Hyper Experts: Language Models With Inference-Time Layer Reallocation

Ilya Deriy
Sakana AI, Japan
ideriy.physics@gmail.com

Edoardo Cetin
Sakana AI, Japan
edo@sakana.ai

Abstract

We present **Hyper Experts**, a new architectural class of language models capable of test-time layer reallocation. By combining all the multi-layer perceptrons in a single cross-layer shared pool of experts, our new architecture is able to construct specialized computational paths for each input token directly at inference time. To validate our approach, we train Hyper Experts models and dense Transformer baselines across different model backbone scales to study and compare the memory-computation trade-offs between old and new designs. Empirically, we show that Hyper Expert models outperform dense baselines, which we attribute to the computational flexibility afforded by our cross-layer expert-sharing principle. We present architectural and training guidelines, together with an analysis of expert similarity and routing efficiency, to identify the key properties and untapped potential of our new architecture.

1 Introduction

Different tasks often require different skill sets. The same is true in language modeling, where optimal token processing across semantic strata may require large language models (LLMs) to possess vast knowledge and capabilities. Endowing a model with a broad spectrum of functions typically requires a commensurate increase in the number of model parameters. However, in architectures with a fixed computational path, such as dense Transformers, a high parameter count may lead to substantial overcomputation because each input token is processed by the full parameter set.

A prominent solution comes in the form of inference-time conditional computation methods like Mixture-of-Experts (MoE) Shazeer et al. (2017); Dai et al. (2024). By selectively routing tokens to specialized parameter subsets—called *experts*—based on their semantic representations, MoE enables highly targeted activation of parameters and improves overall parameter utilization. However, existing MoE architectures are limited to *layer-wise* parameter reallocation, i.e., each token can be assigned to parameters only within the current layer.

In this paper, we present **Hyper Experts** (HX), a new architectural class of language models capable of test-time layer reallocation. Instead of splitting the parameters of the model's multi-layer perceptrons (MLPs), as in traditional Mixture-of-Experts models, our new architecture aggregates them into a single pool of experts, which is shared across all model layers. As a result, Hyper Experts models are capable of learning cross-layer parameter reallocation, enabling deeper test-time optimization of the model's computational paths. We empirically demonstrate that these architectural flexibility and optimization capabilities can enable substantial improvements in the model performance, allowing Hyper Experts models to achieve a lower final training loss and higher evaluation metric scores compared to dense baselines. We show that our cross-layer expert sharing technique improves the parameter utilization, allowing Hyper Experts models to achieve the performance of a larger model, solely by activating each expert more than once per forward pass. Finally, we analyze expert utilization during inference, elucidating key properties of the Hyper Experts architecture and identifying avenues for future advances.

