

Adaptive Mixture-of-Heads: Routing Attention Heads in GPT-2 with Fixed and Dynamic Sparsity

Stanford CS224N Default Project

Ryder Fried

Department of Computer Science
Stanford University
ryfried@stanford.edu

David Stutz

Department of Computer Science
Stanford University
dstutz@stanford.edu

Abstract

We implement GPT-2 from scratch and extend it with an adaptive Mixture-of-Heads (MoH) attention mechanism. Instead of uniformly aggregating all attention heads, our method introduces a learned router that selects a subset of heads for each token and computes a weighted sum of their outputs. We evaluate two routing strategies: a fixed Top- k mechanism and a threshold-based dynamic mechanism that allows the number of active heads to vary per token. We study these models on sentiment analysis, paraphrase detection, and sonnet generation. While both routing methods underperform a vanilla GPT-2 baseline on downstream metrics, threshold-based routing consistently preserves more performance than fixed Top- k routing while using fewer active heads in expectation. Routing statistics further show non-uniform head usage, suggesting that the router learns stable preferences over attention heads rather than acting as random dropout. Overall, our results support the idea that adaptive head routing is a meaningful form of conditional computation, though stronger optimization and sparse kernels are likely required to convert its theoretical efficiency gains into realized performance improvements.

1 Project Information

- Mentor: Luke Bailey
- Late Days (if you are using any):

Team member	Late days remaining
Ryder Fried	5
David Stutz	5
Total pooled	10
Team size	2
Late days applied for the report	3 Each

2 Introduction

Modern large language models rely heavily on the Transformer architecture introduced in *Attention Is All You Need* [1]. A core component of this architecture is multi-head self-attention, where each layer contains a fixed number of attention heads that independently process the same token sequence. The outputs of these heads are then combined to form the final representation.

This design has proven extremely effective, but it is also computationally expensive. In GPT-2 small, each attention layer contains 12 heads, so every token is processed by all 12 heads regardless of whether all of them are equally useful. This raises a natural question: are all heads necessary for every token?

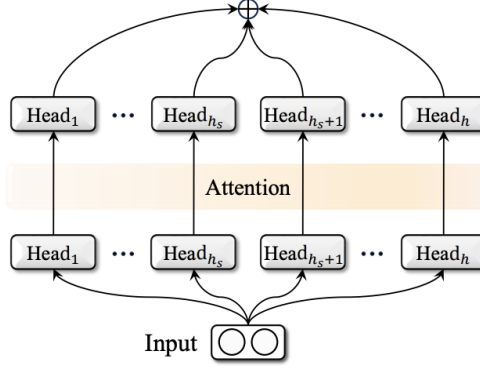


Figure 1: Standard multi-head attention: all heads are always active and their outputs are uniformly aggregated.

Prior work suggests the answer is no. Some heads appear highly specialized, while others capture redundant or generic patterns. If this is true, then forcing every token to activate every head may waste computation and blur specialization. This motivates a conditional-computation view of attention, where only a subset of heads are used for a given token.

In this project, we investigate whether attention heads can be treated as experts and routed dynamically. We implement a Mixture-of-Heads (MoH) extension to GPT-2 that learns to select and weight a subset of attention heads per token. We evaluate two versions: a fixed Top- k router and a threshold-based dynamic router. The first provides a controlled sparse-attention baseline inspired by prior work, while the second is our main extension and allows the number of active heads to vary based on token difficulty.

3 Related Work

Several lines of work aim to improve the efficiency of transformers by modifying attention. Some approaches reduce the number of tokens considered, while others simplify the structure of the attention matrix. For example, Linformer approximates attention with low-rank projections [2], and Longformer uses local sliding-window attention with sparse global tokens [3].

Another direction asks whether all attention heads are necessary. Voita et al. show that some attention heads in trained transformers are highly specialized while others can be pruned with minimal loss in performance [4]. This supports the broader idea that attention heads are not equally important.

The paper most directly motivating our project is *MoH: Multi-head attention as mixture-of-head attention* by Jin et al [6]. Their key idea is to reinterpret attention heads as experts in a Mixture-of-Experts framework. Instead of always evaluating all heads, they designate some heads as shared and always active, while dynamically routing among the remaining heads using a learned scoring mechanism. Their final attention output is a weighted sum over a pre-determined k selected heads rather than a uniform aggregation over all heads. They report strong results across multiple domains, including LLaMA3-8B, where their MoH model activates roughly 75% of the heads while outperforming the vanilla baseline.

