DenseSSM: State Space Models with Dense Hidden Connection for Efficient Large Language Models

Wei He* **Kai Han*†** **Yehui Tang** **Chengcheng Wang**
Yujie Yang **Tianyu Guo** **Yunhe Wang†**
Huawei Noah's Ark Lab
{hewei142, kai.han, yunhe.wang}@huawei.com

Abstract

Large language models (LLMs) face a significant challenge due to the excessive computational and memory requirements of the commonly used Transformer architecture. While state space model (SSM) is a new type of foundational network architecture offering lower computational complexity, their performance has yet to fully rival that of Transformers. This paper introduces DenseSSM, a novel approach to enhance the flow of hidden information between layers in SSMs. By selectively integrating shallow-layer hidden states into deeper layers, DenseSSM retains fine-grained information crucial for the final output. This incremental improvement maintains the training parallelizability and inference efficiency of SSMs while significantly boosting performance. The proposed method is broadly applicable to various SSM types, including RetNet and Mamba, and DenseSSM achieves significant performance improvements on public benchmarks, demonstrating its effectiveness and versatility.

1 Introduction

Since the release of ChatGPT (OpenAI, 2023), large language models (Team, 2023; Bai et al., 2023; Touvron et al., 2023; Zhou et al., 2024; Wang et al., 2023) have entered a new epoch, showcasing outstanding abilities in language comprehension, dialogue, and logical reasoning. Over the past year, the industry has witnessed the emergence of numerous large language models, such as LLaMA (Touvron et al., 2023) and ChatGLM (Zeng et al., 2023). These large language models have given rise to a plethora of practical applications, including conversational bots, code assistants, and AI agents. The foundation of large language models lies in the Transformer network structure (Vaswani et al., 2017), primarily utilizing a multi-head self-attention module for modeling relationships between tokens and a Feed-forward network for non-linear feature transformations. The scaling law (Kaplan et al., 2020) based on the Transformer structure has propelled the continuous development and expansion of large language models.

In the Transformer network, multi-head self-attention (MHSA) plays a crucial role, but it comes with significant computational demands and memory requirements during inference. In terms of computational complexity, for an input sentence of length N, the calculation of self-attention has a complexity of $O(N^2)$ during training and inference. Regarding memory usage, previously encountered keys and values are stored, leading to a memory occupation of $O(ND)$. As a result, recent efforts on network architectures have focused on simplifying Transformer by reducing its computation and space complexity. This includes various approaches, notably convolutional language models (Poli et al., 2023), recurrent unit (Lei, 2021), long context models (Ding et al., 2023), and state space models (SSMs) (Gu et al., 2021; Gu and Dao, 2023). These new models have provided strong alternatives to Transformer for building efficient LLMs.

SSMs propose modeling sequences by introducing an appropriate design of hidden states for handling long-range dependencies with both training parallelizability and inference efficiency. Starting from the continuous mapping system, SSMs are discretized to process discrete inputs in deep learning such as language sequence. The discretized SSMs can be computed in both linear recurrence and global convolution modes. Commonly, convolution mode is used during training to achieve parallel acceleration, while recurrence mode is used during autoregressive inference because it has lower computational complexity.

The core distinction of SSMs from other neu-

*Equal contribution
†Corresponding author

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 9243–9254

ral networks, such as fully-connected neural networks, lies in the design of hidden states. Hidden states enable information to be propagated along the temporal dimension, while avoiding the computation complexity of accessing historical tokens at each step. Through state transition parameters A, hidden states transfer the hidden information from the previous time steps to the current time step, allowing for autoregressive prediction of the next token. Hidden states play a crucial role in SSMs, but have not received sufficient investigation in the past. Weights and hidden features in different layers contain information at various levels from fine-grained to coarse-grained (Gu et al., 2021). However, in previous versions of SSMs, hidden states only flowed within the current layer and could not transmit more information to deeper layers, thus failing to capture more hierarchical information.

In this paper, we propose DenseSSM to facilitate a more comprehensive flow of hidden information between layers in state space models. We first analyze the hidden state degradation in conventional SSMs which will prevent hidden information flow from low levels to high levels. By selectively integrating shallow-layer hidden states into deeper layers, DenseSSM retains fine-grained information that is useful for the final output. The proposed method is applicable to different types of SSMs, such as RetNet (Sun et al., 2023) and Mamba (Gu and Dao, 2023). Our approach maintains the training parallelizability and inference efficiency of SSMs, while achieving a significant improvement with only a slight increase in the number of parameters. For instance, our DenseRetNet model outperforms traditional RetNet with up to 5% accuracy improvement on public benchmarks.

2 Related Works

2.1 Large Language Models

Large language models (LLMs) have seen transformative advancements, enabling them to excel in a diverse array of natural language processing (NLP) tasks, including machine translation, text summarization, and emergent abilities like incontext learning, which were previously unattainable by earlier language models (Devlin et al., 2019; Raffel et al., 2023). The evolution of LLMs has been marked by a monumental shift in scale, exemplified by models like GPT-3 (Brown et al., 2020), with its 175 billion parameters, and the even more expansive PaLM (Chowdhery et al., 2022), packing in a astounding 540 billion parameters. These models have empirically validated the scaling law (Kaplan et al., 2020), which posits that increasing model size leads to improved performance.

