Flashback: 深層系列処理モデルのメモリ効率化・高速化のための記憶機構

関井 大気
サイバーエージェント
{sekii_taiki}@cyberagent.co.jp

概要

本稿では，従来研究における RNN ベースの深層系列処理モデルの持つ課題，（1）記憶の劣化，（2）不正確な勾配の逆伝搬，（3）次トークン予測に対する親和性，の 3 点を同時に解決する DNN 記憶機構の実現に取り組む．具体的には，課題 1〜2 に対処するため，記憶領域に保存されて以降，別の時刻の隠れ状態で上書きされるまで，記憶が保存時の値の恒等写像として完全に保持される Flashback 特性を定義し，その性質を満たす Flashback 機構を提案する．また課題 3 に対して，次トークン予測が可能な方式で，Flashback 機構を従来の Transformers と Mamba それぞれに組み込んだアーキテクチャを提案する．実験では，多様なテキストデータを含む The Pile データセットを学習に用いて，従来手法への Flashback 機構の導入による常識推論精度，処理速度，メモリ使用量のトレードオフを評価することで，Flashback 特性の有効性を検証した．

1 はじめに

ChatGPT [1] や Stable Diffusion [2] をきっかけとして「生成 AI」というワードが急速に広まるとともに，チャット機能 [1] や画像生成 [2]，プログラミング支援 [3] など多岐にわたるアプリケーションが普及し始めている．これらの多くは基盤モデル [4] をベースとしたシステムで構成されており，文章 [5, 6] に限らず動画や音声など複数のモダリティのデータを処理できるものが開発されている [7, 8]．特に，ChatGPT のように自然言語を通じてユーザーの問題を解決する用途では，基盤モデルのアーキテクチャとして，Transformers と呼ばれる Deep Neural Nerworks（DNN）に基づく深層系列処理モデル [5, 6, 9] が代表的に用いられる．学習データの増加に応じて処理精度が向上する Transformers のスケーラビリティによって，2022 年以前では実現が不可能であった高度なタスクを解くことが可能となった．

このような生成 AI の発展に伴い，新たに得られるユーザーの情報や逐次更新されるデータベースのように，学習データに含まれない入力情報（以降，外部記憶 [10, 11] と呼ぶ）を Transformers が一定以上保持できない点が，より高度なアプリケーションを実現する上で問題となる．これは，計算量やメモリ使用量が Attention 機構への入力の系列長に応じて増加する Transformers の設計に由来する [9, 12]．これに対処する方法として，記憶機構をアプリケーション側または DNN アーキテクチャ内に保持する二通りのアプローチが挙げられる．前者は DNN モデルを組み込むアプリケーション側で記憶能力を実現する方法であり，例えば，学習済みの大規模言語モデル（Large Language Models，LLM）で構成されたエージェントがデータベースを検索する方法 [13] や，Retrieval-Augmented Generation [14]（RAG）のように，データベースから関連する情報を抽出する方法が開発されている．これに対し，後者のアプローチは DNN アーキテクチャに記憶能力を持たせる方法 [10, 11, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]（Memory-Capable Neural Networks，MCNN）であり，各時刻の入力情報を隠れ層に保持する Recurrent Neural Networks（RNN）をベースとした手法や，これを発展させた，明示的に記憶領域を設ける手法（Memory-Augmented Neural Networks，MANN）に大別できる．

このうち RNN ベースの手法は，記憶領域を持つアーキテクチャをアプリケーションごとに開発する必要がない．近年では，入力の系列長に制限のない Sub-Quadratic な手法が注目されており，Mamba をはじめとした SSM [19, 21, 22, 25]（State Space Models）や Linear Attention 機構に基づく手法 [23, 24, 26] が目

