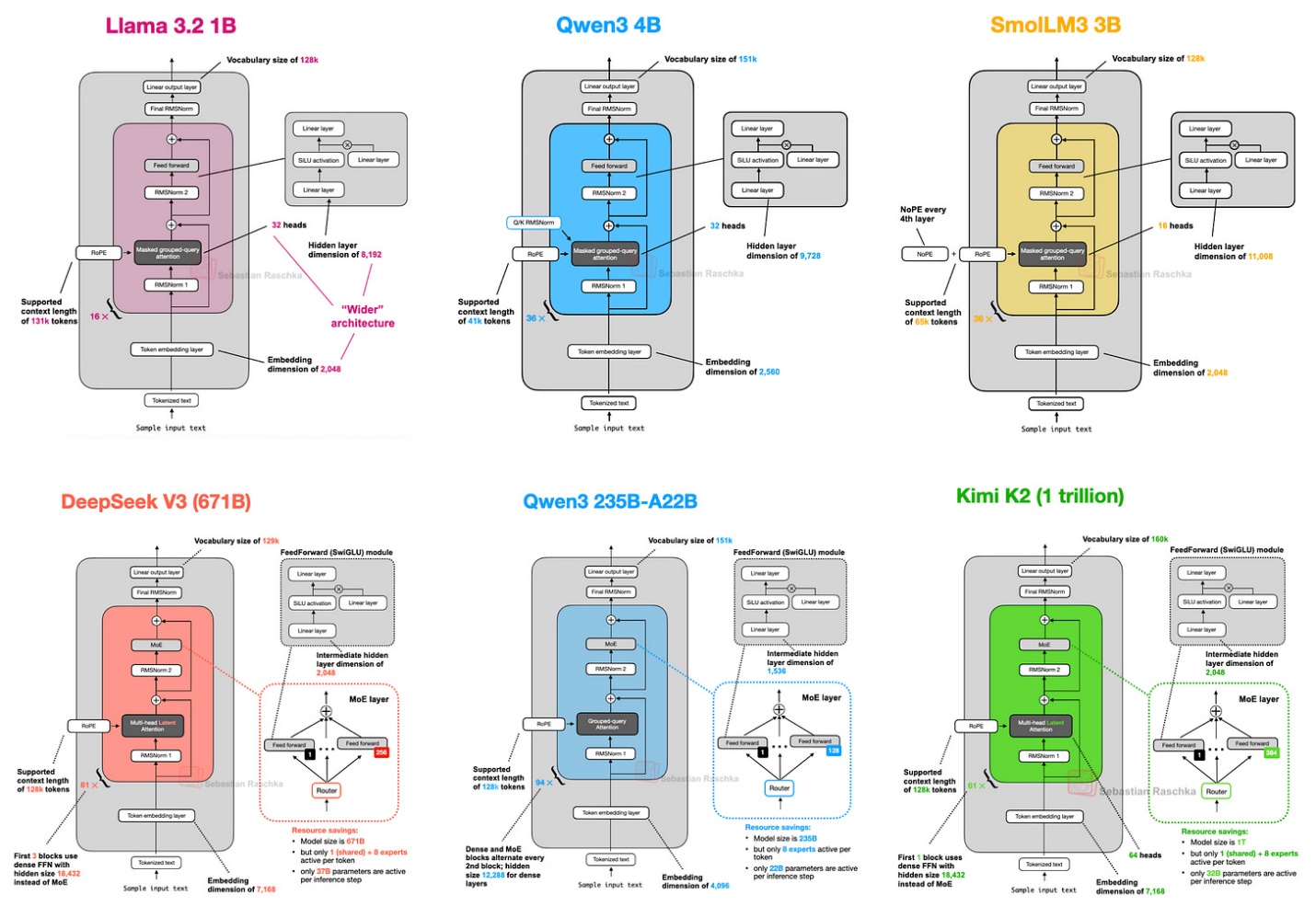
**📚 Comprehensive Study Notes: Modern LLM Architecture Comparison (2025)**

Based on Sebastian Raschka's comprehensive analysis of contemporary Large Language Model architectures, this document provides detailed study notes covering 12 major LLM architectures released in 2024-2025.

**🎯 Overview: The Evolution of LLM Architecture**



*Figure 1: A subset of the architectures covered - showing the diversity of modern LLM designs*

Seven years after the original GPT architecture, modern LLMs maintain structural similarities while introducing key optimizations:

* **Positional Embeddings**: Evolution from absolute → rotational (RoPE) → sometimes none (NoPE)
* **Attention Mechanisms**: Multi-Head Attention → Grouped-Query Attention → Multi-Head Latent Attention
* **Activation Functions**: GELU → SwiGLU for improved efficiency
* **Architecture Patterns**: Dense models → Mixture-of-Experts (MoE) for sparse computation

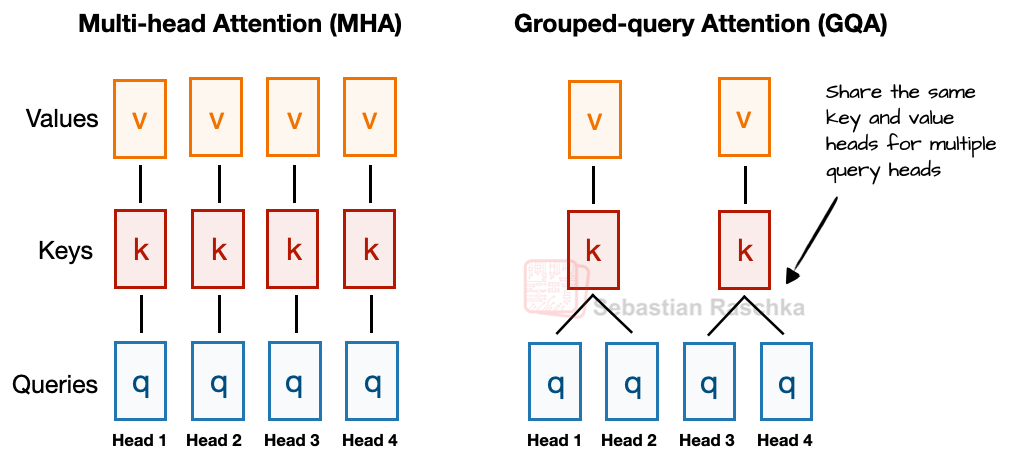
**🏗️ Architecture Deep Dive**

**1. DeepSeek V3/R1 - Multi-Head Latent Attention & MoE Pioneer**

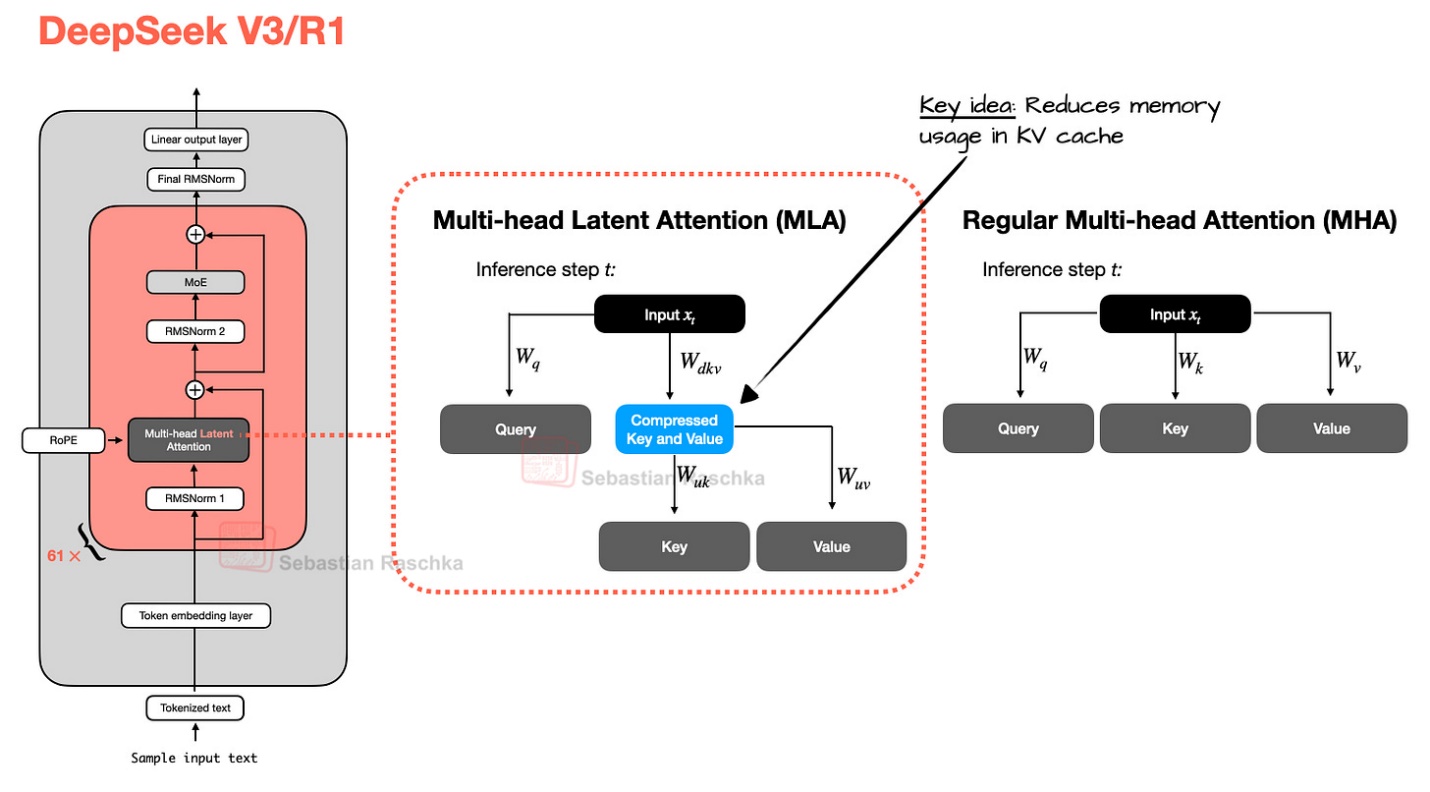
**Key Specifications:**

* **Total Parameters**: 671 billion
* **Active Parameters**: 37 billion (during inference)
* **Architecture Type**: MoE with Multi-Head Latent Attention