In our proposal, we identified a limitation of their formulation: the Top- k routing policy enforces a fixed compute budget for every token. We argued that this misses an important nuance of language. Function words such as *the* or punctuation may require far less reasoning than content words or semantically ambiguous phrases. This led us to propose threshold-based dynamic routing, where the model decides how many heads to activate based on the token’s routing distribution.

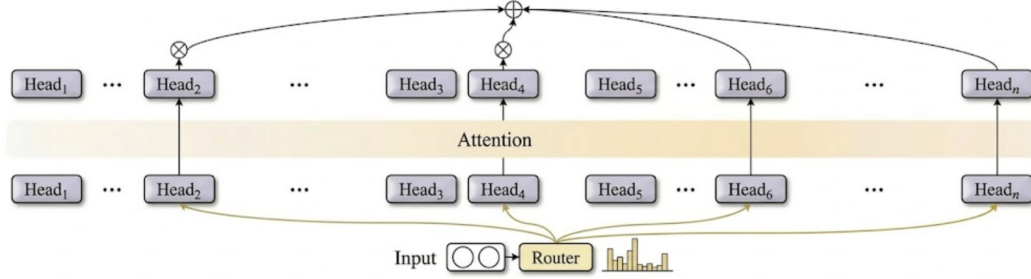


Figure 2: Mixture-of-Heads attention: a learned router dynamically selects which attention heads contribute to each token representation.

4 Approach

4.1 Baseline: GPT-2

We implemented a decoder-only GPT-2 model following the architecture described in [5]. The model consists of token and positional embeddings, 12 transformer layers with masked multi-head self-attention, pre-layer normalization, residual connections, and a position-wise feed-forward network. We tie the output projection weights with the token embedding matrix.

For classification tasks, we followed the project framing and posed the task as next-token prediction. For example, paraphrase detection was formatted as: *Is “s₁” a paraphrase of “s₂”? Answer “yes” or “no”*. We evaluate predictions at the final token position.

4.2 Mixture-of-Heads (MoH)

Inspired by Jin et al., we implemented simplified MoH variants in which attention heads are treated as experts and a learned router determines which heads contribute to the final output.

For token representation $x_t \in \mathbb{R}^d$, the router computes head scores:

$$r_t = W_r x_t$$

where $W_r \in \mathbb{R}^{h \times d}$ and h is the number of heads.

Unlike the original MoH paper, we do not use shared heads or sparse kernels. Instead, we compute all heads densely and then zero out non-selected heads using routing weights. This means that our implementation does *not* realize actual sparse attention speedups, but it still provides a meaningful test of whether conditional head routing improves modeling behavior.

4.2.1 Top- k

The first method we used selects exactly k heads per token. Let S_k denote the top- k heads after applying softmax to the routing logits:

$$S_k = \text{Top-}k(\text{Softmax}(r_t)).$$

We then re-normalize the selected probabilities:

$$g^{(i)} = \begin{cases} \frac{\text{Softmax}(r_t)_i}{\sum_{j \in S_k} \text{Softmax}(r_t)_j} & \text{if } i \in S_k \\ 0 & \text{otherwise.} \end{cases}$$

The final attention output becomes:

$$\text{MoH}(X) = \sum_{i=1}^h g^{(i)} H^{(i)} W_O^{(i)}.$$

Top- k has the advantage of fixed sparsity and straightforward analysis. However, it imposes the same head budget on every token regardless of complexity.

4.2.2 Threshold

The second method uses a threshold τ over the routing probabilities. We compute:

$$p^{(i)} = \text{Softmax}(r_t)_i$$

and define the active set as:

$$S = \{i \mid p^{(i)} > \tau\}.$$

We again re-normalize the selected probabilities:

$$g^{(i)} = \begin{cases} \frac{p^{(i)}}{\sum_{j \in S} p^{(j)}} & \text{if } i \in S \\ 0 & \text{otherwise.} \end{cases}$$

This allows the number of active heads to vary across tokens. In principle, this better matches our original proposal: simple tokens can use fewer heads, while more ambiguous or information-rich tokens can use more.

4.2.3 Load balancing

Following Mixture-of-Experts style approaches, we also implemented a load-balancing penalty intended to prevent the router from collapsing onto a small subset of heads. The auxiliary loss is:

$$\mathcal{L}_b = \sum_i P_i f_i$$

where P_i is the average routing probability for head i and f_i is the fraction of tokens selecting head i .

We experimented with $\beta = 0$ and $\beta = 0.01$ in the final loss:

$$\mathcal{L} = \mathcal{L}_{task} + \beta \mathcal{L}_b.$$

5 Experiments

This section contains the following.