The rapid expansion in model size has underscored the critical need for the development of efficient Transformer algorithms (Dao et al., 2022; Dao, 2023; Gu et al., 2021, 2020; Smith et al., 2023; Fu et al., 2023; Mehta et al., 2022; Sun et al., 2023; Liu et al., 2024), where FlashAttention (Dao et al., 2022; Dao, 2023) has emerged as a significant innovation. This approach enhances the pivotal attention mechanism within Transformers by optimizing softmax computations using a technique known as tiling. By minimizing memory transactions between the GPU's HBM and on-chip SRAM, FlashAttention compute exact attention with fewer memory accesses, resulting in both faster execution and a lower memory footprint compared to standard attention implementations.

2.2 State Space Models

While the Transformer is currently the de facto architecture for large language models (LLMs), providing efficient parallel GPU training, the inference time for single-token inference increases significantly with longer sequence lengths, posing challenges for deployment due to the O(N) complexity per step even with accelerating algorithms like FlashAttention (Dao et al., 2022; Dao, 2023). Efforts have been dedicated to researching the Transformer-Next architecture, aiming to achieve state-of-the-art (SOTA) performance with efficient parallel training and effective inference, particularly for long sequence lengths.

State Space Sequence Models (SSMs) have recently emerged as promising architectures for sequence modeling. HiPPO (Gu et al., 2020) streamlines sequence modeling by compressing lengthy inputs into a dynamic, polynomial-based representation using orthogonal polynomials. S4 (Gu et al., 2021) introduced a novel parameterization through the application of a low-rank structured correction, enabling stable diagonalization and simplifying the process into Cauchy kernel operations. S5 (Smith et al., 2023) further simplifies the S4 layer by employing a single multi-input, multi-output SSM and introducing efficient parallel scan algorithms into the S4 layers. H3 (Fu et al., 2023) narrows the performance gap between SSMs and Transformer language models by designing three projections

(Q, K, V) to simulate the attention mechanism and adopting a fast Fourier transform (FFT) to reduce computation and memory consumption further.

GSS (Mehta et al., 2022) was the first gated neural network architecture incorporating SSMs, it builds upon (Hua et al., 2022) and introducing a compact SSM architecture that contracts model dimensions. Unlike GSS, which emphasizes compressing context into a smaller state, Mamba (Gu and Dao, 2023) diverges by focusing on enhancing the selectivity of the state representation, aiming to balance the tradeoff between efficiency and effectiveness without compromising the model's ability to capture essential information from the context. It achieves this by integrating a selection mechanism which enabling the model to selectively prioritize relevant information while concurrently utilizing a hardware-optimized algorithm.

2.3 Linear Attention

Linear attentions (Katharopoulos et al., 2020; Zhai et al., 2021), which remove the softmax operation from traditional attention, can be seen as a derivative of State Space Models (SSMs). They replace SSMs' convolutions with a variation of Multi-Head Attention (MHA) and eliminate the softmax of the traditional attention mechanism by utilizing a kernel function that operates independently on the queries (Q) and keys (K). These mechanisms also have a parallel form for efficient training and a recurrent form with $O(1)$ complexity.

RetNet (Sun et al., 2023), TransNormer-LLM (Qin et al., 2024), and RWKV (Peng et al., 2023) implement a fixed decay factor to update the previous key-value (KV) states at each recurrent step. This decay mechanism seamlessly integrates with the causal attention mask for efficient parallel computation. However, since this decay factor is preset and independent of the data, it may not be universally applicable across all tasks, especially when prompts or long-range information is particularly important. To address this challenge, GLA (Gated Linear Attention) (Yang et al., 2023) introduces data-dependent gating mechanisms that are practical for both parallel and block-parallel forms. It performs competitively against strong baselines, including the LLaMA-architecture Transformer (Touvron et al., 2023) and Mamba (Gu and Dao, 2023).

3 DenseSSM

In this section, we analyze the hidden state degradation in the deeper layers of SSMs and further introduce dense connection of hidden states to preserve richer information for deeper layers.

3.1 Prelimineries

Transformer Transformer is the widely-used network architecture of large language models which is based on the self-attention mechanism. The self-attention performs as follows:

$$o_t = W_o \frac{\sum_{i=1}^{T} e^{q_t^T k_i} v_i}{\sum_{i=1}^{T} e^{q_t^T k_i}} l, \tag{1}$$

where q, k and v are obtained by fully-connected layers, W_o is the linear transformation weight for the output token o_t at the t-th timestep. Each token will merge information of the other tokens by relationship weights calculated by the self-attention. In addition to self-attention module, the fee-forward network (FFN) module is another key component to transform the token representation and introduces more non-linearity. FFN module is usually composed by two stacked linear layers and non-linear activation function:

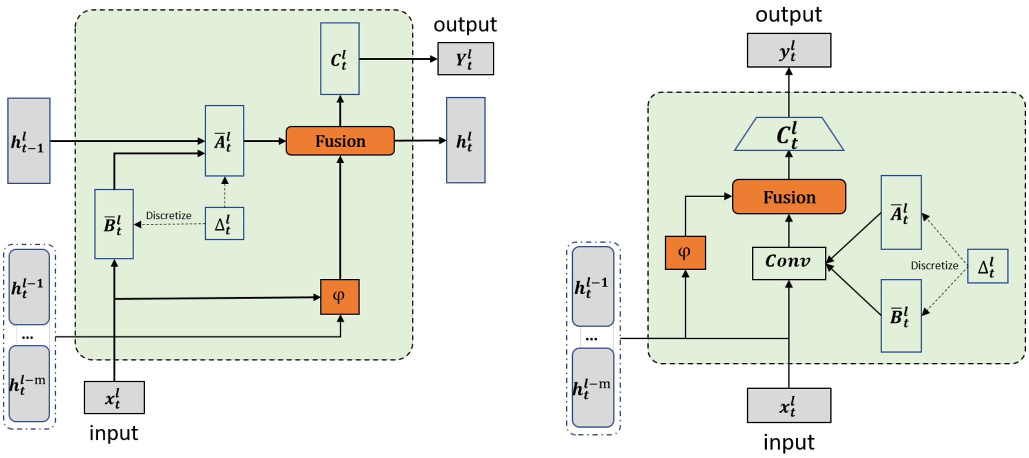
$$y_t = W_{down} \sigma(W_{up} o_t), \tag{2}$$

