Hybrid Linear Attention Done Right: Efficient Distillation and Effective Architectures for Extremely Long Contexts

Yingfa Chen [*1] **Zhen Leng Thai** [*1] **Zihan Zhou** [2] **Zhu Zhang** [2] **Xingyu Shen** [1]
Shuo Wang [1] **Chaojun Xiao** [1] **Xu Han** [1] **Zhiyuan Liu** [1]

Abstract

Hybrid Transformer architectures, which combine softmax attention blocks and recurrent neural networks (RNNs), have shown a desirable performance-efficiency tradeoff for long-context modeling, but their adoption and studies are hindered by the prohibitive cost of large-scale pre-training from scratch. Some recent studies have shown that pre-trained softmax attention blocks can be converted into RNN blocks through parameter transfer and distillation. However, these transfer methods require substantial amounts of training data (more than 10B tokens), and result in poor long-context performance. We present HALO (Hybrid Attention via Layer Optimization), a pipeline for distilling Transformer models into RNN-attention hybrid models. We then present HypeNet, a hybrid architecture with superior length generalization enabled by a novel position encoding scheme (named HyPE) and various architectural modifications. We convert the Qwen3 series into HypeNet using HALO, achieving performance comparable to the original Transformer models while enjoying superior long-context performance and efficiency. The conversion requires just 2.3B tokens, less than 0.01% of their pre-training data.

1. Introduction

Transformer-based language models (Vaswani et al., 2017) rely on softmax attention blocks, which have a quadratic complexity with respect to the context length, making them prohibitively expensive for long contexts. In contrast, recurrent neural networks (RNNs) such as linear attention (Katharopoulos et al., 2020) and state space models (Gu & Dao, 2024) are much faster for long-context modeling due to their linear complexity. However, pure RNN models with fixed-size states generally underperform softmax attention, particularly on recall-intensive tasks (Jelassi et al., 2024; Yang et al., 2025b). To address this gap, there is a surge in interest in hybrid architectures that interleave attention and RNN layers[1], achieving a favorable tradeoff between model performance and inference throughput (Lieber et al., 2024; MiniMax et al., 2025; Qwen, 2025; Kimi et al., 2025; NVIDIA et al., 2025).

Hybrid architectures are typically pre-trained from scratch at a large scale (Qwen, 2025; NVIDIA et al., 2025), placing them beyond the reach of most academic research teams. Hence, some works focus on distilling pre-trained Transformer models into hybrid architectures (Gu et al., 2025; Hoshino et al., 2025). These distillation methods use far fewer training tokens and produce hybrid models that are comparable to their Transformer counterparts on various common-sense reasoning (CSR) tasks. Although distilled hybrid models typically underperform those trained from scratch, they are valuable since they allow teams without resources to scale up pre-training to validate research ideas.

However, these distillation methods still suffer from two limitations. (1) Most methods still require tens to hundreds of billions of training tokens, which is out of reach for most teams. (2) While the resulting models have great short-context performance, they exhibit severe performance degradation on longer context, which is precisely the scenario where they are preferred over Transformer models.

To address these challenges, we first propose HALO (Hybrid Attention via Layer Optimization), a distillation procedure for converting pre-trained Transformer models into hybrid models. It involves an efficient *attention layer selection* method for determining which layers to keep unconverted to ensure the best long-context performance. Then, we propose Hybrid Position Encoding (HyPE), a position encoding scheme with strong length generalization, specifically de-

[1]Department of Computer Science and Technology, Tsinghua University, Beijing China [2]OpenBMB. Correspondence to: Zhiyuan Liu <liuzy@tsinghua.edu.cn>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

[1]We use *hybrid architectures/models* to refer to architectures consisting of full softmax attention and RNN layers.

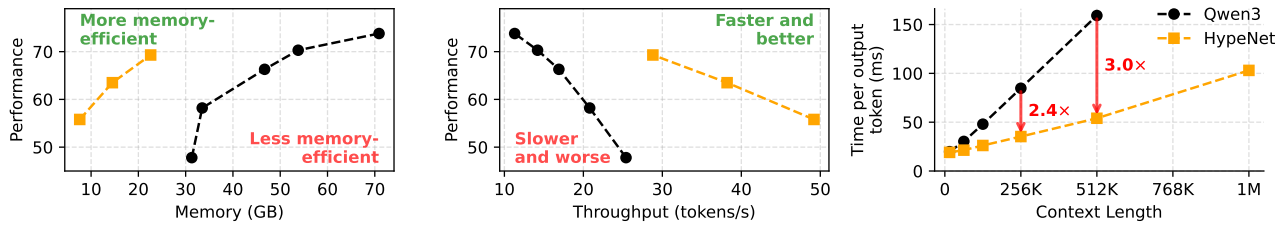

Figure 1. Left & center: the performance-efficiency tradeoff of our model, **HypeNet**, versus the **Qwen3** series, measured with 128K context length and BFloat16 precision. Right: the time per output token of the 1.7B models. Qwen3 runs out of GPU memory at 1M context length. HypeNet is converted from Qwen3 using our distillation procedure, HALO.

signed for hybrid architectures. In addition to HyPE, we propose a series of architectural improvements, validated with extensive ablation experiments. The combination of these improvements results in HypeNet, a series of hybrid models converted from Qwen3, with a much better performance-efficiency tradeoff, as shown in Figure 1.

Our contributions can be summarized as follows:

- We develop a novel cross-architecture distillation procedure that converts Transformer models into RNN-attention hybrid models using fewer than 3B tokens, thereby significantly improving the model's efficiency in long-context scenarios.
- We present HyPE, a novel position encoding scheme that combines RoPE (Su et al., 2023) and NoPE (Kazemnejad et al., 2023), designed for hybrid models. Coupled with an attention scaling mechanism, HyPE achieves superior length generalization.
- Based on HyPE, we propose HypeNet, a novel architecture that incorporates multiple architectural improvements when distilling from Transformer models.

2. Related Works

RNN-Attention Hybrid Models State-of-the-art (SOTA) hybrid models with up to hundreds of billions of parameters have exhibited performance comparable to standard Transformers on both commonsense reasoning and recall-intensive tasks (e.g., needle-in-a-haystack (NIAH) (Hsieh et al., 2024)) while being more efficient for processing long contexts (Lieber et al., 2024; MiniMax et al., 2025; Qwen, 2025; Kimi et al., 2025; NVIDIA et al., 2025). Despite their impressive performance, there are rather few publicly available hybrid models with frontier-level performance, because pre-training from scratch is prohibitively expensive for most teams. To avoid this training cost, we focus on distilling pre-trained Transformer models into hybrid models.

Position Encoding in Hybrid Models Current, most transformer models use RoPE (Su et al., 2023) as the position encoding (PE) (Yang et al., 2025a; Grattafiori et al., 2024). Meanwhile, RNNs usually encode positional information through decay/transition matrices, and do not employ RoPE (Dao & Gu, 2024; Yang et al., 2025b; Kimi et al., 2025). This has remained the case for hybrid models, which means attention layers adopt RoPE while RNN layers do not (i.e., RNNs use NoPE) (Qwen, 2025; MiniMax et al., 2025). In this paper, we propose a novel PE scheme for hybrid models that achieves better long-context performance.

Distilling Transformers into Hybrid Models Many works focus on converting Transformers into pure RNN models via distillation (Kasai et al., 2021; Bick et al., 2025; Zhang et al., 2025; Goldstein et al., 2025), but converting Transformers into hybrid models remains underexplored. When distilling into hybrids, choosing which attention layer to convert to RNN is critical for maintaining performance. Wang et al. (2025b) adopt a simple attention layer selection scheme and show poor performance. Yang et al. (2026) use the output distribution shift when replacing an attention layer with an RNN layer to determine the importance of attention layers. Hoshino et al. (2025) propose a redundancy metric for determining importance, and Gu et al. (2025) rely on the performance drop on certain tasks. Finally, KL-guided layer selection (KL-LS) (Li et al., 2025) uses KL-divergence from the teacher model as the importance metric. These works typically use more than 10B training tokens and have poor long-context recall performance. In contrast, our distillation procedure requires just 2.3B tokens, and our architecture has much stronger long-context performance thanks to its superior length generalization.

3. Preliminaries

Notations All models involved in this study, including both Transformer and hybrid models, consist of a stack of L layers, and the l-th layer can be formalized as

$$\begin{aligned} \mathbf{H}^{(l)} &= \mathrm{Mixer}^{(l)}\left(\mathbf{X}^{(l-1)}\right) + \mathbf{X}^{(l-1)}, \\ \mathbf{X}^{(l)} &= \mathrm{MLP}^{(l)}\left(\mathbf{H}^{(l)}\right) + \mathbf{H}^{(l)}, \end{aligned} \tag{1}$$

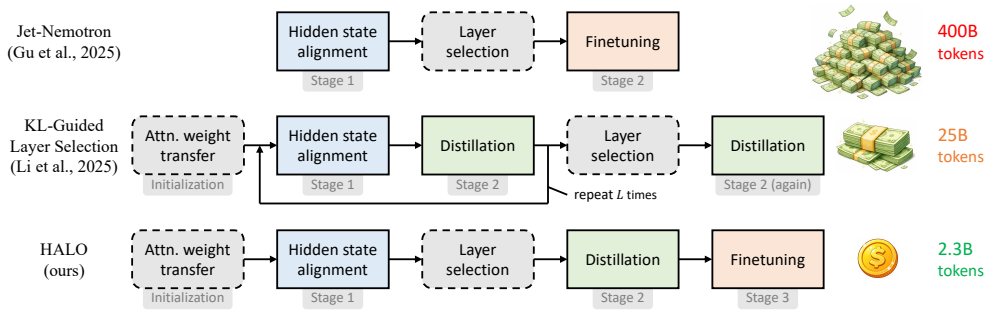

Figure 2. Various pipelines for converting Transformer models into hybrid models. The boxes with dotted lines represent training-free stages, while those with solid lines represent training stages. HALO is much more data-efficient than prior methods.