**Multi-Head Latent Attention (MLA)**



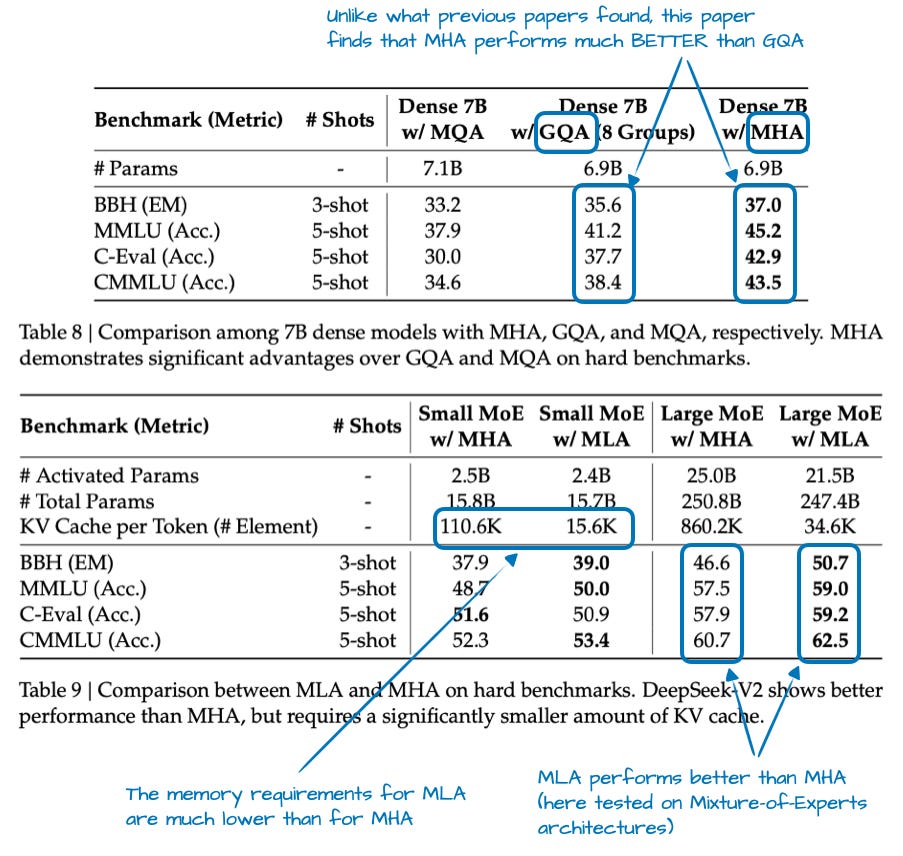
*Figure 2: Multi-Head Attention vs Grouped-Query Attention comparison*



*Figure 3: MLA mechanism - compressing K/V tensors before KV cache storage*

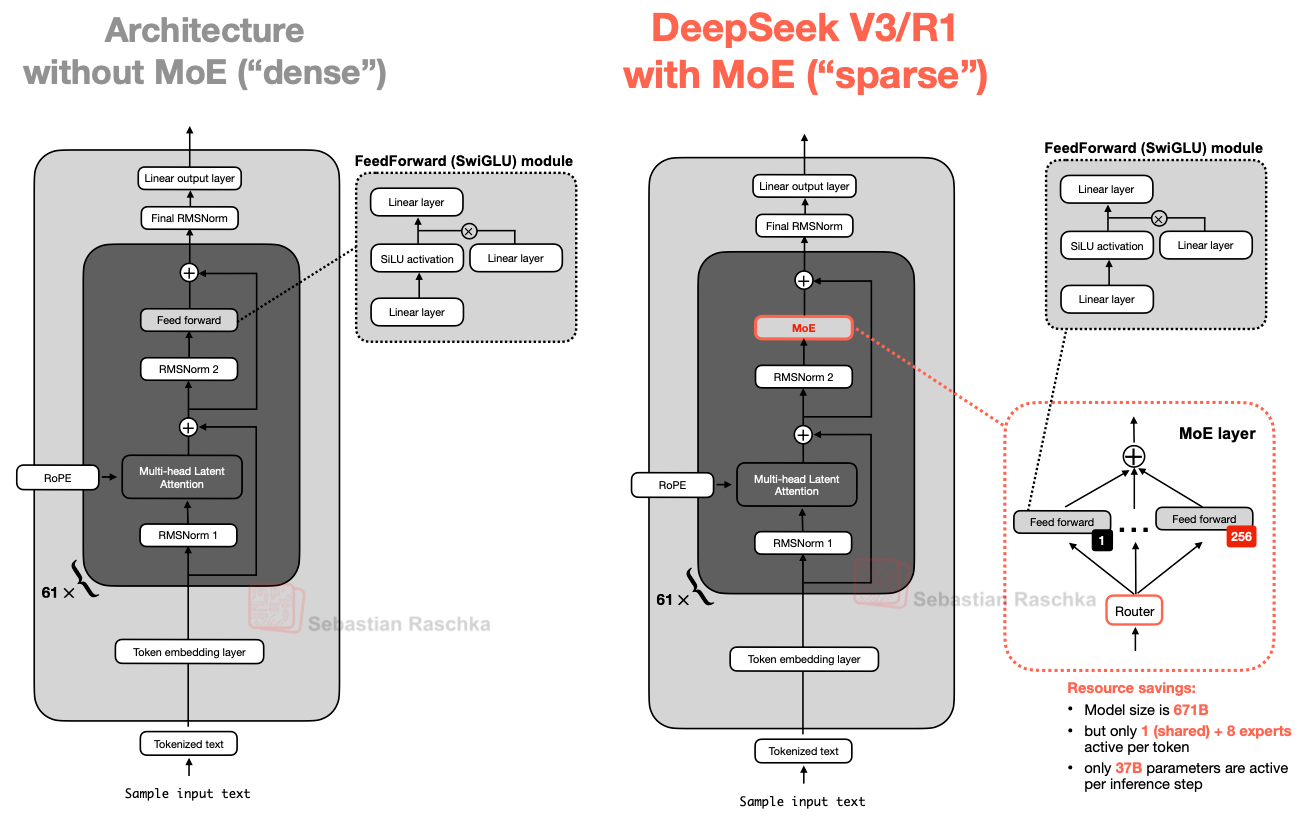
**MLA Innovation:**

* **Memory Compression**: Compresses key/value tensors into lower-dimensional space before KV cache storage
* **Runtime Projection**: Projects compressed tensors back to original size during inference
* **Trade-off**: Extra matrix multiplication vs. significant memory savings
* **Performance**: Outperforms both MHA and GQA in modeling performance



*Figure 4: Ablation study showing MLA's superior performance over GQA*

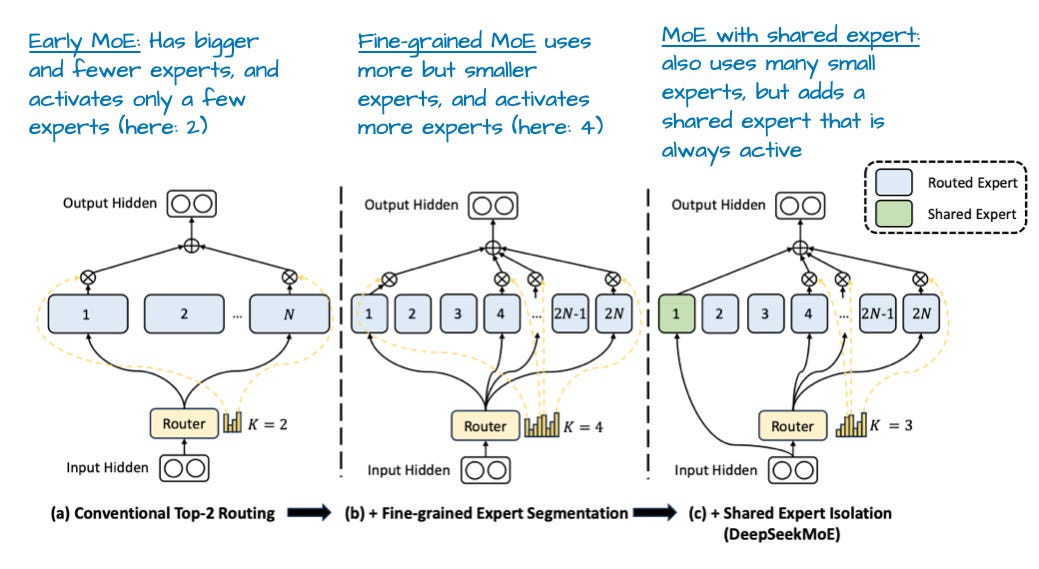
**Mixture-of-Experts (MoE) Architecture**



*Figure 5: MoE architecture comparison - sparse vs dense FeedForward blocks*

**MoE Specifications:**

* **Total Experts**: 256 per MoE module
* **Active Experts**: 9 per token (1 shared + 8 router-selected)
* **Shared Expert**: Always-active expert for common patterns
* **Efficiency**: Only 5.5% of parameters active during inference



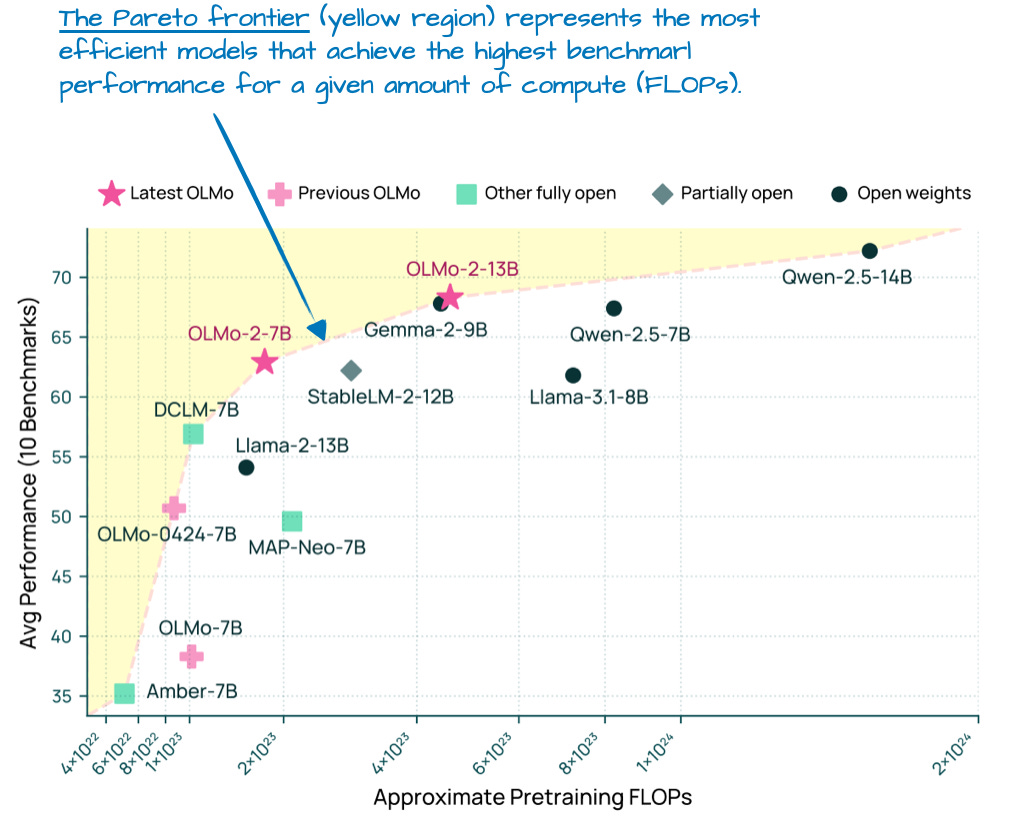
*Figure 6: Shared expert architecture and performance benefits*

**Design Trade-offs:**

* ✅ Massive capacity with efficient inference
* ✅ Superior memory efficiency via MLA
* ❌ Complex implementation
* ❌ Requires specialized hardware optimization

**2. OLMo 2 - Normalization Innovation**

**Key Focus**: Transparency and normalization layer optimization



*Figure 7: OLMo 2's position on the Pareto frontier of performance vs training cost*