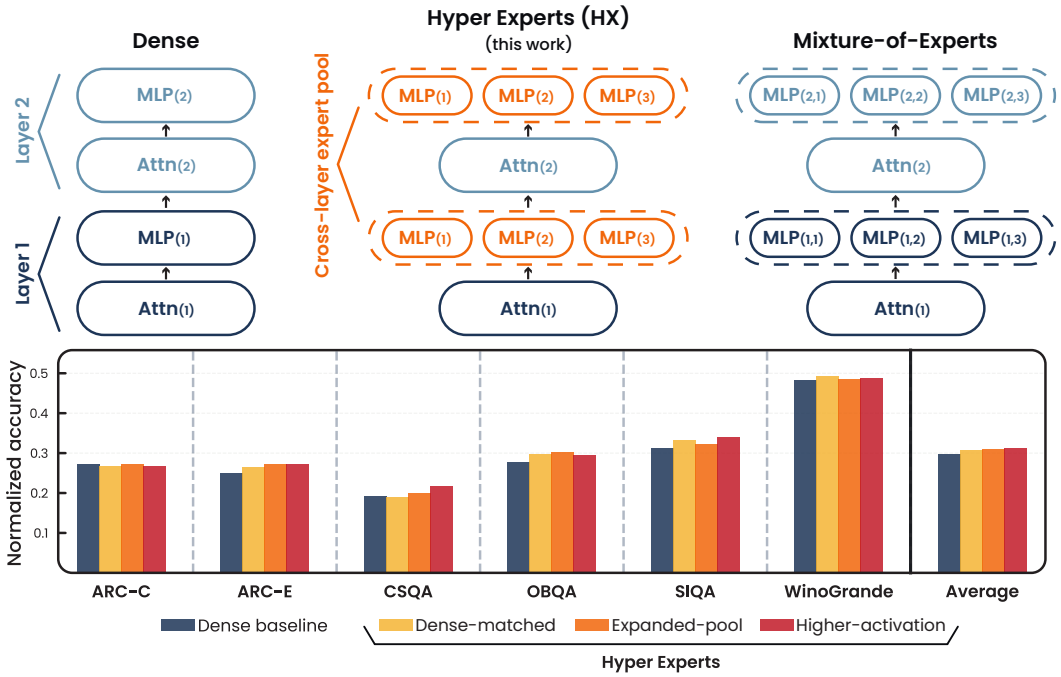

Figure 1: **Hyper Experts** (HX) – our new architectural class capable of learning cross-layer test-time parameter reallocation enabled by the pool of model-wide shared experts.

2 Hyper Experts

A transformer model with L layers is structured as a sequential composition of self-attention layers and feed-forward networks (FFNs) Vaswani et al. (2017). In a standard *dense* Transformer, each FFN is implemented as a single multi-layer perceptron. Mixture-of-Experts architecture expands this idea by splitting the parameters of each feed-forward block into several sub-networks, called "experts" Shazeer et al. (2017); Zoph et al. (2022); Yang et al. (2025a). The FFN output in an MoE model is a router-weighted sum of expert outputs, typically taken over the top-k selected experts.

In *Hyper Experts*, we adapt this idea of expert-based conditional computation, but instead of splitting the parameters of the MLPs into several sub-networks, we gather all MLPs in a single pool of cross-layer shared experts $\mathcal{M}$. The output of a single FFN block in a layer ℓ of the Hyper Experts model, given an input token representation $\hat{x}_t$, is

$$\mathrm{FFN}_\ell\left(\hat{x}_t\right) = \sum_{k \in \mathcal{M}_{\ell,t}} r_{\ell,t,k} \mathrm{MLP}_k\left(\hat{x}_t\right), \quad \mathcal{M}_{\ell,t} = \mathrm{TopK}\left(\mathcal{M}, \hat{r}_{\ell,t}\right), \tag{1}$$

where $r_{\ell,t,k}$ is the routing weight for the token t and expert k at layer ℓ.

Following established MoE practice, we implement the expert router as a linear layer followed by a softmax activation function (Dai et al., 2024; Yang et al., 2025a). A key design choice in the Hyper Experts architecture is that router parameters are *layer-specific*, even though the expert parameters are shared across layers. This choice stems from the fact that token representations in adjacent layers tend to be highly correlated, largely due to the residual connections used in modern Transformers He et al. (2016). Thus, when the router parameters are shared across layers, we find that this correlation can limit the degree of routing flexibility and hinder training convergence.

To mitigate premature router convergence and foster expert explorations, we use the canonical load-balancing-loss (LBL) Fedus et al. (2022); Dai et al. (2024) defined as

$$\mathcal{L}_{\mathrm{LB}} = \frac{\beta E}{L} \hat{f} : \hat{p}, \quad f_{\ell,k} = \frac{1}{T} \sum_t 1\left(k \in \mathcal{M}_{\ell,t}\right), \quad p_{\ell,k} = \frac{1}{T} \sum_t r_{\ell,t,k}, \tag{2}$$

where β is a scaling coefficient, and the tensors $f_{\ell,k}$ and $p_{\ell,k}$ represent the frequency and probability of choosing an expert k at a layer ℓ.

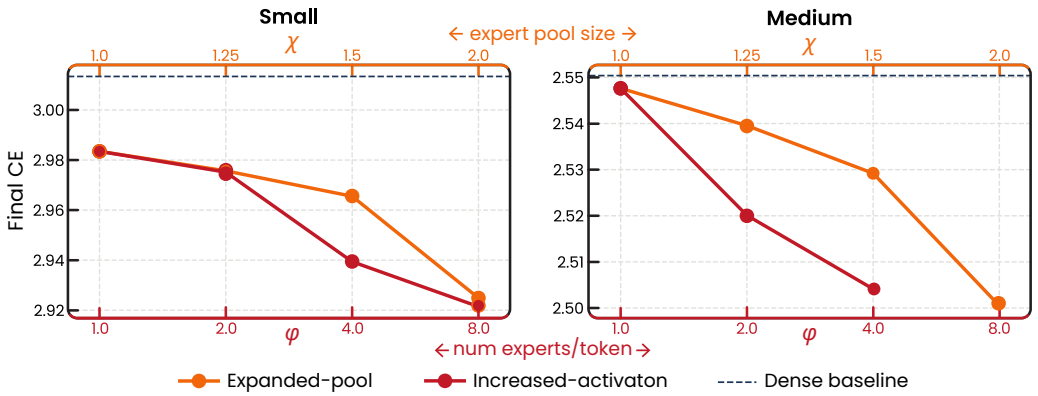

Figure 2: **Hyper Experts outperform dense** Comparison of Hyper Experts models in different configurations: (i) "dense-matched", $\chi = \varphi = 1$; (ii) "expanded-pool", $\chi > 1,\ \varphi = 1$; and (iii) "higher-activation", $\chi = 1,\ \varphi > 1$.