where W_{up} and W_{down} are the weight matrices of up projection and down projection layers, and $\sigma(\cdot)$ is the activation function such as GELU (Hendrycks and Gimpel, 2016).

SSM State space models (SSM) in the literature of deep learning refer to the class of structured SSMs (Gu et al., 2021) and the derivatives such as RWKV (Peng et al., 2023) and RetNet (Sun et al., 2023). Here we briefly describe the structured SSMs as a representative. Structured SSMs define a sequence-to-sequence transformation $x(t) \rightarrow y(t)$ with an implicit latent state $h(t)$. The continuous form is formulated as

$$h'(t) = Ah(t) + Bx(t), \tag{3}$$
$$y(t) = Ch(t), \tag{4}$$

where A, B and C are the parameters. To apply SSM to the real discrete data, we discretize the continuous case and obtain the recurrence formulation and convolution formulation of it. The parameters A and B are transformed to the discrete parameters $\overline{A}$ and $\overline{B}$ with the discretization rule such as



(a) DenseSSM in autoregressive mode.

(b) DenseSSM in parallelizable convolution mode.

Figure 1: Illustrations of DenseSSM framework, where ϕ is the selective transition module and 'Fusion' is the hidden fusion module.

zero-order hold (Gu et al., 2021). The recurrence formulation is

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t, \tag{5}$$
$$y_t = Ch_t. \tag{6}$$

The convolution formulation is

$$\overline{K} = (C\overline{B}, C\overline{AB}, \cdots, C\overline{A}^t\overline{B}), \tag{7}$$
$$y = x * \overline{K}, \tag{8}$$

where $*$ is convolution operation, and $t+1$ is the convolution kernel size. The recurrence mode is usually used for efficient autoregressive inference, while the convolution mode is used for efficient parallelizable training.

3.2 Dense Hidden Connection

Here we analyze the hidden information flow from shallow layers to deep layers. In the following, we use the superscript "l" to represent the l-th block.

$$\begin{aligned} h_t^l =& \overline{A}h_{t-1}^l + \overline{B}x_t^l \\ =& \overline{A}h_{t-1}^l + \overline{B}\Theta(y_t^{l-1}) \\ =& \overline{A}h_{t-1}^l + \overline{B}\Theta(Ch_t^{l-1}) \\ =& \overline{A}h_{t-1}^l + \overline{B}\Theta(C\overline{A}h_{t-1}^{l-1} + C\overline{B}\Theta(Ch_t^{l-2})) \\ =& \overline{A}h_{t-1}^l + \overline{B}\Theta(C\overline{A}h_{t-1}^{l-1} + \cdots \\ & + C\overline{B}\Theta(C\overline{A}h_{t-1}^{l-m+1} + C\overline{B}\Theta(Ch_t^{l-m}\underbrace{))\cdots)}_{m}, \end{aligned} \tag{9}$$

where $\Theta(\cdot)$ is the transformations from the last output to the input of SSM module, such as convolution and FFN. From Eq. 9, we can see that the transmission of hidden information from the $(l-m)$-th layer to the l-th layer requires passing through m transformation blocks and m BC matrix multiplications. Such a complex computational process can lead to significant information loss, meaning that attempting to retrieve certain information from the $(l-m)$-th layer at the l-th layer becomes very challenging and unclear.

Through the above analysis, we have identified a crucial issue in SSM, which is the decay of important hidden states as the layer depth increases. Therefore, we propose a dense connection for hidden states to better preserve fine-grained information from shallow layers, enhancing the ability of deep layers to perceive the original textual information. For the l-th block, we densely connect the hidden states in its previous m blocks. First, we collect the shallow hidden states and introduce a selective transition module ϕ to project them to the subspace of the target layer and select useful parts simultaneously:

$$\mathcal{H}_t^l = [\phi(h_t^{l-1}); \phi(h_t^{l-2}); \cdots; \phi(h_t^{l-m})], \tag{10}$$

Then, the intermediate hidden vectors are injected into the original hidden state of this layer:

$$h'^l_t = Fuse(h_t^l, \mathcal{H}_t^l). \tag{11}$$

The operation $Fuse()$ is the function to fuse the intermediate hidden vectors and the current hidden state. The SSMs with the proposed dense hidden connection is named as DenseSSM (Figure 1(a)). The DenseSSM scheme can be used in any SSM variant such as Mamba (Gu and Dao, 2023). Compared to DenseNet (Huang et al., 2017) for convolutional networks, the proposed DenseSSM densely connect the hidden states in SSMs, and the selective mechanism and fusion manner are more efficient for language modeling.