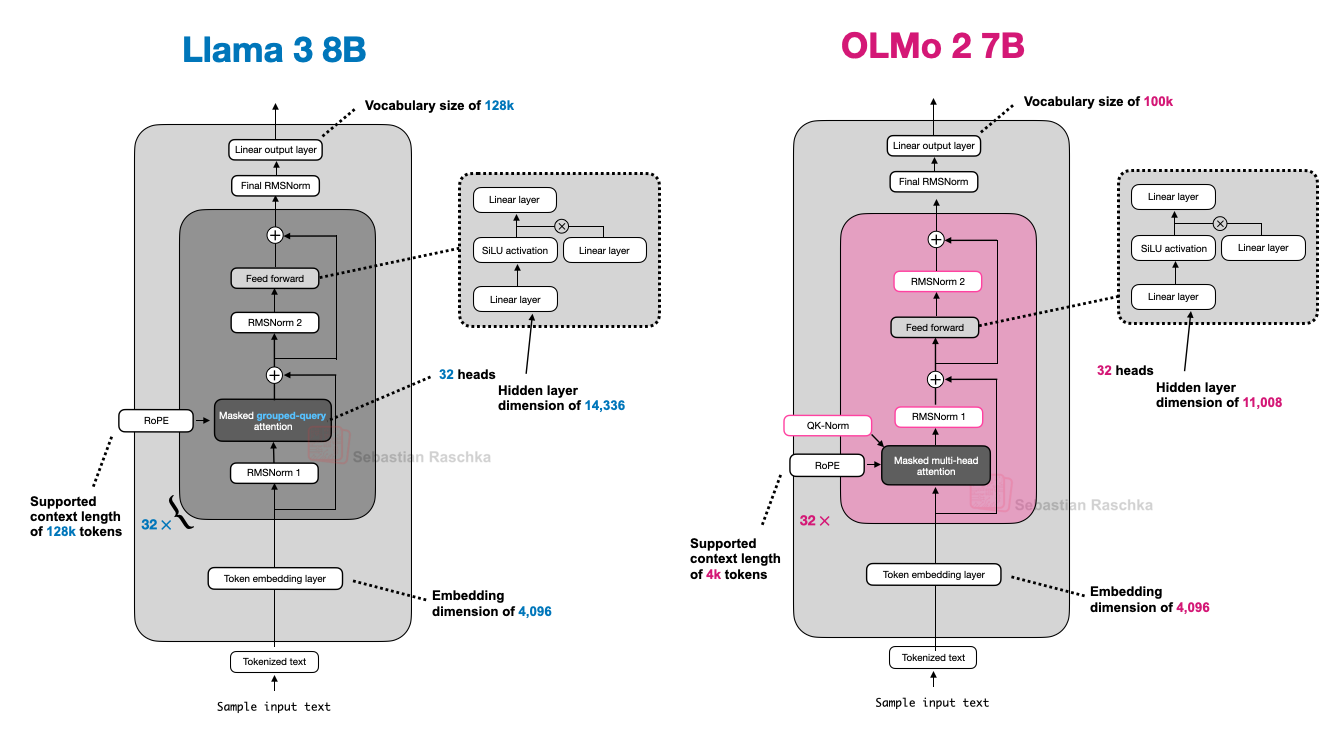
**Normalization Placement Strategy**



*Figure 8: Post-Norm vs Pre-Norm vs OLMo 2's Post-Norm variant*

**Normalization Innovations:**

1. **Post-Norm Placement**: RMSNorm after (not before) attention/FeedForward modules
2. **QK-Norm**: Additional RMSNorm applied to queries and keys before RoPE
3. **Training Stability**: Combined approach significantly improves gradient behavior



*Figure 10: OLMo 2 vs Llama 3 architecture comparison*

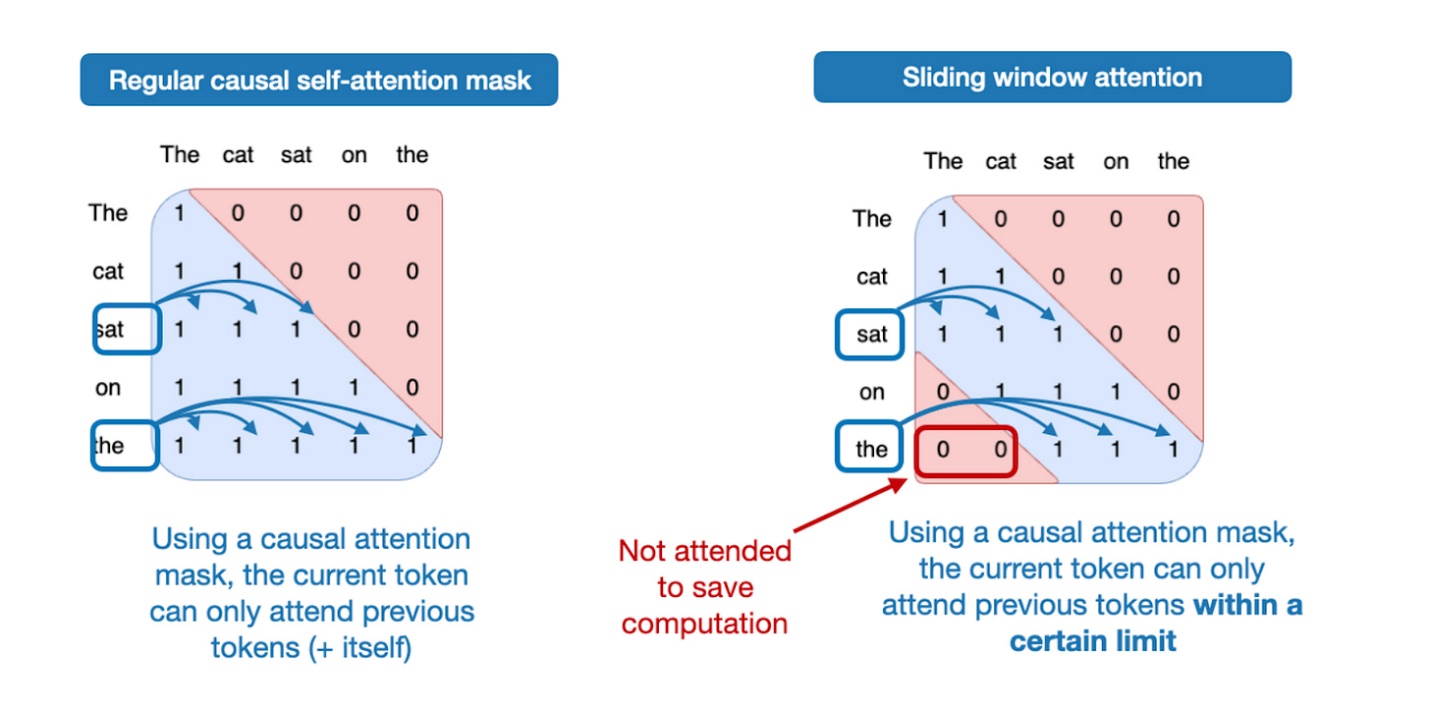
**Design Philosophy:**

* ✅ Full transparency (training data, code, techniques)
* ✅ Excellent training stability
* ✅ Blueprint for LLM development
* ❌ Uses traditional MHA (not GQA/MLA)
* ❌ Not at top of performance leaderboards

**3. Gemma 3 - Sliding Window Attention Master**

**Key Innovation**: Efficient sliding window attention with optimal ratio tuning

**Sliding Window Attention Mechanism**

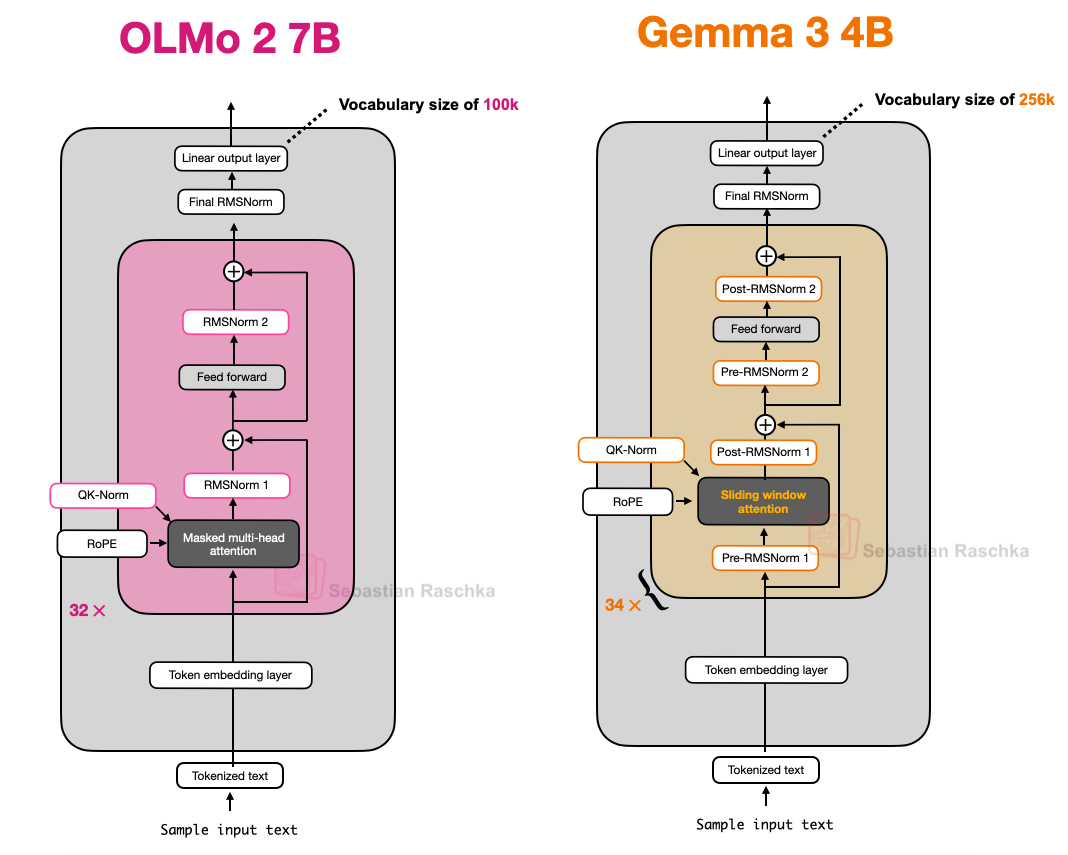


*Figure 12: Global attention vs sliding window (local) attention*

**Sliding Window Specifications:**

* **Ratio**: 5:1 (local:global attention layers)
* **Window Size**: 1024 tokens (reduced from Gemma 2's 4096)
* **Memory Savings**: Substantial KV cache reduction
* **Performance Impact**: Minimal degradation in modeling performance

