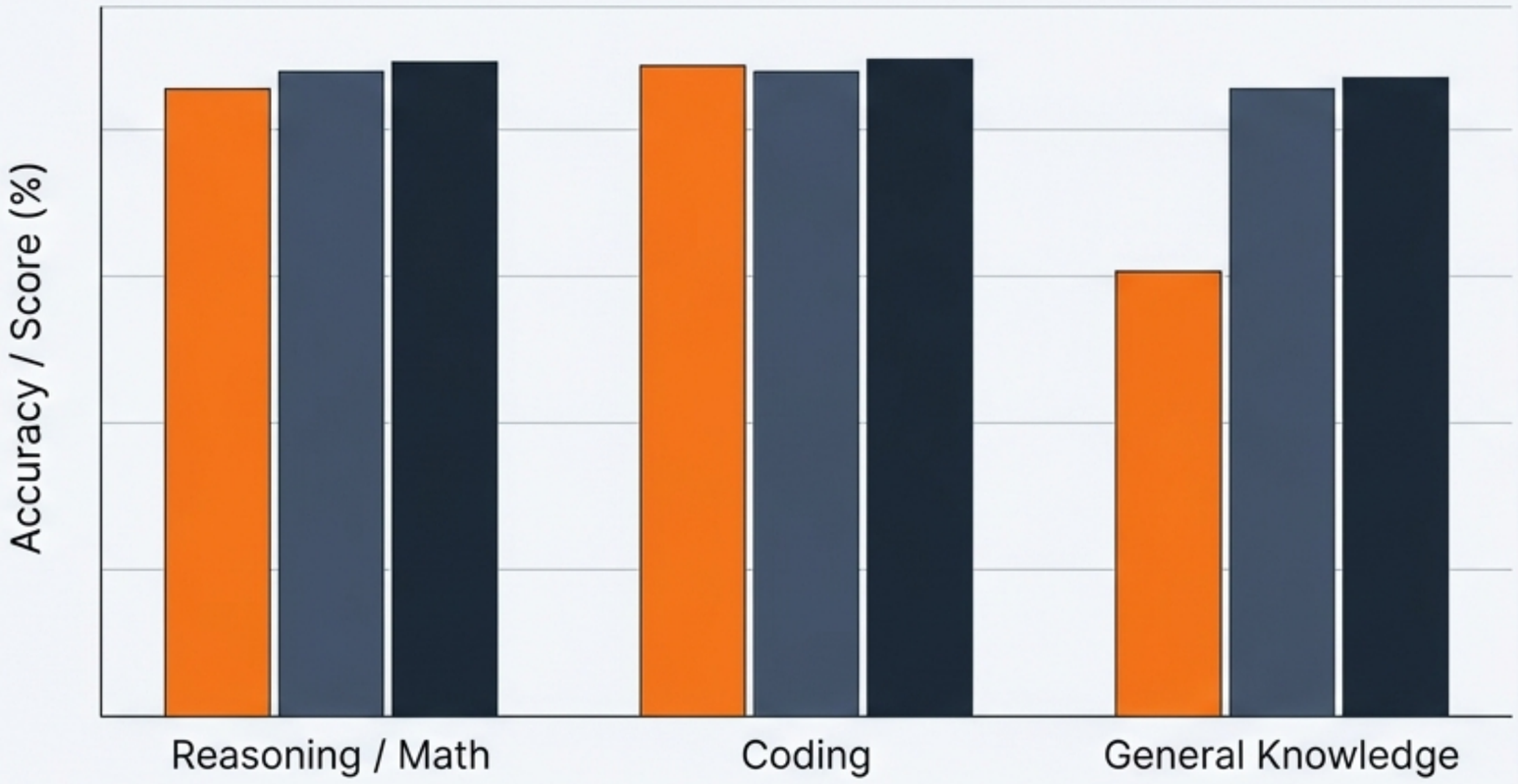
gpt-oss-120b	 NVIDIA H100	**JetBrains Mono** 80GB VRAM Required	ไม่ต้องใช้ Cluster หลายการ์ด ประหยัดค่าเช่า Server
gpt-oss-20b	RTX 50-Series / Patched 4090	**JetBrains Mono** ~16GB VRAM Required	รันได้แม้บน Consumer Hardware

Legacy Hardware: หากไม่รองรับ MXFP4 ต้องใช้ VRAM สูงถึง 48GB สำหรับรุ่น 20b

NotebookLM

ผลการทดสอบ (Benchmarks) และข้อควรระวัง

Model Performance Comparison



Key Insights

Strengths:
ประสิทธิภาพด้าน Reasoning ใกล้เข้าเคียงกับ GPT-5 และ Qwen3 โดยเฉพาะงาน Coding

Weakness (General Knowledge Forgetting):
ความเสี่ยงสูงเรื่อง Hallucination ในคำถามความรู้ทั่วไป เนื่องจากการเทรนเน้นหนักไปทาง STEM

NotebookLM

ภูมิทัศน์ AI ปี 2025: gpt-oss เทียบกับ GPT-5

Context

OpenAI ปล่อย GPT-5 ในสัปดาห์เดียวกับ gpt-oss

Analysis

ผลทดสอบแสดงให้เห็นว่าโมเดล Open-Weights มีประสิทธิภาพไล่จี้โมเดลปิด (Closed Models) มากขึ้นเรื่อยๆ

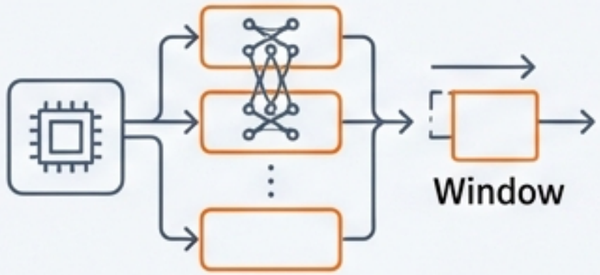
Use Case

gpt-oss ให้ความยืดหยุ่นในการนำไปต่อยอด (Distillation/Fine-tuning) สำหรับนักวิจัยที่ไม่ต้องการพึ่งพา API

NotebookLM

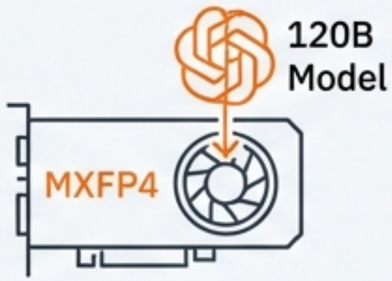
บทสรุป: มาตรฐานใหม่สำหรับการวิจัย AI

1. Architecture



MoE และ Sliding Window คือมาตรฐานใหม่ของความคุ้มค่า **(Efficiency)**

2. Accessibility



MXFP4 เปิดประตูให้รันโมเดลระดับ 120B ได้บนการ์ดใบเดียว

3. Future Direction

เทรนด์ปี 2025 คือ **'Reasoning First'** — คิดได้ลึกซึ้งขึ้น แม้แลกกับการจำความรู้ทั่วไปบ้าง

---

gpt-oss ไม่ใช่แค่โมเดล แต่เป็นพิมพ์เขียว (Blueprint) ที่สำคัญสำหรับการศึกษาและพัฒนาต่อยอดในยุค Open-Weights 2.0

NotebookLM