Fast-K Multi-Header Language Models: Sparse Routing at the LM Head for Efficient Generation

I'm excited to share my latest research on a simple yet effective approach to make language models faster and more accurate.

Replace the single LM head with P parallel heads. During training, update only the head that assigns the highest probability to the gold token (token-aware, sparse). During inference, a pilot head proposes a Top-K shortlist and the other heads score only those K tokens—cutting output projection from O(P·V) to O(V + (P–1)·K).

Architecture at a Glance

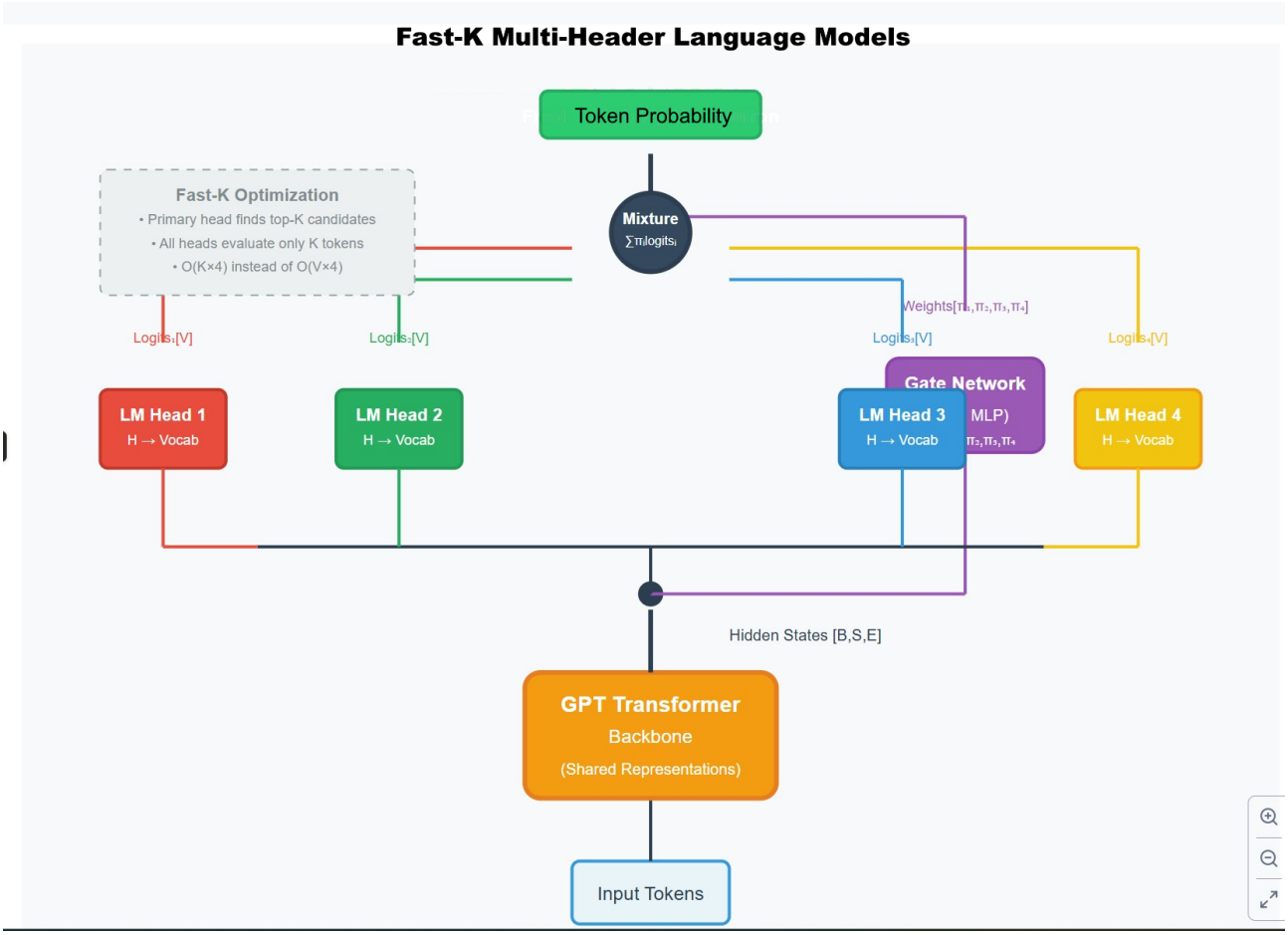

Figure: Fast-K with multiple LM heads. A pilot head selects Top-K tokens; other heads evaluate only those tokens.