The cross-layer shared expert pool in Hyper Experts allows for independent control over the total and active parameter counts of the model. To conveniently describe this design space, we introduce three factors: the *hyper-experts factor* χ (total parameters), the *FLOPs factor* φ (active parameters), and the *experts' granularity factor* γ, which controls how FFN parameters are partitioned across experts. These factors are defined through the number of layers L, the total number of MLP experts M, the hidden and intermediate dimensions H and D, and the number of experts per token K as follows:

$$M = \chi\,\gamma\,L, \qquad D = 3\,H/\gamma, \qquad K = \varphi\,\gamma. \tag{3}$$

For further architectural details, see Appendix B.

3 Results

We consider models of two scales: *small* (82M parameters in the model's backbone), and *medium* (426M). We train all models on the FineWeb-Edu dataset Penedo et al. (2024) and use the final loss as a validation metric because we train sub-epochs. Before the primary set of experiments, we conduct hyperparameter sweeps (see Appendix C for the detailed discussion). We find that optimal values of learning rate are the same for both dense and Hyper Experts architectures, and are $\alpha = 10^{-3}$ for the *small* models and $\alpha = 5 \times 10^{-4}$ for the *medium* models. The optimal values of the LBL coefficient are $\beta = 0$ (*small*) and $\beta = 10^{-5}$ (*medium*). In this work, we focus on three HX configurations: (i) a "dense-matched" configuration ($\chi = \varphi = 1$); (ii) an "expanded-pool" configuration ($\chi > 1,\ \varphi = 1$); and (iii) a "higher-activation" configuration ($\chi = 1,\ \varphi > 1$), each studied within the the "dense-like" architectural regime ($\chi \approx \varphi$). For the discussion of the "MoE-like" overparametrized regime ($\chi \gg \varphi$) see Appendix F.

Training performance. Our results show that Hyper Experts models outperform dense baselines even when their total and active parameter counts are matched (see Fig. 2). We attribute enhanced performance of the Hyper Experts models to their ability to create task- and domain-specific subnetworks during inference, enabled by our experts' sharing principle, which enhances information compression and optimizes parameter usage. Moreover, we find that across both scales, HX models in the "higher-activation" configuration were able to match the performance gains of their expanded-pool" counterparts, solely by activating each expert more than once per forward pass, without increasing the model size. Importantly, the relative gains from higher expert activation rates increase with the model size: *medium*-sized Hyper Experts models required a 30% smaller relative increase in the expert activation rate than *small*-sized HX models. This trend aligns with prior observations that parameter underutilization in LLMs increases with model size Sun et al. (2023); He et al. (2024) and suggests that "higher-activation" HX configurations enhance parameter utilization, providing a path to more compute-efficient models.

Downstream task evaluation. We next test whether the training-loss improvements of Hyper Experts translate into stronger downstream reasoning performance. To do so, we evaluate *medium*-scale Hyper Experts models in all three configurations alongside a dense

