

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

Yingfa Chen¹, Zhen Leng Thai¹, Zihan Zhou², Zhu Zhang², Xingyu Shen¹,
Shuo Wang¹, Chaojun Xiao^{1,†}, Xu Han^{1,†}, Zhiyuan Liu^{1,3}

¹NLP Group, DCST, IAI, Tsinghua University, Beijing, China

²OpenBMB

³Jiangsu Collaborative Innovation Center for Language Ability, Xuzhou, China
yingfa-c24@mails.tsinghua.edu.cn, {xcj, han-xu, liuzy}@tsinghua.edu.cn

Abstract

Hybrid architectures that combine softmax attention blocks and recurrent neural networks (RNNs) have shown a desirable performance-throughput 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 knowledge distillation. However, these transfer methods require substantial amounts of training data (more than 10B tokens), and the resulting hybrid models also exhibit poor long-context performance, which is the scenario where hybrid models enjoy significant inference speedups over Transformer-based models. In this paper, 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 and 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 layers¹, achieving a favorable tradeoff between model performance and inference throughput (Lieber et al., 2024; Mini-Max 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 Transformers into hybrid architectures (Gu et al., 2025; Hoshino et al., 2025; Wang et al., 2025b). 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 for quickly validating research ideas without expensive pre-training.

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

To address these challenges, we first propose HALO (Hybrid Attention via Layer Optimization), a novel cross-architecture distillation procedure for converting pre-trained Transformers into hybrid models. Notably, HALO involves an efficient *atten-*

¹We use *hybrid models* to refer to models that consist of full softmax attention and RNN layers.

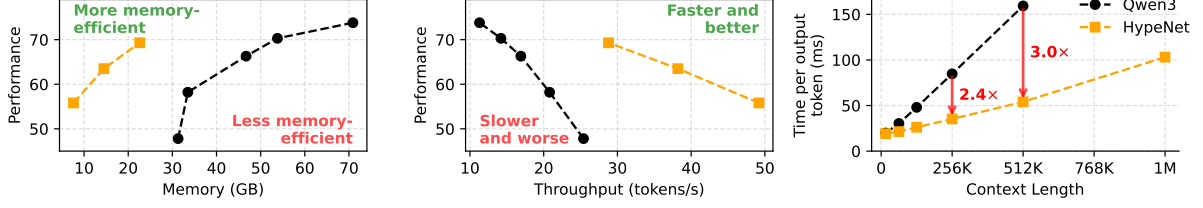


Figure 1: Left & center: the performance-efficiency tradeoff of **HypeNet** (ours), versus the **Qwen3** series, measured with 128K context length and BFloat16 precision. Right: the time per output token of the 1.7B models across different context lengths. At 1M context length, Qwen3 runs out of GPU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

tion layer selection method for determining which attention 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 designed for hybrid architectures. In addition to HyPE, we propose a series of architectural improvements, validated with careful ablation experiments on models with over 1B parameters. The combination of these improvements results in HypeNet, a series of hybrid models converted from the Qwen3 series, with a much better performance-throughput tradeoff, as shown in Figure 1.

Our contributions can be summarized as follows:

- HALO: a novel distillation procedure that converts Transformers into attention-RNN hybrid models using fewer than 3B tokens, thereby significantly improving the model’s efficiency in long-context scenarios.
- HyPE: a novel position encoding scheme designed for hybrid models that achieves superior length generalization.
- HypeNet: a novel hybrid architecture based on HyPE that incorporates multiple architectural improvements when converting from a pre-trained Transformer model.

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

Method	Date	Tokens
SMART (Yang et al.)	May 2025	>7B
RAD (Hoshino et al.)	May 2025	20B
Jet-Nemotron (Gu et al.)	Aug. 2025	400B
KL-LS (Li et al.)	Dec. 2025	25B
HALO (ours)	Jan. 2026	2.3B

Table 1: Existing attention-to-hybrid distillation methods and their release date and training tokens required.

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, RoPE (Su et al., 2023) has become the *de facto* standard position encoding (PE) for Transformer models (Yang et al., 2025a; Grattafiori et al., 2024). On the other hand, RNNs usually encode positional information through decay/transition matrices, and do not employ RoPE (Dao and Gu, 2024; Yang et al., 2025b). 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). Recently, SWAN-GPT (Puvvada et al., 2025) has shown promising long-context generalization by combining RoPE in sliding window attention layers and NoPE in full attention layers, but it is not a hybrid model. Concurrent to this paper, Kimi-Linear (Kimi et al., 2025) has adopted NoPE in both attention and RNN layers. In contrast, our model employs a novel PE scheme and achieves better long-context performance than typical PE methods found in existing hybrid models.

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, especially for tasks that are hard to handle with RNN layers. Wang et al. (2025b) adopt a simple pipeline and attention layer selection scheme and show severe performance degradation. More recent works choose which attention layers to retain more sophisticatedly. 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), a concurrent work, proposes using KL-divergence from the teacher model as the importance metric and requires a thorough search that repeatedly reruns a distillation process for every layer.

Table 1 lists previous distillation works. These works typically use more than 10B training tokens and have poor recall performance compared to Transformer models, especially on long contexts. 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

$$\begin{aligned}\mathbf{H}^{(l)} &= \text{Mixer}^{(l)}(\mathbf{X}^{(l-1)}) + \mathbf{X}^{(l-1)}, \\ \mathbf{X}^{(l)} &= \text{MLP}^{(l)}(\mathbf{H}^{(l)}) + \mathbf{H}^{(l)},\end{aligned}\quad (1)$$

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

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

Softmax Attention Layers In Transformer, the mixer layer uses softmax attention, which can be written as²

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

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o \in \mathbb{R}^{d \times d_h}$ are learnable 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

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

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

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

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 and Gu, 2024), Gated DeltaNet (Yang et al., 2025b), etc. $\mathbf{S}_t$ is named the *recurrent state*³, and Eq. (5) and (6) are named the *update rule* and the *query rule*, respectively.