The above analysis is based on the recurrence mode, in the following we introduce the convolution mode of DenseSSM for efficient training. From Eq. 5, we have

$$\begin{aligned} h_t^l &= \overline{A}h_{t-1}^l + \overline{B}x_t^l \\ &= \overline{A}(\overline{A}h_{t-2}^l + \overline{B}x_{t-1}^l) + \overline{B}x_t^l \\ &= \overline{A}^2 h_{t-2}^l + \overline{AB}x_{t-1}^l + \overline{B}x_t^l \\ &= \overline{A}^t h_0^l + \overline{A}^{t-1}\overline{B}x_1^l + \cdots + \overline{AB}x_{t-1}^l + \overline{B}x_t^l \\ &= \overline{A}^t\overline{B}x_0^l + \overline{A}^{t-1}\overline{B}x_1^l + \cdots + \overline{AB}x_{t-1}^l + \overline{B}x_t^l. \end{aligned} \tag{12}$$

This process can be conducted by a convolution on the input sequence $(x_0^l, x_1^l, \cdots, x_t^l)$:

$$\begin{aligned} h_t^l &= \overline{A}^t\overline{B}x_0^l + \overline{A}^{t-1}\overline{B}x_1^l + \cdots + \overline{AB}x_{t-1}^l + \overline{B}x_t^l \\ &= (x_0^l, x_1^l, \cdots, x_t^l) * (\overline{B}, \overline{AB}, \cdots, \overline{A}^t\overline{B}). \end{aligned} \tag{13}$$

In the proposed DenseSSM, we enhance the hidden states by Eq. 11 and then obtain the outputs of SSM:

$$\begin{aligned} y_t^l = Ch_t'^l = CFuse((x_0^l, x_1^l, \cdots, x_t^l)* \\ (\overline{B}, \overline{AB}, \cdots, \overline{A}^t\overline{B}), \mathcal{H}_t^l). \end{aligned} \tag{14}$$

As shown in Figure 1(b), DenseSSM can be trained in parallelizable convolution mode.

Selective Transition Module The selective transition module $\phi(\cdot)$ is to project inputs to the target subspace and select the useful part of hidden information simultaneously. We implement the selective transition module with projection layer and gate selection mechanism, as shown in Figure 2. First, we project the hidden states in the previous m SSM blocks to the same space:

$$h_t'^{l-m} = Proj(h_t^{l-m}). \tag{15}$$

Then we generate the gate weights based on the input x_t^l and use them to select useful hidden states:

$$\phi(h_t^{l-m}) = h_t'^{l-m} \odot Gate(x_t^l). \tag{16}$$

Please note that the newly introduced modules must not compromise the training parallelizability and inference efficiency of the original SSM framework. Therefore, we maintain a simple and efficient implementation in practice. The projection layer is implemented using a linear transformation, while the gate module is implemented with a two-layer MLP with a SiLU activation (Elfwing et al., 2018).

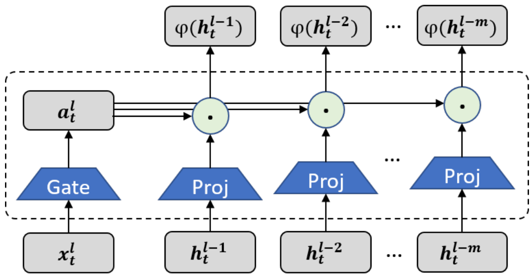


Figure 2: Selective Transition Module.

Hidden Fusion Module After the selective transition module, we obtain the selected hidden states from shallow layers, *i.e.*, $\mathcal{H}_t^L = [\phi(h_t^1); \phi(h_t^2); \cdots; \phi(h_t^{L-1})]$. A hidden fusion module is utilized to integrate shallow hidden states with the current hidden states. Similarly, we keep the implementation simple for efficiency. We add the selected hidden states since they have been projected to the same space:

$$h_t^L = Fuse(h_t^L, \mathcal{H}_t^L) = h_t^L + \sum_{i=1}^{m} h_t^{l-i}. \tag{17}$$

Here, we provide a basic implementation, but of course, there are other implementation approaches such as concatenation and cross-attention.

Extension to RetNet RetNet (Sun et al., 2023) can be viewed as a kind of state space models which uses a variant of self-attention rather than convolution in Eq. 7. Compared to the standard Transformer, RetNet is a RNN-style language model with fast inference and parallelized training. It utilizes linear attention to simplify the computation complexity of self-attention.

$$S_t = \gamma S_{t-1} + k_t^T v_t, \tag{18}$$

$$y_t = q_t S_t, \tag{19}$$

where S_t is the recurrent state, and $0 < \gamma < 1$. The dense KV connection for RetNet is performed as follows. The low-level keys and values are first concatenated:

$$\mathcal{K}_t^l = [\phi(k_t^{l-1}); \phi(k_t^{l-2}); \cdots; \phi(k_t^{l-m})], \tag{20}$$

$$\mathcal{V}_t^l = [\phi(v_t^{l-1}); \phi(v_t^{l-2}); \cdots; \phi(v_t^{l-m})]. \tag{21}$$

Then, the intermediate key (or value) vectors are injected into the original keys (or values) of this layer:

$$k'^L_t = k^L_t + \sum_{i=1}^{m} k^{l-i}_t, \tag{22}$$

$$v'^L_t = v^L_t + \sum_{i=1}^{m} v^{l-i}_t. \tag{23}$$

The RetNet equiped with the proposed dense key-value (KV) connections is named as DenseRetNet, as illustrated as shown in the appendix. In addition, the paralleizable mode of DenseRetNet is formulated as follows:

$$y_t = q_t \sum_{i=1}^{t} \gamma^{t-i} k'^T_i v'_i. \tag{24}$$

Our DenseRetNet can be implemented in parallelizable mode as well, that is, can be trained in parallel on GPUs or NPUs.