The Problem

Traditional language models use a single "head" to predict the next token from a vocabulary of 50,000+ words. This creates a computational bottleneck and struggles with the complexity of human language—where the same word can mean different things in different contexts.

The Solution: Multi-Header Architecture with Fast-K Inference

Instead of one LM head, use P parallel heads. During training, only the head that assigns the highest probability to the gold token is updated (token-aware, sparse). During inference, a pilot head generates a Top-K short list and the remaining heads score only those K tokens, reducing the output-projection cost from O(P·V) to O(V + (P–1)·K).

Key Innovation 1: Multiple Specialized Heads Instead of one prediction head, I use multiple parallel heads that can specialize in different linguistic patterns. During training, only the head that best predicts the target token gets updated—like having multiple experts where only the best one learns from each example.

Key Innovation 2: Fast-K Inference Here's where it gets interesting: Rather than running all heads on the full 50K vocabulary (expensive!), I use a two-stage approach:

1. One "pilot" head identifies the top-K most likely tokens (e.g., top 50)
2. Other heads only evaluate these K candidates
3. Select the best (head, token) combination

This reduces computational complexity from O(P×V) to O(V + (P-1)×K) per step—a massive saving when V=50,000 and K=50.

Results That Matter

- Perplexity drops to 16.11 (HF baseline 25.66).
- Consistent gains in Top-n token accuracy and better rare-token behavior.

✅ **65% faster generation** with 2.9% better quality vs. baseline GPT-2 ✅ **Improved rare token handling** - better top-n accuracy across the board ✅ **Semantic specialization** - different heads learn different meanings of the same word based on context

For example, the word "make" gets routed to different heads based on context:

- "make decisions" (creating/producing) → Head 3
- "make them for themselves" (creating for others) → Head 1
- "make it a problem" (transforming) → Head 3
- "make the solution" (causing change) → Head 2

Implementation notes

- Keep a fast slice-matmul path for shortlisted columns; layout/sharding matters for cache locality.
- Pilot selection can be lowest-entropy, utilization-based, or round-robin; all are simple to try.
- Works with quantized heads (e.g., 4-bit) via slice dequantization.

Why This Matters

Efficiency: Lower computational costs mean more accessible AI

Quality: Better handling of rare words and nuanced meanings

Scalability: Easy to integrate into existing transformer architectures

- Latency-sensitive generation (interactive apps, assistants).
- Edge or cost-constrained deployments where output projection dominates.
- Long-context decoding where cumulative savings add up over many steps.

The Bigger Picture

This work reflects a fundamental principle: just like humans use hierarchical decision-making (screening resumes before interviews), AI systems can be more efficient by first filtering possibilities, then making refined decisions on a smaller set.

What's next

I'm preparing the GitHub repo and arXiv preprint. I'm also exploring roles related to LLM efficiency, inference, and systems—happy to walk through the method and benchmarks.

Currently exploring extensions to larger models and investigating reward-based training methods inspired by how humans learn—because gradient descent might not be the only way forward.

Contact

Songnian Qian — Songnian@gmail.com

Preprint & code: available by request; GitHub link coming soon.

Research available by request - would love to discuss with fellow researchers and practitioners working on efficient AI systems!

#AI #MachineLearning #NLP #LanguageModels #Research #Innovation #DeepLearning #Transformers #AIEfficiency

---

What are your thoughts on specialized vs. generalized approaches in AI architecture? I'd love to hear your perspective in the comments!