**Dual Normalization Strategy**

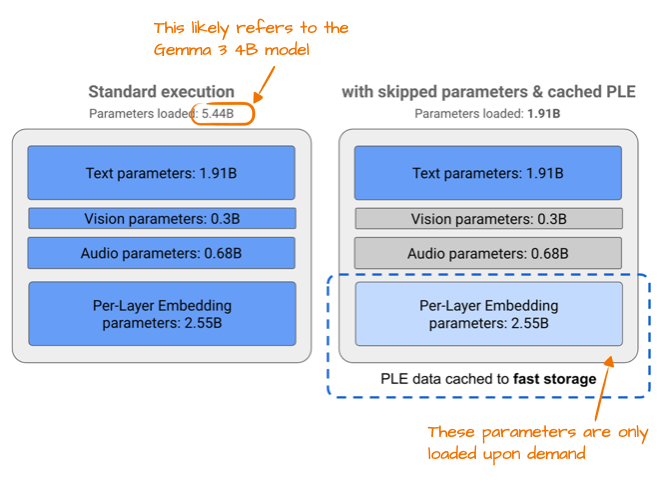


*Figure 14: Gemma 3's unique Pre-Norm + Post-Norm approach*

**Design Highlights:**

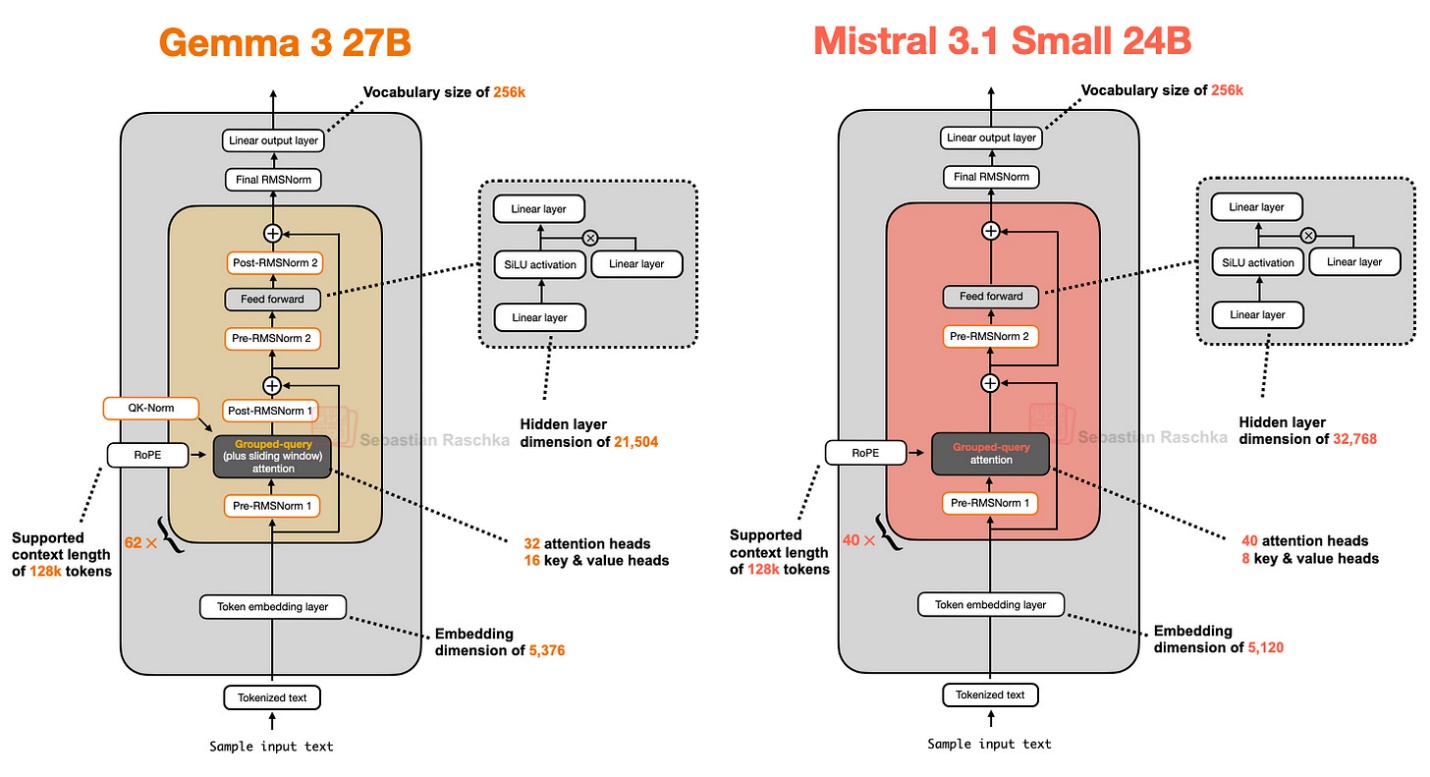
* ✅ Excellent 27B size sweet spot
* ✅ Strong multilingual support (large vocabulary)
* ✅ Efficient memory usage
* ✅ Robust normalization strategy
* ❌ Somewhat underappreciated in community

**Gemma 3n - Mobile Optimization**



*Figure 15: Per-Layer Embedding (PLE) memory savings for mobile deployment*

**4. Mistral Small 3.1 - Speed Optimization**



*Figure 16: Mistral 3.1 Small vs Gemma 3 architecture*

**Performance Focus:**

* **Size**: 24B parameters (vs Gemma 3's 27B)
* **Speed**: Faster inference than Gemma 3
* **Benchmark**: Outperforms Gemma 3 27B (except math)

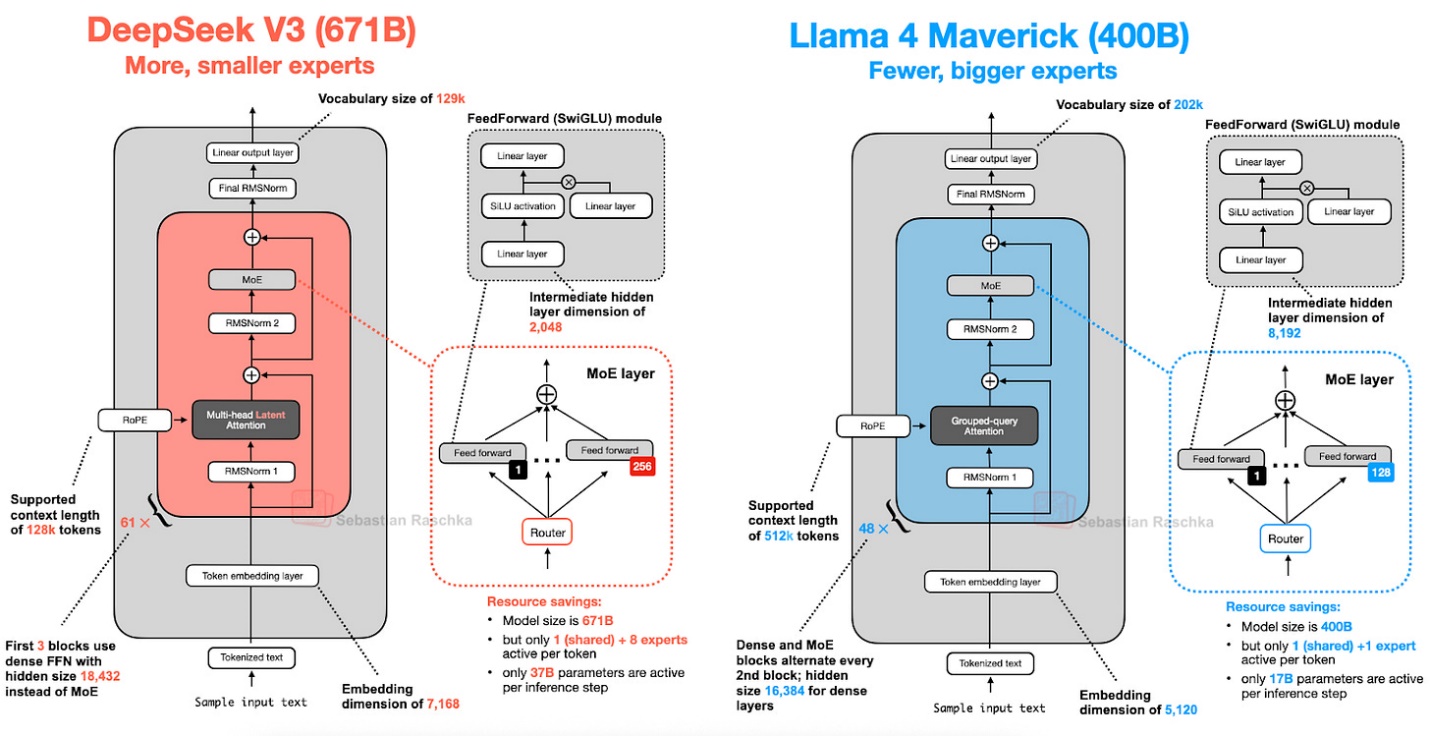
**Speed Optimizations:**

1. **Custom Tokenizer**: Optimized for efficiency
2. **Reduced KV Cache**: Smaller memory footprint
3. **Fewer Layers**: Streamlined architecture
4. **Standard GQA**: Abandoned sliding window for FlashAttention compatibility

**Design Trade-offs:**

* ✅ Superior speed-to-performance ratio
* ✅ Optimized for production deployment
* ❌ Sacrificed some memory efficiency for speed
* ❌ Less innovative architecturally

**5. Llama 4 Maverick - Meta's MoE Entry**



*Figure 17: Llama 4 Maverick vs DeepSeek V3 architecture comparison*

**Specifications:**

* **Total Parameters**: 400 billion
* **Active Parameters**: 17 billion
* **MoE Strategy**: Alternating MoE/dense blocks

**Key Differences from DeepSeek V3:**

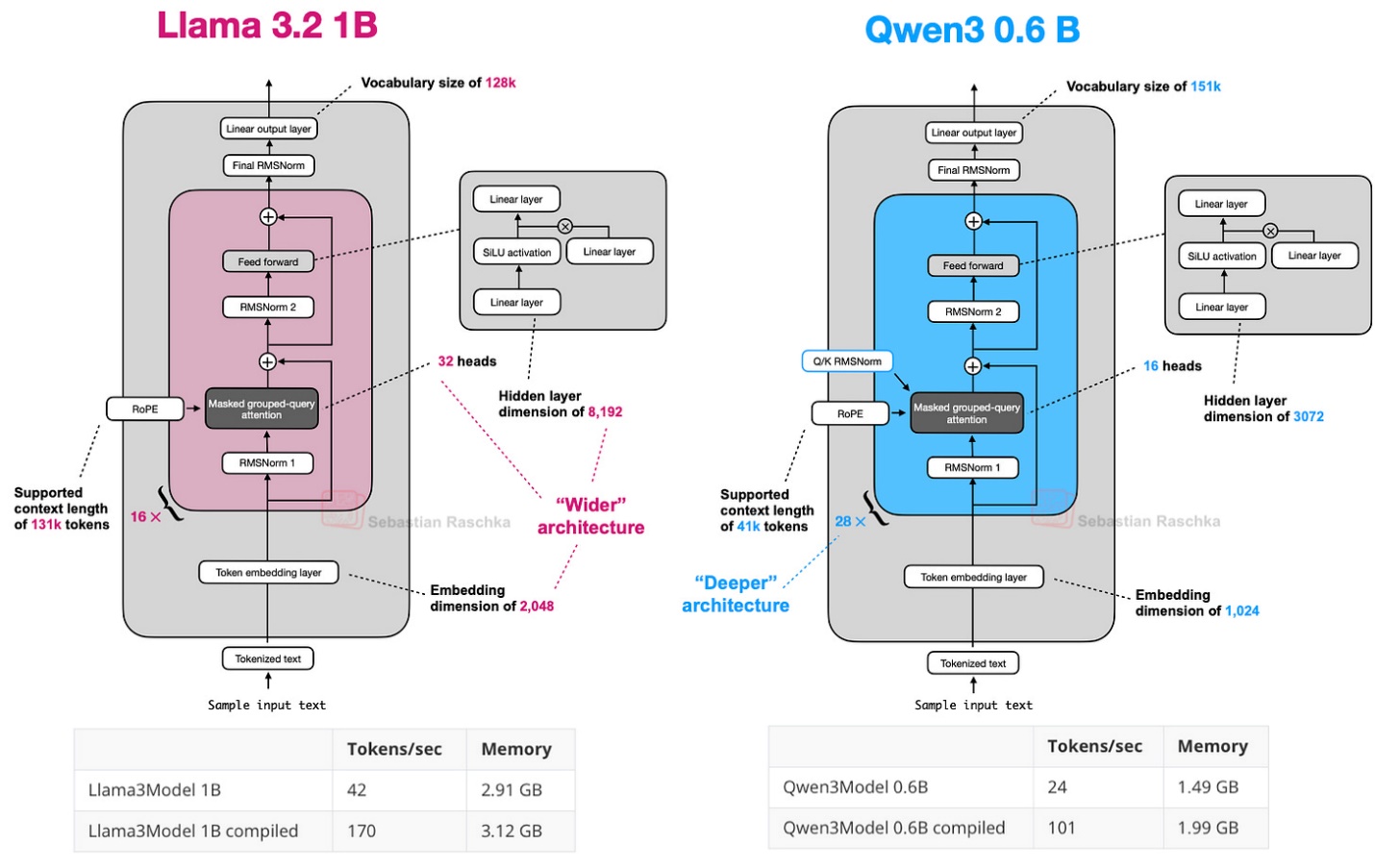
1. **Attention**: Standard GQA (not MLA)
2. **Expert Design**: Fewer, larger experts (2 active vs 9)
3. **Block Layout**: Alternating MoE/dense vs MoE everywhere
4. **Expert Size**: 8,192 hidden vs 2,048 in DeepSeek