3.1 Impact of Attention Layer Selection when Distilling Transformers into Hybrids

When distilling Transformers into hybrid models, one important question is *how to select which attention layers to remain unconverted*, i.e., how to determine the optimal $\mathcal{I}_{\text{attn}}$ for maximizing model performance, without increasing the number of attention layers $|\mathcal{I}_{\text{attn}}|$ (since efficiency is negatively correlated with $|\mathcal{I}_{\text{attn}}|$). Previous works have identified that RNN models underperform attention models on recall-intensive tasks (Shen et al., 2025; Jellasi et al., 2024); thus, our aim to identify which attention layers are most important for modeling recall abilities and leave them unconverted.

4 HALO: Efficient Transformer-to-Hybrid Distillation

Our conversion procedure, HALO, is an adoption and improvement of RADLADS (Goldstein et al., 2025), a distillation method that converts Transformers into pure RNN models (Peng et al., 2025). Figure 2 shows an overview of HALO. It consists

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

³Also named *hidden state* in some papers.

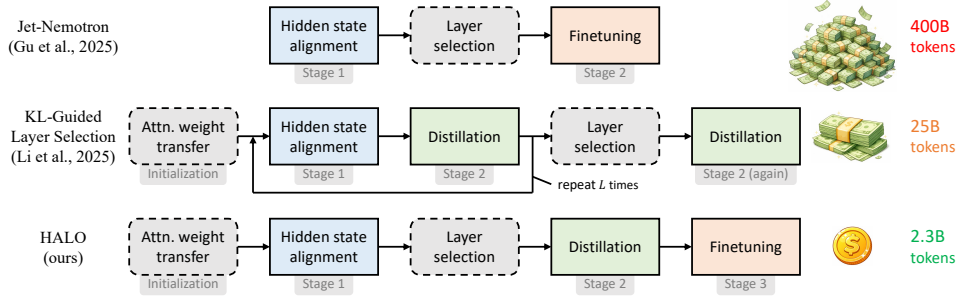


Figure 2: Various pipelines for converting Transformer models into hybrid models. The boxes with dotted lines represent training-free stages, while those with solid lines represent training stages. HALO is much more data-efficient than prior methods.

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.

4.1 Initialization: Attention Weight Transfer

Given a Transformer with only attention layers, for each layer $\text{ATTN}^{(l)}$, 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 $\text{RNN}^{(l)}$. If an RNN layer has other modules that cannot be 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 instantiated RNN layer independently by minimizing the mean squared error (MSE) between its output hidden states and the attention layer used to instantiate it:

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

where $\mathbf{Y}_{\text{teacher}}^{(l)}$ is the output of the l -th layer in the teacher model. During the alignment process, only the RNN layers are trained, and all other weights are frozen. After stage 1, each attention layer has a student RNN layer that can potentially replace it.

4.3 Attention Layer Selection

Here, we perform attention layer selection to determine $\mathcal{I}_{\text{attn}}$. We propose to retain 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 with the i -th layer replaced with the corresponding RNN layer from stage 1. Let $\mathcal{R}(M^{(i)}), \mathcal{C}(M^{(i)}) \in [0, 1]$ denote the recall and CSR performance of $M^{(i)}$,

the importance score of each attention layer is

$$r_i = \frac{\max_i [\mathcal{R}(M^{(i)})] - \mathcal{R}(M^{(i)})}{\max_i [\mathcal{C}(M^{(i)})] - \mathcal{C}(M^{(i)})}, \quad (8)$$

Finally, we simply pick the Top- k most important attention layer as

$$\mathcal{I}_{\text{attn}} = \text{Top-}k(r_i). \quad (9)$$

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 end-to-end knowledge distillation with the original Transformer model f_{orig} as the teacher and the hybrid model as the student. The objective can be formulated as

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

where D_{KL} is KL divergence. The teacher model weights are frozen in this stage. We use 1B training data for knowledge distillation, and adopt a cosine learning rate (LR) scheduler that decays from η_{stage2} to $1e-5$, where η_{stage2} is determined by a separate hyperparameter search for each model size. The effectiveness of this distillation setting is validated in Appendix F.2.

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.

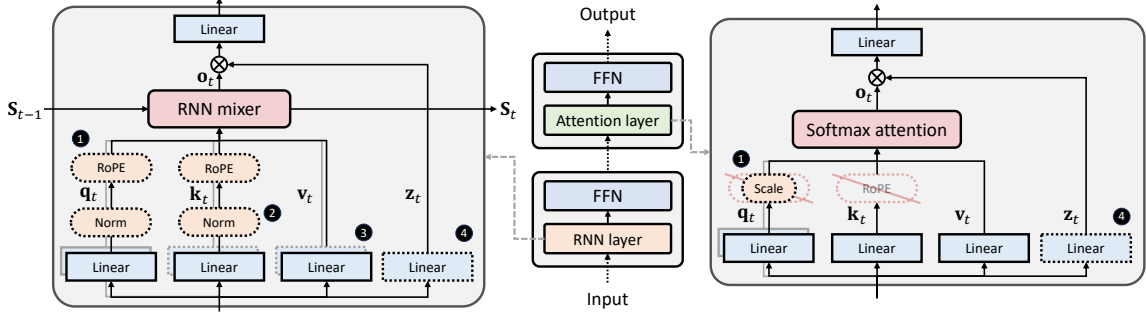


Figure 3: Illustration of HypeNet. The architectural modifications introduced during HALO are marked with ❶, ❷, ❸, and ❹. Red dotted lines indicate components that are removed during HALO, black dotted lines indicate components that are added.