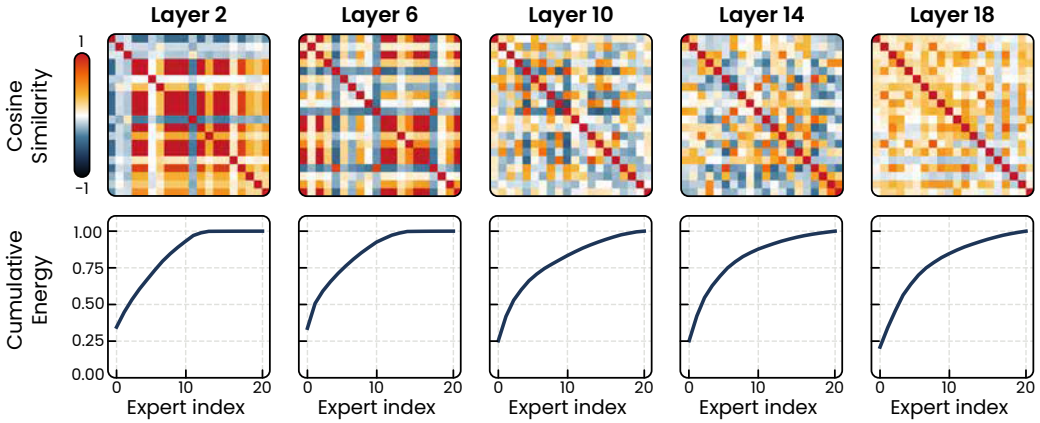

Figure 3: **Expert diversity analysis.** Cosine similarity between expert vectors extracted from the router-parameter matrix of the *medium*-sized Hyper Experts model in the "dense-like" configuration, and the cumulative energy of the matrix's singular spectrum.

Transformer baseline on a suite of multiple-choice reasoning benchmarks (see Appendix D). We find that downstream performance closely tracks final training loss: all three Hyper Experts variants outperform the dense baseline. Among HX variants, the "higher-activation" variant achieved the best aggregate performance, followed by the "expanded-pool", as shown in Fig 1.

Expert diversity analysis We theorize that our cross-layer expert sharing technique can be naturally composed with dynamic compute allocation methods to achieve complementary benefits. We support this claim with an expert diversity analysis: we measure the pairwise cosine similarity between expert vectors extracted from the router parameter matrix of the "dense-matched" *medium*-sized Hyper Experts model and the cumulative energy in the matrix's singular spectrum. We observe substantial redundancy among subsets of experts in earlier layers and almost orthogonal expert preferences in later layers (see Fig. 3). Our findings are in alignment with the fact that earlier layers specialize in simpler syntactic and lexical analysis rather than in processing rich contextual and semantic features Jawahar et al. (2019), and suggest that Hyper Experts models may benefit from dynamic compute allocation both inside and between layers, which is naturally supported by the flexibility of the cross-layer shared expert pool.

Architectural choice study. We find that a single model-global router and increased expert granularity degrade the model's performance, whereas additional post-top-k normalization of routing weights has no effect. We provide a summary and comprehensive discussion of our architectural exploration results in Appendix E.

4 Conclusions

We introduced Hyper Experts, a new class of language models capable of inference-time cross-layer parameter reallocation. This ability is enabled by combining all the model's MLPs into a set of experts shared across all layers. Sharing all MLPs enables dynamic routing via the MLP blocks, supporting complex nonlinear, non-sequential computation paths and allowing the selection of token-specific subnetworks directly during inference, thereby improving the model's expressiveness and performance. We empirically demonstrate that these properties enable HX to outperform dense models even when active and total parameter counts are matched.

HX's architectural flexibility, stemming from the experts sharing principle, allows for a fine-grained control over the model's total capacity and compute. We show that intra-layer expert reuse enhances parameter utilization in the model without increasing its memory footprint, thereby enabling HX to achieve the performance level of larger models. Moreover, our results indicate that HX can be naturally combined with dynamic compute allocation methods to create even more performative models.

Altogether, Hyper Experts represent a principled path toward highly adaptable, both memory- and compute-efficient models, thereby opening a broader design space for future large-scale language model architectures.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Jun Bai, Minghao Tong, Yang Liu, Zixia Jia, and Zilong Zheng. Understanding and leveraging the expert specialization of context faithfulness in mixture-of-experts llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21938–21953, 2025a.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. Diep: Adaptive mixture-of-experts compression through differentiable expert pruning. *arXiv preprint arXiv:2509.16105*, 2025b.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. Moeut: Mixture-of-experts universal transformers. *Advances in Neural Information Processing Systems*, 37:28589–28614, 2024.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

Hongcheng Guo, Juntao Yao, Boyang Wang, Junjia Du, Shaosheng Cao, Donglin Di, Shun Zhang, and Zhoujun Li. Cluster-driven expert pruning for mixture-of-experts large language models. *arXiv preprint arXiv:2504.07807*, 2025a.

