Hungry Hungry Hippos: Towards Language Modeling with State Space Models

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, Christopher Re

第15回最先端NLP勉強会
読み手：牧野 晃平（豊田工大）

https://www.amazon.co.jp/Hasbro-Hungry-Hippos-%E4%B8%A6%E8%A1%8C%E8%BC%B8%E5%85%A5%E5%93%81/dp/B01M24DD0K

H3

Hungry Hungry Hippos: Towards Language Modeling with State Space Models

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, Christopher Re

第15回最先端NLP勉強会
読み手：牧野 晃平（豊田工大）

https://www.amazon.co.jp/Hasbro-Hungry-Hippos-%E4%B8%A6%E8%A1%8C%E8%BC%B8%E5%85%A5%E5%93%81/dp/B01M24DD0K

論文の位置づけ

状態空間モデル
[Brogan+74]

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

深層学習に導入

設計された
A行列

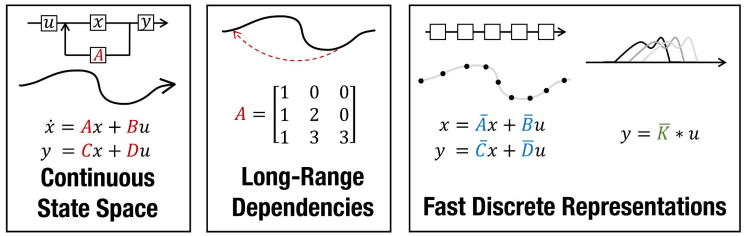
初期化に利用

HiPPO [Gu+20]

$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

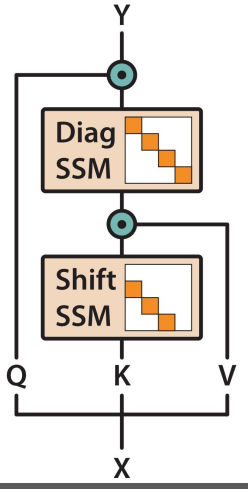
系列を扱える深層学習モデル

S4 [Gu+22]



言語に特化

H3 [Fu+23]



計算速度: S4 >Att
計算時間: S4 ≦Att
CV: S4 ≒ Att
音声: S4 ≒ Att
時系列: S4 ≒ Att
NLP: S4 < Att

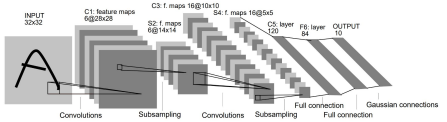
インスピレーション

variant

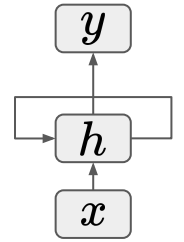
Linear Attention
[Katharopoulos+20]

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

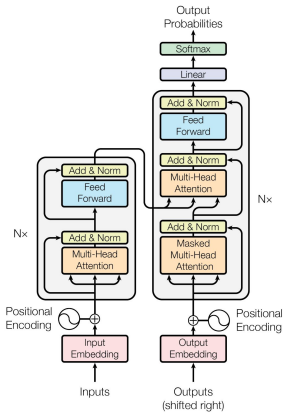
CNN [LeCunn+18]



RNN



Attention
[Vaswani+19]



論文の位置づけ

系列を扱える深層学習モデル

状態空間モデル
[Brogan+74]

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

深層学習に導入

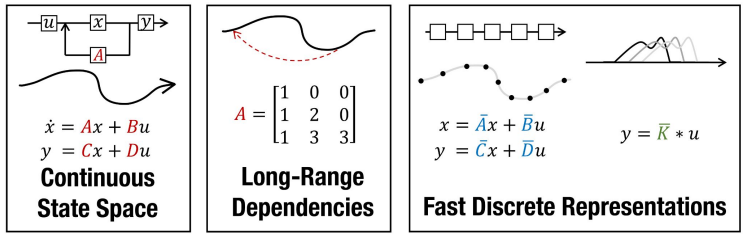
設計された
A行列

初期化に利用

HiPPO [Gu+20]

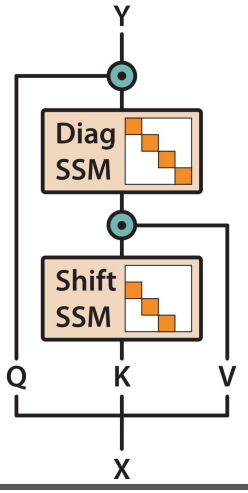
$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

S4 [Gu+22]



言語に特化

H3 [Fu+23]



計算速度: S4 > Att
計算時間: S4 ≦ Att
CV: S4 ≒ Att
音声: S4 ≒ Att
時系列: S4 ≒ Att
NLP: S4 < Att

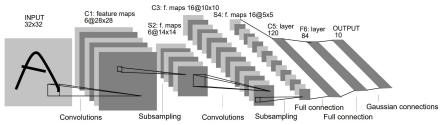
インスピレーション

variant

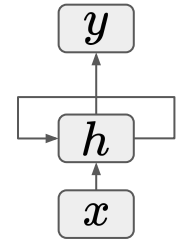
Linear Attention
[Katharopoulos+20]