5 HypeNet: Effective Attention-RNN 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 Section 5.3 for details). Appendix A gives a complete formulation of HypeNet.

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”, meaning 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. Hence, 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 length generalization depends 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:

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

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.1

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

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

$$\mathbf{q}_t = \text{Norm}(\mathbf{x}_t \mathbf{W}_q), \quad \mathbf{k}_t = \text{Norm}(\mathbf{x}_t \mathbf{W}_k). \quad (12)$$

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 (③) 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:

$$\mathbf{W}_{\square}^{(i)} \leftarrow \mathbf{W}_{\square}^{(\lfloor i/g \rfloor)}, \forall i \in \{1, \dots, n_h\}, \square \in \{k, v\} \quad (13)$$

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

Output Gate (④) Many recurrent architectures (Dao and Gu, 2024; Yang et al., 2025b) have an *output gate*, a data-dependent element-wise gating mechanism prior to the output projection:

$$\begin{aligned} \mathbf{o}_t &= \text{Mixer}(\mathbf{x}_t), \quad \mathbf{z}_t = \sigma(\mathbf{x}_t \mathbf{W}_z), \\ \mathbf{y}_t &= (\text{Norm}(\mathbf{o}_t) \odot \mathbf{z}_t) \mathbf{W}_o^T, \end{aligned} \quad (14)$$

where σ is an activation function, and $\mathbf{W}_z \in \mathbb{R}^{d \times d}$ is learnable parameters. 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 ③ and ④, 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.

5.3 RNN Mixer

HypeNet is agnostic to the RNN mixer as long as it takes QKV as the input. Thus, HypeNet can flexibly adopt any of the modern RNN mixers, including Lightning attention (Qin et al., 2024a), Mamba2 (Dao and Gu, 2024), GDN (Yang et al., 2025b), and RWKV-7 (Peng et al., 2025) (Appendix G discusses which RNN mixers are compatible). We experimented with various mixers and concluded that Lightning Attention provides the best balance between CSR and length generalization. Ablation results are reported in Section 6.6.

6 Experiments

We first describe the 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). Afterwards, we present ablation studies for HALO’s architectural modifications (Section 6.3) and attention layer selection method (Section 6.4). Finally, we analyze the inference efficiency of HypeNet (Section 6.5) and verify the effectiveness of various design choices in HypeNet (Section 6.6)

6.1 Experimental Setup

Models We apply HALO to the 1.7B, 4B, and 8B models of Qwen3 (Yang et al., 2025a), one of the most widely-used open-source LLM series.

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 NIAH⁴. More details are given in Appendix E.2.

Efficiency Measurement All efficiency measurements are conducted on servers with one NVIDIA A800 GPU. Softmax attention is implemented with Flash-Attention-2 (Dao, 2024), version 2.8.3. Other RNN mixers are implemented with Triton kernels from Flash-Linear-Attention (Yang and Zhang, 2024), version 0.4.1. Batch size is set to 1 for all models to ensure fair comparison.

6.2 Main Results

Figure 1 shows the CSR performance and efficiency of HypeNet compared to the Qwen3 series, and Table 2 reports the long-context recall performance. Also in Table 2, HypeNet + HALO is compared against recently released SOTA hybrid models that are distilled from pre-trained Transformer models.

⁴By default, NIAH refers to the average of NIAH-Single-1, NIAH-Single-2, and NIAH-Single-3 from RULER.

Model	Param	Token	NIAH-Single-1				NIAH-Single-2				NIAH-Single-3			
			32K	64K	128K	256K	32K	64K	128K	256K	32K	64K	128K	256K
<i>Qwen3 (teacher, no RNNs)</i>	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: Long-context recall performance of HypeNet + HALO versus SOTA hybrid models that are distilled from pre-trained Transformers. Qwen3 is evaluated with YaRN, as suggested by its authors. Best scores are bolded.

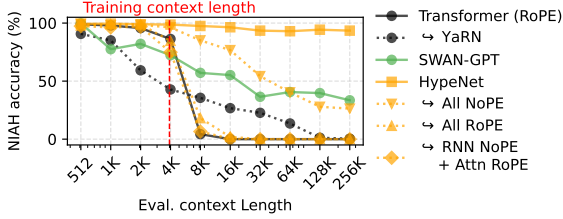


Figure 4: NIAH scores of HypeNet variants based on different position encodings, as a function of context length. The models are trained from scratch with 20B tokens and 500M parameters.

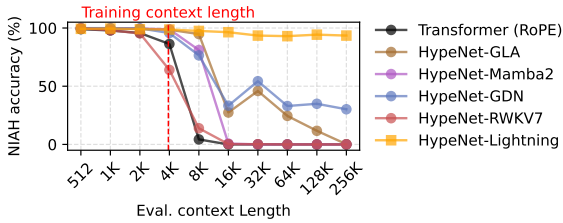


Figure 5: NIAH scores of HypeNet variants based on different RNN mixers, as a function of context length. The models are trained from scratch with 20B tokens and 500M parameters.

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 achieve superior long-context performance, despite using fewer training tokens, training with only open-source data, and being smaller than KL-LS (GDN).