5.1 Data

We evaluate on the datasets provided in the default final project:

- **SST** for sentiment classification
- **CFIMDB** for sentiment classification
- **Quora Question Pairs** for paraphrase detection
- **Shakespeare Sonnets** for generation

These tasks provide diversity across binary classification and sequence generation. We intentionally did not introduce any external datasets or additional preprocessing because we wanted changes in performance to come from architectural modifications rather than data engineering.

5.2 Evaluation method

For SST, CFIMDB, and paraphrase detection, we report development-set accuracy.

For sonnet generation, we report CHRF, the metric provided in the project framework. This is more appropriate than exact-match metrics because it captures partial lexical and character-level overlap between generated sonnets and references.

In addition to task performance (accuracy and loss curves), we also analyze the behavior of the routing mechanism. For all MoH runs, we log per-head selection frequencies, the average number of active heads per token position, and total token-layer routing decisions.

5.3 Experimental details

We use GPT-2 small throughout the main study. All models have:

- hidden size 768
- 12 layers
- 12 attention heads

The MoH router introduces approximately 110k additional parameters relative to the vanilla GPT-2 model, representing less than 0.1% of total parameters. We tuned hyperparameters primarily on paraphrase detection, our slowest and most semantically demanding task, and then transferred the best routing settings to SST, CFIMDB, and sonnet generation. This kept the study manageable while still giving us a principled tuning procedure.

For the vanilla paraphrase baseline, we ran a small learning-rate sweep:

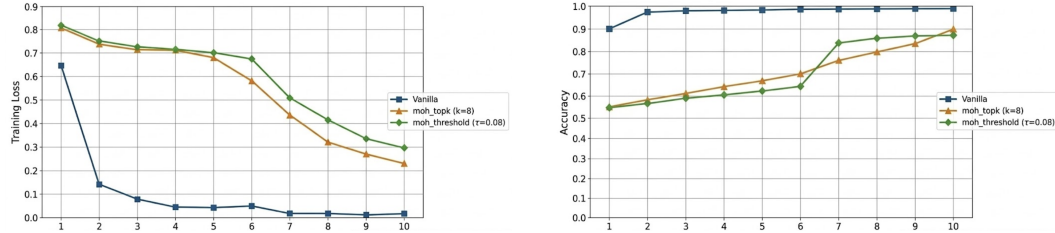
- $5 \times 10^{-6} \rightarrow 0.875$ dev accuracy
- $1 \times 10^{-5} \rightarrow 0.884$ dev accuracy
- $2 \times 10^{-5} \rightarrow 0.891$ dev accuracy

We therefore used 2×10^{-5} for MoH paraphrase experiments.

For Top- k routing, we swept $k \in \{6, 8, 10\}$ and found that $k = 8$ performed best. We also compared $\beta \in \{0, 0.01\}$ and found no meaningful difference for Top- k at $k = 8$.

For threshold routing, we swept $\tau \in \{0.05, 0.08, 0.10\}$. We observed the expected monotonic tradeoff: larger thresholds increased sparsity but reduced accuracy. The best threshold result was $\tau = 0.05$ with 0.848 paraphrase accuracy, while the most informative dynamic-compute run was $\tau = 0.08$, which retained 0.844 accuracy with fewer active heads on average. We also compared $\beta \in \{0, 0.01\}$ at $\tau = 0.08$; the results show that adding the load-balancing term *hurts* performance slightly, reducing dev accuracy from 0.844 to 0.838.

Paraphrase experiments were run for 3 epochs due to their significantly higher training cost. SST, CFIMDB, and sonnet experiments were run for 10 epochs.



(a) Training loss curves for vanilla GPT-2, Top- k MoH, and threshold MoH on the paraphrase task.

(b) Development accuracy over training for vanilla GPT-2, Top- k MoH, and threshold MoH.

Figure 3: Training dynamics comparison between vanilla GPT-2 and Mixture-of-Heads variants. While both MoH approaches converge similarly in loss, threshold routing preserves more downstream accuracy than fixed Top- k routing.

5.4 Results

Across all tasks, both MoH variants underperform the vanilla GPT-2 baseline. However, threshold routing consistently outperforms fixed Top- k routing on the two tasks where the difference matters most: paraphrase detection and sonnet generation. On paraphrase, threshold routing improves over Top- k by 0.007 absolute accuracy (0.848 vs. 0.841). On sonnet generation, threshold routing is substantially stronger than Top- k in CHRF (39.159 vs. 35.985).

These results support the central intuition of our proposal: if head sparsity is going to work, a fixed budget is likely too rigid. Allowing the number of active heads to vary appears to preserve more capacity.