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

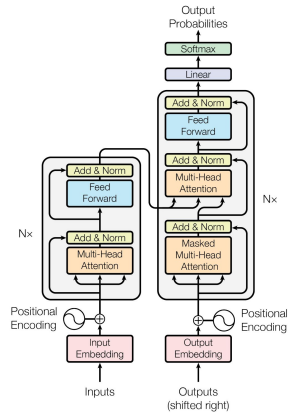
CNN [LeCunn+18]



RNN



Attention
[Vaswani+19]



状態空間モデル（State Space Model; SSM）

中間状態が必要な系を記述するためのモデル

- 時系列解析や制御システムによく利用される

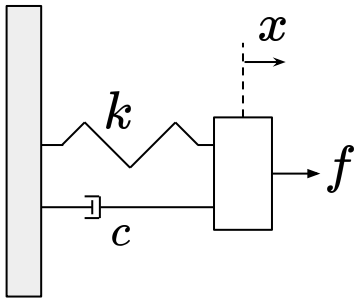
$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

u : 入力
x : 中間状態
y : 出力

例 : 質点—バネ—ダンパー系

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -c & -k \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f$$

$$y = x = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$



離散的な系列を扱えるようにSSMを離散化すると :

$$x_i = \bar{A}x_{i-1} + \bar{B}u_i \quad \bar{A} = (I - \Delta/2 \cdot A)^{-1}(I + \Delta/2 \cdot A)$$
$$y_i = \bar{C}x_i + \bar{D}u_i \quad \bar{B} = (I - \Delta/2 \cdot A)^{-1}\Delta B \quad \bar{C} = C \quad \bar{D} = D$$

SSMの深層学習への導入[Gu+21]

行列をパラメタにして離散化したSSMをニューラルネットに導入

- SSMを特徴量抽出器として使用
  - 順番にx, yを計算していくと，SSMは系列の変換器に見える
  - 前の層で計算されたyを次の層のuだと思って計算
- TransformerのAttention Weightの計算がなくて，系列長に対してほぼ線形に計算できる
- SSMを計算機で並列に扱うための畳み込み形式：

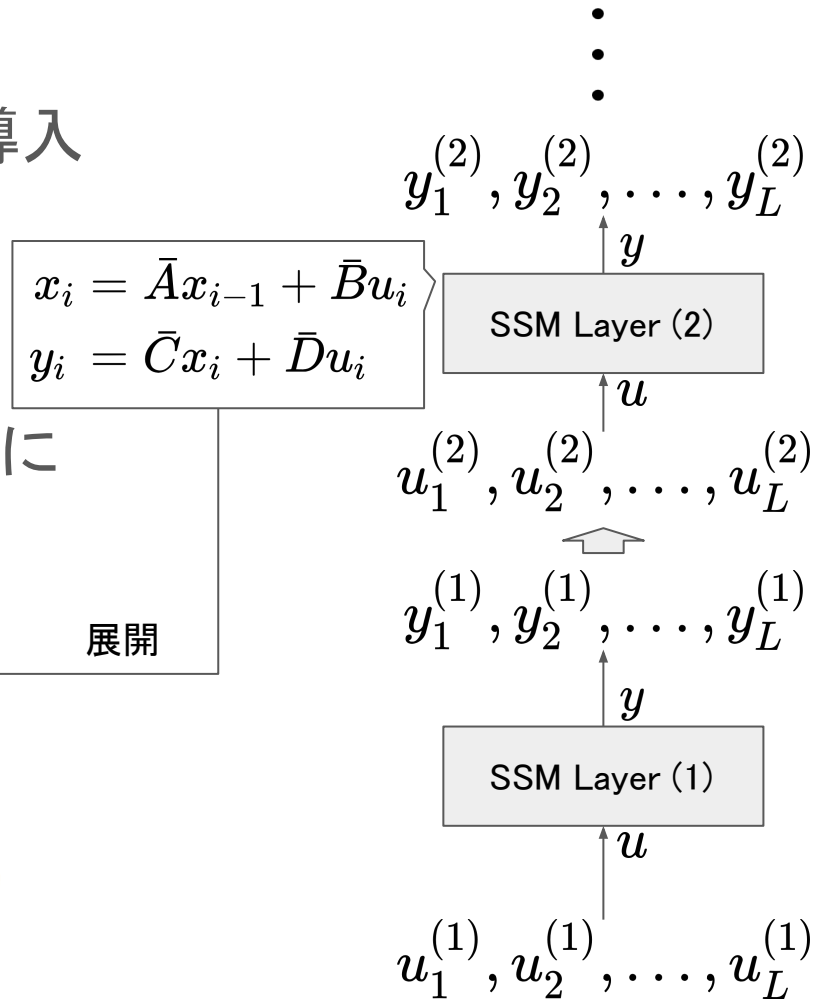
$$y_k = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}}u_k$$

$$y = \overline{\boldsymbol{K}} * u.$$

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L := \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}) := \left(\overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^i\overline{\boldsymbol{B}}\right)_{i\in[L]} = (\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}).$$