Wentao Guo, Mayank Mishra, Xinle Cheng, Ion Stoica, and Tri Dao. Sonicmoe: Accelerating moe with io and tile-aware optimizations. *arXiv preprint arXiv:2512.14080*, 2025b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. What matters in transformers? not all attention is needed. *arXiv preprint arXiv:2406.15786*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

RA Jacobs, MI Jordan, SJ Nowlan, and GE Hinton. [a]adaptive mixtures of local experts, [o] neural computation, vol. 3. 1991.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pioro, Michal Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Krol, Tomasz Odrzygozdz, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. In *Forty-first International Conference on Machine Learning*, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Aribi, Bautista Rozendaal, Diganta Misra, Clémentine Fourrier, Jade Copet, Hady Elsahar, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024. URL `https://arxiv.org/abs/2406.17557`.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Skymizer. Fineweb-edu deduplicated 45b. `https://huggingface.co/datasets/skymizer/fineweb-edu-dedup-45B`, 2024. Accessed: 2026-01-16.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Zheyue Tan, Zhiyuan Li, Tao Yuan, Dong Zhou, Weilin Liu, Yueqing Zhuang, Yadong Li, Guowei Niu, Cheng Qin, Zhuyu Yao, et al. Rexmoe: Reusing experts with minimal overhead in mixture-of-experts. *arXiv preprint arXiv:2510.17483*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yuanhang Yang, Chaozheng Wang, and Jing Li. Umoe: Unifying attention and ffn with shared experts. *arXiv preprint arXiv:2505.07260*, 2025b.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A Notation

All tensors except rank-0 tensors are denoted as italic letters with a hat, i.e., $\hat{x}$. Rank-0 tensors (scalars) are denoted simply as italic letters, i.e., x. When a tensor is indexed, the indexing starts from the leftmost dimension, i.e., if $\hat{x} \in \mathbb{R}^{a\times b\times c\times d}$, $\hat{x}_{i,j} \in \mathbb{R}^{c\times d}$. A single element of a tensor is written in a scalar notation with the same letter, i.e., if $\hat{x} \in \mathbb{R}^{a\times b\times c\times d}$, an element of this tensor will be denoted as $x_{ijkl} \in \mathbb{R}$. Sets are denoted as calligraphic letters, e.g., $\mathcal{A}$ with cardinality of the set being given by the corresponding capital italic letter, i.e. $|\mathcal{A}| = A$.

B Architectural Details

To limit the design space and enable a fair comparison across model sizes, we minimize the number of tunable architectural parameters. Architecture of each model is tuned by: the number of layers, L, the number of attention heads per layer, A, the hyper-experts factor χ, the FLOPs factor φ, and the granularity factor γ.

B.1 Architectural Parameters

The total number of MLPs in the model M, the hidden dimension H, the intermediate dimension D, and (where applicable) the number of experts per token K – are defined as

$$M = \chi\gamma L, \quad H = 64A, \quad D = \frac{3}{\gamma}H, \quad K = \varphi\gamma. \tag{4}$$

Values of the parameters for the dense model are recovered by setting $\chi = \varphi = \gamma = 1$. Consequently, with $\chi = \varphi = \gamma = 1$, Hyper Experts match dense models in the total and active parameter counts, as well as in the intermediate dimension.

B.2 Experts

All models share the same MLP architecture – gated linear unit with a sigmoid linear unit activation function Shazeer (2020),

$$\begin{aligned} \mathrm{MLP}(\hat{x}) &= (\hat{u} \odot \hat{g}) \cdot \hat{W}, \\ \hat{u} = \hat{x} \cdot \hat{U}, \quad \hat{g} &= \mathrm{SiLU}\big(\hat{x} \cdot \hat{G}\big), \end{aligned} \tag{5}$$

where $\hat{U} \in \mathbb{R}^{H\times D}$, $\hat{G}^{H\times D}$, and $\hat{W}^{D\times H}$ are the up, gate, and down-projection weights, respectively.

For efficient expert calculations, we adapt MoE implementations from Megablocks Gale et al. (2023), for models with $H < 512$, and SonicMoE Guo et al. (2025b), for all other models. While SonicMoE is more efficient in terms of the runtime, it sets stricter constraints on the model's dimensions, e.g., $H \leq 512$ or $M \mod 4 = 0$. Due to the latter, the total number of layers was chosen to be a multiple of 4, i.e., $L = 4n$ for all models.

B.3 Factors

The total number of the model's backbone parameters (omitting routers) can be calculated as

$$N_{\mathrm{tot}} = \underbrace{4LH^2}_{\mathrm{SelfAttn}} + \underbrace{3MHD}_{\mathrm{MLPs}} = (4 + 9\chi)\, LH^2. \tag{6}$$

Thus, the ratio between the HX's and the dense model total parameter count is

$$\tilde{N}_{\mathrm{tot}} = \frac{4 + 9\chi}{13} \tag{7}$$

Number of active parameters in the model's backbone (omitting routers)

$$N_{\mathrm{act}} = \underbrace{4LH^2}_{\mathrm{SelfAttn}} + \underbrace{3KLHD}_{\mathrm{MLPs}} = (4 + 9\varphi)\, LH^2. \tag{8}$$