覚ましい発展を遂げている．一方，Transformers をベースとして系列長の大きい入力に対して効率的に Attention を計算する改良 [27, 28, 29, 30, 31] が提案されている．このような Transformers や RNN に基づく手法は，大規模なデータセットでの自己教師あり学習に対する高い処理精度のスケーラビリティから，さまざまな用途において基盤モデルの Backbone として期待されている．しかしながら，携帯端末やロボット上での LLM の実行，テキストに加え動画や音声を含む大容量のデータを扱うマルチモーダル基盤モデルなど，より高度なアプリケーションを実現するためには，Backbone を構成する DNN モデルの高精度化・高速化・省メモリ化が問題となる．以上を踏まえ本研究では，入力データの系列長に制限のない RNN ベース手法の効率化に焦点を当てる．

RNN に基づく MCNN の従来研究では，応用面での性能やスケーラビリティを高める上で，総じて次のいずれかの点が課題となる．

課題 1. 記憶の劣化　外部記憶を保存するために，一定の時間間隔で記憶領域（隠れ状態や DNN 外部に設けたデータ）を必ず更新する．そのため，誤った記憶の改変や継続的な更新による記憶の変化（劣化）を完全に防ぐことは難しい．

課題 2. 不正確な勾配の逆伝搬　RNN が過去に指摘された問題と同様，継続的に記憶を更新するアーキテクチャでは，系列長が大きい場合に勾配が比較的多数の時刻の層を伝って逆伝搬されるため，勾配消失や勾配爆発の問題が生じる．

課題 3. 次トークン予測に対する親和性　外部記憶の保存が各時刻でなく入力を分割したセグメント単位や一定時間間隔でおこなわれる場合，近年 LLM の事前学習で採用されている次トークン予測（Next-Token Prediction）のアーキテクチャとしてそのまま用いることができない．

1.1　本研究の概要と貢献

本研究では，課題 1〜2 を原理的に回避し “記憶を鮮明に保つ” 性質（Flashback 特性）を持ちつつ，次トークン予測のアーキテクチャとして改変なしに導入できる記憶機構（Flashback 機構）を提案することで，課題 1〜3 を同時に解決することを目指す．Flashback 特性では，外部記憶は，保存されて以降の時刻に別の記憶で上書きされるまで，保存時の値の恒等写像として完全に保持される．これにより，ある時刻の記憶領域に対して，後に参照された時刻の処理結果から，勾配が直接逆伝搬される．図 1 に示すように，Flashback 機構における記憶領域は，RNN 同様隠れ状態として表現され，各時刻では，Flashback 特性を満たす方法で隠れ状態と入力データを比較し，必要に応じて記憶領域の各要素を置換する．最後に，次トークン予測の言語モデリングが可能な方式で，Flashback 機構を従来の Transformers と Mamba それぞれに組み込んだアーキテクチャを提案する．実験では，The Pile データセット [32] を学習に用いて，言語モデリングのタスクにおいて提案手法を評価した．具体的には，従来手法への Flashback 機構の導入による常識推論精度，処理速度，メモリ使用量の変化を評価することにより，Flashback 特性の有効性を検証した．

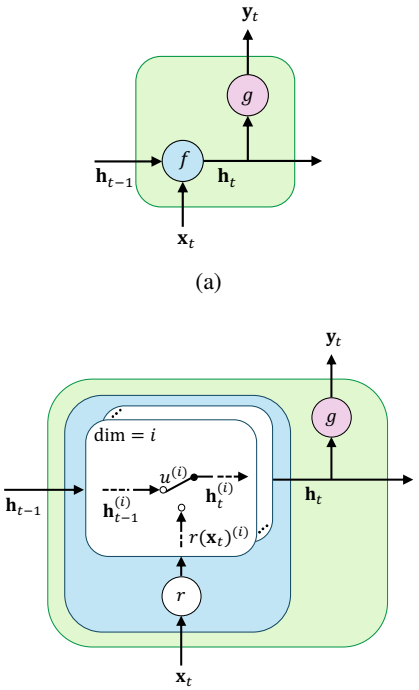

図 1　RNN（a）と Flashback 機構（b）の概念図.