Model	SST Acc	CFIMDB Acc	Paraphrase Acc	Sonnet CHRF
Vanilla GPT-2	0.514	0.980	0.891	41.573
MoH Top- k	0.457	0.906	0.841	35.985
MoH Threshold	0.449	0.869	0.848	39.159

Table 1: Main results across all four tasks. We report the best threshold-routing result on paraphrase ($\tau = 0.05$).

Paraphrase tuning. The paraphrase tuning results are summarized below:

- **Vanilla LR sweep:** $5e^{-6} \rightarrow 0.875$, $1e^{-5} \rightarrow 0.884$, $2e^{-5} \rightarrow 0.891$
- **Top- k sweep:** $k = 6 \rightarrow 0.838$, $k = 8 \rightarrow 0.841$, $k = 10 \rightarrow 0.839$
- **Threshold sweep:** $\tau = 0.05 \rightarrow 0.848$, $\tau = 0.08 \rightarrow 0.844$, $\tau = 0.10 \rightarrow 0.819$

The threshold sweep reveals a clear trend: increasing the threshold too much harms performance. This is exactly what we would expect if the threshold controls access to useful heads. A very high threshold makes routing overly aggressive and forces the model to discard too much information.

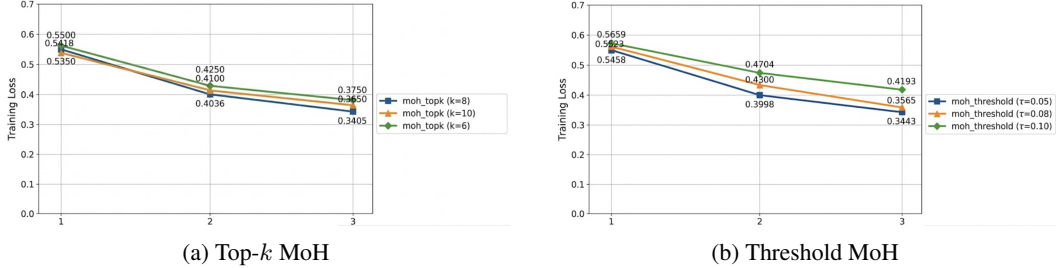


Figure 4: Representative training dynamics for paraphrase. Threshold routing converges similarly to Top- k , while preserving more sparsity at a comparable performance.

As our MoH model was suboptimal compared to the vanilla GPT-2 architecture, our top sonnet generation test leaderboard score came from the small vanilla model [42.253 CHRF score] and our best paraphrase detection test leaderboard score was from the large vanilla model, trained with parameter efficient fine tuning (LoRA) [0.883 accuracy].

6 Analysis

A core motivation for our work was not only performance, but also the possibility of conditional computation. Since our implementation computes all heads densely, we do not claim realized wall-clock savings. However, we can still analyze *theoretical* reductions in head-dependent attention computation.

For standard multi-head self-attention with h heads, the head-dependent dominant terms in attention scale linearly with the number of active heads. This gives a simple FLOP proxy:

$$\text{FLOP Fraction} \approx \frac{\text{Average Active Heads}}{h}.$$

For Top- k with $k = 8$:

$$\frac{8}{12} = 0.667.$$

For threshold routing on paraphrase at $\tau = 0.08$, we observed different sparsity levels depending on the load-balancing coefficient.

Without load balancing ($\beta = 0$):

$$\text{avg active heads per position} = 5.889$$

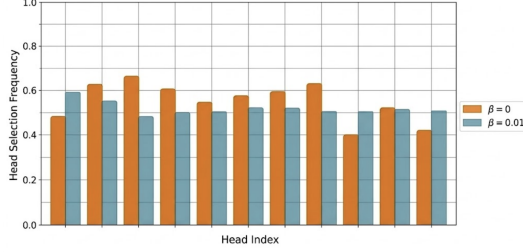


Figure 5: Per head selection frequencies for MoH-Threshold

giving a theoretical attention FLOP fraction of:

$$\frac{5.889}{12} \approx 0.491.$$

With load balancing ($\beta = 0.01$):

$$\text{avg active heads per position} = 6.589$$

giving a theoretical attention FLOP fraction of:

$$\frac{6.589}{12} \approx 0.549.$$

Model	Active Heads	FLOP Fraction	Paraphrase Acc
Vanilla	12	1.00	0.891
Top- k	8	0.67	0.841
Threshold	5.89	0.49	0.844

Table 2: Comparison of average active heads, theoretical attention FLOP fraction, and paraphrase accuracy across routing methods.

