MoEP: Modular Expert Paths for Sample-Efficient Language Modeling

Joonas Tapaninaho
University of Oulu, Faculty of Information Technology and Electrical Engineering, CMVS
Oulu, Finland

Abstract

Training language models under tight compute budgets with small training datasets remains challenging for dense decoder-only Transformers, where every token activates the full stack of model parameters. We introduce *MoEP* (Modular Expert Paths), a sparse decoder-only architecture that enables more selective token activation, which increases model performance and accelerates learning without increasing the total number of parameters. We show that combining model parallelism with Mixture-of-Experts (MoE) style linear projections and a lightweight top-*k* router outperforms the GPT-2 baseline and stabilizes evaluation performance more quickly.

1 Introduction

Despite the strong dominance of dense decoder-only Transformers, there is noticeable growing interest in exploring alternative architectures, which challenge the assumption that every token must pass through the same full stack of layers or route.

Recent and previous work has examined **sparse activation** [9, 13], **routing-based decoder-only language modeling** [1, 6, 10, 15], and compositional approaches, where models are constructed from **modular components** [14]. These efforts highlight a broader trend to improve efficiency and flexibility by enabling tokens to follow different computation paths.

Our previous work, *PaPaformer* [14], introduced method of remodeling Transformer layers into smaller **parallel subpaths**, which can be used as independently trainable modules. Despite being effective for modularity, PaPaformer required pre-trained paths to outperform the baseline architecture and did not fully exploit the sparsity opportunities offered by parallel paths.

This paper presents *MoEP* (Modular Expert Paths), which adds model sparsity by unifying two forms of routing within a decoder-only language model: (i) **Top-k** token routing across parallel Transformer blocks, and (ii) **Mixture-of-Experts** feed-forward layers based on lightweight linear projections and comparison **SwiGLU** variant. As a result, each token activates only a limited set of parallel blocks and experts in forward-pass, creating more diverse computational pathways while reducing redundancy. In training a load-balanced **auxiliary loss** was used to encourage stable expert and block utilization without collapse.

We train MoEP with the **BabyLM** strict-small track [1] data and used the official evaluation pipeline. MoEP was able to outperform all BabyLM strict-small baseline models, not only GPT-2, which layer structure it follows. In addition, MoEP exhibits earlier learning gains in comparison to GPT-2, which suggest faster learning capabilities. This was achieved even though MoEP did not employ the *PaPaformer* [14] style of modularity, in which independent modules were pre-trained separately.

[1] https://babylm.github.io

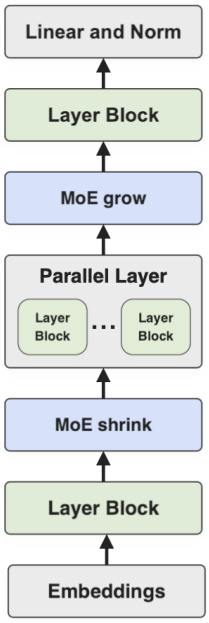

Figure 1: MoEP architecture visualization. *N* **parallel layers are stacked before and after the MoE blocks, whose task is to reduce or increase the hidden dimension to match the layer blocks. In a Parallel layer, the Layer blocks operate on a smaller hidden dimension compared to the individual Layer blocks at the beginning and end of the model.**

This work present following contributions, which are summarized below:

(1) Proposes *MoEP*, a modular sparse decoder-only architecture that integrates top-*k* routing across parallel blocks with MoE style.

(2) Employs the BabyLM evaluation pipeline on the strict-small track to compare MoEP against GPT-2 and other baseline models under matched conditions.

(3) Analyzes fast-eval learning dynamics, showing it earlier stabilization in comparison GPT-2.

(4) Introduces a SwiGLU-based MoEP variant, whose learning behavior is more similar to GPT-2, but which struggles to match its performance.