⇒ 推論はこれで並列に計算できてうれしい

⇒ 訓練では $\bar{A}^k$ を毎イテレーション計算しなきゃならなくて非効率



S4 [Gu+22]：計算の効率化とA行列のいい初期値

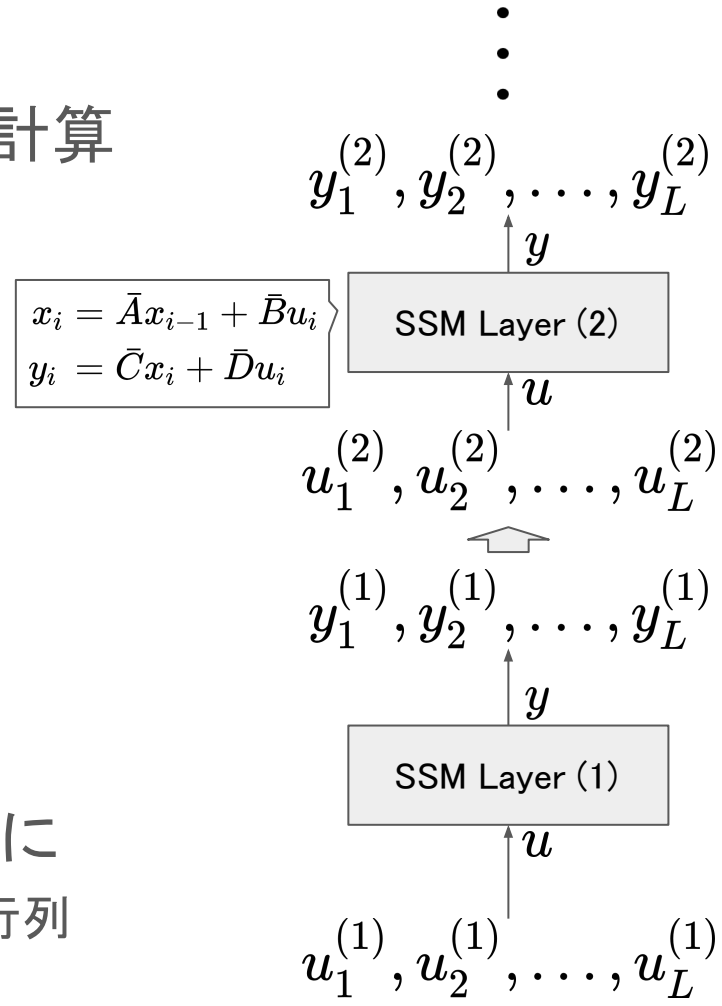
- 計算の効率化：カーネルの $\bar{A}^k$ の計算が重たいので高速に計算

$$y_k = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}}u_k$$
$$y = \overline{\boldsymbol{K}} * u.$$

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L := \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}) := \left(\overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^i\overline{\boldsymbol{B}}\right)_{i\in[L]} = (\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}).$$

  - 方法の概要：
    - 扱うA行列のクラスを扱いやすいクラスに限定：$\Lambda - PQ^*$
      - 対角行列 $\Lambda \in \mathbb{R}^{N\times N}$ – ベクトル（ $P, Q \in \mathbb{R}^{N\times 1}$ ）の積
    - カーネルじゃなくてカーネルの母関数を計算
      - 数学のテクニックがいっぱい使えて高速に計算できる
- 初期値：よく設計されたHiPPO [Gu+20] 行列をA行列の初期値に
  - HiPPO：学習しないで使うSSMで記憶を保持できるように設計されたA行列

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$



論文の位置づけ

状態空間モデル
[Brogan+74]

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

深層学習に導入

設計された
A行列

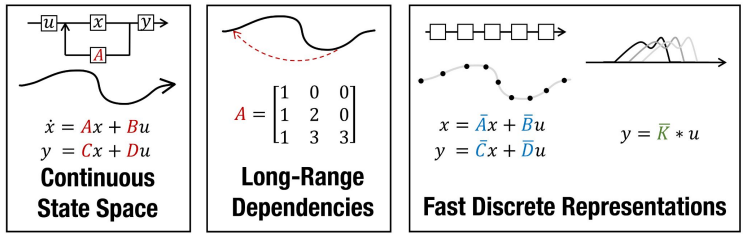
初期化に利用

HiPPO [Gu+20]

$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

系列を扱える深層学習モデル

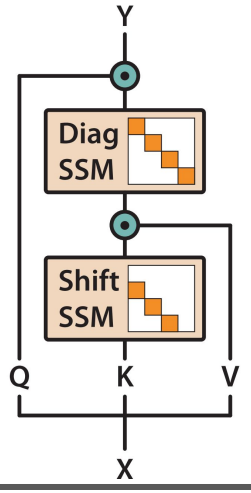
S4 [Gu+22]



言語に特化

計算速度: S4 >Att
計算時間: S4 ≦Att
CV: S4 ≒ Att
音声: S4 ≒ Att
時系列: S4 ≒ Att
NLP: S4 < Att

H3 [Fu+23]