6.3 HALO Ablations: Architectural Modifications

This section validates the effectiveness of the architectural modifications of HALO (①, ②, ③, and ④ in Figure 3). Table 3 reports the ablation results when converting Qwen3-1.7B, and it shows that our ar-

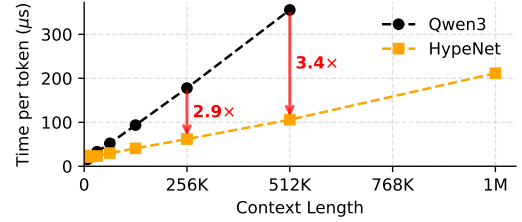


Figure 6: The prefilling time of HypeNet versus Qwen3-1.7B, across different context lengths.

chitectural modifications provide notable gains in CSR and NIAH performance, outperforming common approaches in training hybrid architectures.

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 that evenly distribute attention layers. We do not run the entire distillation procedures of Jet-Nemotron or KL-LS, which involve training on much more data. Our comparison is performed by replacing our attention layer selection method in HALO with these previous methods. The result is reported in Table 4, and it 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 at 128K context length, Figure 1 (right) shows the time per output token (TPOT), and Figure 6 shows the prefill speed results. In brief, HypeNet achieves up to 3.0 $\times$ decoding speedup and 3.4 $\times$ prefilling speedup on 512K context length.

Model	CSR	Needle-in-a-Haystack					
		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 3: Ablation experiment results for various architectural choices in HypeNet-2B, converted from Qwen3-1.7B.

Model	Class	CSR	Needle-in-a-Haystack					
			8K	16K	32K	64K	128K	256K
<i>Qwen3-1.7B (teacher, no RNNs)</i>	Transformer	58.5	99.7	99.9	99.9	99.5	38.6	18.4
HALO (ours)	Hybrid	55.9	94.9	90.3	94.1	90.6	79.9	74.3
Jet-Nemotron-2B (Gu et al., 2025)	Hybrid	55.0	88.7	70.1	70.3	61.9	63.7	56.2
KL-LS (Li et al., 2025)	Hybrid	55.3	85.7	78.4	72.8	68.9	58.3	44.3
Evenly distribute attn. layers	Hybrid	54.0	78.1	77.8	68.2	73.5	61.9	50.9
<i>RADLADS (RNN-only) (Goldstein et al., 2025)</i>	RNN	56.0	64.1	16.4	2.0	0.0	0.0	0.0

Table 4: 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.

6.6 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 H.

Position Encoding We compare HyPE against ordinary RoPE and SWAN-GPT (Puvvada et al., 2025), an architecture with a similar PE but is not a hybrid model. We also compare with HypeNet variants without HyPE (i.e., all RoPE, all NoPE, or attention RoPE + RNN NoPE). The results, reported in Figure 4, show that HyPE has much better length generalization. Notably, in contrast to distillation, from-scratch training achieves even better length generalization (93.5% NIAH accuracy on $64\times$ the training length).

RNN Mixers We also compare the performance of adopting different RNN mixers and report the results in Figure 5. Lightning Attention outperforms more recent RNN variants in terms of length generalization despite having a simpler update rule. One possible explanation for this is that more recent RNN mixers have *data-dependent* forget gates, which may result in poor length generalization, as

shown by Chen et al. (2025b).

Takeaway HyPE and Lightning Attention are essential for HypeNet’s great length generalization.

6.7 More Ablation Experiments

We also conducted experiments to study the effect of attention logits scaling (Appendix F.1), hyperparameters of HALO (Appendix F.2), transferability of HALO + HypeNet to model families (Appendix F.3), the ratio of attention layers (Appendix F.4). We report those results in the Appendix due to space limits.

7 Conclusion

We have proposed HALO, a distillation procedure for converting pre-trained Transformers 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. Our work is valuable for research in cost-efficient long-context LLMs. Our work also fosters research in novel LLM architectures by making it cheaper to empirically validate hybrid architectures at scale.

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.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (2024YFB4505603), National Natural Science Foundation of China (No. 62576186), supported by Tsinghua University Initiative Scientific Research Program.

References

- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. [GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints](#). *Preprint*, arXiv:2305.13245.
- Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2025. [Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes](#). *Preprint*, arXiv:2304.09433.
- Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. 2025. [Transformers to SSMs: Distilling Quadratic Knowledge to Subquadratic Models](#). *Preprint*, arXiv:2408.10189.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. [PIQA: Reasoning about Physical Commonsense in Natural Language](#). *Preprint*, arXiv:1911.11641.
- Yingfa Chen, Yutong Wu, Chenyang Song, Zhen Leng Thai, Xingyu Shen, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025a. [Cost-Optimal Grouped-Query Attention for Long-Context Modeling](#). *Preprint*, arXiv:2503.09579.
- Yingfa Chen, Xinrong Zhang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025b. [Stuffed Mamba: Oversized States Lead to the Inability to Forget](#). *Preprint*, arXiv:2410.07145.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge](#). *Preprint*, arXiv:1803.05457.
- Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*.
- Tri Dao and Albert Gu. 2024. [Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality](#). *Preprint*, arXiv:2405.21060.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. [The language model evaluation harness](#).
- Daniel Goldstein, Eric Alcaide, Janna Lu, and Eugene Cheah. 2025. [RADLADS: Rapid Attention Distillation to Linear Attention Decoders at Scale](#). *Preprint*, arXiv:2505.03005.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 34 others. 2024. [The Llama 3 Herd of Models](#). *Preprint*, arXiv:2407.21783.
- Albert Gu and Tri Dao. 2024. [Mamba: Linear-Time Sequence Modeling with Selective State Spaces](#). *Preprint*, arXiv:2312.00752.
- Yuxian Gu, Qinghao Hu, Shang Yang, Haocheng Xi, Junyu Chen, Song Han, and Han Cai. 2025. [Jet-Nemotron: Efficient Language Model with Post Neural Architecture Search](#). *Preprint*, arXiv:2508.15884.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring Massive Multitask Language Understanding](#). *Preprint*, arXiv:2009.03300.
- Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. 2020. [Query-Key Normalization for Transformers](#). *Preprint*, arXiv:2010.04245.
- Yuichiro Hoshino, Hideyuki Tachibana, Muneyoshi Inahara, and Hiroto Takegawa. 2025. [RAD: Redundancy-aware distillation for hybrid models via self-speculative decoding](#). *Preprint*, arXiv:2505.22135.