where $\mathbf{X}^{(l)} = \left[\mathbf{x}_1^\top, \cdots, \mathbf{x}_T^\top\right]^\top \in \mathbb{R}^{T\times d}$ denotes the T d-dimensional output embeddings. In an RNN-attention hybrid model, the set of attention layers is specified by $\mathcal{I}_{\text{attn}} \in \{l_{\text{attn},i} \mid i = 1, \cdots, L_{\text{attn}}\}$, where L_{attn} is the number of attention layers and $l_{\text{attn},i} \in \{1, \cdots, L\}$ is the index of the i-th attention layer. The mixers are defined as

$$\text{Mixer}^{(l)} = \begin{cases} \text{ATTN}^{(l)} & \text{if } l \in \mathcal{I}_{\text{attn}}, \\ \text{RNN}^{(l)} & \text{otherwise.} \end{cases} \tag{2}$$

Softmax Attention Layers In Transformer, the mixer layer uses softmax attention, which can be written as[2]

$$\begin{aligned} &\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_k, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_v, \\ &\mathbf{Y} = \text{softmax}\left(\frac{1}{\sqrt{d_h}}\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}\right)\mathbf{V}\mathbf{W}_o^\top, \end{aligned} \tag{3}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o \in \mathbb{R}^{d\times d_h}$ are learnble parameters, and $\mathbf{M}$ is the attention mask. We use row-vector representation, so $\mathbf{x}^\top\mathbf{x}$ denotes an outer product.

Modern RNN Layers There are many variants of RNN layers, but we focus on RNNs that can be written as

$$\mathbf{q}_t = \mathbf{x}_t\mathbf{W}_q, \quad \mathbf{k}_t = \mathbf{x}_t\mathbf{W}_k, \quad \mathbf{v}_t = \mathbf{x}_t\mathbf{W}_v, \tag{4}$$

$$\mathbf{S}_t = \mathbf{F}_t\mathbf{S}_{t-1} + \mathbf{k}_t^\top\mathbf{v}_t \in \mathbb{R}^{d_h\times d_h}, \tag{5}$$

$$\mathbf{y}_t = \mathbf{q}_t\mathbf{S}_t\mathbf{W}_o^\top \in \mathbb{R}^d, \tag{6}$$

where $\mathbf{F}_t \in \mathbb{R}^{d_h\times d_h}$ is named the *transition matrix* and is a function of $\mathbf{x}_t$. The above formulas include SOTA RNN variants such as Mamba2 (Dao & Gu, 2024), Gated DeltaNet (Yang et al., 2025b), etc.

4. HALO: An Efficient Pipeline to Distill Transformers into Hybrids

Our conversion procedure, HALO, is an adoption and improvement of RADLADS (Goldstein et al., 2025), a distillation method that converts Transformer models into pure RNN models. Figure 2 shows an overview of HALO. It consists of an attention weight transfer process, three training stages, and an attention layer selection process. Appendix B shows the training configuration of each stage in HALO.

[2]Here, we ignore the multi-head mechanism for simplicity.

4.1. Initialization Stage: Attention Weight Transfer

Given a Transformer model, for each attention layer $\text{ATTN}^{(l)}(\cdot)$, we use its configuration and pre-trained projection weights $(\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o)$ to instantiate an RNN layer $\text{RNN}^{(l)}(\cdot)$. If an RNN layer has other modules that are not covered by the weights of the attention layer, we initialize the weights of these modules using the empirical implementation of RNN layers.

4.2. Stage 1: Hidden State Alignment

We train each RNN layer independently by minimizing the mean squared error (MSE) between its output and the attention layer used to instantiate it:

$$\mathcal{L}_{\text{stage 1}}^{(l)} = \text{MSE}\left(\mathbf{Y}_{\text{teacher}}^{(l)}, \text{RNN}^{(l)}\left(\mathbf{X}^{(l-1)}\right)\right), \tag{7}$$

where $\mathbf{Y}_{\text{teacher}}^{(l)}$ is the output of the l-th attention layer in the teacher model. During alignment, only the RNN layers are trained, and all other weights are frozen.

4.3. Attention Layer Selection

Here, we perform attention layer selection to determine $\mathcal{I}_{\text{attn}}$. We propose to select attention layers that, when replaced by RNN layers, exhibit **a large drop in recall performance and a small drop in CSR**. Let $M^{(i)}$ denote the original model but with the i-th layer replaced with the corresponding RNN layer from stage 1. Let $\mathcal{R}(M), \mathcal{C}(M) \in [0, 1]$ denote the recall and CSR performance of the model M, then, the importance score of each attention layer is

$$s_i = \frac{\max_i\left[\mathcal{R}\left(M^{(i)}\right)\right] - \mathcal{R}\left(M^{(i)}\right)}{\max_i\left[\mathcal{C}\left(M^{(i)}\right)\right] - \mathcal{C}\left(M^{(i)}\right) + \epsilon}, \tag{8}$$

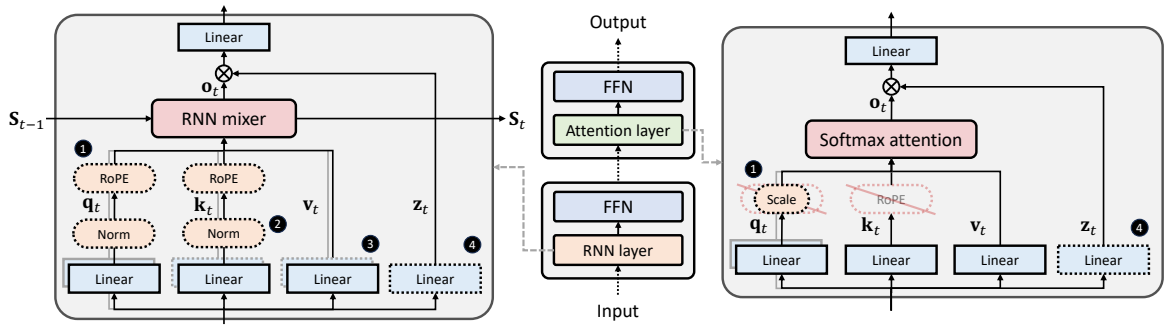

Figure 3. Illustration of HypeNet. The architectural modifications introduced during HALO are marked with ❶, ❷, ❸, and ❹. Red dotted lines indicate components that are removed during HALO, black dotted lines indicate components that are added.

where $\epsilon = 10^{-6}$ is a small constant to avoid division by zero. Finally, we simply pick the Top-k most important attention layer as

$$\mathcal{I}_{\text{attn}} = \underset{i}{\text{Top-}k}(s_i). \tag{9}$$

Based on Wang et al. (2025a), we always use $k = \lfloor L/4 \rfloor$ in this paper, which means that 25% of the layers in the final model are attention layers. The actual layer indices $\mathcal{I}_{\text{attn}}$ selected by our approach are reported in Appendix C.

4.4. Stage 2: Knowledge Distillation

In stage 2, we construct the final hybrid model f_{hybrid} using $\mathcal{I}_{\text{attn}}$ and conduct standard end-to-end knowledge distillation, with the original Transformer model f_{orig} as the teacher and the hybrid model as the student:

$$\mathcal{L}_{\text{stage 2}} = D_{\text{KL}}\left(f_{\text{orig}}(\mathbf{X}) \| f_{\text{hybrid}}(\mathbf{X})\right), \tag{10}$$

where D_{KL} is KL divergence. The teacher model weights are frozen in this stage. Appendix F.3.1 provides some ablation experiments for stage 2 configurations.

4.5. Stage 3: Finetuning

Finally, to optimize the hybrid model's capabilities, we finetune the hybrid model with greater context length and a smaller learning rate. We use 1B training data for long-context finetuning.

5. HypeNet: An Effective RNN-Attention Hybrid Architecture

HypeNet is illustrated in Figure 3. It incorporates a novel PE scheme called HyPE (described in Section 5.1) and some other architectural modifications (described in Section 5.2). These architectural improvements are agnostic to the RNN mixer. Therefore, HypeNet is compatible with most modern RNNs (see Appendix G). A complete formulation of HypeNet can be found in Appendix A.

5.1. HyPE: Hybrid Positional Encoding (❶)

In brief, HyPE applies RoPE in RNN layers and NoPE in attention layers. This scheme allows the model to combine the length generalization power of NoPE and the rich positional information of RoPE, getting the best of both worlds.

Motivation HyPE is motivated by the finding that RNNs have a limited "receptive field", which means they struggle to model long-context dependencies (Chen et al., 2025b). This implies that in hybrid models, RNN layers primarily model short-distance dependencies while attention layers model long-distance dependencies. Therefore, when the context length exceeds the RNNs' receptive field, RNN layers are agnostic to the context length, implying that length generalization is unaffected by these layers. Consequently, the model's length generalization depends only on attention layers, which use NoPE, allowing it to generalize well beyond its training context length. In the meantime, RNN layers with RoPE provide rich positional information, allowing the model to outperform a NoPE-only model.

Attention Logits Scaling As the context length increases, the entropy of attention scores increases, resulting in poor length generalization. To mitigate this, we adopt the dynamic attention scaling from Puvvada et al. (2025), where the attention logits are scaled with a *position-dependent scaling factor* s_t during inference:

$$\text{softmax}\left(\frac{s_t \mathbf{q}_t \mathbf{K}}{\sqrt{d_h}}\right), \quad s_t = \log_a(t + a), \tag{11}$$

where a is a hyperparameter determined after training by minimizing loss on a set of pre-training documents. The actual value of each model is reported in Appendix C. This scaling can be applied prior to the attention operator. Therefore, it has a negligible effect on the runtime. The effectiveness of this scaling mechanism is validated in Appendix F.3

Conversion Details In HALO, attention layers are not trained/modified during stage 1. Therefore, the removal of

Table 1. Long-context recall performance of HypeNet + HALO versus SOTA hybrid models that are distilled from pre-trained Transformer models. Qwen3 is evaluated with YaRN, as suggested by its authors. Best scores are bolded.