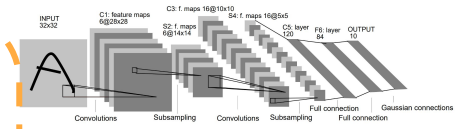
インスピレーション

variant

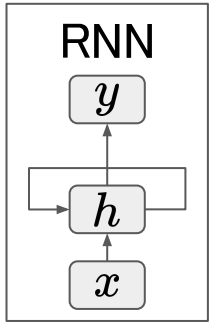
Linear Attention
[Katharopoulos+20]

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

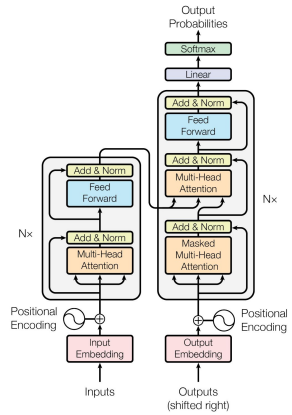
CNN [LeCunn+18]



RNN



Attention
[Vaswani+19]



事前調査：人工言語を利用したSSMの性能調査

二種類の人工的なタスクでSSMができないことを見つける

- タスク：

Task	Input	Output	Sequence Length	Vocab Size
Induction Head	$a\ b\ c\ d\ e \vdash f\ g\ h\ i \ldots x\ y\ z \vdash$	f	30	20
Associative Recall	$a\ 2\ c\ 4\ b\ 3\ d\ 1\ a$	2	20	10

[Olson+22]・[Ba+16]
を参考に作成

- 評価：2層のモデルでタスクを解いた時の正解率で評価
  - Attentionでできることが既存のSSMではできない

Task	Random	S4D	Gated State Spaces	Attention
Induction Head	5.0	35.6	6.8	**100.0**
Associative Recall	25.0	86.0	78.0	**100.0**

- この調査でわかったこと：SSMで何ができないか
  - トークン同士を比較できない　　∵ トークン同士を比べる計算をしていない
  - ※　AttentionではAttention weightの計算でトークンの比較が可能

提案手法：H3 layer

二つのSSMで入力トークン同士の比較を実現

- $\mathrm{SSM}_{\mathrm{diag}}$・$\mathrm{SSM}_{\mathrm{shift}}$　：A行列をそれぞれ対角行列とシフト行列としたSSM

※　S4で扱う行列の範囲にシフト行列は含まれない

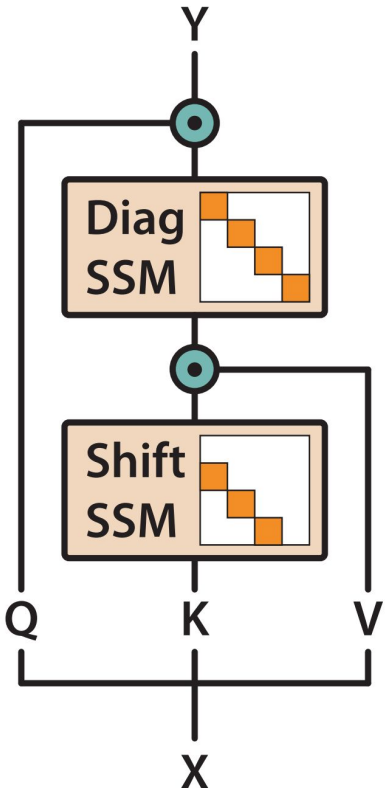
- $\mathrm{SSM}_{\mathrm{shift}}(\mathbf{K}) \odot \mathbf{V}$　：シフトした系列と元の系列の要素積
  - 過去の特徴を集約した表現が作れる：直感的には以下のような項が作れる

$$\left[0, x_1^{\mathrm{key}}, \ldots, \sum_{k=1}^{l} x_k^{\mathrm{key}}, \ldots, \sum_{k=1}^{L} x_L^{\mathrm{key}}\right] \odot \left[x_1^{\mathrm{val}}, x_2^{\mathrm{val}}, \ldots, x_l^{\mathrm{val}} \ldots, x_L^{\mathrm{val}}\right]$$