本研究の貢献は，(1) MCNN の従来手法が持つ先述の 3 つの課題を解決するため，Flashback 特性を定義し，(2) Flashback 特性を持つアーキテクチャを，次トークン予測に基づく LLM に導入可能な Flashback 機構として提案することの 2 点である．

2　提案手法

本章では，はじめに RNN の原理を説明した上で，従来研究の課題を明らかにする．次に，Flashback 特性を定義し，その性質を満たす Flashback 機構とそれを組み込んだ DNN アーキテクチャを提案する．

2.1 RNN に基づく従来研究の課題

RNN では，時刻 $t-1$ における隠れ状態 $\mathbf{h}_{t-1} \in \mathbb{R}^D$ と入力 $\mathbf{x}_t \in \mathbb{R}^D$ から時刻 t の隠れ状態 $\mathbf{h}_t$ を次式のように求める．

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t), \tag{1}$$

ただし，$f(\cdot)$ は状態更新関数である．RNN の出力 $\mathbf{y}_t$ は，$\mathbf{h}_t$ を用いて次式のように計算される．

$$\mathbf{y}_t = g(\mathbf{h}_t), \tag{2}$$

ただし，$g(\cdot)$ は出力関数である．一般に，時刻 t から T 経過した時刻の隠れ状態 $\mathbf{h}_{t+T}$ は次式のように表される．

$$\mathbf{h}_{t+T} = f(f(f(\cdots f(\mathbf{h}_{t-1}, \mathbf{x}_t)\cdots), \mathbf{x}_{t+T-1}), \mathbf{x}_{t+T}). \tag{3}$$

このように，$\mathbf{h}_t$ は時刻 $t+T$ まで $f(\cdot)$ により再起的に変換され続けるため，誤った変換や誤差の蓄積により記憶が劣化するリスクがある（1 章の課題 1 を参照）．このとき，時刻 $t+T$ で得られた損失 L_{t+T} の，t の隠れ状態 $\mathbf{h}_t$ に対する勾配は，誤差逆伝搬法により次式のように計算できる．

$$\frac{\partial L_{t+T}}{\partial \mathbf{h}_t} = \frac{\partial L_{t+T}}{\partial \mathbf{h}_{t+T}} \cdot \frac{\partial \mathbf{h}_{t+T}}{\partial \mathbf{h}_{t+T-1}} \cdot \cdots \cdot \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}. \tag{4}$$

ここで，次トークン予測では全ての時刻に対して損失を計算することに注意されたい．このような連鎖律における多くのヤコビ行列の積が，勾配が消失または爆発する原因となる（1 章の課題 2 を参照）．

外部記憶の情報が隠れ状態に保存されると考える場合，MANN の従来研究の多くは，複雑ではあるものの本質的に RNN の基本原理にしたがう．例えば，DNC の Controller を用いた Memory 行列の操作，Recurrent Memory Transformer [20] の記憶トークンへの write 操作は，$f(\cdot)$ による隠れ状態の更新と捉えられる．また，SSM や Linear Attention ベースの手法は，RNN のコンセプトに基づいて設計されている．

2.2 Flashback 特性

隠れ状態 $\mathbf{h}_t$ の計算を次式のように表す．

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}) = \mathrm{Update}(r(\mathbf{x}_t), \mathbf{h}_{t-1}), \tag{5}$$

ただし，$r(\mathbf{x}_t) \in \mathbb{R}^D$ は線形または非線形に $\mathbf{x}_t$ を変換する関数である．$\mathrm{Update}(r(\mathbf{x}_t), \mathbf{h}_{t-1})$ は，次式のように，$\mathbf{h}_t$ の i 番目の要素に対して，$r(\mathbf{x}_t)$，$\mathbf{h}_{t-1}$ のいずれか一方の要素を選択する操作である．

$$\mathbf{h}_t^{(i)} = \begin{cases} r^{(i)}(\mathbf{x}_t) & \text{if } u_i = 1 \\ \mathbf{h}_{t-1}^{(i)} & \text{if } u_i = 0 \end{cases}, \tag{6}$$