			NIAH-Single-1				NIAH-Single-2				NIAH-Single-3			
Model	**Param**	**Token**	**32K**	**64K**	**128K**	**256K**	**32K**	**64K**	**128K**	**256K**	**32K**	**64K**	**128K**	**256K**
Qwen3 (teacher, no RNNs)	1.7B	-	100	100	96.4	17.0	100	98.8	24.8	19.2	100	98.4	14.8	19.0
Jet-Nemotron (Gu et al., 2025)	**2B**	400B	**99.8**	56.0	0.0	0.0	94.2	65.0	0.0	0.0	84.0	15.4	0.0	0.0
KL-LS (GDN) (Li et al., 2025)	3B	25B	**99.8**	99.4	68.4	14.8	**99.4**	49.6	28.2	10.4	**99.0**	51.0	24.8	11.0
HypeNet + HALO (ours)	**2B**	**2.3B**	**99.8**	**99.6**	**99.8**	**99.8**	95.2	**99.6**	**97.8**	**86.2**	87.2	**72.6**	**44.8**	**48.8**

Table 2. Comparison of different *attention layer selection* methods on CSR and NIAH tasks. All models are converted from Qwen3 with HALO, but use different layer selection methods. The best scores are bolded.

		Needle-in-a-Haystack					
Model	**CSR**	**8K**	**16K**	**32K**	**64K**	**128K**	**256K**
Qwen3-1.7B (teacher, no RNNs)	58.5	99.7	99.9	99.9	99.5	38.6	18.4
HALO (ours)	**55.9**	**94.9**	**90.3**	**94.1**	**90.6**	**79.9**	**74.3**
Jet-Nemotron-2B (Gu et al., 2025)	55.0	88.7	70.1	70.3	61.9	63.7	56.2
KL-LS (Li et al., 2025)	55.3	85.7	78.4	72.8	68.9	58.3	44.3
Evenly distribute attn. layers	54.0	78.1	77.8	68.2	73.5	61.9	50.9

RoPE in attention layers occurs at the start of stage 2, when we instantiate the final hybrid model.

5.2. Other Architectural Modifications

In addition to HyPE, we make the following architectural modifications (marked with ❷, ❸, and ❹ in Figure 3) to further boost the performance and length generalization. To save space, we briefly explain them and the complete formulation is described in Appendix A.1. Notably, these are not required modifications, but they together constitute an effective hybrid architecture.

QK-Normalization (❷) Proposed by Henry et al. (2020), this normalizes $\mathbf{q}_t$ and $\mathbf{k}_t$ before the mixer. We find that adding them in RNN layers improves the hybrid model's performance. Thus, when converting models without QK-normalization, we add QK-normalization to the RNN layer.

GQA to MHA (❸) Most Transformer models employ grouped-query attention (GQA) (Ainslie et al., 2023), but most RNNs do not employ GQA. When initializing RNN layers before stage 1, we decouple KV heads by cloning the attention KV projection weights.

Output Gate (❹) Many recurrent architectures (Dao & Gu, 2024; Yang et al., 2025b) have an *output gate*, a data-dependent gating prior to the output projection. We add this mechanism in HALO because we found that it gives consistent performance gains with little increase in inference costs. $\mathbf{W}_z$ is randomly initialized.

6. Experiments

We first describe our experimental setup (Section 6.1). Then, we compare HypeNet + HALO against Qwen3 and SOTA hybrids that are also converted from pre-trained models (Section 6.2). Then, we verify the effectiveness of various design choices in HypeNet (Section 6.3). Afterwards, we present ablation studies for HALO's architectural modifications (Section F.1) and attention layer selection method (Section 6.4). Finally, we analyze the inference efficiency of HypeNet (Section 6.5).

6.1. Experimental Setup

Models We apply HALO to the 1.7B, 4B, and 8B models of Qwen3 (Yang et al., 2025a).

Training Configurations In HALO, we use FineWeb-edu (Penedo et al., 2024) for training. It is a popular open-source, high-quality Internet-scale pre-training corpus. All data are randomly sampled from the 10B subset. The concrete hyperparameters that we use for each stage in HALO are reported in Appendix B.

Evaluation We mainly evaluate CSR and long-context recall performance. For CSR, we use a suite of zero-shot downstream tasks that are common in related literature. To measure long-context performance, we report accuracy on NIAH[3]. More details are given in Appendix E.2.

[3]By default, NIAH refers to the average of NIAH-Single-1, NIAH-Single-2, and NIAH-Single-3 from RULER.

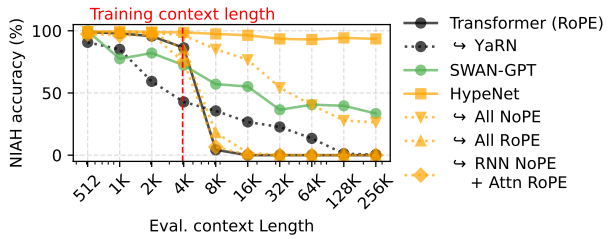

Figure 4. NIAH scores of HypeNet variants based on different position encodings, as a function of context length. The models are trained from scratch with 20B tokens and 500M parameters.

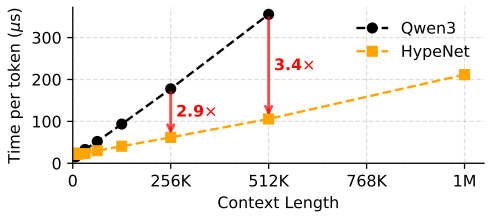

Figure 5. The prefilling time of HypeNet versus Qwen3-1.7B, across different context lengths.

Evaluation Data for Layer Selection Our layer selection method relies on measuring the performance change in CSR and recall (see Eq. (8)). Inspired by Gu et al. (2025), we use the normalized accuracy on HellaSwag (Zellers et al., 2019), ARC-Easy, and ARC-Challenge (Clark et al., 2018) as the CSR performance, the average score on SQuAD (Rajpurkar et al., 2016), FDA (Arora et al., 2025), and SWDE (Lockard et al., 2019) as the recall performance.

Efficiency Measurement All efficiency measurements are conducted on servers with a single NVIDIA A800 GPU. Softmax attention is implemented with Flash-Attention-2 (Dao, 2024), version 2.8.3. Batch size is set to 1 for all models to ensure fair comparison.

6.2. Main Results: Distilling from Qwen3

Figure 1 shows the CSR performance and efficiency of HypeNet compared to the Qwen3, and Table 1 reports the long-context recall performance. In Table 1, HypeNet + HALO is compared against recently SOTA hybrid models that are distilled from pre-trained Transformer models.

Takeaway 1 Under 128K context length, HypeNet is much more efficient than Qwen3 in terms of memory and throughput due to the reduced number of attention layers, and this tradeoff advantage increases with the context length.

Takeaway 2 Compared to SOTA Transformer-to-hybrid methods, HypeNet + HALO achieves superior long-context performance, despite training with fewer tokens and only open-source data, and being smaller than KL-LS (GDN).

6.3. HypeNet Ablations: Training From Scratch

To validate the effectiveness of HypeNet independent of HALO, we pre-train 500M HypeNet variants from scratch with 20B tokens and compare them against common baselines. The experimental details are reported in Appendix I. We compare HyPE against ordinary Transformer with RoPE and SWAN-GPT (Puvvada et al., 2025), which is an architecture with a similar PE but is not a hybrid model. The result, reported in Figure 4, demonstrates that HyPE's length generalization abilities outperform existing PE by a large margin. Notably, compared to conversion from pre-trained checkpoints, training HyPE from scratch achieves even better length generalization (having 93.5% NIAH accuracy on 64× the training context length), demonstrating the great potential of HyPE.

6.4. HALO Ablations: Attention Layer Selection

Here, we compare our proposed layer selection method (described in Section 4.3) with two SOTA approaches for determining layer importance, Jet-Nemotron (Gu et al., 2025) and KL-LS (Li et al., 2025), as well as some naive baselines. Our comparison is performed by replacing our attention layer selection method in HALO with these previous methods. The result, reported in Table 2, shows that our selection method achieves considerably better overall performance in terms of CSR and recall.

6.5. Efficiency Results

Figure 1 (center) shows the throughput of HypeNet models of different sizes (2B, 5B, and 9B) at 128K context length, and Figure 1 (right) shows the time per output token (TPOT) across different context lengths. Figure 5 shows the prefill speed results. We also provide a comparison of the runtime of various RNN mixers in Appendix E.1. In brief, HypeNet achieves up to 3.0× decoding speedup and 3.4× prefilling speedup on 512K context length, before Qwen3-1.7B runs out of GPU memory on 1M context length.

7. Conclusion

We have proposed HALO, a novel distillation procedure for converting pre-trained Transformer models into RNN-attention hybrid architectures with less than 3B tokens. We also proposed HypeNet, a hybrid architecture based on a novel PE scheme called HyPE, and it achieves superior length generalization. Applying our methods to Qwen3 produces a series of hybrid models with much better performance-efficiency tradeoff on long-context scenarios.

Limitations

Our hybrid models are obtained through a conversion process trained on the FineWeb-Edu corpus, which primarily consists of pre-training-style data. As a result, instruction-following and alignment behaviors of the pre-training model introduced by post-training may be diminished by our conversion process. However, this is a common shortcoming of all existing distillation methods for converting into hybrid architectures. How to efficiently recover the base models' capabilities remains an open question.