**Design Philosophy:**

* ✅ Proven GQA reliability
* ✅ Balanced MoE/dense approach
* ❌ Less memory efficient than MLA
* ❌ Fewer active parameters for given size

**6. Qwen3 - Flexible Dense/MoE Options**

**Dense Model Architecture**

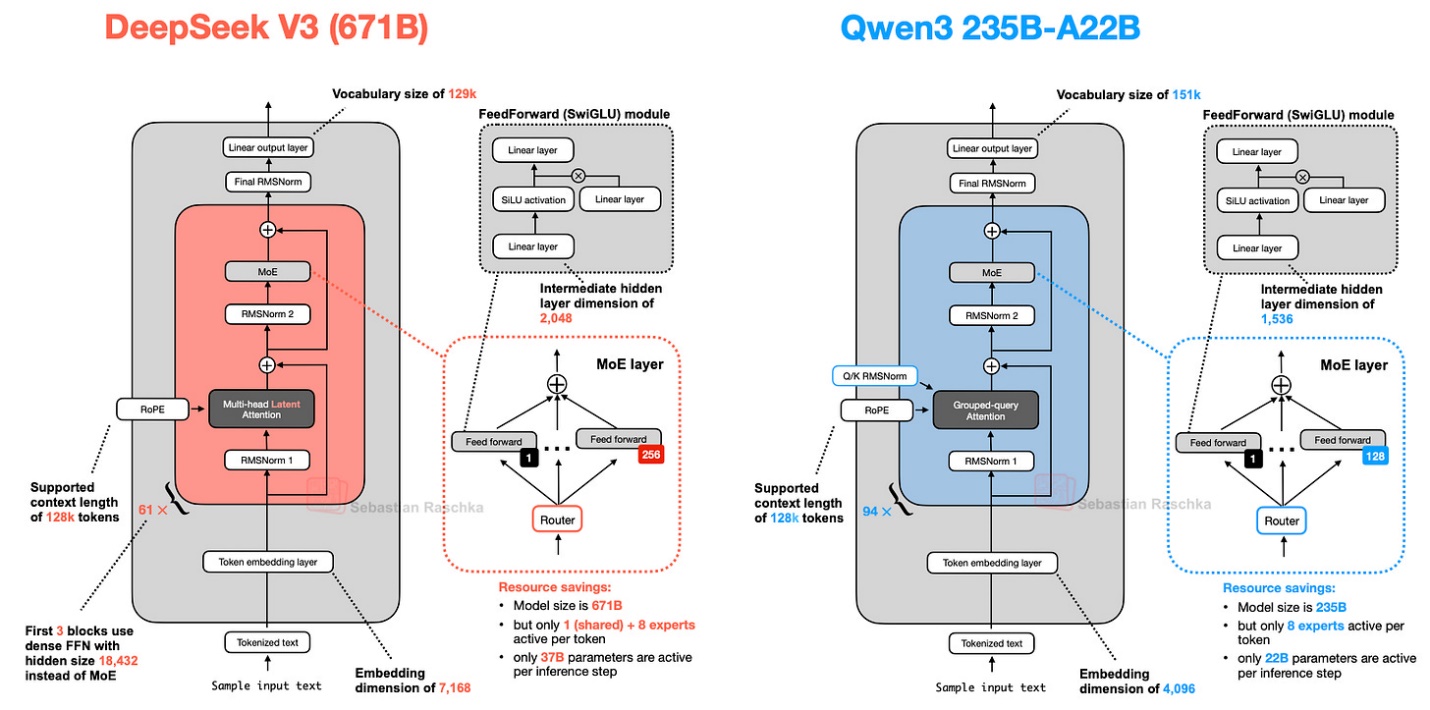


*Figure 18: Qwen3 0.6B vs Llama 3 1B - depth vs width comparison*

**Model Variants:**

* **Dense Models**: 0.6B, 1.7B, 4B, 8B, 14B, 32B
* **MoE Models**: 30B-A3B, 235B-A22B
* **Architecture Strategy**: Deeper (more layers) vs wider (more heads)

**MoE Architecture Comparison**



*Figure 19: Qwen3 235B-A22B vs DeepSeek V3 MoE comparison*

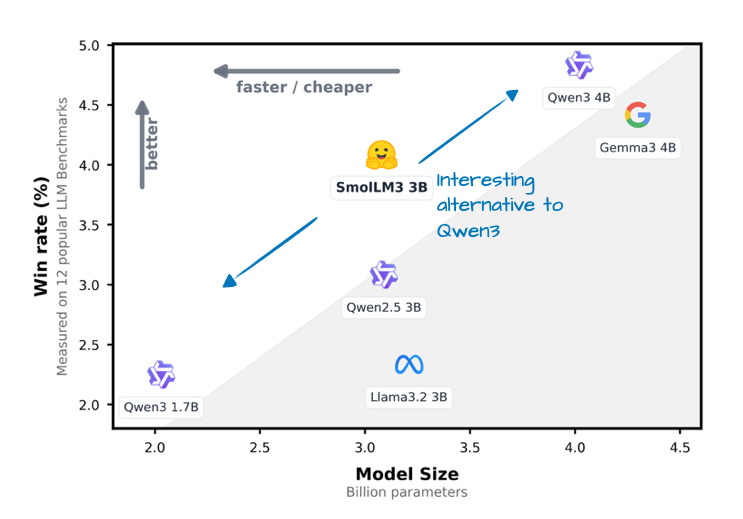
**Notable Design Choice:**

* **No Shared Expert**: Unlike DeepSeek V3, Qwen3 MoE abandoned shared experts
* **Reasoning**: Possibly unnecessary with 8 experts vs complexity reduction

**Use Case Strategy:**

* **Dense Models**: Fine-tuning, local deployment, educational use
* **MoE Models**: Large-scale inference, production serving
* ✅ Flexibility for different use cases
* ✅ Excellent small model performance (0.6B)
* ✅ Consistent quality across sizes

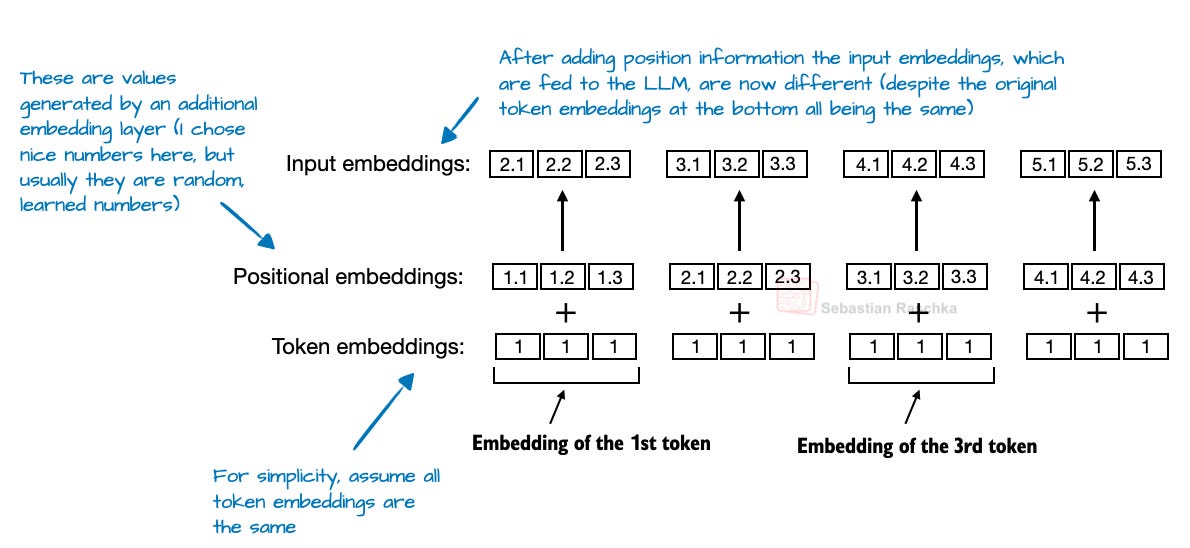
**7. SmolLM3 - No Positional Embeddings (NoPE)**



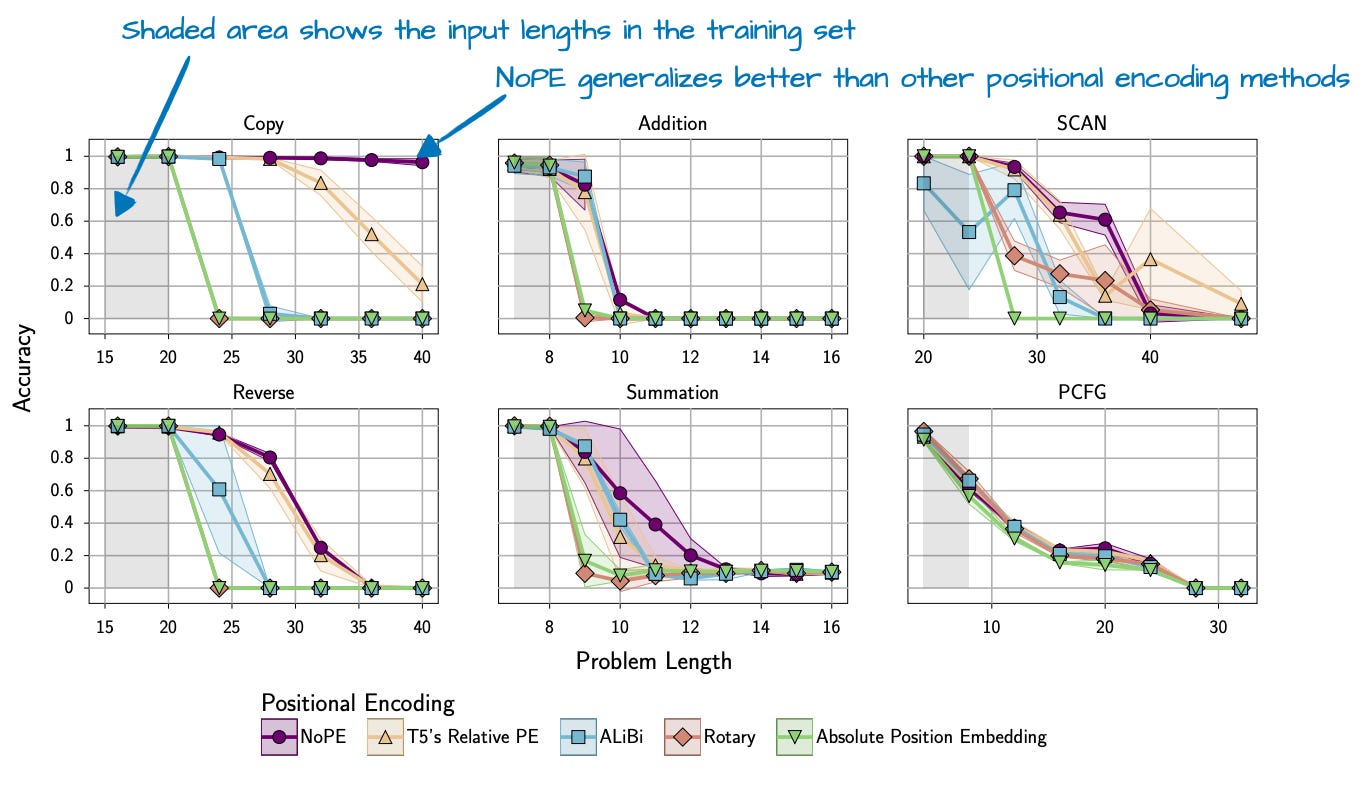
*Figure 20: SmolLM3 win rate against comparable models*