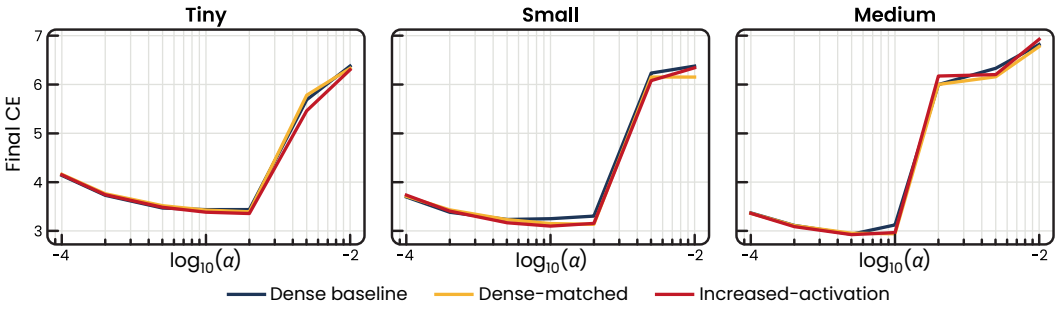

Figure 4: Learning rate tuning results across different architectures and model sizes. Tiny model was trained for $T = 3000$ (chinchilla-optimal $\times$ 10), small model for $T = 3280$ (chinchilla-optimal $\times$ 2), and medium for $T = 2840$ (chinchilla-optimal $\times$ 1/2).

Thus, the ratio between the HX's and the dense model active parameter count is

$$\tilde{N}_{\text{act}} = \frac{4 + 9\varphi}{13} \tag{9}$$

Model's theoretical number of FLOPs, for a single sequence, assuming that multiplication of $a \times b$ by $b \times c$ matrix costs $2abc$ FLOPs, and omitting the cost of activation functions and routers, is

$$F = \underbrace{4LSH\,(2H + S)}_{\text{SelfAttn}} + \underbrace{6KLSHD}_{\text{MLPs}} = 2LSH\,(2S + (4 + 9\varphi)\,H)\,. \tag{10}$$

Thus, the ratio between the HX's and the dense model FLOPs is, assuming $S = sH$,

$$\tilde{F} = \frac{2s + 4 + 9\varphi}{2s + 13}. \tag{11}$$

In addition, it is important to note that values of such quantities as numbers of hyper-experts or number of experts per token, etc., are restricted to integer numbers, which, in turn, sets restrictions on the allowed values of the factors. In practice, we allow all factors to be instances of the `float` and calculate integer-valued quantities defined via factors as `q = int(round(...))`.

C Training Details

We consider models of three size groups: tiny models with $L = 8, A = 6$, small models with $L = 24, A = 8$, and medium models with $L = 20, A = 20$. Training results for the tiny models mirror those of the small- and medium-sized models: HX outperforms the dense baseline in all configurations. However, we find the tiny model's results too noisy; therefore, we omit them from the main text and present them only in the Appendix.

We train all models on the FineWeb-Edu Penedo et al. (2024) dataset, specifically on the deduplicated version Skymizer (2024). We use a sequence length of $S = 2048$ tokens and a batch size of $B = 512$ sequences. While this sequence length may be suboptimal for the tiny model, using the same sequence length is important for a fair comparison across model sizes. Total training steps T, maximal learning rate α, and LBL coefficient β (where applicable) were tuned for each model size. Each model was trained with a constant (max) learning rate for $0.9T$ steps. Over the remaining $0.1T$ steps, we apply a cosine learning rate decay schedule. Final learning rate was calculated as $\alpha_{\text{final}} = \alpha \cdot 10^{-4}$.

We use the AdamW optimizer Loshchilov & Hutter (2017) with following parameters: weight decay $\lambda = 0$, first-moment decay $\beta_1 = 0.9$, second-moment decay $\beta_2 = 0.999$, and numerical stability term $\epsilon = 10^{-8}$. We also apply gradient norm clipping Pascanu et al. (2013) with a clipping value $c = 1.0$.