Proceedings of the First BabyLM Workshop, pages 540–547

2 Related Works

2.1 Sparse and Routing-Based Models

Mixture-of-Experts (MoE) architectures [9, 13] introduced sparse token-wise routing within feed-forward layers, enabling models to increase capacity without a proportional increase in computational density. Follow-up works such as GLaM [6], DeepSeek-V2 [15], and OLMoE [10] extended this idea with improved routing strategies. More recently, approaches like MoR [1] explored layer-level routing, where different tokens may skip or use fewer layers. These works reflect a broader trend toward architectures that diversify token computation paths beyond uniform dense stacks.

Our work aligns with this trajectory but integrates routing both across parallel Transformer blocks and within MoE experts.

2.2 Parallel Architectures

As alternative to dense Transformer architecture design, some works have explored parallelization as purpose to increase expressiveness or efficiency. PaLM [4] introduced pathway-based scaling, while Branchformer [11] combined MLP and attention to parallel components. Our prior work, PaPaformer [14], proposed a alternative approach, combining independently trained parallel paths into larger composite models. MoEP is build on this line by maintaining parallelism but coupling it with more MoE style top k routing.

2.3 Tiny Language Models

Evaluating new architectures at small scale has become increasingly important, as recent results show that novel architectural methods can notably improve model performance, while the size of current Large Language Models (LLMs) limits the threshold for exploring such innovations. However, small datasets such as **TinyStories** [8] and **BabyLM** [3] enable rapid iteration with models under 100M parameters. The BabyLM challenge explicitly emphasizes architectural innovations under a 100M and 1B-word budget and provides a comprehensive evaluation-pipeline[2].

MoEP is designed within this paradigm: small enough for fast training, yet still large enough to be reasonably evaluated on benchmark suites such as BLiMP [19] and SuperGLUE [18].

2.4 Decoder-Only Baselines

Dense decoder-only Transformers have long been the standard for autoregressive modeling, exemplified by **GPT-2** [12], **GPT-3** [2], **LLaMA-2** [17], and **LLaMA-3** [7], as well as Google's **Gemini** models [16]. These baseline models provide strong performance, but process every token through the same layers and routes. MoEP is directly compared against GPT-2 under matched data, optimization settings, and training conditions to isolate the effect of modular sparse routing.

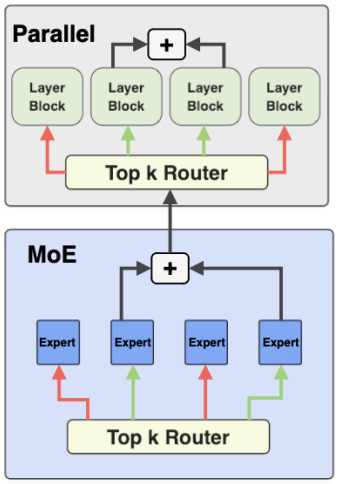

Figure 2: Overview of MoEP routing structure. Each token is routed through a sparse subset of experts in a Mixture-of-Experts (MoE) block, followed by a top-k routed selection of Layer Blocks in a Parallel stack. The routers select k components, whose outputs are summed. This design allows different tokens to follow distinct computation paths through both experts and parallel layers.

3 Methodology

3.1 Overview of MoEP architecture

MoEP and **MoEP-SwiGLU** (see Figure 1), is a decoder-only model that interleaves two standard (dense) *Large Layers* with a sparse middle stack: **Layer Block** (full size) → **MoE** (shrink) → **Parallel Layer** N times with top-k routing → **MoE** (grow) → **Layer Block**.

The first full size **Layer Block** operates at a higher hidden dimension d_{L}. A shrinking **MoE Block** uses E experts with top-k routing, where experts are either simple Linear layer or SwiGLU, which map projection in to smaller hidden dimension d_{P} suited for the routed parallel stack (see Figure 2). **Parallel Layer** uses top-k routing among P **Layer Blocks**, which uses hidden dimension d_{P}. After N Parallel Layers, a growing **MoE Block** projection maps back from d_{P} to d_{L} before the second Large Layer.