4 Experiments

In this section, we conducted comprehensive experiments to validate the effectiveness of the proposed DenseSSM. The verification was carried out on different architectures, including RetNet and Mamba

4.1 Data and Experimental Settings

Pretraining Data In our empirical analysis, we trained multiple models from scratch. Our experiments involved training on a dataset tokenized with the LLaMA tokenizer (Touvron et al., 2023), comprising 56GB of raw data sourced from 91 files sampled from The Pile (Gao et al., 2020). This dataset was randomly sampled from the full Pile dataset, excluding data from the DM_Mathematics and GitHub subsets, resulting in a cached dataset containing a total of 15 billion tokens. For a detailed list of the 15B data files sampled from the Pile in our analysis, see Ref. A.3.

Additionally, we conducted experiments with DenseMamba-1.4B, trained on the entire Pile dataset, extending to 300 billion tokens and utilizing the GPT-NeoX tokenizer (Black et al., 2022). This approach ensured a fair comparison with other models, such as Mamba and Pythia (Biderman et al., 2023).

Evaluation Datasets In our experiment, we investigate models performance across a spectrum of downstream tasks, focusing on zero-shot and 4-shot learning capabilities. The tasks, We benchmarked in Table 1, encompass a range of datasets designed to test common-sense reasoning and question-answering, such as HellaSwag (Zellers et al., 2019), BoolQ (Clark et al., 2019), COPA (Ponti et al., 2020), PIQA (Bisk et al., 2019), Winograd (Muennighoff et al., 2022), Winogrande (Sakaguchi et al., 2019), StoryCloze (Lin et al., 2021), OpenBookQA (Mihaylov et al., 2018), SciQ (Welbl et al., 2017), ARC_E (ARC-easy) and ARC_C (ARC-challenge) (Clark et al., 2018). Words Perplexity results of WikiText (Merity et al., 2016) and LAMBADA (LAMBADA_OPENAI) (Paperno et al., 2016) are also reported. All evaluations are executed using the LM evaluation harness (Gao et al., 2023), ensuring a standardized approach to assessing the models' capabilities.

4.2 Training Setup and Model's Architectures

To validate the effectiveness of our proposed method, we trained 350M and 1.3B DenseSSM models from scratch for one epoch. For experiments with 15 billion training tokens, we utilized a training batch size of 0.5 million tokens and training context length is set to 2048. The AdamW optimizer (Loshchilov and Hutter, 2019) was employed, featuring a polynomial learning rate decay and warm-up ratio is set to 1.5% of total training steps. We set the weight decay to 0.01 and applied gradient clipping at 1. In experiments conducted on The Pile (300B), we adhered to the training settings and model hyperparameters used in Mamba (Gu and Dao, 2023). Additionally, we designed our DenseRetNet model to be fully comprised of GAU-like blocks, which will be detailed in the subsequent paragraph.

Transformer-based language models We evaluate our proposed select dense mechanism against popular large language models like LLaMA (Touvron et al., 2023) and OPT (Zhang et al., 2022), comparing with LLaMA for 350M size models and with OPT for 1.3B size models. Their hyperparameters are reported in the appendix A.2.

Mamba In our experiments with a dataset containing 15 billion tokens, we followed the model structure in each Mamba layer and the training settings outlined in Mamba's paper. Specifically, we set the learning rate to 3e-4 for training the Mamba-360M model and 2e-4 for the Mamba-1.3B model, with no dropout applied in either case. Two additional layers were added to ensure a fair compar-

Models / Tasks	Wikitext↓	LAMBADA↓	ARC_C	ARC_E	BoolQ	COPA	HellaSwag	PIQA	WinoGrande	StoryCloze	Winograd	OpenBookQA	SciQ	Avg.↑
Zero-Shot														
LLaMA-350M	26.79	22.50	22.95	**46.13**	**59.27**	64	33.19	64.36	49.09	57.64	62.02	29.6	75.3	51.23
RetNet-350M	36.88	35.53	21.25	40.99	48.35	61	29.86	62.30	51.07	55.59	59.05	28.4	75.8	48.51
DenseRetNet-350M	31.35	19.92	23.72	45.03	58.50	**69**	32.31	64.04	52.09	58.04	60.82	**30.4**	76.6	51.87
Mamba-360M	26.60	17.55	23.98	45.83	55.78	61	**34.89**	64.31	**52.88**	58.90	62.92	29.2	**79.8**	51.77
DenseMamba-360M	**26.41**	**17.03**	**24.32**	46.0	59.20	66	34.68	**64.80**	51.14	**59.03**	**63.23**	29.8	**79.8**	**52.55**
Four-Shot														
LLaMA-350M	-	-	23.81	**47.26**	53.00	65	33.71	**64.15**	51.14	57.38	64.25	28.2	**81.2**	51.73
RetNet-350M	-	-	23.04	40.91	50.37	63	29.49	62.08	51.78	55.66	59.61	27.4	77.4	49.16
DenseRetNet-350M	-	-	24.74	45.66	54.89	**69**	32.14	63.70	52.01	57.58	59.23	28.2	78.3	51.41
Mamba-360M	-	-	**25.26**	46.51	45.41	63	34.25	65.13	**52.80**	**58.97**	62.88	**29.0**	81.0	51.29
DenseMamba-360M	-	-	24.83	46.97	**58.26**	66	**34.74**	64.69	52.01	58.37	**63.44**	28.6	80.3	**52.56**
Zero-Shot														
OPT-1.3B	22.04	13.79	24.66	48.65	58.07	63	37.00	65.89	**52.80**	61.02	65.51	29.6	81.1	53.39
RetNet-1.3B	27.90	23.41	22.61	46.34	48.75	58	32.25	63.44	49.96	57.71	60.65	23.4	77.3	49.13
DenseRetNet-1.3B	21.55	**10.88**	24.49	50.88	**58.62**	63	38.72	67.25	49.96	60.82	65.85	**31.8**	**82.7**	54.01
Mamba-1.3B	21.79	12.46	**25.09**	50.84	53.15	**67**	38.34	67.19	50.59	60.29	65.25	30.0	79.8	53.41
DenseMamba-1.3B	**21.39**	12.47	**25.09**	**51.89**	58.59	**67**	**39.26**	**67.90**	52.01	**61.28**	**66.11**	30.6	79.9	**54.51**
Four-Shot														
OPT-1.3B	-	-	25.94	50.46	52.35	63	36.97	64.64	52.33	60.09	**66.58**	28.2	**89.4**	53.63
RetNet-1.3B	-	-	24.66	46.30	47.49	67	31.96	63.22	52.09	57.51	61.42	26.6	80.3	50.78
DenseRetNet-1.3B	-	-	25.68	**53.07**	56.3	67	38.56	66.97	53.59	**62.08**	65.12	27.8	86.7	54.81
Mamba-1.3B	-	-	**26.96**	52.69	49.56	**69**	39.25	66.27	52.96	61.15	66.06	30.4	82.3	54.24
DenseMamba-1.3B	-	-	26.54	52.99	**58.59**	67	**39.26**	**67.08**	**53.67**	61.48	65.89	**31.0**	82.1	**55.05**