ただし，u_i はあらかじめ設定された基準にしたがい，$\mathbf{h}_t$ の要素ごとに $\mathrm{Update}(\cdot)$ によって決定されるフラグである．

本研究では，隠れ状態 $\mathbf{h}_t$ の計算において，$\mathbf{h}_t$ の各要素が t 以前の時刻の入力や隠れ状態の要素の恒等写像となる性質に着目する．このような性質を持つ場合，隠れ状態が更新時の値で維持されるため，前節で述べた再起的な変換による記憶の劣化のリスクが少ない．また，時刻 $t+T$ の隠れ状態 $\mathbf{h}_{t+T}$ に，t における隠れ状態の i 番目の要素 $\mathbf{h}_t^{(i)}$ が保持されている場合，

$$\frac{\partial \mathbf{h}_{t+T}^{(i)}}{\partial \mathbf{h}_t^{(i)}} = 1 \tag{7}$$

が成立するため，式 (4) で述べた $t+T$ における損失の勾配 $\partial L_{t+T}/\partial \mathbf{h}_t^{(i)}$ が，連鎖率による $\mathbf{h}_{t+T}$ から $\mathbf{h}_t$ までのヤコビ行列の積を用いることなく計算できる．したがって，前節で述べた RNN における勾配の消失や爆発のリスクが小さくなる．本研究では，以上に述べた RNN の問題を回避する性質を Flashback 特性と呼ぶ．これにより，記憶領域の情報とそれに対する勾配が，長期間にわたってそれぞれ正確に順・逆伝搬されることで，汎化と学習の安定化が促進されることを狙う．

2.3 Flashback 機構

本研究では，Flashback 特性を持つ記憶機構を Flashback 機構として提案する．図 1 に Flashback 機構のアーキテクチャを示す．はじめに，$\mathbf{x}_t$ を線形変換した $r(\mathbf{x}_t)$ と隠れ状態 $\mathbf{h}_{t-1}$ を用いて，$\mathbf{h}_t$ を次式のように更新する．

$$\mathbf{h}_t = \mathrm{MaxPool}(r(\mathbf{x}_t); \mathbf{h}_{t-1}), \tag{8}$$

ただし，Flashback 特性に準じた最も単純な Update 関数として Max-Pooling を採用し，$\mathrm{MaxPool}(\cdot)$ は，チャンネル方向の要素から最大値を選択する Max-Pooling 関数である．次に，次式のように出力 $\mathbf{y}_t$ を得る[1)]．

$$\mathbf{z}_t = \mathrm{LayerNorm}(\mathbf{x}_t + \mathrm{GeLU}(q(\mathbf{x}_t)) + \mathrm{Norm}(\mathbf{h}_{t-1})), \tag{9}$$

$$\mathbf{y}_t = \mathrm{LayerNorm}(\mathbf{z}_t + \mathrm{FFN}(\mathbf{z}_t)), \tag{10}$$

ただし，$\mathrm{LayerNorm}(\cdot)$ は Layer Normalization をおこなう関数，$\mathrm{GeLU}(\cdot)$ は Gaussian Error Linear Unit を表