⇒ Attention weightのような気持ちで過去のトークンと比較

- $\mathbf{Q} \odot \mathrm{SSM}_{\mathrm{diag}}(\cdot)$　：過去をまとめた表現を現在の表現と比較

⇒ トークンの比較を実現



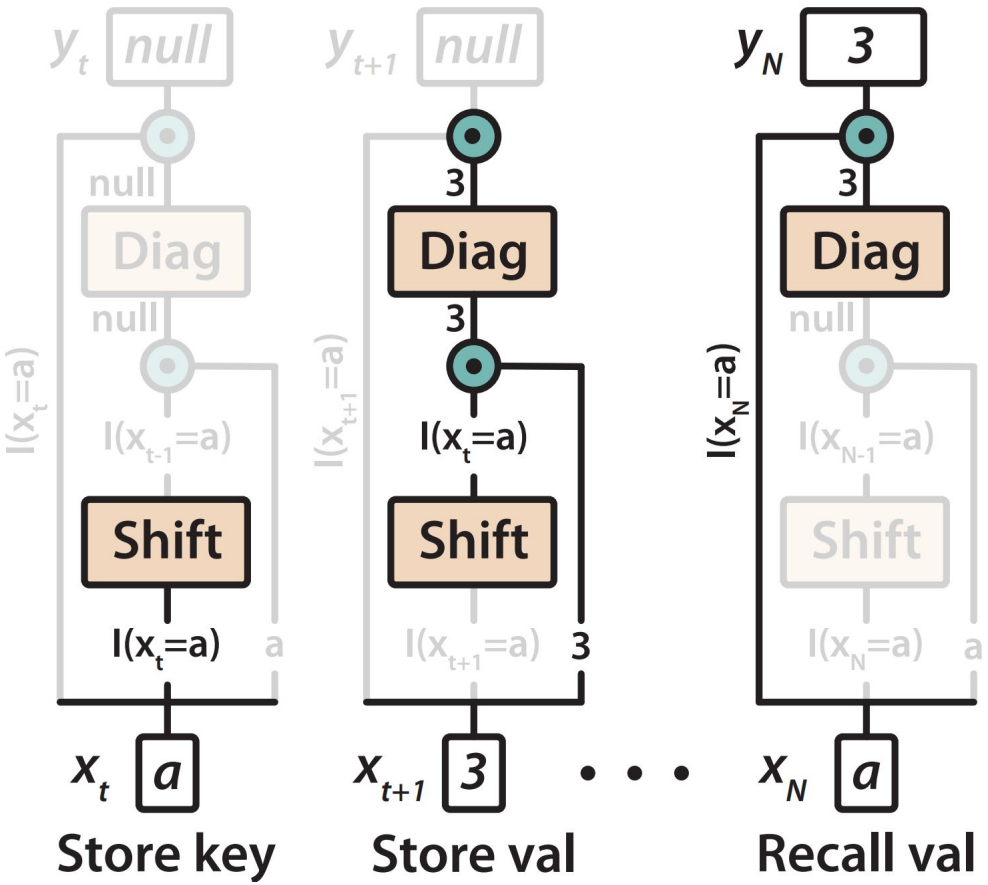
$$\mathbf{Q} \odot \mathrm{SSM}_{\mathrm{diag}}(\mathrm{SSM}_{\mathrm{shift}}(\mathbf{K}) \odot \mathbf{V})$$

提案手法：H3 layerの動作例

- Store key：シフトした系列を作成
- Store val：シフトされた系列と元の系列を比較

  ⇒ a と 3 の積が計算できて，ペアとして扱える

- Recall val：入力中からaに対応するトークンを参照する



$$\ldots a 3 \ldots a \underline{?}$$

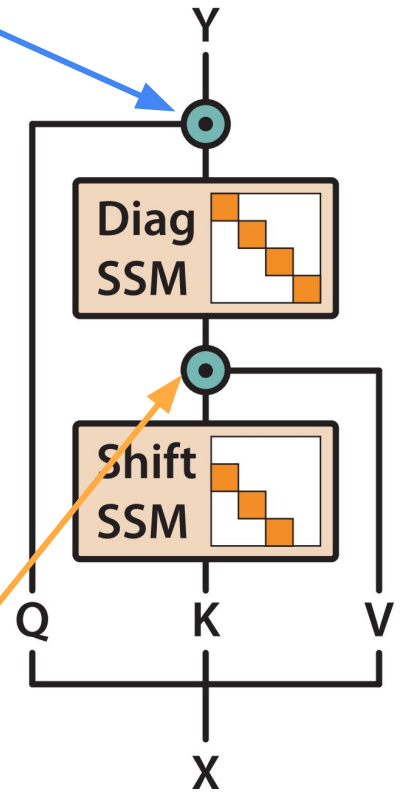
提案手法：正確なH3 layer

Algorithm 1 H3 Layer

Require: Input sequence $u \in \mathbb{R}^{N \times d}$ from the previous layer, weight matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O \in \mathbb{R}^{d \times d}$, a shift SSM $\mathrm{SSM}_{\mathrm{shift}}$, a diagonal SSM $\mathrm{SSM}_{\mathrm{diag}}$, head dimension d_h.