Moreover, our conversion protocol is designed specifically for Transformer-based architectures. Hence, its applicability to other model architectures requires further investigation, although the vast majority of publicly available LLMs are Transformer-based.

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 of which we feel must be specifically highlighted here.

References

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Arora, S., Yang, B., Eyuboglu, S., Narayan, A., Hojel, A., Trummer, I., and Ré, C. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes, 2025. URL https://arxiv.org/abs/2304.09433.

Bick, A., Li, K. Y., Xing, E. P., Kolter, J. Z., and Gu, A. Transformers to SSMs: Distilling Quadratic Knowledge to Subquadratic Models, 2025. URL https://arxiv.org/abs/2408.10189.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: Reasoning about Physical Commonsense in Natural Language, 2019. URL https://arxiv.org/abs/1911.11641.

Chen, Y., Wu, Y., Song, C., Thai, Z. L., Shen, X., Han, X., Liu, Z., and Sun, M. Cost-Optimal Grouped-Query Attention for Long-Context Modeling, 2025a. URL https://arxiv.org/abs/2503.09579.

Chen, Y., Zhang, X., Hu, S., Han, X., Liu, Z., and Sun, M. Stuffed Mamba: Oversized States Lead to the Inability to Forget, 2025b. URL https://arxiv.org/abs/2410.07145.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Dao, T. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Dao, T. and Gu, A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Goldstein, D., Alcaide, E., Lu, J., and Cheah, E. RADLADS: Rapid Attention Distillation to Linear Attention Decoders at Scale, 2025. URL https://arxiv.org/abs/2505.03005.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., and more. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Gu, A. and Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Gu, Y., Hu, Q., Yang, S., Xi, H., Chen, J., Han, S., and Cai, H. Jet-Nemotron: Efficient Language Model with Post Neural Architecture Search, 2025. URL https://arxiv.org/abs/2508.15884.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring Massive

Multitask Language Understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Henry, A., Dachapally, P. R., Pawar, S., and Chen, Y. Query-Key Normalization for Transformers, 2020. URL https://arxiv.org/abs/2010.04245.

Hoshino, Y., Tachibana, H., Inahara, M., and Takegawa, H. RAD: Redundancy-aware distillation for hybrid models via self-speculative decoding, 2025. URL https://arxiv.org/abs/2505.22135.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. RULER: What's the Real Context Size of Your Long-Context Language Models?, 2024. URL https://arxiv.org/abs/2404.06654.

Hu, S., Tu, Y., Han, X., He, C., Cui, G., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhao, W., Zhang, X., Thai, Z. L., Zhang, K., Wang, C., Yao, Y., Zhao, C., Zhou, J., Cai, J., Zhai, Z., Ding, N., Jia, C., Zeng, G., Li, D., Liu, Z., and Sun, M. MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies, 2024. URL https://arxiv.org/abs/2404.06395.

Jelassi, S., Brandfonbrener, D., Kakade, S. M., and Malach, E. Repeat After Me: Transformers are Better than State Space Models at Copying, 2024. URL https://arxiv.org/abs/2402.01032.

Kasai, J., Peng, H., Zhang, Y., Yogatama, D., Ilharco, G., Pappas, N., Mao, Y., Chen, W., and Smith, N. A. Finetuning Pretrained Transformers into RNNs, 2021. URL https://arxiv.org/abs/2103.13076.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention, 2020. URL https://arxiv.org/abs/2006.16236.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The Impact of Positional Encoding on Length Generalization in Transformers, 2023. URL https://arxiv.org/abs/2305.19466.

Kimi, T., Zhang, Y., Lin, Z., Yao, X., Hu, J., Meng, F., Liu, C., Men, X., Yang, S., Li, Z., Li, W., Lu, E., Liu, W., Chen, Y., Xu, W., Yu, L., Wang, Y., Fan, Y., Zhong, L., Yuan, E., Zhang, D., Zhang, Y., Liu, T. Y., Wang, H., Fang, S., He, W., Liu, S., Li, Y., Su, J., Qiu, J., Pang, B., Yan, J., Jiang, Z., Huang, W., Yin, B., You, J., Wei, C., Wang, Z., Hong, C., Chen, Y., Chen, G., Wang, Y., Zheng, H., Wang, F., Liu, Y., Dong, M., Zhang, Z., Pan, S., Wu, W., Wu, Y., Guan, L., Tao, J., Fu, G., Xu, X., Wang, Y., Lai, G., Wu, Y., Zhou, X., Yang, Z., and Du, Y. Kimi Linear: An Expressive, Efficient Attention Architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Li, Y., Yang, S., Tan, S., Mishra, M., Panda, R., Zhou, J., and Kim, Y. Distilling to Hybrid Attention Models via KL-Guided Layer Selection, 2025. URL https://arxiv.org/abs/2512.20569.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., Abend, O., Alon, R., Asida, T., Bergman, A., Glozman, R., Gokhman, M., Manevich, A., Ratner, N., Rozen, N., Shwartz, E., Zusman, M., and Shoham, Y. Jamba: A Hybrid Transformer-Mamba Language Model, 2024. URL https://arxiv.org/abs/2403.19887.

Lockard, C., Shiralkar, P., and Dong, X. L. OpenCeres: When Open Information Extraction Meets the Semi-Structured Web. 2019. URL https://aclanthology.org/N19-1309/.

MiniCPM, T., Xiao, C., Li, Y., Han, X., Bai, Y., Cai, J., Chen, H., Chen, W., Cong, X., Cui, G., Ding, N., Fan, S., Fang, Y., Fu, Z., Guan, W., Guan, Y., Guo, J., Han, Y., He, B., Huang, Y., Ji, B., Kong, C., Li, Q., Li, S., Li, W., Li, X., Li, Y., Li, Y., Li, Z., Liu, D., Lin, B., Lin, Y., Long, X., Lu, Q., Lu, Y., Luo, P., Lyu, H., Ou, L., Pan, Y., Pu, L., Qu, Z., Shi, Q., Song, Z., Su, J., Su, Z., Sun, A., Sun, X., Tang, P., Wang, F., Wang, F., Wang, S., Wang, Y., Wang, Z., Wu, Y., Xiao, Z., Xie, J., Xie, Z., Xu, X., Yan, Y., Yuan, J., Zhang, J., Zhang, K., Zhang, L., Zhang, L., Zhang, X., Zhang, Y., Zhao, H., Zhao, W., Zhao, W., Zhao, Y., Zheng, Z., Zhou, C., Zhou, G., Zhou, J., Zhou, W., Zhou, Y., Zhou, Z., Zhou, Z., Liu, Z., Zeng, G., Jia, C., Li, D., and Sun, M. MiniCPM4: Ultra-Efficient LLMs on End Devices, 2025. URL https://arxiv.org/abs/2506.07900.

MiniMax, Li, A., Gong, B., Yang, B., Shan, B., Liu, C., Zhu, C., Zhang, C., Guo, C., Chen, D., Li, D., Jiao, E., Li, G., Zhang, G., Sun, H., Dong, H., Zhu, J., Zhuang, J., Song, J., Zhu, J., Han, J., Li, J., Xie, J., Xu, J., Yan, J., Zhang, K., Xiao, K., Kang, K., Han, L., Wang, L., Yu, L., Feng, L., Zheng, L., Chai, L., Xing, L., Ju, M., Chi, M., Zhang, M., Huang, P., Niu, P., Li, P., Zhao, P., Yang, Q., Xu, Q., Wang, Q., Wang, Q., Li, Q., Leng, R., Shi, S., Yu, S., Li, S., Zhu, S., Huang, T., Liang, T., Sun, W., Sun, W., Cheng, W., Li, W., Song, X., Su, X., Han, X., Zhang, X., Hou, X., Min, X., Zou, X., Shen, X., Gong, Y., Zhu, Y., Zhou, Y., Zhong, Y., Hu, Y., Fan, Y., Yu, Y., Yang, Y., Li, Y., Huang, Y., Li, Y., Huang, Y., Xu, Y., Mao, Y., Li, Z., Li, Z., Tao, Z., Ying, Z., Cong, Z., Qin, Z., Fan, Z., Yu, Z., Jiang, Z., and Wu, Z. MiniMax-01: Scaling Foundation Models with Lightning Attention, 2025. URL https://arxiv.org/abs/2501.08313.

NVIDIA, Blakeman, A., Grattafiori, A., Basant, A., Gupta, A., Khattar, A., Renduchintala, A., Vavre, A., Shukla, A.,

Bercovich, A., Ficek, A., Shaposhnikov, A., Kondratenko, A., Bukharin, A., Milesi, A., Taghibakhshi, A., Liu, A., Barton, A., Mahabaleshwarkar, A. S., Klein, A., Zuker, A., Geifman, A., Shen, A., Bhiwandiwalla, A., Tao, A., Agrusa, A., Verma, A., Guan, A., Mandarwal, A., Mehta, A., Aithal, A., Poojary, A., Ahamed, A., Mishra, A., Thekkumpate, A. K., Dattagupta, A., Zhu, B., Sadeghi, B., Simkin, B., Lanir, B., Schifferer, B., Nushi, B., Kartal, B., Rouhani, B. D., Ginsburg, B., Norick, B., Soubasis, B., Kisacanin, B., Yu, B., and more. NVIDIA Nemotron 3: Efficient and Open Intelligence, 2025. URL https://arxiv.org/abs/2512.20856.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale, 2024. URL https://arxiv.org/abs/2406.17557.

Peng, B., Zhang, R., Goldstein, D., Alcaide, E., Du, X., Hou, H., Lin, J., Liu, J., Lu, J., Merrill, W., Song, G., Tan, K., Utpala, S., Wilce, N., Wind, J. S., Wu, T., Wuttke, D., and Zhou-Zheng, C. RWKV-7 "Goose" with Expressive Dynamic State Evolution, 2025. URL https://arxiv.org/abs/2503.14456.