This is an important result. Relative to vanilla GPT-2, the threshold model at $\tau = 0.08$ uses only about 50% of the head-dependent attention computation in theory, while still retaining 0.844 development accuracy when $\beta = 0$. In contrast, the Top- k model uses about 67% of the head-dependent attention computation and achieves slightly worse performance. This means the threshold router is not only more adaptive, but also more compute-efficient in theory.

Why threshold helps. This result supports the key idea from our original proposal: language does not have a uniform compute budget across tokens. Threshold routing naturally implements a variable budget. Tokens whose routing distributions are concentrated activate fewer heads, while tokens with broader uncertainty or multiple moderately useful heads can activate more. This flexibility allows threshold routing to avoid both under-computing difficult tokens and over-computing easy ones.

Head specialization. The per-head selection frequencies from the threshold paraphrase runs further illustrate this effect. For $\beta = 0$, the router shows a wider spread in head usage:

$$[0.483, 0.626, 0.399, 0.352, 0.302, 0.580, 0.442, 0.579, 0.510, 0.518, 0.531, 0.569].$$

With load balancing enabled ($\beta = 0.01$), the selection frequencies become noticeably more uniform:

$$[0.592, 0.553, 0.522, 0.555, 0.537, 0.574, 0.558, 0.534, 0.552, 0.540, 0.509, 0.563].$$

This confirms that the load-balancing loss encourages more even usage across heads. However, since the more uniform distribution corresponds to both higher compute and slightly worse accuracy, our results suggest that mild head specialization is beneficial for this task.

Effect of load balancing. We also compared threshold routing with and without the load-balancing term at $\tau = 0.08$. Without load balancing ($\beta = 0$), the model achieved 0.844 development accuracy while activating an average of 5.889 heads per token position. With load balancing ($\beta = 0.01$), development accuracy dropped slightly to 0.838 while the average number of active heads increased to 6.589.

This indicates that the auxiliary loss successfully pushed the router toward more uniform head usage, but doing so both increased theoretical compute and slightly degraded performance. In our setting, allowing the router to specialize more strongly appears to produce better results than enforcing evenly distributed head usage.

Why performance still drops. Although threshold routing is more promising than Top- k , both routing models still underperform vanilla GPT-2. We believe there are several reasons:

- Introducing randomly initialized routing parameters into a pre-trained model may interfere with the representations learned by the original attention heads.
- We removed heads from computation without increasing capacity elsewhere, so performance losses may simply reflect reduced expressive power.
- The original MoH paper included shared heads and a more elaborate routing design, while our implementation deliberately simplified these features.

Thus, our results should not be interpreted as disproving MoH. Rather, they suggest that dynamic head routing is feasible and behaves meaningfully, but requires more careful integration to match or exceed the strong vanilla baseline.

7 Conclusion

In this project, we investigated whether attention heads in GPT-2 can be dynamically routed using a Mixture-of-Heads (MoH) framework. We implemented two routing strategies: a fixed Top- k mechanism and a threshold-based mechanism that allows the number of active heads to vary per token. These models were evaluated on sentiment classification, paraphrase detection, and sonnet generation.

While neither MoH variant outperformed the vanilla GPT-2 baseline, threshold-based routing consistently preserved more performance than fixed Top- k routing while activating fewer attention heads in expectation. On the paraphrase task, the threshold model at $\tau = 0.08$ activated an average of 5.889 heads per token position, corresponding to roughly 49% of the head-dependent attention computation used by the vanilla model in theory. Despite this reduction, the model retained much of the baseline performance. The best threshold configuration ($\tau = 0.05$) achieved 0.848 development accuracy and outperformed the best Top- k configuration, which required a fixed budget of eight heads per token.

These results suggest that adaptive routing is more effective than fixed routing for selectively allocating attention computation. Threshold routing naturally enables variable attention budgets, allowing the model to activate fewer heads for simpler tokens while retaining flexibility for more complex inputs. However, our experiments also show that routing alone is not sufficient to surpass the vanilla transformer architecture. Future work could explore integrating shared heads, hardware-aware sparse execution, or larger-scale models to better realize the potential efficiency benefits of adaptive attention routing.

Team contributions (Required for multi-person team)

Both team members contributed to the overall direction and execution of the project. Ryder focused primarily on implementing the routing mechanisms, including Top- k and threshold-based MoH, and on setting up the experimental framework for the different model branches. David focused primarily on training infrastructure, structured logging, result aggregation, and analysis of routing statistics such as head frequencies and active-head counts. Both team members collaborated on tuning experiments, interpreting results, and writing the final report.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [2] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv preprint arXiv:2006.04768*, 2020.
- [3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- [4] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.
- [5] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Technical Report*, 2019.
- [6] Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. MoH: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*, 2025.