1: Compute $\mathbf{Q} = u\mathbf{W}_Q, \mathbf{K} = u\mathbf{W}_K, \mathbf{V} = u\mathbf{W}_V \in \mathbb{R}^{N \times d}$.
2: Pass $\mathbf{K}$ through the shift SSM: $\overline{\mathbf{K}} = \mathrm{SSM}_{\mathrm{shift}}(\mathbf{K}) \in \mathbb{R}^{N \times d}$.
3: Split $\mathbf{Q}, \overline{\mathbf{K}}, \mathbf{V}$ into H "heads" ($\mathbf{Q}^{(h)}, \overline{\mathbf{K}}^{(h)}, \mathbf{V}^{(h)}$ for $h = 1, \ldots, H$), each a sequence of N vectors of size $d_h = d/H$.
4: **for** $1 \le h \le H$ **do**
5: Take the batched outer product $\overline{\mathbf{K}}^{(h)} (\mathbf{V}^{(h)})^\top \in \mathbb{R}^{N \times d_h \times d_h}$ (batched in the N-dimension) and pass it through a diagonal SSM: $\mathbf{K}\mathbf{V}^{(h)} = \mathrm{SSM}_{\mathrm{diag}}(\overline{\mathbf{K}}^{(h)} (\mathbf{V}^{(h)})^\top) \in \mathbb{R}^{N \times d_h \times d_h}$.
6: Batch-multiply by $\mathbf{Q}$: $\mathbf{O}^{(h)} = [\mathbf{Q}_1^{(h)} \mathbf{K}\mathbf{V}_1^{(h)}, \ldots, \mathbf{Q}_N^{(h)} \mathbf{K}\mathbf{V}_N^{(h)}] \in \mathbb{R}^{N \times d_h}$ (batched in the N-dimension).
7: **end for**
8: Concatenate the output $\mathbf{O}^{(h)}$ of each head, and multiply by the output projection matrix $\mathbf{W}_O \in \mathbb{R}^{d \times d}$.

行列積



外積

$$\mathbf{Q} \odot \mathrm{SSM}_{\mathrm{diag}}(\mathrm{SSM}_{\mathrm{shift}}(\mathbf{K}) \odot \mathbf{V})$$

論文の位置づけ

状態空間モデル
[Brogan+74]

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

深層学習に導入

設計された
A行列

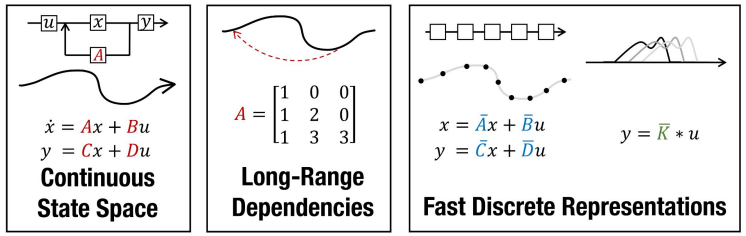
初期化に利用

HiPPO [Gu+20]

$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

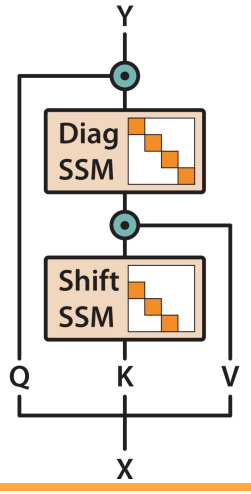
系列を扱える深層学習モデル

S4 [Gu+22]



言語に特化

H3 [Fu+23]



計算速度: S4 >Att
計算時間: S4 ≦Att
CV: S4 ≒ Att
音声: S4 ≒ Att
時系列: S4 ≒ Att
NLP: S4 < Att

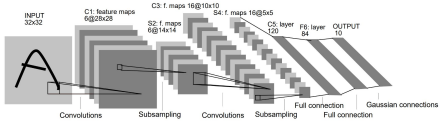
インスピレーション

variant

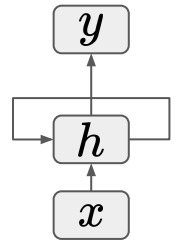
Linear Attention
[Katharopoulos+20]

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

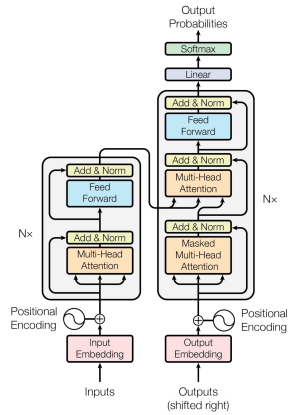
CNN [LeCunn+18]



RNN



Attention
[Vaswani+19]



提案手法：H3 layerとAttentionの比較

- Attentionの定式化：
  - 一般的なAttention [Vaswani+19] ： $\mathrm{Sim}(q,k) = e^{q^\top k}$
  - Linear Attention： $\mathrm{Sim}(q,k) = \phi(q)^\top \phi(k)$
  - Linear AttentionのRNN表記：
    $z_i = \sum_{j=1}^{i} \phi(K_j) \in \mathbb{R}^d \quad S_i = \sum_{j=1}^{i} \phi(K_j) V_j^\top \in \mathbb{R}^{d \times d}$

$$O_i = \frac{\sum_{j=1}^{i} \mathrm{Sim}(Q_i, K_j) V_j}{\sum_{j=1}^{i} \mathrm{Sim}(Q_i, K_j)} \in \mathbb{R}^d$$

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

$$O_i = \frac{\phi(Q_i)^\top S_i}{d_i}$$

- 線形時変システム：パラメタを時間に合わせて用意するSSM→

$$x_i = \mathbf{A}_i x_{i-1} + \mathbf{B}_i u_i,$$
$$y_i = \mathbf{C}_i x_i + \mathbf{D}_i u_i.$$