1) 本節の定式化では，入力から出力関数までの定義が，簡略化された前節の RNN の定式化に厳密に一致しないものの，その原理においては等価である．

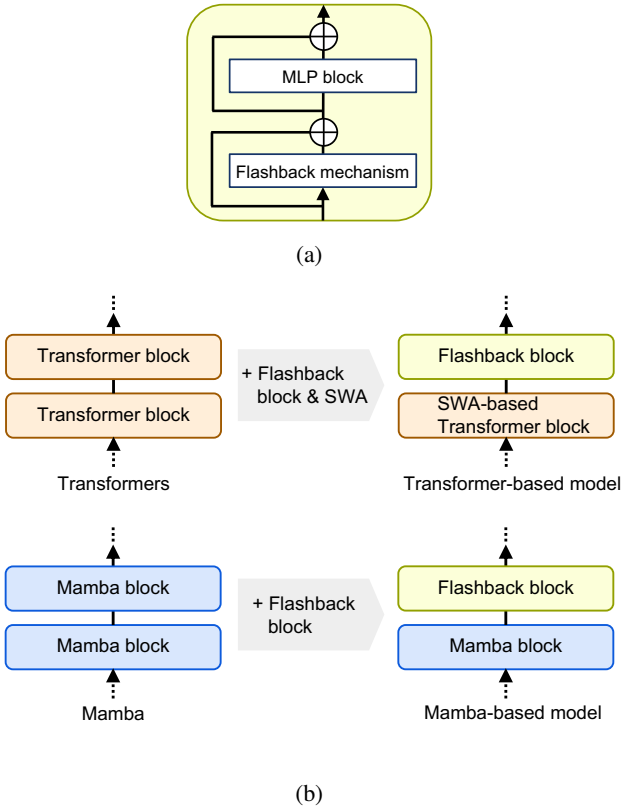

図 2　提案手法における DNN アーキテクチャの概要図．Flashback 機構（a）を Transformer++・Mamba（b）に導入．

す関数，Norm(·) は正規化関数，また，FFN(·) は Feed Forward Networks（FFN）を表す関数である．

2.4　Flashback 特性に基づく DNN アーキテクチャ

本稿では，Transformers と Mamba それぞれに Flashback 機構を統合した DNN アーキテクチャを提案する．Transformers アーキテクチャのベースとして，Transformer++ [5]（LLaMa アーキテクチャ）を用いる．Transformers アーキテクチャでは，FFN を含む Attention 機構を一組のブロックとして扱う．また Mamba では，SSM アーキテクチャを一組のブロックとする．図 2 に Flashback 機構を統合した Transformer++と Mamba のアーキテクチャを示す．Transformer++・Mamba の偶数番目のブロック位置に Flashback 機構を挿入することで，アーキテクチャに Flashback 特性を導入する．このとき，Transformers ベースのアーキテクチャでは Sliding Window Attention [29]（SWA）を採用することで，計算量を Sub-Quadratic な範囲に抑え，処理効率が入力の系列長に依存して悪化しないようにする．Flashback 機構の入出力のトークンや隠れ状態の次元は，組み込む従来手法の設定を踏襲する．

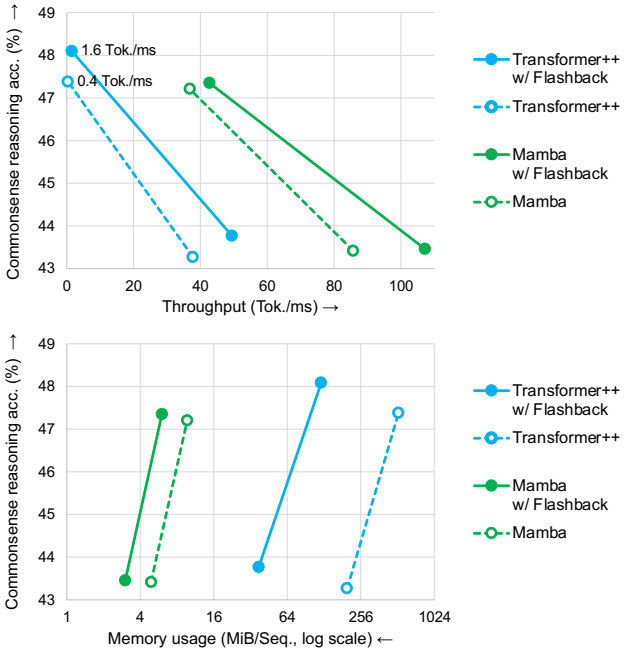

図 3　Transformer++・Mamba への Flashback 機構の導入による常識推論精度，処理速度，メモリ使用量の変化．1.3B と 360M の 2 パターンのパラメータ数で実験．

3　評価実験

本実験では，DNN の学習に The Pile データセット [32] を用いる．また，LM-Eval harness [33] を用いて，複数のデータセットに渡って常識推論精度を評価する．その他の実験の設定の詳細については，付録 A を参照されたい．