3.2 Parallel Layers (Block Routing at Smaller Dimension)

Each **Parallel Layer** contains P Transformer blocks $\{B_1, B_2, \ldots, B_K\}$, which are architecturally equivalent to the full size **Layer Block**, but operates at the reduced dimension d_{P} (same sublayer structure, distinct parameters) and **Router**, which is simple Linear Layer size d_{P} x P. In token-level, **Router** applies **top-k** selection among P **Layer Block** and routed inputs are summed together. This routing method allows

[2]https://github.com/babylm/evaluation-pipeline-2025/

different tokens traverse with different subsets of blocks within each Parallel Layer.

Stacking N **Parallel Layers** yields a deep routed path in compact dimensions.

3.3 MoE Projections (Shrink and Grow)

The two **MoE Block** projections implement the dimensionality transitions:

$$\text{shrink: } d_{\text{L}} \to d_{\text{P}}, \qquad \text{grow: } d_{\text{P}} \to d_{\text{L}}.$$

Each **MoE Block** consists of E experts and a token-level top-k routing over experts. In the base **MoEP** model, experts are simple *linear* projections and in **MoEP-SwiGLU**, experts use *SwiGLU*-based feed-forward projections.

3.4 Routing Objective and Training Loss

To avoid expert and block collapse, in training phase we used a standard load-balancing regularizer.

Let p_i denote the average routing probability assigned to block or expert i over a batch. The balancing term is

$$\mathcal{L}_{\text{balance}} = -\sum_i p_i \log p_i,$$

computed separately for block routing and expert routing. The total objective is

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda^{\text{block}} \mathcal{L}^{\text{block}}_{\text{balance}} + \lambda^{\text{expert}} \mathcal{L}^{\text{expert}}_{\text{balance}},$$

where $\mathcal{L}_{\text{CE}}$ is the next-token cross-entropy loss and λ learning weight.

4 Training and Experimental Setup

Training Data

For training, we used only the **BabyLM** [3] strict-small dataset, without any additional text preprocessing. The corpus contains a little over 10 million words, drawn from curated English sources including **CHILDES**, **BNC Spoken**, **Gutenberg**, **OpenSubtitles**, **Simple Wikipedia**, and **Switchboard**. No external data were added, in order to ensure direct comparability with the baseline submissions in the track.

Tokenization

All models were trained with the same **GPT-2** style **byte-pair encoding** (BPE) tokenizer. We use a fixed vocabulary of 16K tokens, trained on the **BabyLM** strict-small corpus. This size balances compactness with adequate coverage of rare subwords. The tokenizer follows a similar pattern-recognition strategy as **babylm-baseline-10m-gpt2** [3], avoiding the need for training data preprocessing and ensuring maximal similarity with the **BabyLM** baseline models.

Training Procedure

MoEP, **MoEP-SwiGLU**, and **GPT-2** baseline model were trained from scratch under identical training settings and with causal language modeling objective. We used **AdamW** with **cosine learning rate decay** for stable model training with standard **dropout** and **weight decay** regularization. Initially, we pre-tokenized the training data with a **stride of 128**. During training, examples were randomly sampled from the full pre-tokenized dataset using an epoch-based shared seed, ensuring that all models were trained on the same examples.

Each model was trained for 10 epochs, with training in each epoch stopped after the model had seen approximately 10M words.

Checkpoints were saved every 1M words up to 9M words, and subsequently every 10M words up to 100M words. After training, we ran fast evaluation on all checkpoints, and the final model weights were taken from the checkpoint with the best evaluation performance. These weights were then used for full evaluation. **MoEP** and **GPT-2** achieved their best accuracy at 30M words, while **MoEP-SwiGLU** reached its peak after 80M words.

A model hyperparameters (hidden dimension, number of layers, parameter counts) were selected to match with BabyLM baseline models and these are listed in Appendix A and Appendix B.

Evaluation Protocol

Evaluation followed the official BabyLM pipeline [3]. Zero-shot evaluation included **BLiMP**, **EWOK**, **WUG**, and other tasks, with the full list available in the evaluation pipeline documentation[4]. For tasks involving finetuning (e.g., **MNLI**, **QQP**, **RTE**), the **BabyLM** evaluation pipeline supplied both training data and default finetuning parameters, which we adopted directly.