Table 1: Benchmarking results on the 15B Pile subset, comparing DenseSSM models with baseline models like RetNet (Sun et al., 2023) and Mamba (Gu and Dao, 2023), as well as Transformer-based models LLaMA-350M (Touvron et al., 2023) and OPT-1.3B (Zhang et al., 2022). DenseSSM models demonstrate lower perplexity and higher accuracy, enhancing the performance of SSM models and surpassing that of Transformer-based models.

ison in terms of parameter count. Details of the model's hyperparameters are provided in the appendix A.2. For experiments scaling up to the Pile dataset with 300 billion tokens, we used the same architecture as the original Mamba-1.4B model, with negligible increases in parameters and computational costs for dense hidden connections thanks to the relatively small hidden size of the Mamba architecture.

RetNet Model sizes and hyperparameters for our RetNet variants with DenSSM methods are shown in the appendix A.2. We further utilize Gated Attention Unit (GAU) (Hua et al., 2022) in our DenseRetNet. GAU combine Attention and FFN block into one, so a single block can perform both channel mixing and token mixing: $Y = (XW_u \odot A\hat{V})W_o$, where A is attention weight cauculated though Eq. 24. Also, multiple attention heads with different exponential decay rates are utilized to perform multi-scale decay instead of GAU's single-head strategy. In our experiments, we have observed that our architecture surpasses the origin RetNet structure in terms of training stability and performance.

4.3 Experiment Results

Experiment Results on 15B Pile-Subset Table 1 presents the experimental results from training with the 15B pile-subset, comparing DenseRetNet and DenseMamba with LLaMA (Touvron et al., 2023), OPT (Zhang et al., 2022), Mamba (Gu and Dao, 2023), and RetNet (Sun et al., 2023). DenseRetNet achieves lower perplexity on the Wikitext and LAMBADA, demonstrating clear advantages in downstream tasks in both zero-shot and few-shot settings, and significantly outperforms RetNet. Additionally, DenseMamba shows superior perplexity and accuracy on the test set, outperforming Mamba and other Transformer-based models.

Experiment Results on 300B Pile In our experiments with the 300B Pile dataset, we assessed the performance of DenseMamba-1.4B trained from scratch. We compared benchmark results from the original Mamba-1.4B (Gu and Dao, 2023), Pythia-1.4B (Biderman et al., 2023) and RWKV-1.5B (Peng et al., 2023), which were sourced from the Mamba paper. As illustrated in Table 2, DenseMamba-1.4B demonstrated a clear advantage over the original Mamba-1.4B and other models. This highlights the effectiveness of the DenseSSM approach in handling data at scale.

4.4 Ablation Studies

We conduct an ablation study to assess the impact of various design choices in our Selective Transition Module and Hidden Fusion Module. Word Perplexity results are reported for in-domain and out-of-domain corpora (Merity et al., 2016). We adjust model parameters to ensure fair comparisons across all studies under similar computational costs, using a 350M RetNet model as the baseline. Metrics are In-domain evaluation loss and out-of-domain Wikitext word perplexity, with training data consisting of 5B tokens tokenized using LLaMA tokenizer.

Model	Pile↓	LAMBADA↓	LAMBADA	HellaSwag	PIQA	Arc_E	Arc_C	WinoGrande	Avg. ↑
Pythia-1.4B	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	55.2
RWKV-1.5B	7.70	7.04	56.4	52.5	72.4	60.5	29.4	54.6	54.3
Mamba-1.4B	6.80	5.04	64.9	59.1	74.2	65.5	32.8	61.5	59.7
DenseMamba-1.4B	6.68	4.85	66.4	60.6	74.0	66.7	33.2	62.9	**60.6**

Table 2: Zero-shot benchmarking results when training the Pile(300B), comparing DenseSSM models with Pythia-1.4B (Biderman et al., 2023), RWKV-1.5B (Peng et al., 2023) and Mamba-1.4B (Gu and Dao, 2023).