- Cheng-Ping Hsieh, Simeng Sun, Samuel Krizan, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. [RULER: What's the Real Context Size of Your Long-Context Language Models?](#) *Preprint*, arXiv:2404.06654.
- Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, and 6 others. 2024. [MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies](#). *Preprint*, arXiv:2404.06395.
- Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. 2024. [Repeat After Me: Transformers are Better than State Space Models at Copying](#). *Preprint*, arXiv:2402.01032.
- Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021. [Finetuning Pretrained Transformers into RNNs](#). *Preprint*, arXiv:2103.13076.
- Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. [Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention](#). *Preprint*, arXiv:2006.16236.
- Team Kimi, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, and 41 others. 2025. [Kimi Linear: An Expressive, Efficient Attention Architecture](#). *Preprint*, arXiv:2510.26692.
- Yanhong Li, Songlin Yang, Shawn Tan, Mayank Mishra, Rameswar Panda, Jiawei Zhou, and Yoon Kim. 2025. [Distilling to Hybrid Attention Models via KL-Guided Layer Selection](#). *Preprint*, arXiv:2512.20569.
- Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirum, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, and 3 others. 2024. [Jamba: A Hybrid Transformer-Mamba Language Model](#). *Preprint*, arXiv:2403.19887.
- Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. [OpenCeres: When Open Information Extraction Meets the Semi-Structured Web](#).
- Team MiniCPM, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, Ning Ding, Shengda Fan, Yewei Fang, Zixuan Fu, Wenyu Guan, Yitong Guan, Junshao Guo, Yufeng Han, Bingxiang He, and 64 others. 2025. [MiniCPM4: Ultra-Efficient LLMs on End Devices](#). *Preprint*, arXiv:2506.07900.
- MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, Enwei Jiao, Gengxin Li, Guojun Zhang, Haohai Sun, Houze Dong, Jiadai Zhu, Jiaqi Zhuang, Jiayuan Song, and 71 others. 2025. [MiniMax-01: Scaling Foundation Models with Lightning Attention](#). *Preprint*, arXiv:2501.08313.
- NVIDIA, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Kondratenko, Alexander Bukharin, Alexandre Milesi, Ali Taghibakhshi, Alisa Liu, Amelia Barton, Ameya Sunil Mahabaleshwarkar, and 31 others. 2025. [NVIDIA Nemotron 3: Efficient and Open Intelligence](#). *Preprint*, arXiv:2512.20856.
- Denis Paperno, Germán Kruszewski, Angeliki Lazari-dou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. [The LAMBADA dataset: Word prediction requiring a broad discourse context](#). *Preprint*, arXiv:1606.06031.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben al-lal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. [The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale](#). *Preprint*, arXiv:2406.17557.
- Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. 2025. [RWKV-7 "Goose" with Expressive Dynamic State Evolution](#). *Preprint*, arXiv:2503.14456.
- Krishna C. Puvvada, Faisal Ladhak, Santiago Aklé Serrano, Cheng-Ping Hsieh, Shantanu Acharya, Somshubra Majumdar, Fei Jia, Samuel Krizan, Simeng Sun, Dima Rekesh, and Boris Ginsburg. 2025. [SWAN-GPT: An Efficient and Scalable Approach for Long-Context Language Modeling](#). *Preprint*, arXiv:2504.08719.
- Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. 2024a. [Various Lengths, Constant Speed: Efficient Language Modeling with Lightning Attention](#). *Preprint*, arXiv:2405.17381.
- Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. 2024b. [HGRN2: Gated Linear RNNs with State Expansion](#). *Preprint*, arXiv:2404.07904.
- Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. [Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free](#). *Preprint*, arXiv:2505.06708.
- Qwen. 2025. [Qwen3-Next: Towards Ultimate Training & Inference Efficiency](#).

- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#). *Preprint*, arXiv:1606.05250.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. [Winogrande: An adversarial winograd schema challenge at scale](#). *Preprint*, arXiv:1907.10641.
- Xingyu Shen, Yingfa Chen, Zhen Leng Thai, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. [StateX: Enhancing RNN Recall via Post-training State Expansion](#). *Preprint*, arXiv:2509.22630.
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. [RoFormer: Enhanced Transformer with Rotary Position Embedding](#). *Preprint*, arXiv:2104.09864.
- Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. 2025. [Learning to \(Learn at Test Time\): RNNs with Expressive Hidden States](#). *Preprint*, arXiv:2407.04620.
- Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. [Retentive Network: A Successor to Transformer for Large Language Models](#). *Preprint*, arXiv:2307.08621.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention Is All You Need](#). *Preprint*, arXiv:1706.03762.
- Dustin Wang, Rui-Jie Zhu, Steven Abreu, Yong Shan, Taylor Kergan, Yuqi Pan, Yuhong Chou, Zheng Li, Ge Zhang, Wenhao Huang, and Jason Eshraghian. 2025a. [A Systematic Analysis of Hybrid Linear Attention](#). *Preprint*, arXiv:2507.06457.
- Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. 2025b. [The Mamba in the Llama: Distilling and accelerating hybrid models](#). *Preprint*, arXiv:2408.15237.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 Technical Report](#). *Preprint*, arXiv:2505.09388.
- Mingyu Yang, Mehdi Rezagholizadeh, Guihong Li, Vikram Appia, and Emad Barsoum. 2026. [Zebra-Llama: Towards Extremely Efficient Hybrid Models](#). *Preprint*, arXiv:2505.17272.
- Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025b. [Gated Delta Networks: Improving Mamba2 with Delta Rule](#). *Preprint*, arXiv:2412.06464.
- Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024. [Gated Linear Attention Transformers with Hardware-Efficient Training](#). *Preprint*, arXiv:2312.06635.
- Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2025c. [Parallelizing Linear Transformers with the Delta Rule over Sequence Length](#). *Preprint*, arXiv:2406.06484.
- Songlin Yang and Yu Zhang. 2024. [FLA: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism](#).
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a Machine Really Finish Your Sentence?](#) *Preprint*, arXiv:1905.07830.
- Biao Zhang and Rico Sennrich. 2019. [Root mean square layer normalization](#). *Preprint*, arXiv:1910.07467.
- Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. 2025. [LoLCATs: On Low-Rank Linearizing of Large Language Models](#). *Preprint*, arXiv:2410.10254.
- Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. 2024. [Gated Slot Attention for Efficient Linear-Time Sequence Modeling](#). *Preprint*, arXiv: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:

$$\begin{aligned}\mathbf{H}^{(l)} &= \text{Mixer}^{(l)} \left(\text{Norm} \left(\mathbf{X}^{(l-1)} \right) \right) + \mathbf{X}^{(l-1)} \in \mathbb{R}^{T \times d} \\ \mathbf{X}^{(l)} &= \text{MLP}^{(l)} \left(\text{Norm} \left(\mathbf{H}^{(l)} \right) \right) + \mathbf{H}^{(l)} \in \mathbb{R}^{T \times d}\end{aligned}\quad (15)$$

where $l \in \{1, \dots, L\}$ is the layer index and $\text{Norm}(\cdot)$ represents an RMSNorm (Zhang and Senrich, 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 $\mathcal{I}_{\text{attn}}$:

$$\text{Mixer}^{(l)} = \begin{cases} \text{ATTN}^{(l)} & \text{if } l \in \mathcal{I}_{\text{attn}} \\ \text{RNN}^{(l)} & \text{otherwise} \end{cases} \quad (16)$$

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:

$$\begin{aligned}\mathbf{q}_t &= \mathbf{x}_t \mathbf{W}_q \in \mathbb{R}^{1 \times d_h} \\ \mathbf{k}_t &= \mathbf{x}_t \mathbf{W}_k \in \mathbb{R}^{1 \times d_h} \\ \mathbf{v}_t &= \mathbf{x}_t \mathbf{W}_v \in \mathbb{R}^{1 \times d_h} \\ \tilde{\mathbf{q}}_t &= \frac{s_t \mathbf{q}_t}{\sqrt{d_h}} \in \mathbb{R}^{1 \times d_h}, \quad s_t = \log_a(t + a) \in \mathbb{R} \\ \mathbf{o}_t &= \sum_{i=1}^t \frac{\exp(\tilde{\mathbf{q}}_t \mathbf{k}_i^\top) \mathbf{v}_i}{\sum_{j=1}^t \exp(\tilde{\mathbf{q}}_j \mathbf{k}_j^\top)} \in \mathbb{R}^{1 \times d_h} \\ \mathbf{z}_t &= \text{sigmoid}(\mathbf{x}_t \mathbf{W}_z) \in \mathbb{R}^{1 \times d_h} \\ \mathbf{y}_t &= (\text{Norm}(\mathbf{o}_t) \odot \mathbf{z}_t) \mathbf{W}_o^\top \in \mathbb{R}^{1 \times d}\end{aligned}\quad (17)$$

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:

$$\begin{aligned}\mathbf{q}_t &= \text{Norm}(\mathbf{x}_t \mathbf{W}_q) \in \mathbb{R}^{1 \times d_h} \\ \mathbf{k}_t &= \text{Norm}(\mathbf{x}_t \mathbf{W}_k) \in \mathbb{R}^{1 \times d_h} \\ \mathbf{v}_t &= \mathbf{x}_t \mathbf{W}_v \in \mathbb{R}^{1 \times d_h} \\ \tilde{\mathbf{q}}_t &= \text{RoPE}_t(\mathbf{q}_t) \in \mathbb{R}^{1 \times d_h} \\ \tilde{\mathbf{k}}_t &= \frac{\text{RoPE}_t(\mathbf{k}_t)}{\sqrt{d_h}} \in \mathbb{R}^{1 \times d_h} \\ \mathbf{S}_t &= \mathbf{S}_{t-1} \gamma + \tilde{\mathbf{k}}_t^\top \mathbf{v}_t \in \mathbb{R}^{d_h \times d_h} \\ \mathbf{o}_t &= \tilde{\mathbf{q}}_t \mathbf{S}_t \in \mathbb{R}^{1 \times d_h} \\ \mathbf{z}_t &= \text{sigmoid}(\mathbf{x}_t \mathbf{W}_z) \in \mathbb{R}^{1 \times d_h} \\ \mathbf{y}_t &= (\text{Norm}(\mathbf{o}_t) \odot \mathbf{z}_t) \mathbf{W}_o^\top \in \mathbb{R}^{1 \times d}\end{aligned}\quad (18)$$

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:

$$\gamma_h = \exp\left(-2^{-8h/H}\right) \in (0, 1) \quad (19)$$

where $h \in \{1, \dots, H\}$ 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:

0.43132370	0.49306869	0.55178127	0.60653067
0.65675239	0.70218850	0.74281985	0.77880078
0.81040263	0.83796686	0.86186993	0.88249690
0.90022365	0.91540533	0.92836951	0.93941306
0.94880116	0.95676816	0.96351933	0.96923323
0.97406423	0.97814524	0.98159021	0.98449644
0.98694694	0.98901224	0.99075234	0.99221796
0.99345203	0.99449096	0.99536544	0.99610137

B HALO Training Configurations

Table 5 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.

