Domain-Specialised Mixture of Experts for Better Language Modelling

Atharva Bhutani | 16 yo | Self-taught AI Dev

ML Collective Research Jam #27

Introduction

Hi everyone 👋! I’m Atharva, a 16 year old high school student learning deep learning by building things from scratch. I don’t have formal training, just curiosity and a lot of trial and error. Right now I’m building a transformer-based LLM in NumPy and expanding it into a domain-specialized Mixture of Experts.

Motivation

- Current LLMs are massive and generalized, but real world needs are often domain-specific.
- Can we scale intelligently instead of brutally?
- MoEs are promising, but most implementations are compute hungry.
- My goal: a light, explainable, domain-specialized MoE, built from scratch.

References

- Shazeer et al. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer.
- Fedus et al. (2022). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity.

Foundations - LLM from Scratch

- Tokenization → Embedding → LayerNorm → Multi-Head Attention → MLP → Output
- Built everything in NumPy with custom initialization (He/Xavier), manual forward passes.
- Learned core ideas by implementation—not by lecture.

Takeaways

- Debugging teaches more than reading.
- Data shapes, matrix flow, and attention mechanisms become intuitive.

References

- Vaswani et al. (2017). Attention is All You Need.
- Karpathy's minGPT for interpretability inspiration.

Transition to MoE

- After a working transformer, I wanted modularity.
- Instead of larger models, what if we selectively activate parts of the network?
- MoEs do this by routing tokens to “experts” based on input.
- My twist: domain tags help route inputs, not just token patterns.

Domain Specialisation

- Instead of training on a generic corpus, I segment data by domain: science, law, tech, etc.
- Each domain has its own expert MLP.
- Routing is done via a lightweight classifier that predicts the domain.

Why this matters:

- Makes routing interpretable.
- Each expert learns something meaningful and domain-specific.
- Reduces unnecessary computation.

Architecture Overview

- Embedding → Shared Attention Stack → Domain Classifier → Route to Domain Expert MLP → Output Head
- Experts are only activated when their domain is predicted.
- Optimizes for both memory and performance.

Sparsity: ~10% of experts active per token

Scalability: Add more experts, keep compute constant.

Challenges

- Routing accuracy: Wrong domain predictions can lead to poor generations.
- Balancing domain distribution in training.
- Getting enough tokens per domain to avoid underfitting.

Extras: Visualize attention and classifier confidence.

What's Next?

- Add diffusion-based pretraining for smoother text representations.
- Train on real-world domain data (Reddit science, Stack Overflow, arXiv abstracts).
- Try collaborative training.

Key Papers:

- Vaswani et al. (2017) – Transformers
- Shazeer et al. (2017) – Sparsely-Gated MoEs
- Fedus et al. (2022) – Switch Transformer
- Ramesh et al. (2022) – Hierarchical Mixture of Experts (for inspiration)

Thank You for listening!

Twitter: @gradanis

GitHub: @gradanii

Mail: gradanis@proton.me
atharvaxcode@proton.me