Projection	**Select**	**#Param**	**In domain**	**Wikitext**
Vanilla RetNet	-	356M	2.524	46.82
None	None	346M	2.459	43.76
Identity	MLP	353M	**2.428**	**42.08**
Identity	Linear	357M	2.443	43.54
Linear	MLP	353M	2.460	43.37
Linear	Linear	356M	2.469	44.23

Table 3: Ablation on Selective Transition Module

Fusion Layers (m)	**Diff. gates**	**#Param**	**In domain**	**Wikitext**
Vanilla RetNet	-	356M	2.524	46.82
1	✗	353M	2.463	44.17
2	✗	353M	2.428	**42.05**
2	✓	360M	2.431	42.12
4	✗	353M	**2.420**	42.10
4	✓	374M	2.447	43.91

Table 4: Ablation on Different Fusion Layers and Gates

Ablations on Selective Transition Module The selective transition module projects shallow hidden states to a common subspace and selects useful parts, which can be implemented in various ways. Table 3 examines different settings for Projection and Select. With variables controlled (dense layers fixed at 2 and 'Add' operation used as fusion module), the results show that Identity Projection combined with a selection gate, learned from hidden states via a parameter-efficient MLP, optimally balances parameter efficiency and performance.

Ablations on Dense Layers We also conducted an ablation analysis on the depth of stored fusion layers (denoted as m). Our results, shown in Table 4, indicate that both two-layer(m=2) and four-layer (m=4) fusion architectures improve performance. Considering computational cost, the two-layer fusion is more optimal. Additionally, we explored the necessity of different selection gates for various stored dense layers m, different selection gates do not significantly impact performance, benefiting the development of lightweight dense connection architectures.

Ablations on Hidden Fusion Module In Table 5, we evaluate the efficiency and effectiveness of different hidden fusion module methods. Feature fusion, achieved either by concatenation followed by dimension reduction or by employing Cross-Attention, tends to increase the model's parameter count or computational cost. We opted for the addition (Add) method over Cross-Attention for our fusion strategy, prioritizing computational efficiency while maintaining performance comparability.

Fusion	**#Param**	**In domain**	**Wikitext**
Vanilla RetNet	356M	2.524	46.82
Concat	354M	2.440	43.75
Add	353M	2.428	**42.05**
Cross-Attention	353M	**2.422**	42.31

Table 5: Ablation on HiddenFusion module.

In Table 6, we investigate the performance of feature fusion when applied at different intervals across layers or at each layer using the same previously stored dense features ($m = 2$). Fusing at each layer facilitates information transfer from lower to higher layers more effectively.

Fuse Frequency	**#Param**	**In domain**	**Wikitext**
Vanilla RetNet	356M	2.524	46.82
Every layer	353M	**2.428**	**42.05**
Every 2 layers	353M	2.441	42.76
Every 4 layers	353M	2.455	44.20

Table 6: Ablation on Fusion Frequency.

5 Conclusion

In this paper, we propose **DenseSSM**, a framework designed to enhance the hidden information flow in SSMs. By selectively integrating hidden states from shallow layers into deeper layers, DenseSSM improves the model's ability to capture low-level textual information. This approach preserves the key advantages of SSMs, such as efficient autoregressive inference and parallelizable training. Experiments on Pile have validated the effectiveness of the DenseSSM method on both RetNet and Mamba, demonstrating its applicability to various SSM architectures.

6 Limitations

In this paper, our experiments primarily compare pure SSM methods, while comparisons involving hybrid architectures could be part of our future work. We have not yet tested larger-scale models and datasets, and the hyperparameters we propose for DenseSSM are optimized for models with sizes of 350M and 1.3B. It is important to note that as we scale the model, different hyperparameter strategies may prove more optimal, as they can impact the stability and efficiency of training.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

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.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *Preprint*, arXiv:1905.10044.

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. *ArXiv*, abs/1803.05457.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *Preprint*, arXiv:2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Preprint*, arXiv:2205.14135.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. 2023. Hungry hungry hippos: Towards language modeling with state space models. *Preprint*, arXiv:2212.14052.

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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487.

Albert Gu, Karan Goel, and Christopher Re. 2021. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. 2022. Transformer quality in linear time. *Preprint*, arXiv:2202.10447.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. *Preprint*, arXiv:2006.16236.

Tao Lei. 2021. When attention meets fast recurrence: Training language models with reduced compute. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7633–7648.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin Stoyanov, and Xian Li. 2021. Few-shot learning with multilingual language models. *CoRR*, abs/2112.10668.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. 2024. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2022. Long range language modeling via gated state spaces. *Preprint*, arXiv:2206.13947.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

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

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2022. Crosslingual generalization through multitask finetuning. *Preprint*, arXiv:2211.01786.

OpenAI. 2023. Chatgpt (mar 14 version). https://chat.openai.com/chat.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. 2023. Rwkv: Reinventing rnns for the transformer era. In *Findings of EMNLP 2023*.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*.

Edoardo M. Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and Yiran Zhong. 2024. Transnormerllm: A faster and better large language model with improved transnormer. *Preprint*, arXiv:2307.14995.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. 2023. Simplified state space layers for sequence modeling. *Preprint*, arXiv:2208.04933.