Puvvada, K. C., Ladhak, F., Serrano, S. A., Hsieh, C.-P., Acharya, S., Majumdar, S., Jia, F., Kriman, S., Sun, S., Rekesh, D., and Ginsburg, B. SWAN-GPT: An Efficient and Scalable Approach for Long-Context Language Modeling, 2025. URL https://arxiv.org/abs/2504.08719.

Qin, Z., Sun, W., Li, D., Shen, X., Sun, W., and Zhong, Y. Various Lengths, Constant Speed: Efficient Language Modeling with Lightning Attention, 2024a. URL https://arxiv.org/abs/2405.17381.

Qin, Z., Yang, S., Sun, W., Shen, X., Li, D., Sun, W., and Zhong, Y. HGRN2: Gated Linear RNNs with State Expansion, 2024b. URL https://arxiv.org/abs/2404.07904.

Qiu, Z., Wang, Z., Zheng, B., Huang, Z., Wen, K., Yang, S., Men, R., Yu, L., Huang, F., Huang, S., Liu, D., Zhou, J., and Lin, J. Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free, 2025. URL https://arxiv.org/abs/2505.06708.

Qwen. Qwen3-Next: Towards Ultimate Training & Inference Efficiency, 2025. URL https://qwen.ai/blog?id=4074cca80393150c248e508aa62983f9cb7d27cd.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text, 2016. URL https://arxiv.org/abs/1606.05250.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive Network: A Successor to Transformer for Large Language Models, 2023. URL https://arxiv.org/abs/2307.08621.

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., Hashimoto, T., and Guestrin, C. Learning to (Learn at Test Time): RNNs with Expressive Hidden States, 2025. URL https://arxiv.org/abs/2407.04620.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need, 2017. URL https://arxiv.org/abs/1706.03762.

Wang, D., Zhu, R.-J., Abreu, S., Shan, Y., Kergan, T., Pan, Y., Chou, Y., Li, Z., Zhang, G., Huang, W., and Eshraghian, J. A Systematic Analysis of Hybrid Linear Attention, 2025a. URL https://arxiv.org/abs/2507.06457.

Wang, J., Paliotta, D., May, A., Rush, A. M., and Dao, T. The Mamba in the Llama: Distilling and accelerating hybrid models, 2025b. URL https://arxiv.org/abs/2408.15237.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 Technical Report, 2025a. URL https://arxiv.org/abs/2505.09388.

Yang, M., Rezagholizadeh, M., Li, G., Appia, V., and Barsoum, E. Zebra-Llama: Towards Extremely Efficient Hybrid Models, 2026. URL https://arxiv.org/abs/2505.17272.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated Linear Attention Transformers with Hardware-Efficient Training, 2024. URL https://arxiv.org/abs/2312.06635.

Yang, S., Kautz, J., and Hatamizadeh, A. Gated Delta Networks: Improving Mamba2 with Delta Rule, 2025b. URL https://arxiv.org/abs/2412.06464.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. Parallelizing Linear Transformers with the Delta Rule over Sequence Length, 2025c. URL https://arxiv.org/abs/2406.06484.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a Machine Really Finish Your Sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zhang, B. and Sennrich, R. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Zhang, M., Arora, S., Chalamala, R., Wu, A., Spector, B., Singhal, A., Ramesh, K., and Ré, C. LoLCATs: On Low-Rank Linearizing of Large Language Models, 2025. URL https://arxiv.org/abs/2410.10254.

Zhang, Y., Yang, S., Zhu, R., Zhang, Y., Cui, L., Wang, Y., Wang, B., Shi, F., Wang, B., Bi, W., Zhou, P., and Fu, G. Gated Slot Attention for Efficient Linear-Time Sequence Modeling, 2024. URL https://arxiv.org/abs/2409.07146.

A. Complete Formulation of HypeNet

Here, we present a complete formulation of HypeNet for clarity. Recall that the model consists of a stack of L layers that consists of a token mixer and MLP:

$$\begin{aligned}\mathbf{H}^{(l)} &= \text{Mixer}^{(l)}\left(\text{Norm}\left(\mathbf{X}^{(l-1)}\right)\right) + \mathbf{X}^{(l-1)} \in \mathbb{R}^{T\times d}\\ \mathbf{X}^{(l)} &= \text{MLP}^{(l)}\left(\text{Norm}\left(\mathbf{H}^{(l)}\right)\right) + \mathbf{H}^{(l)} \in \mathbb{R}^{T\times d}\end{aligned} \tag{12}$$

where $l \in \{1, \cdots, L\}$ is the layer index and $\text{Norm}(\cdot)$ represents an RMSNorm (Zhang & Sennrich, 2019). Then, each mixer is either an attention layer $\text{ATTN}(\cdot)$ or an RNN layer $\text{RNN}(\cdot)$, specified by an attention index set $\mathcal{I}_{\text{attn}}$:

$$\text{Mixer}^{(l)} = \begin{cases}\text{ATTN}^{(l)} & \text{if } l \in \mathcal{I}_{\text{attn}}\\ \text{RNN}^{(l)} & \text{otherwise}\end{cases} \tag{13}$$

Since the MLP layer is exactly the same as the one in the base model, we omit its formulation. Each attention layer and RNN layer consists of n_h heads, which are identical (except for the KV sharing mechanism in GQA). Thus, in the following formulations, we omit the head index and only give the formulation for a single head for simplicity. The output of the layer is the sum of the outputs of all heads.

Attention Layers Each attention layer can be written as follows:

$$\begin{aligned}\mathbf{q}_t &= \mathbf{x}_t\mathbf{W}_q \in \mathbb{R}^{1\times d_h}\\ \mathbf{k}_t &= \mathbf{x}_t\mathbf{W}_k \in \mathbb{R}^{1\times d_h}\\ \mathbf{v}_t &= \mathbf{x}_t\mathbf{W}_v \in \mathbb{R}^{1\times d_h}\\ \tilde{\mathbf{q}}_t &= \frac{s_t\mathbf{q}_t}{\sqrt{d_h}} \in \mathbb{R}^{1\times d_h}, \quad s_t = \log_a(t+a) \in \mathbb{R}\\ \mathbf{o}_t &= \sum_{i=1}^{t}\frac{\exp\left(\tilde{\mathbf{q}}_t\mathbf{k}_i^\top\right)\mathbf{v}_i}{\sum_{j=1}^{t}\exp\left(\tilde{\mathbf{q}}_j\mathbf{k}_j^\top\right)} \in \mathbb{R}^{1\times d_h}\\ \mathbf{z}_t &= \text{sigmoid}(\mathbf{x}_t\mathbf{W}_z) \in \mathbb{R}^{1\times d_h}\\ \mathbf{y}_t &= \left(\text{Norm}(\mathbf{o}_t)\odot\mathbf{z}_t\right)\mathbf{W}_o^\top \in \mathbb{R}^{1\times d}\end{aligned} \tag{14}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o, \mathbf{W}_z \in \mathbb{R}^{d\times d_h}$ are learnable parameters, and s_t is the position-dependent scaling factor, and a is a hyperparameter. Depending on the base model, there may be a QK-norm in attention layers.

RNN Layers Each RNN layer can be written as follows:

$$\begin{aligned}\mathbf{q}_t &= \text{Norm}\left(\mathbf{x}_t\mathbf{W}_q\right) \in \mathbb{R}^{1\times d_h}\\ \mathbf{k}_t &= \text{Norm}\left(\mathbf{x}_t\mathbf{W}_k\right) \in \mathbb{R}^{1\times d_h}\\ \mathbf{v}_t &= \mathbf{x}_t\mathbf{W}_v \in \mathbb{R}^{1\times d_h}\\ \tilde{\mathbf{q}}_t &= \text{RoPE}_t(\mathbf{q}_t) \in \mathbb{R}^{1\times d_h}\\ \tilde{\mathbf{k}}_t &= \frac{\text{RoPE}_t(\mathbf{k}_t)}{\sqrt{d_h}} \in \mathbb{R}^{1\times d_h}\\ \mathbf{S}_t &= \mathbf{S}_{t-1}\gamma + \tilde{\mathbf{k}}_t^\top\mathbf{v}_t \in \mathbb{R}^{d_h\times d_h}\\ \mathbf{o}_t &= \tilde{\mathbf{q}}_t\mathbf{S}_t \in \mathbb{R}^{1\times d_h}\\ \mathbf{z}_t &= \text{sigmoid}(\mathbf{x}_t\mathbf{W}_z) \in \mathbb{R}^{1\times d_h}\\ \mathbf{y}_t &= \left(\text{Norm}(\mathbf{o}_t)\odot\mathbf{z}_t\right)\mathbf{W}_o^\top \in \mathbb{R}^{1\times d}\end{aligned} \tag{15}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o, \mathbf{W}_z \in \mathbb{R}^{d\times d_h}$ are learnable parameters, and γ is the head-specific slope rate of Lightning Attention (Qin et al., 2024a), which is a *data-independent forget gate*. RoPE_t is the rotational matrix of RoPE (Su et al., 2023) for position t.

Forget Gate The forget gate of Lightning Attention in HypeNet is defined as:

$$\gamma_h = \exp\left(-2^{-8h/H}\right) \in (0, 1) \tag{16}$$

where $h \in \{1, \cdots, H\}$ is the head index and H is the number of heads. Notably, we do not rescale this value with a layer-specific factor as in the original implementation, because our preliminary results show that it does not yield performance gains in a hybrid model. The γ_h values for each head when $H = 32$ is:

```
0.4313237   0.4930687   0.5517813   0.60653067  0.6567524   0.7021885   0.74281985
0.7788008   0.81040263  0.83796686  0.86186993  0.8824969   0.9002237   0.91540533
0.9283695   0.9394131   0.94880116  0.95676816  0.96351933  0.9692332   0.97406423
0.97814524  0.9815902   0.9844964   0.98694694  0.98901224  0.99075234  0.99221796
0.993452    0.994491    0.99536544  0.9961014
```

A.1. Other Architectural Modifications in HypeNet (❷, ❸, and ❹)

In addition to HyPE, we make the following architectural modifications (marked with ❷, ❸, and ❹ in Figure 3) to further boost the performance and length generalization.