*Figure 21: SmolLM3 vs Qwen3 4B architecture*

**NoPE (No Positional Embeddings) Innovation**



*Figure 22: Traditional absolute positional embeddings (abandoned in NoPE)*



*Figure 23: NoPE's superior length generalization performance*

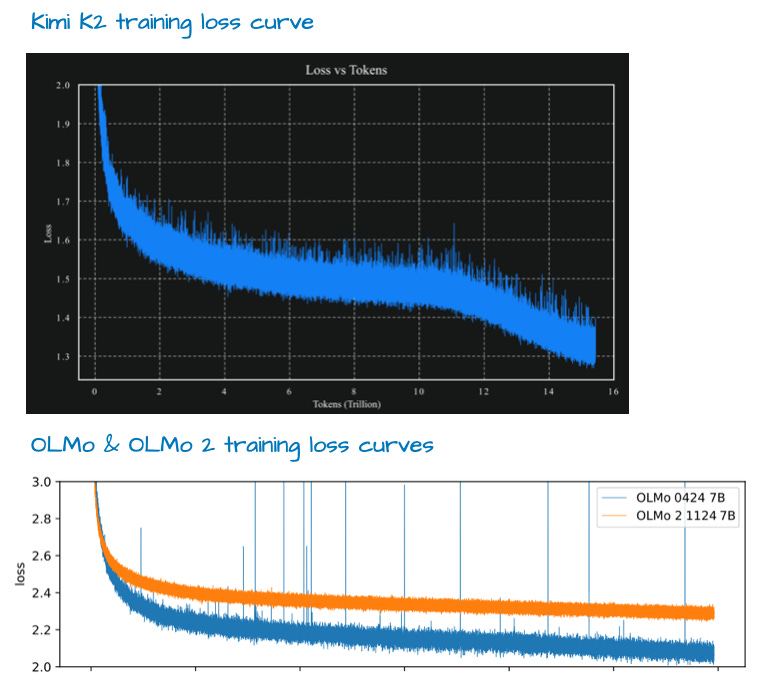
**NoPE Mechanism:**

* **No Explicit Positional Info**: Removes RoPE, absolute embeddings
* **Causal Masking**: Relies solely on attention mask for sequence order
* **Partial Implementation**: Used in every 4th layer (not everywhere)
* **Benefits**: Better length generalization, simpler architecture

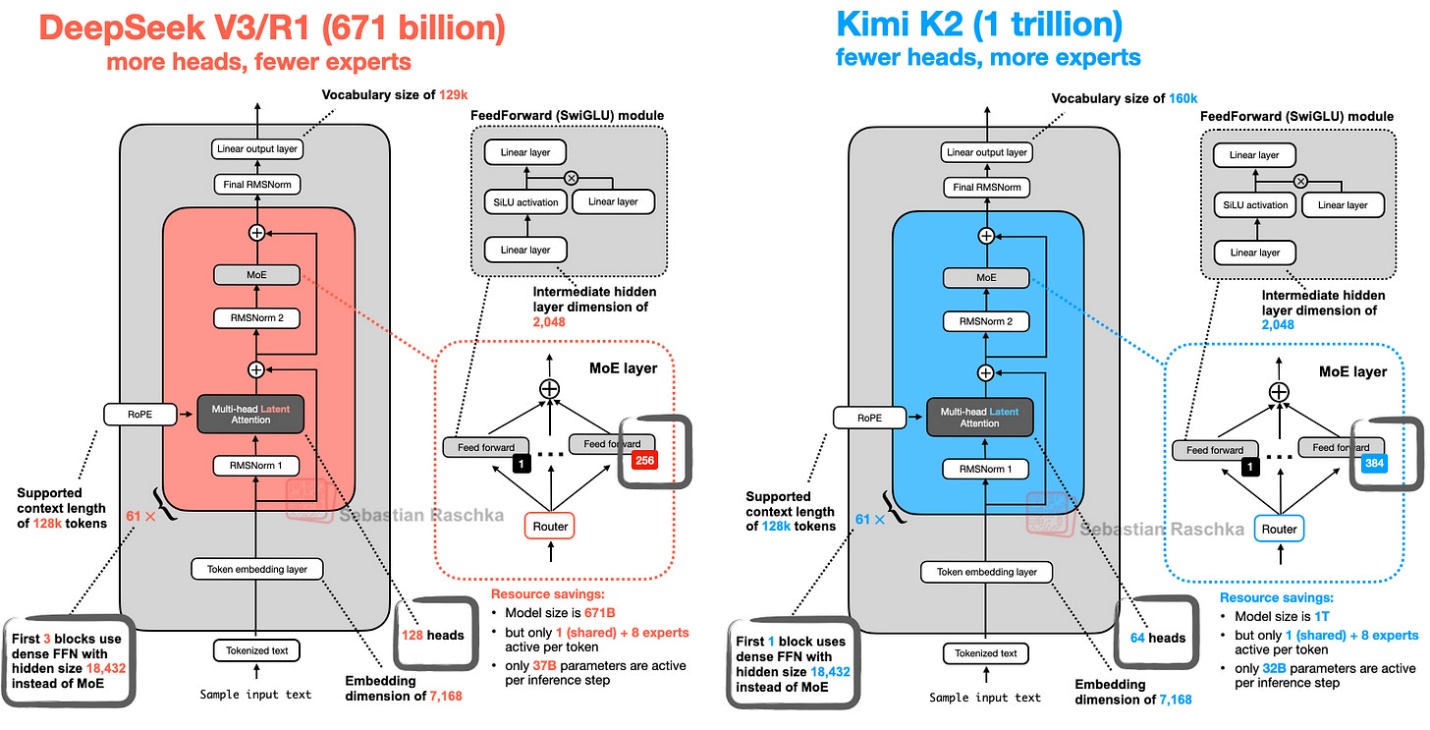
**Design Trade-offs:**

* ✅ Excellent performance for 3B parameters
* ✅ Better length generalization
* ✅ Simplified architecture
* ❌ Limited to smaller models (scaling unclear)
* ❌ Conservative partial implementation

**8. Kimi 2 - Trillion Parameter Scale**



*Figure 24: Kimi K2's smooth training loss curve vs OLMo 2*



*Figure 25: Kimi K2 vs DeepSeek V3 scaled architecture*

**Scale Achievements:**

* **Size**: 1 trillion parameters (largest current-generation open model)
* **Base Architecture**: DeepSeek V3 scaled up
* **Training**: Smooth loss curves with Muon optimizer
* **Performance**: Top-tier benchmark results

**Scaling Modifications:**

* **More Experts**: Increased expert count per MoE module
* **Fewer Heads**: Reduced MLA heads for efficiency
* **Same Principles**: MLA + MoE foundation maintained

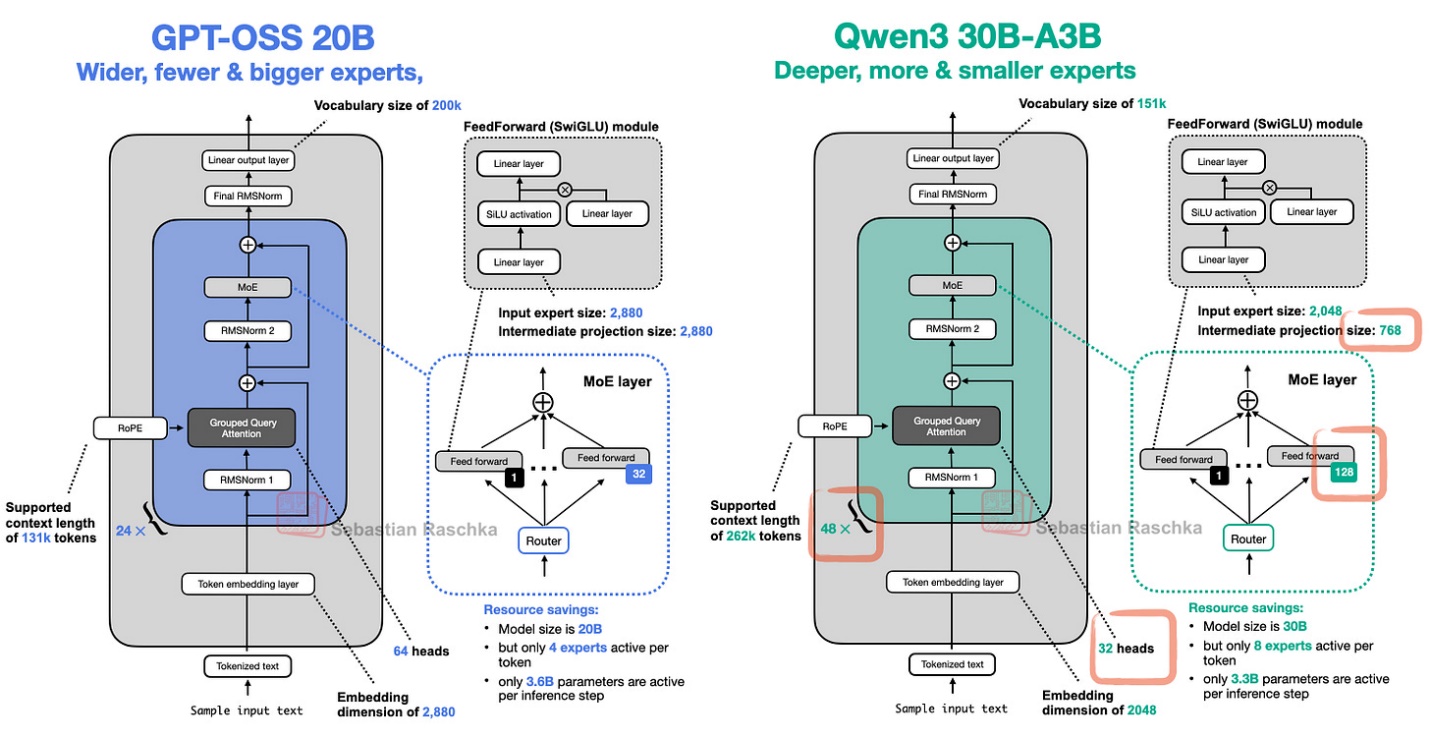
**Significance:**

* ✅ Demonstrates extreme scaling viability
* ✅ Excellent benchmark performance
* ✅ Open-weight accessibility
* ❌ Massive computational requirements
* ❌ Limited accessibility for most users