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.

C HypeNet Model Configurations

Table 6 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 7.

D Addition Notes on the Model Architecture

Short Convolution Many recent RNNs (Dao and 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 8 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 Training 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 $700M \times L + 600M$ tokens, where L is the number of layers in the base Transformer. In the main content (Table 1 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

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 7. “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 `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:

- LAMBADA (Paperno et al., 2016)
- MMLU (Hendrycks et al., 2021)
- 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)

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

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	$\eta_{\text{stage2}} \rightarrow$ 1e-5	512	96	20K
3	1B	Constant	1e-5	16K	128	500

Table 5: Hyperparameters for each training stage in HALO. η_{stage2} is a hyperparameter that depends on the model (reported in Table 6).

Hyperparameter	HypeNet-2B	HypeNet-5B	HypeNet-9B
Vocab size	151936	151936	151936
Layers	28	36	36
Hidden size	2048	2560	4096
Attention 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
α (in Eq. (11))	500	600	900
η_{stage2} (see Table 5)	1e-4	5e-5	3e-5

Table 6: Hyperparameters of various HypeNet models.

Method	Layer indices (most important $\rightarrow$ least important)
<i>Qwen3-1.7B</i>	
HALO (ours)	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)	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)	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
<i>Qwen3-4B</i>	
HALO (ours)	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
<i>Qwen3-8B</i>	
HALO (ours)	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

Table 7: 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.

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 8: 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.

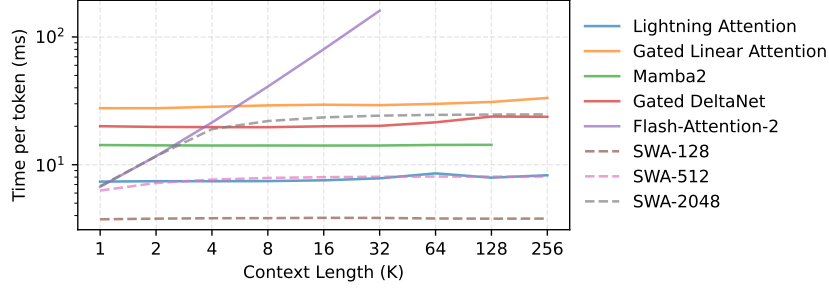


Figure 7: The inference prefilling time of various mixers as a function of context lengths, measured on one A800-80GB GPU using BFloat16. The sliding window mixers are implemented with Flash-Attention-2, Mamba2 is implemented with its official mamba_ssm library, and all other RNN mixers are taken from the widely used Flash-Linear-Attention. Mamba2 ran out of CUDA Memory on 256K context length. The y-axis is on log scale.

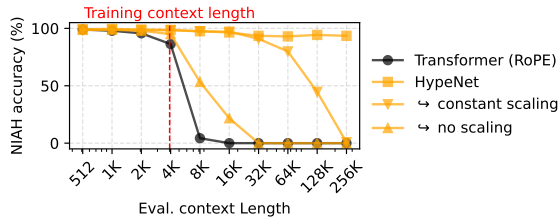


Figure 8: Results for validating attention logits scaling (see Eq. 11). The plot shows the NIAH performance of HypeNet without attention logits scaling, HypeNet with constant scaling (which is common in RoPE-based length extrapolation methods), and HypeNet with the attention logits scaling defined in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

F More Experimental Results

F.1 Ablation: Attention Logits Scaling

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:

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

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.2 Ablation: HALO Configuration

Table 9 presents the ablation experiments on the training configurations in our conversion procedure, HALO.

Stage 1 Results For stage 1, surprisingly, increasing the amount of training data beyond 320M tokens does not result in strong final performance.

Stage 2 Results 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.

Stage 3 Results For stage 3, the results show that the fine-tuning stage (stage 3) has a very small impact on the CSR performance but is essential for ensuring long-context recall capabilities.

F.3 Application to Other Model Families

In this section, we validate that our methods, HALO + HypeNet can be applied to model families beyond Qwen3 in the main content. We apply to the base model of MiniCPM4 (MiniCPM et al., 2025) using the same training configurations as those for Qwen3-8B. MiniCPM4 has 8B parameters and was pre-trained on 4K context lengths. Since the context length is much shorter than Qwen3, we evaluate using shorter context lengths than those in Section 6.2. The results are reported in Table 10, and they show that our method is transferable to other model families.

F.4 Attention Proportion

In this section, we report the results of HALO + HypeNet on Qwen3-1.7B when using varying proportion of attention layers. The results are shown in Table 11, and it shows that as the increase the proportion of attention layers, the CSR performance drops, but the recall performance increases. The 25% proportion, which is what we use by default, gives a good balance between CSR and recall performance.

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)	55.9	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)	55.9	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	70.1	46.1	27.1	14.8	4.3	0.3
w/ fine-tuning (ours)	55.9	95.9	94.9	90.3	94.1	90.6	79.9

Table 9: Ablation experiment results for different stages of HALO, applied to Qwen3-1.7B to convert into HypeNet.

Model	CSR	NIAH			
		4K	8K	16K	32K
MiniCPM4	63.5	100	96.0	0.0	0.0
HypeNet	61.2	85.7	75.5	74.1	76.2

Table 10: The results of converting MiniCPM4 into HypeNet with HALO.