QK-Normalization (❷) Proposed by Henry et al. (2020), this normalizes $\mathbf{q}_t$ and $\mathbf{k}_t$:

$$\mathbf{q}_t = \text{Norm}(\mathbf{x}_t \mathbf{W}_q), \;\; \mathbf{k}_t = \text{Norm}(\mathbf{x}_t \mathbf{W}_k). \tag{17}$$

This has been adopted by some open-source Transformer LLMs (e.g., Qwen3 and Gemma3 (Henry et al., 2020)), but is not usually used in RNN layers. However, we find that adding them in RNN layers improves the hybrid model's performance. Thus, when converting models without QK-normalization, we add QK-normalization to the RNN layer.

GQA to MHA (❸) Most Transformer models employ grouped-query attention (GQA) (Ainslie et al., 2023), where groups of attention heads share the same set of KVs, reducing KV cache size. However, RNN layers do not have a KV cache, and sharing KVs may reduce the expressivity of RNN layers. Thus, when initializing RNN layers before stage 1, we decouple KV heads by cloning the attention KV projection weights:

$$\mathbf{W}_{\square}^{(i)} \leftarrow \mathbf{W}_{\square}^{(\lfloor i/g \rfloor)}, \; \forall i \in \{1, \cdots, n_h\}, \square \in \{k, v\} \tag{18}$$

where g is the query group size and $\mathbf{W}_{\square}^{(i)}$ is the KV projection weights for the i-th head.

Output Gate (❹) Many recurrent architectures (Dao & Gu, 2024; Yang et al., 2025b) have an *output gate*, a data-dependent element-wise gating mechanism prior to the output projection:

$$\begin{aligned} \mathbf{o}_t &= \text{Mixer}(\mathbf{x}_t), \quad \mathbf{z}_t = \sigma(\mathbf{x}_t \mathbf{W}_z), \\ \mathbf{y}_t &= (\text{Norm}(\mathbf{o}_t) \odot \mathbf{z}_t)\, \mathbf{W}_o^\top, \end{aligned} \tag{19}$$

where σ is an activation function, and $\mathbf{W}_z \in \mathbb{R}^{d \times d}$ is learnable parameters. We found that adding this component during conversion gives consistent performance gains with little increase in inference costs. Hence, during initialization, we add this mechanism by randomly initializing $\mathbf{W}_z$.

Qiu et al. (2025) have shown that adding an output gate to softmax attention improves model quality and length generalization. Thus, we also add a randomly initialized output gate to attention layers, but at the start of stage 2 instead of stage 1, since attention layers are not trained in stage 1.

Increased Model Size Due to the introduction of ❸ and ❹, HypeNet is roughly 10% larger than the model it is distilled from. However, according to Chen et al. (2025a), increasing model size while reducing the KV size is more cost-effective in long-context scenarios. HypeNet is much more efficient than the base model, due to a much smaller KV cache despite having slightly more parameters.

B. HALO Training Configurations

Table 3 reports the hyperparameters used for each stage in our conversion procedure, HALO. By default, we use the AdamW optimizer with beta values of $(0.9, 0.95)$ and without weight decay. Each stage uses an LR linear warmup from 0 to maximum LR, consisting of 50 steps. We train all models with BFloat16 precision.

Table 3. Hyperparameters for each training stage in HALO. η_{stage2} is a hyperparameter that depends on the model (reported in Table 4).

Stage	Tokens	LR Scheduler	LR	Context len.	Batch size	Train steps
1	320M	Cosine	1e-3 $\rightarrow$ 1e-5	512	32	20K
2	1B	Cosine	η_{stage2} $\rightarrow$ 1e-5	512	96	20K
3	1B	Constant	1e-5	16K	128	500

C. HypeNet Model Configurations

Table 4 reports the configuration of each model in this study. We also report the actual indices of the attention layers for each HypeNet model in Table 5.

Table 4. Hyperparameters of various HypeNet models.

Hyperparameter	HypeNet-2B	HypeNet-5B	HypeNet-9B
Vocab size	151936	151936	151936
Layers	28	36	36
Hidden size	2048	2560	4096
Attn. layers	7	8	8
RNN layers	21	24	24
FFN width	6144	9728	12288
Attention heads	16	32	32
Attention KV heads	8	8	8
RNN heads	16	32	32
Tie embeddings	Yes	Yes	Yes
RoPE theta	1M	1M	1M
RoPE scaling	None	None	None
a (in Eq. (11))	500	600	900
η_{stage2} (see Table 3)	1e-4	5e-5	3e-5

Table 5. Layer selection results. Here are the attention layers indices sorted by importance score computed using different layer selection methods. The top-k attention layers that are kept in the final model are highlighted with a box. The red indices in the box indicate layers that are not selected by our approach.

Method	Layer indices (most important $\rightarrow$ least important)
	Qwen3-1.7B
HALO (ours)	$\boxed{3, 21, 2, 9, 25, 6, 8,}$ 19, 16, 24, 12, 26, 23, 11, 27, 14, 18, 4, 7, 17, 13, 15, 20, 10, 22, 1, 0, 5
Jet-Nemotron (Gu et al., 2025)	$\boxed{0, 21, 25, 19, 6, 11, 9,}$ 24, 12, 2, 26, 16, 17, 23, 18, 4, 7, 3, 14, 20, 1, 27, 10, 13, 8, 22, 15, 5
KL-guided layer selection (Li et al., 2025)	$\boxed{21, 16, 25, 24, 0, 18, 19,}$ 20, 8, 1, 2, 11, 12, 26, 13, 17, 14, 15, 10, 9, 22, 23, 6, 7, 4, 3, 27, 5
	Qwen3-4B
HALO (ours)	$\boxed{0, 7, 1, 33, 24, 15, 34, 22,}$ 14, 31, 5, 21, 23, 16, 20, 2, 18, 19, 32, 27, 13, 25, 30, 6, 29, 17, 11, 35, 8, 12, 9, 10, 26, 28, 4, 3
	Qwen3-8B
HALO (ours)	$\boxed{10, 6, 7, 24, 33, 2, 4, 1, 34,}$ 22, 13, 26, 35, 20, 31, 15, 9, 29, 14, 5, 3, 17, 23, 28, 30, 21, 25, 18, 8, 11, 32, 12, 0, 19, 27, 16

D. Addition Notes on the Model Architecture

Short Convolution Many recent RNNs (Dao & Gu, 2024; Yang et al., 2025b; Gu et al., 2025) incorporate a "short convolution" layer, which is a per-channel 1D convolutional layer with a small kernel size (typically from 2 to 4). Most transformer models do not have this layer. Consistent with Goldstein et al. (2025), we found that adding this component

through post-training does not provide performance gains for the 8B model and even failed to converge when applied to the 1.7B model. Moreover, short convolutional layers require another dedicated CUDA kernel and more implementation overhead. Thus, we do not incorporate short convolutional layers in HypeNet.

E. Computational Cost of Each Stage

Table 6. The number of FLOPs and training/inference tokens required by each stage in HALO, applied to Qwen3-1.7B. The layer selection stage spends fewer FLOPs per token because it performs only inference and does not require backward passes. Stage 3 has greater FLOPs per token because it uses a greater context length. * indicates inference tokens, while other entries are training tokens.

Stage	Tokens	FLOPs / token	FLOPs	GPU hours (A800)
Stage 1	320M	4.15B	2.7e18	10.0
Layer selection	234M*	1.38B	6.5e17	N/A
Stage 2	1B	4.15B	8.3e18	43.4
Stage 3	1B	6.88B	1.4e19	37.7
Total	2.3B	16.6B	2.5e19	91.1

Table 6 reports the computational cost (in the number of FLOPs) of each stage in HALO, our distillation process. The layer selection process requires the model to perform inference on our evaluation tasks. These tasks contain 8.36M tokens in total, and the FLOPs per token for inference is notably fewer than that of training.

The Number of Tokens Required by KL-Guided Layer Selection The number of training tokens used for the attention selection method, KL-guided layer selection (KL-LS) (Li et al., 2025), depends on the number of layers in the model. Specifically, their method requires $700\text{M} \times L + 600\text{M}$ tokens, where L is the number of layers in the base model. In the main content (Table **??** and Figure 2), we report the number of tokens used for converting Qwen2.5-3B into RNNs with KL-LS, which is the model used in their paper. That model has 36 layers.

E.1. RNN Mixer Efficiency Measurement

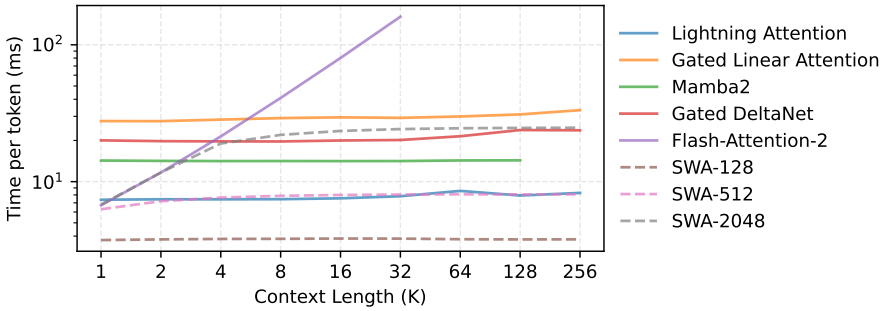

Figure 6. The inference prefilling time of various mixers as a function of context lengths, measured on one A800-80GB GPU using BFLoat16. The sliding window mixers are implemented with Flash-Attention-2, Mamba2 is implemented with its official `mamba_ssm` library, and all other RNN mixers are taken from the widely used Flash-Linear-Attention[4]. Mamba2 ran out of CUDA Memory on 256K context length. The y-axis is on log scale.