**9. GPT-OSS - OpenAI's Return to Open Source**



*Figure 26: GPT-OSS architecture overview (20B and 120B variants)*



*Figure 27: GPT-OSS vs Qwen3 MoE architecture comparison*

**Width vs Depth Philosophy**



*Figure 28: Evolution toward more, smaller experts (GPT-OSS uses fewer, larger)*

**Architectural Decisions:**

* **Width Over Depth**: 24 layers vs Qwen3's 48
* **Larger Embedding**: 2880 vs 2048 dimensions
* **Fewer, Larger Experts**: 32 experts vs 128 in Qwen3
* **Expert Count**: 4 active vs 8 in Qwen3

**Unique Features**



*Figure 31: Attention sinks implementation in GPT-OSS*

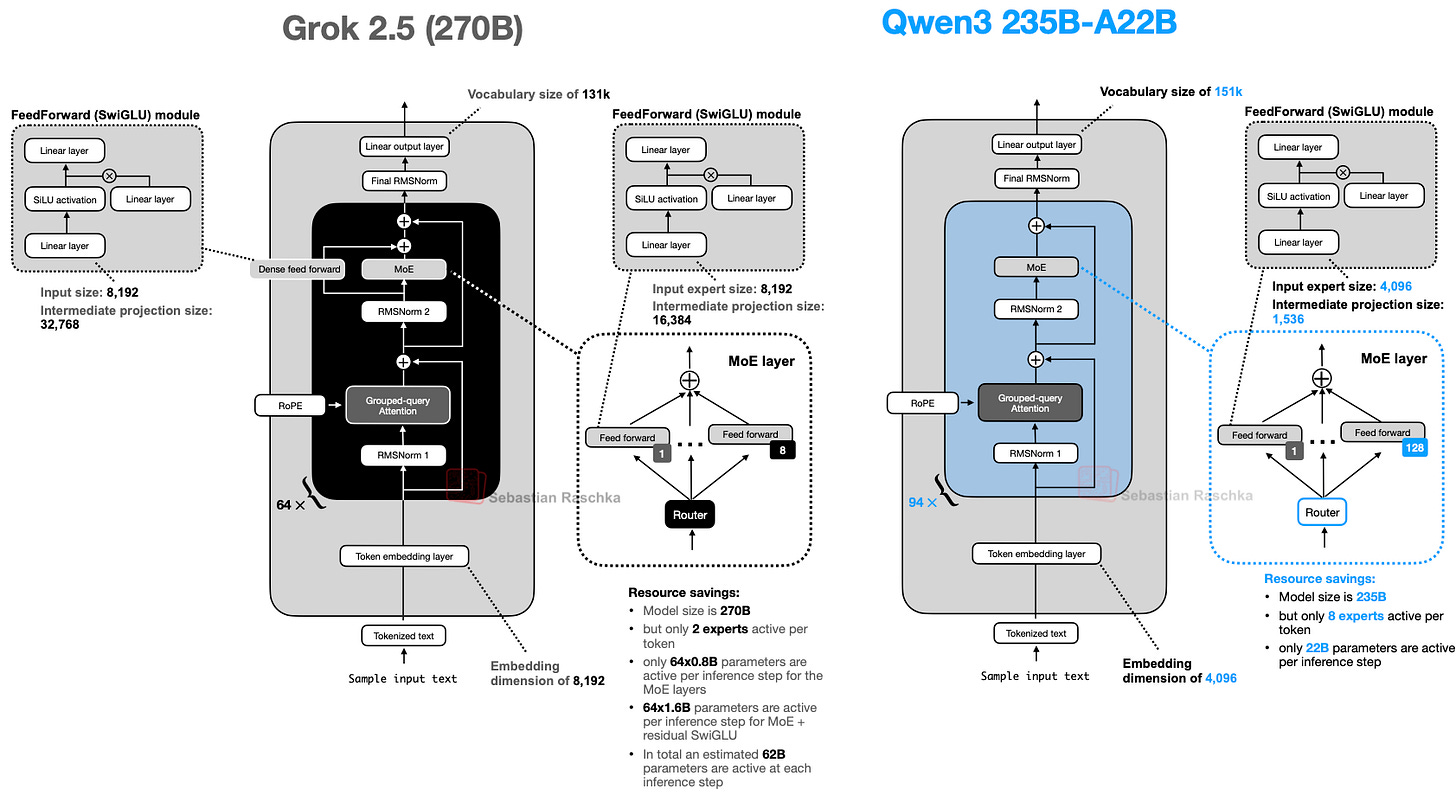
**Special Mechanisms:**

1. **Attention Bias**: Rare bias units in attention layers
2. **Attention Sinks**: Learned bias logits for long context stability
3. **Sliding Window**: Every other layer (vs Gemma 3's 5:1)

**Design Trade-offs:**

* ✅ Higher inference throughput (width advantage)
* ✅ OpenAI's proven design principles
* ✅ Novel attention sink mechanism
* ❌ Higher memory requirements
* ❌ Contrarian expert design (fewer, larger vs trend)

**10. Grok 2.5 - Production Model Insights**



*Figure 32: Grok 2.5 architecture compared to Qwen3*

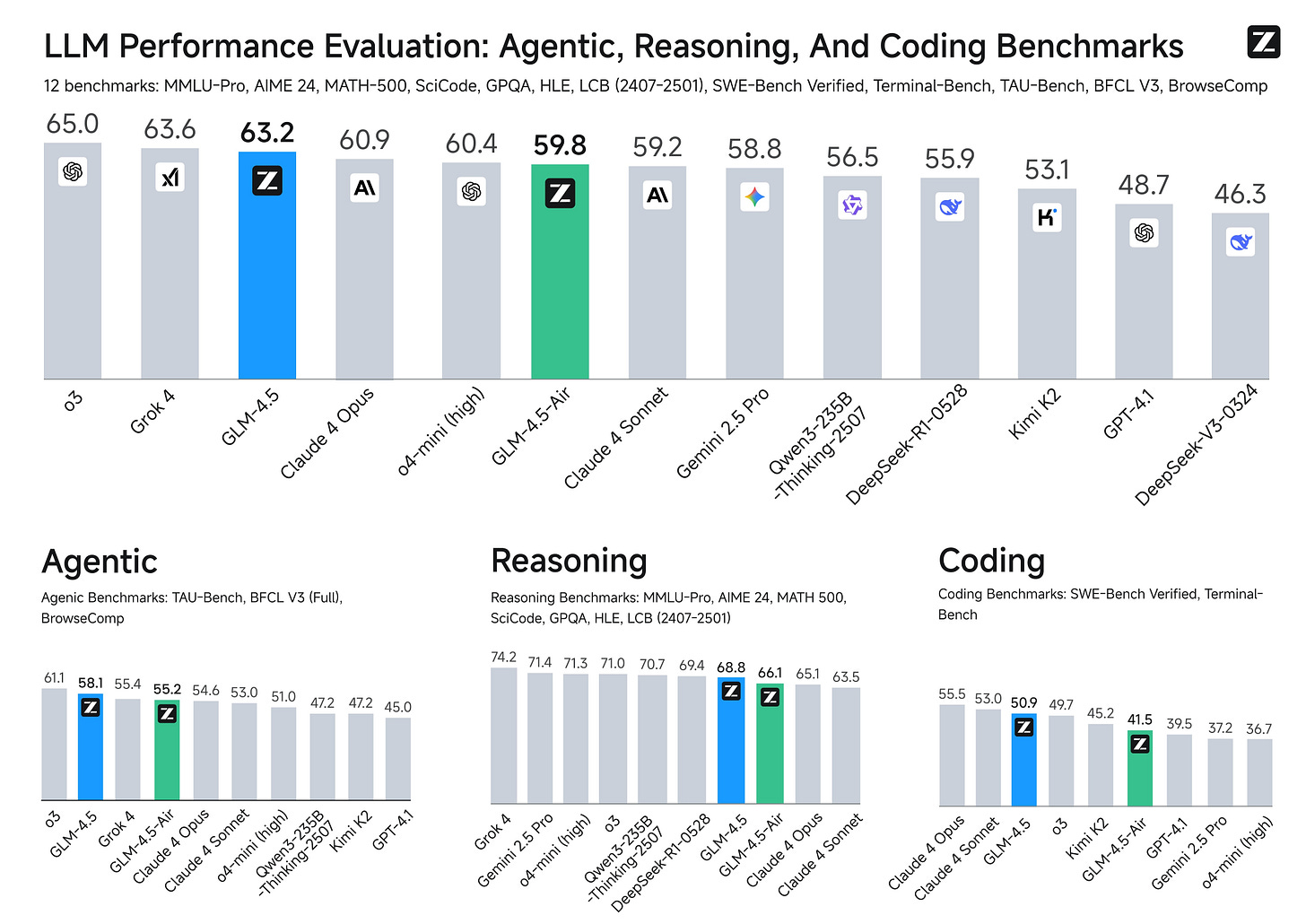
**Production Model Characteristics:**

* **Size**: 270B parameters
* **Expert Strategy**: 8 large experts (older MoE trend)
* **Shared Expert**: SwiGLU-based always-active expert
* **Real-World Usage**: Actual production system (not research model)

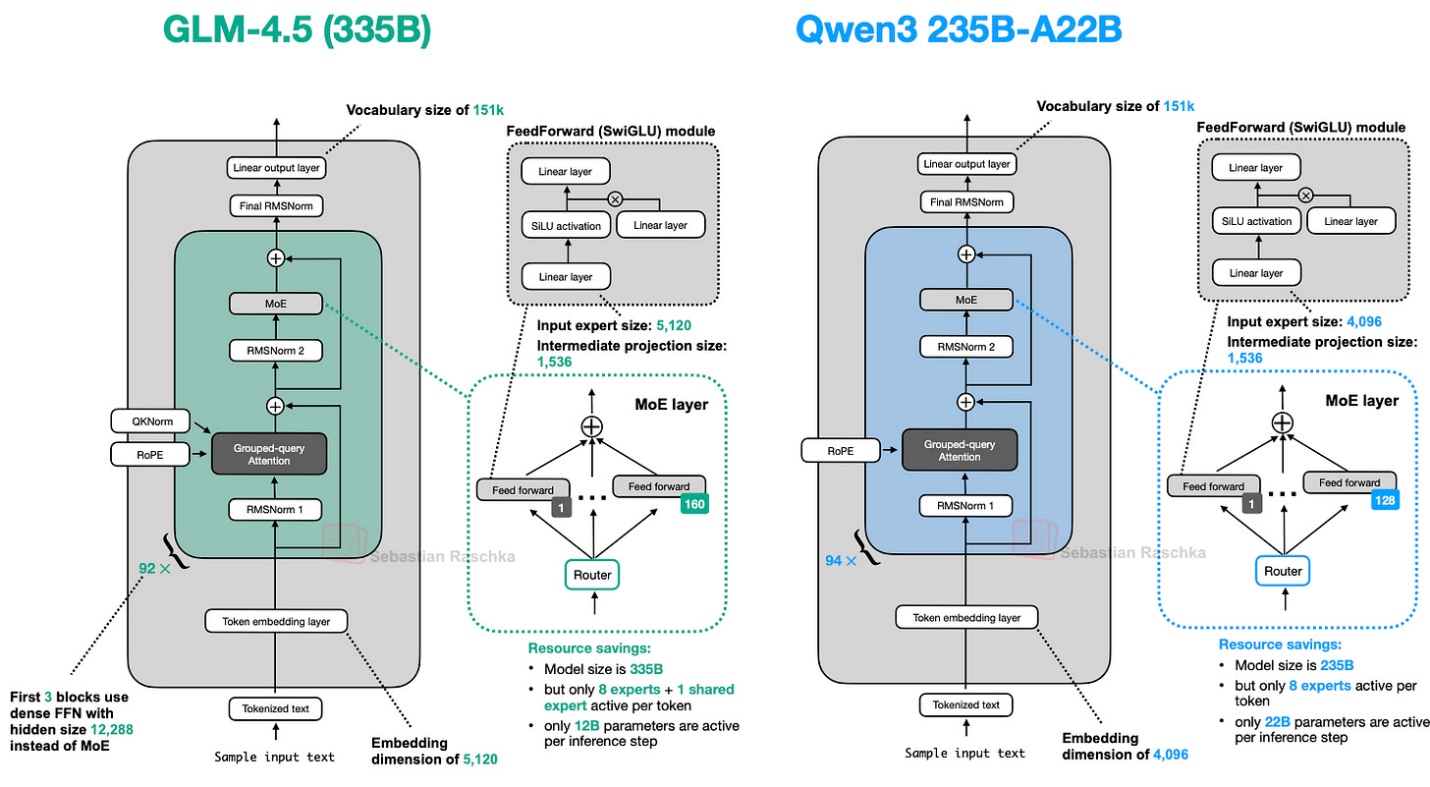
**Design Insights:**

* ✅ Proven production stability
* ✅ Conservative, reliable approach
* ✅ Shared expert benefits
* ❌ Older expert design philosophy
* ❌ Not cutting-edge architecturally