  - Linear Attention： $d_i = 1$ とすると $S_{i+1} = S_i + \phi(K_{i+1}) V_{i+1}^T \quad O_{i+1} = \phi(Q_{i+1})^T S_{i+1}$
    ⇒ $\mathbf{A} = I \quad \mathbf{B} = I, u_i = \phi(K_i) V_i^T, C_i = \phi(Q_i)^T$ としたときのSSM

$$x_{i+1} = \mathbf{A} x_i + \mathbf{B} \phi(K_i) V_i^T$$
$$y_{i+1} = \mathbf{C} x_i,$$

  - H3 layer： $\phi(K_i) = \mathrm{SSM}_{\mathrm{shift}}(K_i)$ としたときのLinear Attention

⇒ Linear Attention と似たような特徴を持てる！

「Hungry」HiPPO の意味

HiPPO：記憶を保持できるように設計されたA行列

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

H3で使用している行列：

- 対角行列
- シフト行列

} HiPPOの一部 → hungry HiPPO × 2 →Hungry Hungry HiPPO → H3

ただし… 著者本人はブレインストーミングで出てきた遊びみたいな名前だって言っている

- https://www.youtube.com/watch?v=x_Z9fzYCIB0&t=1191s

H3の評価：言語モデル

Transformer・既存のSSMと比較して評価する

- 人工言語を利用したタスク（再掲）

Task	Random	S4D	Gated State Spaces	H3	Attention
Induction Head	5.0	35.6	6.8	**100.0**	**100.0**
Associative Recall	25.0	86.0	78.0	99.8	**100.0**

⇒ H3は入力のコピー＆トークンの比較が可能

- 自然言語（OpenWebText）の言語モデリング
  - GPT-2 small [Radford+19] と同等のパラメタ（12層）で評価
  - Hybrid：Transformer層を2・8層目（2・2+L/2層目）に挿入したモデル

H3	H3 Hybrid (2 Attn)	S4D	GSS	GSS Hybrid (2 Attn)	Transformer
21.0	**19.6**	24.9	24.0	19.8	20.6

⇒ H3でTransformerと同等・Hybridにするように利用すると性能向上

H3の評価：事前学習モデル

言語モデルで事前学習して評価

- 設定：
  - 同一サイズのHybridモデルのH3を用意してPileコーパス [Gao+20] で事前学習
  - 他のコーパスについてはZero-shotで評価のみ行った場合のスコア
  - ベースラインは事前学習済みの
    チェックポイントを使用
- 結果：
  - GPT-Neoを超える性能が得られた

  ⇒ SSMの事前学習がうまくできている

  - Zero-shotの評価についても優れている

Zero-shot

Model	Pile	OpenWebText	WikiText103
GPT-2 small (125M)	19.0*	22.6	29.9
GPT-Neo-125M	9.4	22.6	26.3
Hybrid H3-125M	**8.8**	**20.9**	**23.7**
GPT-2 medium (355M)	13.9*	17.0	21.8
Hybrid H3-355M	**7.1**	**15.9**	**16.9**
GPT-2 XL (1.5B)	12.4*	12.9	17.0
GPT-Neo-1.3B	6.2	13.1	13.3
Hybrid H3-1.3B	**6.0**	**12.4**	**12.5**
GPT-Neo-2.7B	5.7	11.7	11.5
Hybrid H3-2.7B	**5.4**	**11.0**	**10.6**

* GPT-2はPileを事前学習に使っていない

H3の評価：SuperGLUE [Wang+19]

- 設定：
  - 選択肢のうち尤度が高いものを選択して出力とする
  - Few-shotはGPT-3 [Brown+20]の設定同様，プロンプトで与える
- 結果：
  - Zero/Few-shot の場合で共に過半数のタスクで既存モデルを上回った

Zero-shot

Model	WSC	WIC	RTE	CB	MultiRC	ReCoRD	BoolQ	COPA	Average
OPT-125M	**39.4**	52.0	48.7	37.4	58.9	44.9	59.6	60.0	50.1
GPT-Neo-125M	36.5	**53.6**	53.1	41.1	**59.9**	39.6	**62.2**	60.0	50.8
Hybrid H3-125M	**39.4**	51.4	**59.2**	**48.2**	51.4	**55.0**	59.6	**67.0**	**53.9**
GPT-2 medium (355M)	50.0	**52.0**	51.3	28.6	**59.5**	53.3	61.0	65.0	52.6
OPT-350M	**53.5**	50.8	53.4	35.7	58.9	51.4	60.9	60.0	53.1
Hybrid H3-355M	37.5	51.7	**55.2**	**41.1**	**59.5**	**62.3**	**61.5**	**69.0**	**54.7**
OPT-1.3B	36.5	49.5	**53.4**	**39.3**	**58.3**	61.8	55.0	69.0	52.9
GPT-Neo-1.3B	41.3	50.0	52.3	33.9	57.9	55.5	59.9	66.0	52.1
Hybrid H3-1.3B	**52.9**	**50.3**	**53.4**	33.9	58.2	**67.8**	**61.7**	**74.0**	**56.5**
OPT-2.7B	**51.0**	50.8	50.5	41.1	57.4	65.9	60.9	66.0	55.5
GPT-Neo-2.7B	37.5	50.0	52.3	**50.0**	**59.1**	60.0	**61.1**	67.0	54.6
Hybrid H3-2.7B	36.5	**51.3**	**57.0**	37.5	58.7	**71.3**	**61.1**	**81.0**	**56.8**