Before the main training runs, we perform learning-rate tuning in two stages across all architectures and model sizes. In the first stage, we conduct short training runs for approximately 3000 steps. During these runs, we sweep the learning rate from 10^{-4} to 10^{-2} using

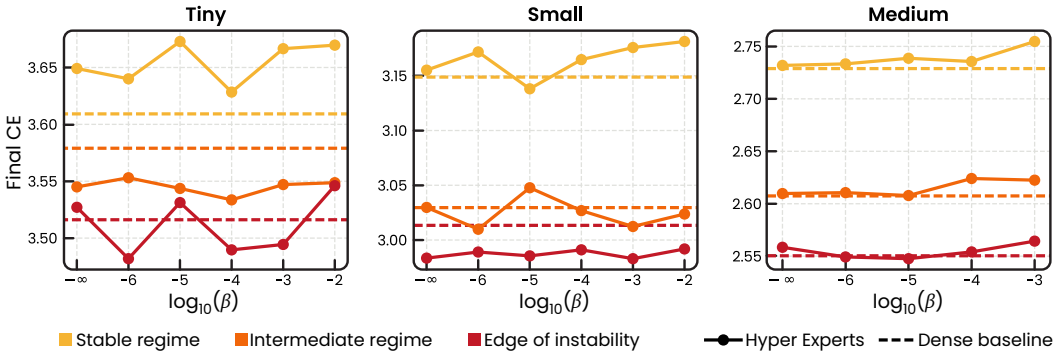

Figure 5: Results of the second stage of hyperparameter tuning for the HX model in $\varphi = \chi = 1$ configuration and the dense baseline. Chosen values of the learning rates correspond to the stable training (yellow), the intermediate regime (orange), and the edge of instability (red). Values of all learning rates for all model sizes are presented in Table 1.

Table 1: Learning-rate values corresponding to stable training, the intermediate regime, and the edge of instability.

	Tiny	**Small**	**Medium**
Stable training	5×10^{-4}	1×10^{-3}	2×10^{-3}
Intermediate	2×10^{-4}	5×10^{-4}	1×10^{-3}
Edge of instability	1×10^{-4}	2×10^{-4}	5×10^{-4}

the "1–2–5" schedule to identify the edge of instability. Throughout this phase, the LBL coefficient is set to 0. Results of the first stage are presented in Fig. 4.

In the second stage, we perform full-length training runs for both the dense baseline and the HX model. Because during the first tuning stage, different HX configurations have shown similar results, during the second stage, we consider only the $\varphi = \chi = \gamma = 1$ HX configuration, and later apply the results to all HX configurations. We train the tiny model for $T = 2400$ steps ($8 \times$ chinchilla), the small model for $T = 6560$ steps ($4 \times$ chinchilla), and the medium model for $T = 17040$ steps ($2 \times$ chinchilla). During the second stage, we again sweep the learning rates near the stability boundary and tune the LBL coefficient for the HX model (see Fig. 5). We consider three learning-rate values, corresponding to fully stable training, the intermediate regime, and the edge of instability. Values of learning rates for each model size are presented in the Table 1.

D Evaluation Details

For evaluation we use the AI2 Reasoning Challenge (ARC; Easy and Challenge splits) Clark et al. (2018), CommonsenseQA Talmor et al. (2019), OpenBookQA Mihaylov et al. (2018), Social IQa Sap et al. (2019), and WinoGrande Sakaguchi et al. (2021) datasets. We use the LightEval framefork **?**, adopting the corresponding default prompt templates provided by LightEval for all datasets. Each benchmark is evaluated as a multiple-choice likelihood task, where the model scores a fixed set of candidate answers given the input context and question. We report the normalized loglikelihood accuracy, following LightEval's built-in evaluation metrics.

E Explored Configurations

In this section, we present a controlled study of various design choices in the Hyper Experts architecture and analyze their effect on model performance. We note that all design-related experiments were performed with a medium-sized HX model in $\varphi = \chi = 1$ configuration.

Table 2: Impact of various design choices on the performance of the *medium*-sized Hyper Experts model in the "dense-matched" configuration.

	Final CE	ΔPPL (↓ %)
Base model	2.54	0.00
Renormalization	2.54	0.00
Increased granularity	2.55	1.01
Single router	2.57	3.05

To measure performance drop, we calculate the relative perplexity increase as

$$\Delta\text{PPL}(\%) = 100 \left(\frac{\exp(\text{CE}_{\text{variant}})}{\exp(\text{CE}_{\text{base}})} - 1 \right). \tag{12}$$

We summarize our findings in Table 2 and discuss their implications below.

E.1 Increased Granularity

Prior works on Mixture-of-Experts models report benefits from increased expert granularity Ludziejewski et al. (2024); Dai et al. (2024). To test whether the same holds for Hyper Experts, we evaluate HX configuration in which each expert is split in half along the intermediate dimension ($\gamma = 2$). We observe small but consistent performance degradation in HX models with such a configuration. However, in traditional Mixture-of-Experts literature, benefits from the increased expert granularity are reported to grow with model size Ludziejewski et al. (2024). Namely, in smaller models, splitting experts can make them too narrow, thus making their expressivity the primary bottleneck. We therefore conjecture that increased granularity may become beneficial in larger Hyper Experts models ($\geq$ 1B parameters).