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.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yunhe Wang, Hanting Chen, Yehui Tang, Tianyu Guo, Kai Han, Ying Nie, Xutao Wang, Hailin Hu, Zheyuan Bai, Yun Wang, et al. 2023. Pangu-π: Enhancing language model architectures via nonlinearity compensation. *arXiv preprint arXiv:2312.17276*.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *NUT@EMNLP*.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2023. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. 2021. An attention free transformer. *Preprint*, arXiv:2105.14103.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Hang Zhou, Yehui Tang, Haochen Qin, Yujie Yang, Renren Jin, Deyi Xiong, Kai Han, and Yunhe Wang. 2024. Star-agents: Automatic data optimization with llm agents for instruction tuning. *arXiv preprint arXiv:2411.14497*.

A Appendix

A.1 Illustration of DenseRetNet

RetNet (Sun et al., 2023) can be viewed as a kind of state space models which uses a variant of self-attention rather than convolution. The autoregressive mode of DenseRetNet is shown in Figure 3. In addition, the paralleizable mode of DenseRetNet is formulated as follows:

$$y_t = q_t \sum_{i=1}^{t} \gamma^{t-i} {k'}_i^T {v'}_i. \tag{25}$$

Our DenseRetNet can be implemented in parallelizable mode as well, that is, can be trained in parallel on GPUs or NPUs.

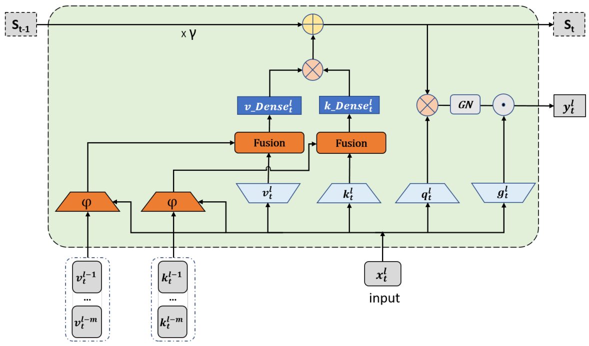


Figure 3: DenseRetNet in autoregressive mode.

Hyperparam	LLaMA 350M	RetNet 350M	DenseRetNet 360M	Mamba 360M	DenseMamba 360M
Layers	18	16	16	50	50
Hidden Size	1024	1216	1536	1024	1024
FFN Size	4096	2052	-	-	-
Heads	8	4	2	-	-
Dense Layers	-	-	2	4	-
Query & Key Size	-	-	768	-	-
Value & Gate Size	-	2052	3072	-	-
Learning-rate	6×10^{-4}	6×10^{-4}	6×10^{-4}	3×10^{-4}	3×10^{-4}
Adam β	$(0.9, 0.98)$	$(0.9, 0.98)$	$(0.9, 0.98)$	$(0.9, 0.95)$	$(0.9, 0.95)$
Dropout	0.0	0.1	0.1	0.0	0.0

Table 7: Key hyperparameters for 350M models

Hyperparam	OPT 1.3B	RetNet 1.3B	Mamba 1.3B	DenseRetNet 1.3B	DenseMamba 1.3B
Layers	24	24	50	25	50
Hidden Size	2048	2048	2048	2560	2048
FFN Size	8192	3456	-	-	-
Heads	32	8	-	4	-
Dense Layers	-	-	-	2	4
Query & Key Size	-	-	-	1280	-
Value & Gate Size	-	3456	-	5120	-
Learning-rate	6×10^{-4}	6×10^{-4}	2×10^{-4}	6×10^{-4}	2×10^{-4}
Adam β	$(0.9, 0.98)$	$(0.9, 0.98)$	$(0.9, 0.95)$	$(0.9, 0.98)$	$(0.9, 0.95)$
Dropout	0.1	0.1	0.0	0.1	0.0

Table 8: Key hyperparameters for 1.3B models

A.2 Details of the Compared Models

There are two model specifications, i.e., 350M and 1.3B, to verify the validity of our proposed dense mechanism. The details of the compared models including Mamba and RetNet a re listed in Table 7 and 8.

A.3 Details of the 15B Pile-Subset

Here are the details of the sampled files in the 15B Pile-Subset:

- `pile_ArXiv_{025, 069, 070, 092, 098, 123, 124, 133, 134, 157}.json`
- `pile_Books3_{015, 016, 052, 057, 071, 083, 084, 093, 115, 134, 173, 197, 203, 235, 242, 247}.json`
- `pile_Enron_Emails_004.json`
- `pile_FreeLaw_{031, 083, 104}.json`
- `pile_Gutenberg_PG-19_{044, 049}.json`
- `pile_OpenSubtitles_{008, 031, 037}.json`
- `pile_OpenWebText2_{011, 050, 063, 108, 118, 132, 157, 162, 212, 216, 242, 245, 256}.json`
- `pile_Pile-CC_{001, 024, 069, 076, 106, 120, 133, 181, 209, 211, 237, 254, 259}.json`
- `pile_PubMed_Abstracts_{037, 049, 054}.json`
- `pile_PubMed_Central_{028, 053, 067, 069, 085, 123, 125, 132, 149, 165, 173, 215, 220}.json`
- `pile_Stack_Exchange_055.json`
- `pile_USPTO_Backgrounds_{012, 027, 031, 051}.json`
- `pile_Ubuntu_IRC_{001, 017, 021}.json`
- `pile_Wikipedia_en_{006, 009, 043, 053, 070}.json`
- `pile_YoutubeSubtitles_008.json`