Environment

All experiments were conducted with single NVIDIA A100 GPU in CSC's Puhti supercomputing environment [5]. Training a single model for 10 epochs required approximately 1–2 hours, a duration that could be further reduced with code optimizations. All code is implemented using **PyTorch** and **Hugging Face** libraries and released for reproducibility [5] and model is directly downloadable in Hugging Face [6].

5 Results

5.1 Evaluation Scores

As table 1 shows, MoEP achieved the highest performance across all models, including the official **BabyLM** baselines under the strict-small track, when the **AoA** task score was included in the **Macro Average**. Even when excluding AoA from the macro average, MoEP still outperformed the our and the official BabyLM GPT-2 baseline, which we consider our primary comparison point due to the similarity (MoEP-SwiGLU) or full correspondence (MoEP) in layer architecture. MoEP also obtained the best score in five individual tasks, the highest count among all models evaluated.

Among our models, the GPT-2 variant slightly outperformed the **BabyLM GPT-2 baseline** in macro average without **AoA**, reaching performance comparable to **MoEP**. However, subsequent analysis revealed a key distinction - **MoEP** extracted useful patterns earlier during training. This indicates that modular sparse routing can provide better sample efficiency, even if final scores converge to similar levels.

[3] https://huggingface.co/BabyLM-community/babylm-baseline-10m-gpt2

[4] https://github.com/babylm/evaluation-pipeline-2025/

[5] https://github.com/Jtapsa/BabyLM-2025

[6] https://huggingface.co/Jtapsa/moep

Model	Zero-shot Tasks							Finetuned Tasks							Macro
	BLiMP	EWOK	Entity	WUG	Comps	Reading	AoA	BoolQ	MNLI	MRPC	MultiRC	QQP	RTE	WSC	Avg
Our Models															
GPT-2	59.70	**57.85**	13.15	36.00	51.20	6.40	–	67.50	49.10	69.60	66.70	71.55	**62.60**	63.45	48.10 –
MoEP [7]	59.15	50.20	**35.65**	33.00	50.70	**6.70**	53.70	66.20	48.10	70.10	64.50	70.75	**62.60**	**67.30**	49.00 **44.50**
MoEP (SwiGLU) [8]	60.35	49.50	17.10	36.50	51.35	6.60	–	66.30	48.30	70.60	67.25	69.40	54.70	61.55	47.70 –
HF Baselines															
GTP-2 [9]	61.75	49.90	13.90	30.55	51.70	6.50	11.7	52.10	33.10	67.60	57.50	63.60	56.10	61.50	46.60 37.40
GPT-BERT (causal) [10]	**67.45**	49.50	34.60	36.05	52.80	**6.70**	-3.90	**68.10**	46.90	74.50	**68.30**	76.70	56.10	65.40	**54.10** 41.20
GPT-BERT (focus-causal) [11]	62.35	49.5	31.10	32.70	**52.90**	6.50	3.8	67.60	51.80	**78.90**	67.40	**77.40**	57.60	61.50	53.65 40.00
GPT-BERT (mixed-causal) [12]	65.60	50.20	25.40	**48.50**	25.00	6.40	14.50	66.70	**53.30**	77.50	67.00	76.60	55.40	63.50	52.40 39.20

Table 1: Evaluation scores on BabyLM tasks for our models (top) and Hugging Face baseline models (bottom). Two macro averages are reported: the first excludes the AoA result obtained from the Hugging Face leaderboard, while the second represents the overall text-average. In table, BLiMP refers to the average over BLiMP and BLiMP-supplement, WUG corresponds to the average of Wug Adjacency and Wug Past Tense, and Readings is the average of Eye Tracking and Self-Paced Reading tasks.