In this section, we compare the runtime of each RNN mixer across different context lengths, measured on one NVIDIA A800-80GB GPU. The inference throughput results are shown in Figure 6. "time-dep." means that forget gates (or, memory decay multiplier) depend on the current time step, while "time-indep." means that forget gates are fixed. We find that Lightning Attention with data-independent forget gates is significantly faster than other RNN mixers and comparable to SWA with a 512 window size, thanks to its highly simple update rule. This result further validates the superiority of Lightning Attention on HypeNet.

E.2. More Evaluation Details

We use the popular evaluation framework, LM-Evaluation-Harness (Gao et al., 2024), for all of our evaluations, and the version we use is 0.4.10.dev0. Before evaluation, we export each checkpoint such that it can be loaded with

Table 7. Ablation experiment results for various architectural and training choices in HypeNet-2B, converted from Qwen3-1.7B.

		Needle-in-a-Haystack					
Model	**CSR**	**4K**	**8K**	**16K**	**32K**	**64K**	**128K**
HypeNet	**55.9**	**95.9**	94.9	**90.3**	**94.1**	**90.6**	79.9
↪ w/o RNN RoPE (❶)	53.8	82.3	82.7	79.1	76.1	72.4	47.9
↪ w/ attention RoPE (❶)	55.8	95.3	**95.3**	87.0	67.1	37.2	19.7
↪ w/o RNN QK-norm (❷)	55.3	91.7	92.3	89.1	73.9	53.5	17.3
↪ w/o RNN GQA to MHA (❸)	55.8	89.7	90.0	87.9	89.5	88.9	**83.5**
↪ w/o RNN output gate (❹)	55.6	91.1	89.3	84.6	84.9	81.3	74.5
↪ w/o attention output gate (❹)	55.4	95.5	93.3	88.2	92.5	87.3	80.9

Table 8. Ablation experiment results on the number of retained attention layers.

Attn. Proportion	**CSR**	**NIAH-16K**	**NIAH-32K**	**NIAH-64K**	**NIAH-128K**
0%	**56.4**	0.5	0.3	0.1	0.3
14%	56.0	75.2	80.4	60.2	64.3
25%	55.8	90.3	**94.1**	90.6	79.9
50%	55.1	**97.2**	92.4	**96.7**	**90.9**

`AutoModelForCausalLM.from_pretrained` with the HuggingFace transformers library. Then, we run LM-Evaluation-Harness with the HuggingFace API. We use BFloat16 during evaluation.

Qwen3 YaRN By default, we evaluate Qwen3 models without any modifications to the official model configuration file. But for long-context tasks that exceed their default maximum context length, which is 40,960 tokens, we apply the YaRN method as described in the official model card adding a `"rope_scaling"` entry in the configuration file.

Downstream Tasks for CSR The downstream tasks for measuring CSR performance are as follows:

- ARC-Easy (Clark et al., 2018)
- ARC-Challenge (Clark et al., 2018)
- HellaSwag (Zellers et al., 2019)
- WinoGrande (Sakaguchi et al., 2019)
- PIQA (Bisk et al., 2019)
- LAMBADA (Paperno et al., 2016)
- MMLU (Hendrycks et al., 2021)

We always use normalized accuracy by default, which is more common according to the authors of LM-Evaluation-Harness.

F. More Experimental Results

F.1. HALO Ablations: Architectural Modifications

This section validates the effectiveness of various architectural modifications of HALO (those marked with ❶, ❷, ❸, and ❹ in Figure 3). Table 7 reports the architectural ablation results when converting Qwen3-1.7B, and it shows that our architectural modifications provide effective gains in CSR and NIAH performance, considerably outperforming common approaches in training hybrid architectures.

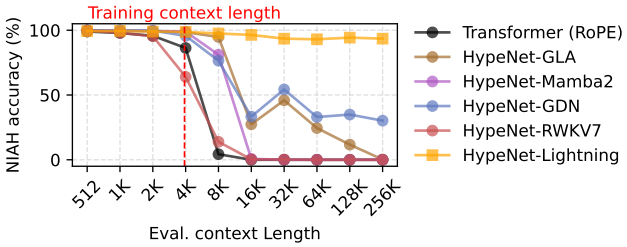

Figure 7. NIAH scores of HypeNet variants based on different RNN mixers, as a function of context length. The models are trained from scratch with 20B tokens and 500M parameters.

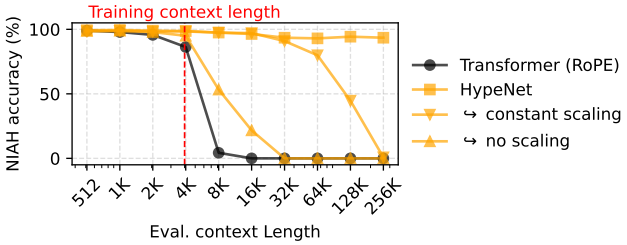

Figure 8. Results for validating attention logits scaling (see Eq. 11). The plot shows the NIAH performance of HypeNet without attention logits scaling, HypeNet with constant scaling (which is common in RoPE-based length extrapolation methods), and HypeNet with the attention logits scaling defined in Eq. 11.

F.2. HypeNet Ablations: Training From Scratch

Different RNN Mixers Moreover, we also compare the performance of incorporating different RNN mixers (those mentioned in Section **??**), and report the results in Figure 7. Perhaps surprisingly, Lightning Attention outperforms more recent RNN variants in terms of length generalization despite having a simpler update rule. One possible explanation is that Lightning Attention employs *data-independent* forget gates. In contrast, the other RNN mixers have *data-dependent* forget gates, which may result in poor length generalization, as shown by Chen et al. (2025b).

F.3. Attention Logits Scaling Validation

Figure 8 reports the results of HypeNet without attention logits scaling in HyPE, which is described in Eq. 11, but is repeated here for convenience:

$$\text{softmax}\left(\frac{s_t \mathbf{q}_t \mathbf{K}}{\sqrt{d_h}}\right), \quad s_t = \log_a(t + a), \tag{20}$$

As one can see from Figure 8, without logits scaling (i.e., setting $s_t = 1$), HyPE exhibits limited length generalization abilities. Constant scaling (setting $s_t = 1.5$ for all positions) improves length generalization to a decent degree. But the full potential of HyPE is unlocked with a position-dependent scaling factor, setting $s_t = \log_a(t + a)$.

F.3.1. HALO Configuration Ablation Experiments

Table 9 presents the ablation experiments on the training configurations in our conversion procedure, HALO. For stage 1, surprisingly, increasing the amount of training data beyond 320M tokens does not result in strong final performance. For stage 2, we can see that the default constant LR from RADLADS (Goldstein et al., 2025) is highly suboptimal. This discrepancy might be a result of the fact that RADLADS employs a different network architecture than ours and/or that their model sizes are different.

Table 9. Ablation experiment results for different training stages of HALO, applied to Qwen3-1.7B.

Model	CSR	Needle-in-a-Haystack					
		4K	8K	16K	32K	64K	128K
Stage 1 ablations							
100M tokens (RADLADS)	55.2	91.7	89.9	80.9	87.8	84.1	79.9
320M tokens (ours)	55.4	95.5	93.3	88.2	92.5	87.3	80.9
625M tokens	55.4	95.1	94.5	90.0	92.0	86.7	75.6
1.3B tokens	55.1	90.5	89.5	81.0	91.4	83.1	61.2
Stage 2 ablations							
Max LR = 1e-5 (RADLADS)	46.9	89.2	72.7	71.2	88.1	65.1	60.7
Max LR = 3e-5	55.5	67.0	70.1	66.4	64.5	54.2	54.9
Max LR = 1e-4 (ours)	56.4	79.9	75.4	76.9	78.7	70.1	68.7
Max LR = 3e-4	46.0	71.1	61.2	36.4	39.8	36.1	36.1
Max LR = 1e-3	36.8	79.2	73.9	75.5	75.1	84.5	75.1
Stage 3 ablations							
w/o fine-tuning	55.7	–	–	27.1	14.8	4.3	0.3
w/ fine-tuning (ours)	55.8	–	–	90.3	94.1	90.6	79.9

G. Which RNN Mixers are Compatible with HypeNet?

Here, we describe a more comprehensive (but not exhaustive) list of RNN mixers that are compatible with HypeNet. In other words, they can be expressed as Eq. (5) and (6), which we rewrite here for convenience.

$$\begin{aligned} \mathbf{q}_t &= \mathbf{x}_t\mathbf{W}_q, \quad \mathbf{k}_t = \mathbf{x}_t\mathbf{W}_k, \quad \mathbf{v}_t = \mathbf{x}_t\mathbf{W}_v, \\ \mathbf{S}_t &= \mathbf{F}_t\mathbf{S}_{t-1} + \mathbf{k}_t^\top\mathbf{v}_t \in \mathbb{R}^{d_h\times d_h}, \\ \mathbf{y}_t &= \mathbf{q}_t\mathbf{S}_t\mathbf{W}_o^\top \in \mathbb{R}^d, \end{aligned} \tag{21}$$

This formulation includes (but is not limited to) the RNN mixers listed in Table 10. Table 11 describes how the notations from each of the mixers studied in this paper correspond to our notations for RNN mixers (i.e., Eq. (21)). It also illustrates which components in these RNN mixers inherit the attention weights in HALO.