従来手法に Flashback 機構を導入した際の性能の変化を評価した結果を図 3 に示す．同図より，Transformer++（1.3B）に Flashback 機構を導入し，SWA により計算量を抑制することで，常識推論精度の向上とともに，4 倍以上の高速化と 77 [%] のメモリ使用量の削減の効果が現れている．また，Mamba（1.3B）に Flashback 機構を導入することで，16 [%] の高速化と 38 [%] のメモリ使用量の削減を同時に実現している．以上を踏まえると，1 章の MCNN の課題 1〜3 に対し，提案手法が次トークン予測のための記憶機構として効率的に動作することがわかる．

4　まとめ

本稿では，MCNN の従来研究の持つ 3 つの課題の同時解決に取り組んだ．記憶の劣化と勾配消失・爆発を原理的に回避するための性質を Flashback 特性として定義するとともに，その性質を満たす Flashback 機構と DNN アーキテクチャを提案した．

参考文献

[1] OpenAI. ChatGPT.

[2] Stability AI. Stable Diffusion.

[3] GitHub, Inc. GitHub Copilot.

[4] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, et al. On the opportunities and risks of foundation models. **arXiv preprint arXiv:2108.07258**, 2021.

[5] Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. **arXiv preprint arXiv:2307.09288**, 2023.

[6] OpenAI, Josh Achiam, Steven Adler, et al. GPT-4 technical report. **arXiv preprint arXiv:2303.08774**, 2024.

[7] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In **CVPR**, 2023.

[8] Neelu Madan, Andreas Moegelmose, Rajat Modi, Yogesh S. Rawat, and Thomas B. Moeslund. Foundation models for video understanding: A survey, 2024.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In **NeurIPS**, 2017.

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. **arXiv preprint arXiv:1410.5401**, 2014.

[11] Alex Graves, Greg Wayne, Malcolm Reynolds, et al. Hybrid computing using a neural network with dynamic external memory. **Nature**, Vol. 538, pp. 471–476, 2016.

[12] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In **ACL**, 2019.

[13] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In **EMNLP**, 2023.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In **NeurIPS**, 2020.

[15] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In **ICLR**, 2020.

[16] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In **ICLR**, 2015.

[17] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In **Advances in Neural Information Processing Systems**, 2015.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. **Neural Computation**, Vol. 9, No. 8, pp. 1735–1780, 1997.

[19] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers. **NeurIPS**, 2021.

[20] Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In **NeurIPS**, 2022.

[21] Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards language modeling with state space models. In **ICLR**, 2023.

[22] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In **ICLR**, 2022.

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

[24] Bo Peng, Eric Alcaide, Quentin Anthony, et al. RWKV: Reinventing RNNs for the transformer era. In **EMNLP**, 2023.

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

[26] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In **ICML**, 2024.

[27] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In **ICML**, 2018.

[28] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. **arXiv preprint arXiv:1904.10509**, 2019.

[29] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. **arXiv preprint arXiv:2004.05150**, 2020.

[30] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. **arXiv preprint arXiv:2307.08621**, 2023.

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

[32] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. **arXiv preprint arXiv:2101.00027**, 2020.

[33] 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. A framework for few-shot language model evaluation, 07 2024.

[34] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In **ICLR**, 2024.