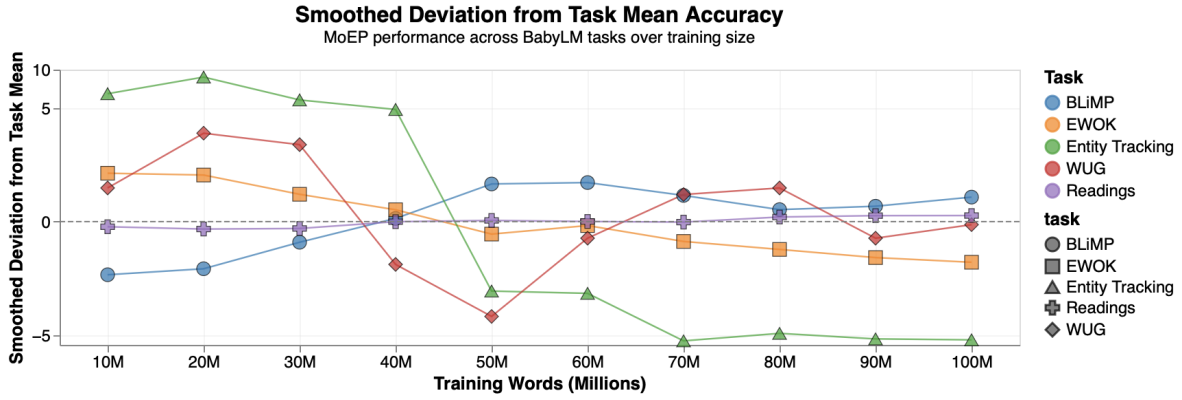

Figure 3: Smoothed deviation from task mean accuracy for MoEP. The dashed origin line represents the average result, while smoothed deviation shows the task accuracy at specific checkpoint relative to the mean.

By contrast, **MoEP-SwiGLU** did not reach the same level of performance. This suggests that lightweight linear experts are more effective at the small scale, whereas **SwiGLU** based feed-forward experts require longer training to stabilize and still achieve lower overall scores compared to the other models.

Note that our **GPT-2** and **MoEP-SwiGLU** results do not include **AoA** scores, which are provided in the official **BabyLM** leaderboard.

5.2 Analysis of Training Development

To better understand how each model architecture learns over training, we analyze their fast-evaluation scores across checkpoints. In the following training dynamics analysis, **BLiMP** refers to the average over BLiMP and BLiMP-supplement, **WUG** corresponds to the average of Wug Adjacency and Wug Past Tense, and **Readings** is the average of Eye Tracking and Self-Paced Reading tasks.

MoEP

Figure 3 presents results for the **MoEP** model. Compared to **GPT-2** (see Figure 4), **MoEP** exhibits more comprehensive early learning, reaching peak performance at the 30M checkpoint, where nearly all task scores are at or above their task-specific means. After 90M words, deviations regress toward zero, with **Entity Tracking** in particular stabilizing well below the mean. This indicates that **MoEP** quickly learns to achieve near-optimal evaluation performance but later begins to overfit, leading to diminished generalization. The