E.2 Shared Router

We find that sharing router weights across layers consistently degrades performance relative to using layer-specific routers. We hypothesize that this effect is caused primarily by residual connections in the model backbone, which keep token representations highly correlated across adjacent layers. As a result, a shared router may repeatedly select the same subset of experts for a given token across many layers. This behavior is likely suboptimal, as it reduces the diversity of the expert routing decisions.

E.3 Router Softmax and Renormalization

In our default configuration, routing weights $\hat{r}$ are obtained by applying softmax to the router logits *before* top-k selection, with *no additional renormalization* afterward. We do not consider the popular alternative design in which the softmax is applied *after* the top-k operation Shazeer et al. (2017) because, under top-1 routing, this design yields zero gradients with respect to router weights, preventing the router from learning. A plausible alternative is to apply softmax *before* top-k, as in our default settings, but then *renormalize* the selected routing weights with gradient stop, for example, as `r = topk(r)/norm().detach()` in PyTorch-like pseudocode. The gradient stop (`detach` method here) is necessary to avoid zero gradients to router weights in the top-1 case. Empirically, however, this additional renormalization step has no measurable effect on performance.

F Comparison with Mixture-of-Experts

In our preliminary experiments, we compared *medium*-sized Hyper Experts models against the traditional Mixture-of-Experts architecture, matching the total and active parameter counts of both architectures. We find that, under the optimal training regime, which

we found for both architectures by performing the hyperparameter sweeps, MoE models outperformed HX models. However, removing the load balancing loss resulted in the reversed outcome with Hyper Experts outperforming MoE. Thus, we theorize that in the overparametrized regime, the potential benefits of the cross-layer expert sharing are nullified by the increased complexity of the expert routing, which is known to become increasingly difficult as the number of experts increases Bai et al. (2025a). Namely, in our experiments, we consider MoE models with 16 experts in each layer, resulting in the total model size $\approx 10\times$ the parameters of the "dense-matched" configuration. Thus, the cross-layer shared expert pool in the Hyper Experts model had 320 experts (number of experts in one layer $\times$ number of layers), making the expert routing and load balancing much harder than in the MoE model. However, we note that due to the high redundancy of experts across neighboring layers Guo et al. (2025a); Bai et al. (2025b), cross-layer expert sharing can potentially improve the performance of modern MoE models, given that routing strategies scale efficiently to large expert pools.

G Related Work

Weight sharing. Weight sharing is a standard route to parameter efficiency and recurrence-like inductive biases. In Transformers, it is commonly realized by reusing the same block over multiple depth "steps," effectively trading depth for recurrence Dehghani et al. (2018); Lan et al. (2019). Deep Equilibrium Models (DEQs) push this idea to implicit (infinite) depth by solving for a fixed point and backpropagating via implicit differentiation Bai et al. (2019). Iterative parameter reuse also appears in diffusion and score-based models, where a single model is applied repeatedly across timesteps conditioned on a timestep embedding Ho et al. (2020); Song et al. (2020).

Mixture-of-Experts. MoE models originate from gated ensembles in which an input-dependent router selects among specialized submodels Jacobs et al. (1991); Jordan & Jacobs (1994). Modern MoE LMs typically replace FFNs with a large pool of sparsely activated experts to obtain high capacity at the computational cost of a smaller model Shazeer et al. (2017); Lepikhin et al. (2020). A lot of effort was dedicated to improving routing and scaling of MoE models Zhou et al. (2022), domain extensions Riquelme et al. (2021), and systems support for expert parallelism and efficient training/inference Yang et al. (2025a); Dai et al. (2024); Liu et al. (2025). A central limitation of large sparse ensembles is the underutilization of parameters. In our work, we address this issue by sharing experts across layers and targeting regimes with $N_{\text{act}} > N_{\text{tot}}$.

Expert sharing. Prior expert-sharing methods either distribute experts across multiple submodules within a layer Yang et al. (2025b) or share them across a small number of neighboring layers Csordás et al. (2024); Tan et al. (2025), largely in conventional overparameterized MoE regimes. Interestingly, these works suggest that a single global expert pool yields poorer performance. In contrast, we study dense-like settings with tightly controlled active-to-total ratios and show that a model-wide shared expert pool is beneficial.