Method	w/ Flashback	Param.	Pile Ppl. ↓	LAMBADA Acc. ↑	HellaSwag Acc. Norm. ↑	PIQA Acc. ↑	ARC-E/-C Acc./Acc. Norm. ↑	WinoGrande Acc. ↑	Average Acc. ↑
H3		360M	10.60	23.58	30.62	63.11	45.20/23.29	50.28	39.35
RWKV v5			9.79	—	—	—	—	—	—
GLA			9.12	—	—	—	—	—	—
BASED			8.65	38.13	33.17	64.58	46.97/24.40	50.59	42.97
Mamba			8.64	39.12	**33.87**	**64.69**	47.85/**24.57**	50.43	43.42
	✓		**8.60**	**40.36**	33.75	64.58	**48.36**/22.35	**51.38**	**43.46**
Transformer++			8.39	38.81	33.61	**64.69**	46.63/24.32	**51.54**	43.27
	✓		**8.38**	**40.89**	**34.13**	64.64	**47.43/25.26**	50.28	**43.77**
Mamba		1.3B	7.48	**46.85**	39.36	**67.57**	51.89/26.11	51.46	47.21
	✓		**7.43**	46.17	**39.46**	67.08	**52.10/26.54**	**52.80**	**47.35**
Transformer++			7.26	48.22	39.08	67.63	51.09/26.11	**52.17**	47.38
	✓		**7.19**	**50.13**	**41.60**	**67.74**	**52.86/26.28**	49.96	**48.09**

表 1　The Pile データセットを用いた常識推論精度の比較結果．従来研究 [25] の実験と同じデータセットを利用．

A　評価実験の詳細

A.1　データセット

A.1.1　学習データ

The Pile データセット [32] は、さまざまなソースから集められた高品質なテキストの集合であり，深層系列処理モデルの従来研究において近年広く利用されている．具体的には，22 のサブセットから構成されており，総容量は 800 [GB] に及ぶ．これらのサブセットには，論文，書籍，プログラム，技術文書などが含まれ，多岐にわたる種類のテキストデータを用いて言語モデルの汎化性を評価することができる．

A.1.2　評価データ

LM-Eval Harness [33] は，EleutherAI によって提供されている標準化されたベンチマークである．本実験では，表 1 に示す複数のデータセットを評価に用いた．なお，図 3 に示した常識推論精度は，これらのデータセットに対する精度の平均である．

A.2　実験の設定

BASED [26] の公開実装をベースに従来手法の評価と提案手法の実装をおこなった．従来手法と提案手法ともに，The Pile データセットから 10 億トークン分のテキストを同じ順序で学習した．これらのテキストは，GPT-2 BPE トークナイザーを用いてトークン化された．Attention 機構を用いる手法では，学習と評価に FlashAttention-2 [34] を用いた．学習時のバッチサイズは各手法でメモリ使用量が最大になるよう調整した．学習時の入力の系列長は各手法共通の 2048 に設定した．

提案手法を組み込む Transformer++と Mamba のモデルサイズとして，パラメータ数がおおよそ 360M と 1.3B の 2 パターンを用いた．提案手法を組み込む Transformer++の Sliding Window Attention の窓サイズを 512（学習する入力の系列長の 1/4）に設定した．2.4 節で述べた提案手法のアーキテクチャにおけるブロックの総数を，トークンの予測精度（Perplexity）が提案手法を導入する前後でおおよそ一致するよう調整した．

A.2.1　評価方法

各データセットで用いられる評価指標については，Arora ら [26] と同一のものを用いているため，詳細は文献を参照されたい．各手法の性能を公平に比較するため，学習と評価に同一の NVIDIA A100 GPU を用いた．各手法の処理速度の比較には，単位時間あたりのトークン生成のスループット（Tok./ms）を用いる．処理速度の計測時には，各手法のバッチサイズを，実行可能な範囲でスループットが最大になるよう調整した．これは，手法間の計算特性やハードウェア利用効率の違いを考慮し，各手法の性能を最大限に引き出すことで公平な比較を実現するためである．また，各手法のメモリ効率の比較には，処理速度計測時の GPU メモリの総使用量をバッチサイズで除算した量（MiB/Seq.）を用いる．

A.2.2　ベースライン手法

本実験では，従来研究における主要なベースライン手法 [5, 25] を選択し，その性能を提案手法と比較した．Transformer++ [5] は Rotary Encoding による位置埋め込みと Gated Linear Unit を Transformers に追加したものである．Mamba [25] は，重要な情報を入力に依存して選択的に隠れ状態に保存できる SSM の最新の手法である．