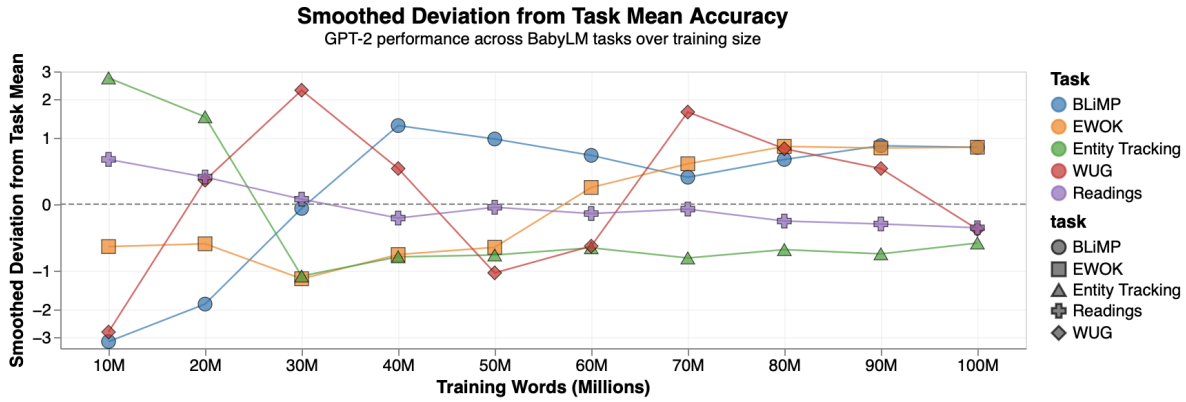

Figure 4: Smoothed deviation from task mean accuracy for GPT-2. Where The dashed origin line represents the average result, while smoothed deviation shows the task accuracy at specific checkpoint relative to the mean.

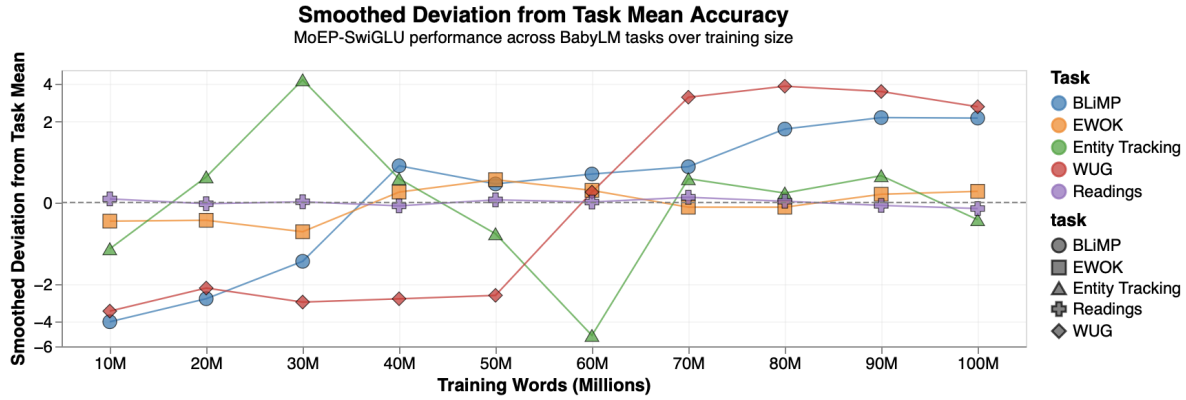

Figure 5: Smoothed deviation from task mean accuracy for MoEP-SwiGLU. The dashed origin line represents the average result, while smoothed deviation shows the task accuracy at specific checkpoint relative to the mean.

pattern highlights that modular routing accelerates initial pattern discovery but may not sustain improvements throughout training.

GPT-2

Figure 4 shows the **GPT-2** baseline smoothed task-mean fast-evaluation results. Unlike **MoEP**, once **GPT-2** reaches its best performance at the 30M checkpoint, it does not stabilize as quickly but continues to improve on certain tasks. On the other hand, after the 70M checkpoint, **WUG** begins to decline and shows no clear signs of stabilization. This reflects a key tradeoff of dense architectures: the model reaches its best scores on different evaluation tasks at different checkpoints rather than converging to a consistent stable state.

MoEP-SwiGLU

Unlike **MoEP**, **MoEP-SwiGLU** (see Figure 5) shows a development more similar to **GPT-2**. The model exhibits strong late-phase improvements on **WUG** and **BLiMP**, with performance rising steadily after 60M words, while other tasks begin to stabilize. **MoEP-SwiGLU** reaches its best performance at the 80M checkpoint, much later than the other models. As with **MoEP** and **GPT-2**, **Entity Tracking** shows strong instability, where early gains at the first checkpoints collapse sharply afterward. These results suggest that SwiGLU-based experts can improve performance on certain tasks, while other evaluations stabilize without the declines observed elsewhere.

Comparative Trends

MoEP learns rapidly during its early checkpoints, showing early specialization particularly on **Entity Tracking** and **WUG**. After the peak at the 30M checkpoint, however, subsequent checkpoints achieve notably lower evaluation scores.