**11. GLM-4.5 - Function Calling Specialist**



*Figure 33: GLM-4.5 benchmark performance across multiple evaluations*



*Figure 34: GLM-4.5 architecture comparison with Qwen3*

**Specialized Focus:**

* **Primary Use**: Function calling and agent applications
* **Variants**: 355B (flagship) and 106B (compact)
* **Performance**: Competes with Claude 4 Opus, trails o3 slightly

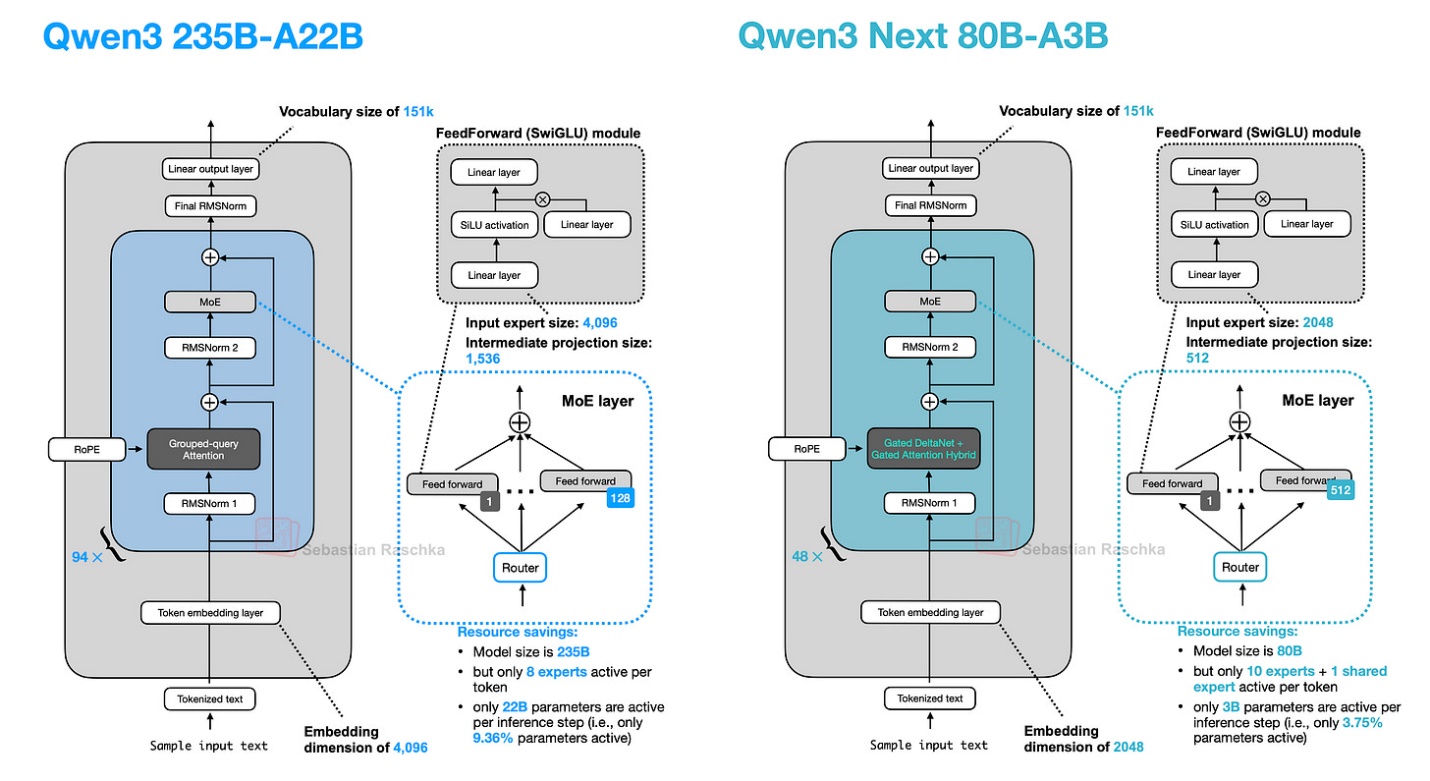
**Architectural Features:**

1. **Dense Prefix**: 3 dense layers before MoE (like DeepSeek V3)
2. **Shared Expert**: Always-active expert maintained
3. **Attention Bias**: Retains GPT-2 style bias units
4. **Stability Focus**: Dense prefix improves MoE convergence

**Design Trade-offs:**

* ✅ Excellent agent/function calling performance
* ✅ Stable MoE training approach
* ✅ Two-variant strategy
* ❌ Specialized for specific use cases
* ❌ Not general-purpose optimized

**12. Qwen3-Next - Hybrid Attention Revolution**



*Figure 35: Qwen3 vs Qwen3-Next architecture evolution*

**Revolutionary Changes:**

* **Size**: 80B-A3B (3x smaller than predecessor)
* **Experts**: 4x more experts + reintroduced shared expert
* **Context**: Native 262K token support (vs 32K)
* **Attention**: Hybrid Gated DeltaNet + Gated Attention

**Hybrid Attention Mechanism**

**Component Ratio**: 3:1 (DeltaNet:Attention)

**Gated Attention Enhancements:**

* Output gate for scaled attention results
* Zero-centered RMSNorm for QK normalization
* Partial RoPE implementation
* Stability improvements over standard GQA

**Gated DeltaNet Innovation:**

* Fast-weight delta rule updates
* Linear-time, cache-free operation
* Alternative to Mamba for sequence modeling
* Lightweight convolutions for q/k/v/α/β generation

**Multi-Token Prediction (MTP)**

* Predicts multiple future tokens per step
* Accelerates training convergence
* Enables speculative decoding
* Multi-step inference optimization

**Design Trade-offs:**

* ✅ Massive context length capability
* ✅ Improved efficiency vs size
* ✅ Cutting-edge attention mechanisms
* ❌ Complex hybrid implementation
* ❌ Limited real-world testing

**📊 Key Trends and Trade-offs Summary**

**Major Architectural Trends (2024-2025)**

| **Trend** | **Models** | **Benefits** | **Trade-offs** |
| --- | --- | --- | --- |
| **MoE Adoption** | DeepSeek V3, Llama 4, Qwen3, GLM-4.5, Kimi 2 | Massive capacity, inference efficiency | Implementation complexity, hardware requirements |
| **Attention Evolution** | MLA (DeepSeek), Sliding Window (Gemma), Hybrid (Qwen3-Next) | Memory efficiency, long context | Varying implementation complexity |
| **Normalization Innovation** | Post-Norm (OLMo), QK-Norm (Multiple), Dual-Norm (Gemma) | Training stability, gradient behavior | Minimal overhead, architecture complexity |
| **Positional Encoding** | RoPE (Most), NoPE (SmolLM3), Partial RoPE (Qwen3-Next) | Length generalization, simplification | Limited scaling evidence |

**Performance vs Efficiency Matrix**

| **Model** | **Size Class** | **Efficiency Focus** | **Performance Tier** | **Best Use Case** |
| --- | --- | --- | --- | --- |
| **DeepSeek V3** | Ultra-Large (671B) | Memory (MLA) | Top Tier | Research, Production |
| **Kimi 2** | Extreme (1T) | Scale | Top Tier | Benchmark Competition |
| **Llama 4** | Large (400B) | Balanced MoE | High Tier | General Production |
| **GLM-4.5** | Large (355B) | Agent Tasks | High Tier | Function Calling |
| **Gemma 3** | Medium (27B) | Memory (Sliding) | High Tier | Local Deployment |
| **Qwen3** | Multi-Size | Flexibility | High Tier | Diverse Use Cases |
| **Mistral 3.1** | Medium (24B) | Speed | High Tier | Fast Inference |
| **OLMo 2** | Medium | Transparency | Mid Tier | Research/Education |
| **SmolLM3** | Small (3B) | Simplicity | Mid Tier | Edge/Mobile |
| **Qwen3-Next** | Large (80B) | Long Context | High Tier | Document Processing |

**Architecture Decision Framework**

**Choose MoE When:**

* Need massive capacity with efficient inference
* Have hardware for sparse computation
* Serving at scale with varied queries

**Choose Dense When:**

* Fine-tuning is priority
* Simple deployment required
* Limited specialized hardware

**Choose Sliding Window When:**

* Memory constraints are critical
* Long sequences are common
* Acceptable slight performance trade-off

**Choose Advanced Attention When:**

* Pushing efficiency boundaries
* Research/experimental deployment
* Specific use case optimization

**🎓 Study Guide: Key Concepts to Master**

**Essential Attention Mechanisms**

1. **Multi-Head Attention (MHA)**: Foundation mechanism
2. **Grouped-Query Attention (GQA)**: Memory-efficient via K/V sharing
3. **Multi-Head Latent Attention (MLA)**: Compression-based efficiency
4. **Sliding Window Attention**: Local context for memory savings
5. **Gated DeltaNet**: Linear-time alternative with fast weights

**Critical Normalization Strategies**

1. **Pre-Norm**: Before attention/FFN (GPT-2 style)
2. **Post-Norm**: After attention/FFN (original Transformer)
3. **QK-Norm**: Additional normalization for queries/keys
4. **Dual-Norm**: Both pre- and post-normalization

**MoE Design Patterns**

1. **Expert Count**: Trend toward more, smaller experts
2. **Shared Expert**: Always-active for common patterns
3. **Router Strategy**: Top-k selection mechanisms
4. **Dense Prefix**: Stable layers before MoE introduction

**Efficiency Optimization Techniques**

1. **KV Cache Management**: Critical for inference speed
2. **Memory Compression**: MLA-style tensor compression
3. **Sparse Computation**: MoE activation patterns
4. **Context Length**: Sliding windows vs hybrid mechanisms

**📚 Recommended Further Reading**

* [DeepSeek V3 Paper](https://arxiv.org/abs/2412.19437) - MLA and MoE innovations
* [OLMo 2 Paper](https://arxiv.org/abs/2501.00656) - Normalization strategies
* [Gemma 3 Paper](https://arxiv.org/abs/2503.19786) - Sliding window attention
* [Original MoE Paper](https://arxiv.org/abs/1701.06538) - Foundational concepts
* [Attention Is All You Need](https://arxiv.org/abs/1706.03762) - Transformer foundation
* [RoPE Paper](https://arxiv.org/abs/2104.09864) - Rotary positional embeddings
* [GQA Paper](https://arxiv.org/abs/2305.13245) - Grouped-query attention