Table 10. Non-exhaustive list of representative RNN mixers that are compatible with HypeNet and HALO.

Linear Attention (Katharopoulos et al., 2020)	RetNet (Sun et al., 2023)
Lightning Attention (Qin et al., 2024a)	HGRN-2 (Qin et al., 2024b)
GLA (Yang et al., 2024)	Mamba2 (Dao & Gu, 2024)
GSA (Zhang et al., 2024)	DeltaNet (Yang et al., 2025c)
GDN (Yang et al., 2025b)	RWKV-7 (Peng et al., 2025)
TTT (Sun et al., 2025)	Kimi DeltaAttention (Kimi et al., 2025)

Table 11. List of how various state-of-the-art RNNs can be expressed as outer-product-based RNNs (i.e., Eq (21)), using the notations from their respective original paper. "-" indicates that these variables are never described in their original papers, but they can be found in the implementations. Our code for converting each of these RNN mixers is publicly available.

Mixer	$\mathbf{F}_t$	$\mathbf{q}_t$	$\mathbf{k}_t$	$\mathbf{v}_t$	$\mathbf{W}_q$	$\mathbf{W}_k$	$\mathbf{W}_v$	$\mathbf{W}_o$
Lightning Attention	λ	$\mathbf{q}_t$	$\mathbf{k}_t$	$\mathbf{v}_t$	$\mathbf{W}_q$	$\mathbf{W}_k$	$\mathbf{W}_v$	-
Mamba2	$\alpha_t I$	C_t	$\Delta_t B_t$	x_t	-	-	$W^{(x)}$	$W^{(o)}$
GLA	$\mathrm{diag}\left(\alpha_t\right)$	$\boldsymbol{q}_t$	$\boldsymbol{k}_t$	$\boldsymbol{v}_t$	$\boldsymbol{W}_Q$	$\boldsymbol{W}_K$	$\boldsymbol{W}_V$	$\boldsymbol{W}_O$
GDN	$\alpha_t\left(I-\beta_t\boldsymbol{k}_t^\top\boldsymbol{k}_t\right)$	$\boldsymbol{q}_t$	$\boldsymbol{k}_t$	$\boldsymbol{v}_t$	$\boldsymbol{W}_Q$	$\boldsymbol{W}_K$	$\boldsymbol{W}_V$	-
RWKV-7	$\left(\mathrm{diag}(\omega_t)-\hat{\kappa}_t k_t^\top\left(a_t\odot\hat{\kappa}_t\right)\right)$	r_t	$\tilde{k}_t$	ν_t	$\boldsymbol{W}_r$	$\boldsymbol{W}_k$	$\boldsymbol{W}_v$	$\boldsymbol{W}_o$

Table 12. Results on MiniCPM4-8B. Although the MiniCPM4-8B base model was trained with only a 4K context length, applying HALO enables strong extrapolation to substantially longer contexts. In particular, HypeNet retains non-trivial NIAH performance up to 32K tokens, whereas the original MiniCPM4 model fails beyond 4K.

Model	CSR	NIAH-2K	NIAH-4K	NIAH-8K	NIAH-16K	NIAH-32K
MiniCPM4	63.5	100.0	96.0	0.0	0.0	0.0
HypeNet	61.2	87.9	78.8	77.0	75.1	68.9

G.1. HypeNet's Compatibility with Mamba2

Mamba2 is derived from the perspective of state space models (SSMs), which is not based on QKV as the input. State space models may not always be expressible as Eq (21). Fortunately, Mamba and Mamba2 are special cases of SSMs that can be expressed as gated linear attention (Yang et al., 2024). The Mamba2 paper (Dao & Gu, 2024) provides an in-depth discussion of how (gated) linear attention is related to SSMs. In brief, both of these state-of-the-art SSMs are compatible with HypeNet.

Multi-Head Mechanism However, from the perspective of linear attention, Mamba2 adopts a multi-value mechanism in which all heads share the same set of queries and keys. This is not the usual configuration for softmax attention models. Therefore, in order to utilize the pre-trained model weights of softmax attention models, we use multi-head Mamba2 in this paper. This change has a negligible impact on the model's throughput.

G.2. A Note on Kimi Delta Attention

Here, we discuss a failed attempt at converting Qwen3's attention into KDA (Kimi et al., 2025), in order to facilitate more effective research. We have tried to use HALO to convert Qwen3's attention layers into KDA layers using the same configurations as described in Appendix B. However, the training process could not converge with the gradient norm becoming `inf` after a few steps in stage 2. We tried reducing the learning rate, but it did not help.

H. Applying HALO to Other Model Families

To assess whether the effectiveness of HALO generalizes beyond the Qwen3 model family studied in the main paper, we additionally apply HALO to the MiniCPM4.0 (MiniCPM et al., 2025), whch is a widely used open-source LLM with 8B parameters. Since the MiniCPM4-8B base model was originally trained with a context length of only 4K tokens, we evaluate it using shorter context lengths than those considered in the main experiments.

As shown in Table 12, the original MiniCPM4-8B model performs well within its trained context window, achieving near-perfect accuracy on NIAH-2K and NIAH-4K. However, its performance collapses once the context length exceeds the training length, with zero accuracy from 8K onward. In contrast, the HALO-extended HypeNet model maintains strong NIAH performance across all tested long-context settings, achieving 77.0, 75.1, and 68.9 accuracy at 8K, 16K, and 32K tokens, respectively. This result suggests that HALO is not limited to a single model family and can effectively improve length extrapolation for other open-source LLMs as well.

I. Training and Model Configurations for Training From Scratch Experiments

Here, we describe the training and model configurations for the experiments in Section 6.3.

I.1. Training Configurations

All models are trained on 20 billion tokens from the FineWeb-edu dataset (Penedo et al., 2024). We use 8 NVIDIA A800 GPUs to train each model. The training code is based on the HuggingFace Accelerate framework. The specific training hyperparameters are detailed in Table 13. The hyperparameters are chosen to best match standard practice in LLM pre-training.

Table 13. Training configurations and hyperparameters used when training from scratch (Section 6.3).

Hyperparameter	Value
Total tokens	20B
Context length	4096
Batch size	128
Training steps	40,000
LR scheduler	WSD (Hu et al., 2024)
Max. Learning rate	5×10^{-4}
Min. learning rate	5×10^{-5}
LR warmup steps	1,000
LR decay steps	8,000
Optimizer	AdamW, $\beta = (0.9, 0.95)$
Weight decay	0.1

Table 14. Model architecture configurations for the from-scratch training experiments (Section 6.3). The tokenizer for all models is the GPT-2 tokenizer[5].

Hyperparameter	Transformer	SWAN-GPT	HypeNet
Tokenizer	GPT-2	GPT-2	GPT-2
Vocabulary size	50,304	50,304	50,304
Layers	28	28	28
Hidden size	1024	1024	1024
RNN layers	0	0	21
Full Attn. layers	28	7	7
SWA layers	0	21	0
SWA Window size	–	512	–
FNN width	3072	3072	3072
Head dim	128	128	128
Attention heads	16	16	16
Attention KV heads	8	8	8
RNN heads	–	–	16
Tie embeddings	Yes	Yes	Yes
QK Norm in attention	Yes	Yes	Yes
RoPE θ	50k	50k	50k

I.2. Model Configurations

To ensure fair comparison, the parameter count for all models is controlled at approximately 500M. We also try to keep the implementation as similar as possible to its official implementation released by the respective authors. For HypeNet models, 25% of the layers are attention layers, interleaved with RNN layers in a repeating pattern of one attention layer followed by three RNN layers (i.e., Attn $\rightarrow$ RNN $\rightarrow$ RNN $\rightarrow$ RNN)[6]. The MLP blocks after each attention/RNN block are always a SwiGLU block with the same hyperparameters. Table 14 reports the detailed configuration for each model, Table 15 reports the attention logits scaling (see Section 5.1) for each model, and Table 16 reports the configurations for each RNN mixer. To ensure fair comparison with the Transformer model and SWAN-GPT, and also to better compare with our HypeNet models that are distilled from pre-trained Transformer models, we do not employ short convolutions in RNN mixers.

J. Use of AI

AI tools was used during implementation and paper-draft quality-checking, but was never used to explicitly write any content on this paper.

[6]Since we are training from scratch, we do not need to handle attention layer selection as in HALO.

Table 15. The logits scaling hyperparameter of various models in the from-scratch training experiments (Section 6.3).

Model	**Logit scaling base a (Eq. 11)**
Transformer	None
HypeNet-Lightning	300
HypeNet-Lightning (all NoPE)	1000
HypeNet-GDN	200
HypeNet-GLA	500
HypeNet-RWKV7	5000
HypeNet-Mamba2	1000
SWAN-GPT	1000

Table 16. Hyperparameters for of the RNN layers in the HypeNet variants of the from-scratch training experiments (Section 6.3). ✓ denotes that the feature is enabled, ✗ denotes disabled, and "–" means that the hyperparameter is not applicable.

Hyperparameter	**HypeNet-Lightning**	**HypeNet-GDN**	**HypeNet-GLA**	**HypeNet-RWKV7**	**HypeNet-Mamba2**
Gating & Normalization					
Output gate	✓	✓	✓	✓	✓
Output norm	✓	✓	✓	✓	✓
QK norm	✓	✓, L_2-norm	✗	✗	✗
QKV activation	✗	✓, SiLU	✗	✗	✗
Short Convolution	✗	✗	✗	✗	✗
$\mathbf{F}_t$ neg. eigenvalue	✗	✓	✗	✗	✗
Low-Rank Parametrization					
Gate low-rank dim.	–	–	16	160	–
Value low-rank dim.	–	–	–	96	–
Decay low-rank dim.	–	–	–	160	–
A low-rank dim.	–	–	–	160	–