MoEP-SwiGLU achieves the strongest late-phase gains on **WUG**, but this comes at the cost of weaker performance on **Entity Tracking** and **BLiMP**.

GPT-2 shows steadier learning and after reaching its peak performance it experiences fewer dramatic changes in later checkpoints compared to the MoEP variants.

In conclusion, these metrics illustrate that sparse modular routing can accelerate early learning but also introduces instability. The choice of expert type (**linear** vs. **SwiGLU**) further shifts the balance between stability and specialization.

6 Discussion

Limitations

Despite promising results, **MoEP** and **MoEP-SwiGLU** were trained only on a small dataset. It therefore remains unclear whether scaling up the model size and training data would preserve their relative performance compared to **GPT-2**. Within **BabyLM**, where the training corpus and patterns to be learned are relatively simple, smaller-dimensional parallel blocks can capture these patterns as effectively as dense **GPT-2** layers. With more complex data, however, parallel layers may no longer operate effectively at reduced dimensionality, forcing an increase in total parameters that could exceed those required by a dense **GPT-2** to learn the same patterns.

This work also did not include a detailed analysis of expert and block routing. Routing dynamics may have influenced the fast-evaluation results, in way that faster learning observed at early checkpoints could be a consequence of more flexible routing, while the current load-balancing regularizer may have forced overly uniform usage, negatively impacting final evaluation scores. Finally, due to the small model and dataset scale, the present study focused on evaluation benchmarks only and did not investigate generation capabilities. Such analysis might reveal additional differences between sparse **MoEP** and dense **GPT-2** architectures.

Architectural Takeaways

The experiments suggest three main lessons:

- Sparse block and expert routing accelerates early learning, but overall evaluation scores decline afterwards and do not fully recover. This drop is driven by permanent degradation on certain tasks, which lowers the aggregate performance.
- SwiGLU experts increase task specialization and yield late-phase gains, but also amplify volatility and fail to achieve overall results comparable to linear experts.
- Parallel models can match or even outperform dense architectures in the BabyLM strict-small setting. This shows that lower-dimensional sparse paths are sufficient to capture relatively simple language patterns.

Future Work

Future extensions of MoEP could explore:

- Scaling the number of parallel blocks and MoE experts beyond the current four to further increase model sparsity.
- Testing alternative expert architectures in the MoE projections.
- Exploring different load-balancing regularization strategies and analyzing their effects on learning dynamics and evaluation performance.
- Extending evaluation to larger and more complex training datasets to test whether MoEP retains its ability for fast learning and stable evaluation performance.

7 Conclusion

We presented *MoEP*, a sparse decoder-only architecture that combines top-k routing across parallel blocks with linear and feed-forward Mixture-of-Experts projections, allowing the model to flexibly adjust dimensionality across layers.

Within the BabyLM strict-small track, **MoEP** outperformed all official BabyLM baseline models, not only **GPT-2** (the architecture on which it is based), even though **GPT-2** itself was the weakest among the BabyLM baselines.

Our analysis demonstrates a tradeoff in which sparse modular routing accelerates early learning but also introduces higher training variance, with performance often peaking early and then declining. The **MoEP-SwiGLU** variant further showed that expert design directly influences both learning speed and stability. This may be due to the increased parameter size, which is an effect of the **MoEP-SwiGLU** expert design, although the task based learning behavior is neither similar nor stable compared to **MoEP**.

These findings suggest that layer-level sparse, routing-based architectures provide a viable path toward sample-efficient language modeling, even under small-scale budgets. Future work will focus on improving learning stability, optimizing sparse computation, and extending modular expert routing to larger-scale settings.

Acknowledgments

This work was made possible through computation environments provided by the University of Oulu ICT services. I would like to thank Prof. Mourad Oussalah and MSc. Moinul Islam for their valuable feedback and helpful suggestions, which contributed to the development of this research.