Attn. prop.	CSR	NIAH			
		16K	32K	64K	128K
0%	56.4	0.5	0.3	0.1	0.3
14%	56.0	75.2	80.4	60.2	64.3
25%	55.9	90.3	94.1	90.6	79.9
50%	55.1	97.2	92.4	96.7	90.9

Table 11: Performance under different attention proportions when converting from Qwen3-1.7B, which has 28 layers.

F.5 Per-Task Results in Common-Sense Reasoning

For reproducibility, we report the results on each of the tasks in common-sense reasoning in Table 12.

F.6 The Inclusion of Test Data Attention Layer Selection

In this section, we discuss the possibility of test data leakage during attention layer selection (Section 4.3). The evaluation data that we used for determining attention layer importance includes ARC-easy, ARC-challenge, and HellaSwag, which is also a part of the CSR task suite. We emphasize that *this does not result in test data leakage*. This is because in our algorithm, the importance of an at-

tention layer (Eq. 8) is inversely proportional to the CSR performance drop when it is replaced with an RNN layer. This implies that the retained attention layers are those that lead to large CSR performance drop when they are replaced by RNNs. In other words, our attention replacement algorithm maximized the CSR performance drop measured with those three tasks. Therefore, using these tasks for attention layer selection does not lead to test data leakage, because the attention layer selection process is not optimizing the model’s performance on these tasks.

We choose to use these tasks for CSR evaluation to be consistent with prior works (Gu et al., 2025), but our preliminary results show that using other CSR tasks would have minimal impact on the final performance.

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.

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

This formulation includes (but is not limited to) the RNN mixers listed in Table 13. Table 14 describes how the notations from each of the mixers studied

Model	Param	Token	LMBD	MMLU	ARC-e	ARC-c	HS	WG	PIQA
<i>Qwen3-1.7B</i>	1.7B	–	46.9	55.5	69.7	43.1	60.4	61.8	72.1
HypeNet	2B	2.3B	39.5	47.0	68.9	42.5	57.1	63.1	72.5
KL-LS	3B	25B	40.4	46.1	68.2	40.6	57.1	62.4	72.4
Jet-Nemotron	2B	400B	40.7	45.6	67.8	41.0	56.8	61.5	71.9

Table 12: Model performance on each common-sense reasoning task.

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.

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 and 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, randomly initializing the RNN projection weights, adding more normalization layers. None of these helped.

H Training and Model Configurations for Training From Scratch Experiments

Here, we describe the training and model configurations for the experiments in Section 6.6.

H.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 15. The hyperparameters are chosen to best match standard practice in LLM pre-training.

H.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., $\text{Attn} \rightarrow \text{RNN} \rightarrow \text{RNN} \rightarrow \text{RNN}$). Since we are training from scratch, we do not need to handle attention layer selection as in HALO.

The MLP blocks after each attention/RNN block are always a SwiGLU block with the same hyperparameters. Table 16 reports the detailed configuration for each model, Table 17 reports the attention logits scaling (see Section 5.1) for each model, and Table 18 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.

I AI Assistants in Writing

We have used AI for quality checking this paper, but we never instructed AI to directly generate any content.

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 and 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 13: Non-exhaustive list of representative RNN mixers that are compatible with HypeNet and HALO.

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	$\text{diag}(\alpha_t)$	$\mathbf{q}_t$	$\mathbf{k}_t$	$\mathbf{v}_t$	$\mathbf{W}_Q$	$\mathbf{W}_K$	$\mathbf{W}_V$	$\mathbf{W}_O$
GDN	$\alpha_t (I - \beta_t \mathbf{k}_t^\top \mathbf{k}_t)$	$\mathbf{q}_t$	$\mathbf{k}_t$	$\mathbf{v}_t$	$\mathbf{W}_Q$	$\mathbf{W}_K$	$\mathbf{W}_V$	-
RWKV-7	$(\text{diag}(\omega_t) - \hat{\kappa}_t \mathbf{k}_t^\top (a_t \odot \hat{\kappa}_t))$	r_t	$\tilde{\mathbf{k}}_t$	ν_t	$\mathbf{W}_r$	$\mathbf{W}_k$	$\mathbf{W}_v$	$\mathbf{W}_o$

Table 14: 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.

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 15: Training configurations and hyperparameters used when training from scratch (Section 6.6).

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

Table 16: Model architecture configurations for the from-scratch training experiments (Section 6.6). The tokenizer for all models is the GPT-2 tokenizer.

Model	Logit scaling base (a in 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 17: The logits scaling hyperparameter of various models in the from-scratch training experiments (Section 6.6).

Hyperparameter	HypeNet-Lightning	HypeNet-GDN	HypeNet-GLA	HypeNet-RWKV7	HypeNet-Mamba2
<i>Gating & Normalization</i>					
Output gate	✓	✓	✓	✓	✓
Output norm	✓	✓	✓	✓	✓
QK norm	✓	✓, L_2 -norm	✗	✗	✗
QKV activation	✗	✓, SiLU	✗	✗	✗
Short Convolution	✗	✗	✗	✗	✗
F_t neg. eigenvalue	✗	✓	✗	✗	✗
<i>Low-Rank Parametrization</i>					
Gate low-rank dim.	–	–	16	160	–
Value low-rank dim.	–	–	–	96	–
Decay low-rank dim.	–	–	–	160	–
A low-rank dim.	–	–	–	160	–

Table 18: Hyperparameters for of the RNN layers in the HypeNet variants of the from-scratch training experiments (Section 6.6). ✓ denotes that the feature is enabled, ✗ denotes disabled, and “–” means that the hyperparameter is not applicable.