References

[1] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. 2025. Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation. arXiv:2507.10524 [cs.CL] https://arxiv.org/abs/2507.10524

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[3] Lucas Charpentier, Leshem Choshen, Ryan Cotterell, Mustafa Omer Gul, Michael Hu, Jaap Jumelet, Tal Linzen, Jing Liu, Aaron Mueller, Candace Ross, Raj Sanjay Shah, Alex Warstadt, Ethan Wilcox, and Adina Williams. 2025. BabyLM Turns 3: Call for papers for the 2025 BabyLM workshop. arXiv:2502.10645 [cs.CL] https://arxiv.org/abs/2502.10645

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, and et al. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs.CL] https://arxiv.org/abs/2204.02311

[5] CSC – IT Center for Science. [n. d.]. Puhti Supercomputer. https://docs.csc.fi/computing/systems-puhti/. Accessed: 2025-05-06.

[6] Nan Du, Le Hou, Aitor Zhang, Anton Bakhtin, Nathan Scales, Zhifeng Dai, Xin Li, Shixiang Xie, William Fedus, Mostafa Dehghani, and et al. 2022. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=k7K6kB9td9

[7] Abhimanyu Dubey and et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[8] Ronen Eldan and Yuanzhi Li. 2023. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759 [cs.CL] https://arxiv.org/abs/2305.07759

[9] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 8473–8483. https://proceedings.neurips.cc/paper/2021/hash/2c5dc10619a37b0c79ef595e0bda0592-Abstract.html

[10] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe

Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. 2025. OLMoE: Open Mixture-of-Experts Language Models. arXiv:2409.02060 [cs.CL] https://arxiv.org/abs/2409.02060

[11] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. 2022. Branchformer: Parallel MLP-Attention Architectures to Capture Local and Global Context for Speech Recognition and Understanding. arXiv:2207.02971 [cs.CL] https://arxiv.org/abs/2207.02971

[12] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* (2019).

[13] Noam Shazeer, Azalia Mirhoseini, Andrew Maziarz, Krzysztof Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=B1ckMDqlg

[14] Joonas Tapaninaho and Mourad Oussala. 2025. PaPaformer: Language Model from Pre-trained Parallel Paths. arXiv:2508.00544 [cs.CL] https://arxiv.org/abs/2508.00544

[15] DeepSeek Team. 2024. DeepSeek V2: Scaling Vision-Language Models with Mixture of Experts. arXiv:2401.00733 [cs.CL]

[16] Gemini Team and et al. 2023. Gemini: A Family of Highly Capable Multimodal Models. *Google DeepMind Technical Report* (2023).

[17] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Sharan Batra, Akshat Bhargava, Shruti Bhosale, et al. 2023. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[18] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems*, Vol. 32.

[19] Alex Warstadt, Yining Cao, Jun Ho, Ellie Pavlick, and Samuel R Bowman. 2020. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics* 8 (2020), 377–392.

A Model Hyperparametrs

Comparison of architectural hyperparameters across model variants.

Hyperparameter	GPT-2	MoEP	MoEP SwiGLU
Vocabulary size	~ 16K	~ 16K	~ 16K
d_{model}	384	384 / 192	384 / 192
Layers	12	2 / 10	2 / 10
Parallel blocks	-	4	4
Heads	6	6 / 3	6 / 3
Head dimension	64	64	64
FF multiplier	4	4	4
FF type	MLP	MLP	SwiGLU
MoE FF type	-	Liner	SwiGLU
N experts	-	4	4
Top k	-	2	2
Normalization	LN	LN	LN
Attention	MHA	MHA	MHA
Train seq len	512	512	512
Total Parameter (millions)	28M	28M	38M

Table 2: Architectural hyperparameters of GPT-2, MoEP, and MoEP-SwiGLU.

B Training Setup

Detailed training configurations for all models.

Hyperparameter	Value
Optimizer	AdamW
Learning rate	3×10^{-4}
Batch size	16
Training epochs	10
Gradient accumulation steps	1
Weight decay	0.1
Adam betas	(0.9, 0.95)
Adam epsilon	1×10^{-8}
Scheduler type	Cosine
Warmup steps	800
Random seed	42

Table 3: Training setup and optimization parameters.