3-shot

Model	WSC	WIC	RTE	CB	MultiRC	ReCoRD	BoolQ	COPA	Average
OPT-125M	36.5	**50.2**	47.3	44.6	**57.9**	44.9	41.9	60.0	47.9
GPT-Neo-125M	38.5	50.0	53.1	17.9	56.3	39.6	**62.1**	60.0	47.2
Hybrid H3-125M	**43.3**	49.1	**58.1**	**51.8**	48.9	**55.0**	56.1	**67.0**	**53.7**
GPT-2 medium (355M)	36.5	**50.5**	48.0	8.9	43.5	53.3	58.8	65.0	45.6
OPT-350M	37.5	50.0	45.8	**44.6**	49.8	51.4	**61.7**	60.0	50.1
Hybrid H3-355M	**42.3**	47.5	**50.5**	28.6	**59.7**	**62.3**	60.5	**69.0**	**52.6**
OPT-1.3B	**44.2**	**51.1**	53.4	16.1	**59.9**	62.1	38.3	70.0	49.4
GPT-Neo-1.3B	35.6	50.6	47.3	**32.1**	**59.9**	55.7	**61.2**	67.0	51.2
Hybrid H3-1.3B	36.5	49.2	**55.2**	23.2	59.3	**67.6**	56.9	**76.0**	**53.0**
OPT-2.7B	44.2	50.5	**53.4**	17.9	59.2	66.0	**62.0**	71.0	53.0
GPT-Neo-2.7B	**49.0**	**51.9**	51.6	21.4	57.0	60.0	56.0	68.0	51.9
Hybrid H3-2.7B	36.5	45.6	47.3	**46.4**	**59.4**	**71.1**	60.6	**77.0**	**55.5**

H3の評価：スループット

単位時間あたりに出力可能なトークン数（スループット）を計測

- 目的：再帰的なH3モデルとAuto RegresiveなTransformerと比べてどれだけ早く生成が可能かを確認
- 結果：スループットは約2.4倍 ⇒ より高速に言語モデルが可能

Table 7: Inference throughput on A100 80GB, 1.3B models. Batch size 64, prompt length 512, 1024, or 1536, and generating 128 tokens per sequence in the batch (i.e., 64 × 128 tokens in a batch). Hybrid H3 is up to 2.4× faster than a Transformer of similar size in inference. The difference is larger for longer sequences.

Tokens/s	Prompt length 512	Prompt length 1024	Prompt length 1536
Transformer-1.3B	1340	770	520
Hybrid H3-1.3B	1980	1580	1240

論文の位置づけ

状態空間モデル
[Brogan+74]

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

深層学習に導入

設計された
A行列

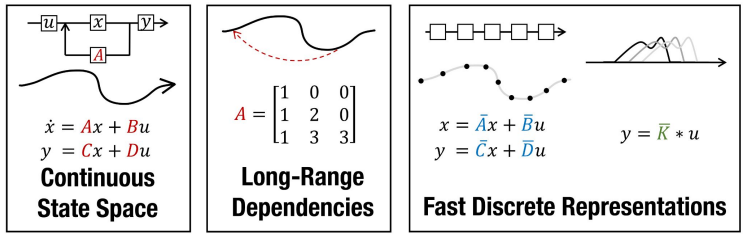
初期化に利用

HiPPO [Gu+20]

$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

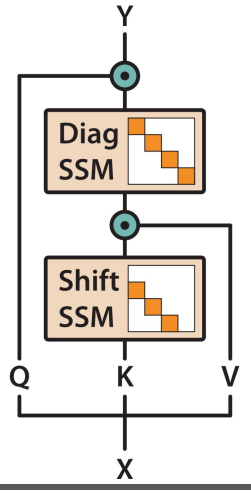
系列を扱える深層学習モデル

S4 [Gu+22]



言語に特化

H3 [Fu+23]



計算速度： S4 >Att
計算時間： S4 ≦Att
CV： S4 ≒ Att
音声： S4 ≒ Att
時系列： S4 ≒ Att
NLP： S4 < Att

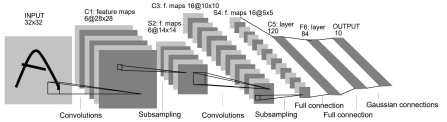
インスピレーション

variant

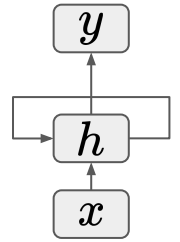
Linear Attention
[Katharopoulos+20]

$$O_i = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}$$

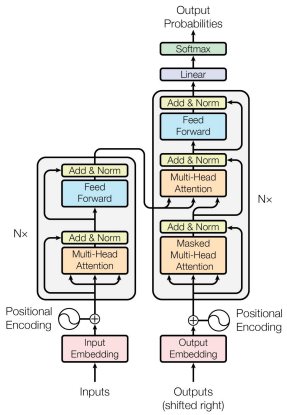
CNN [LeCunn+18]



RNN



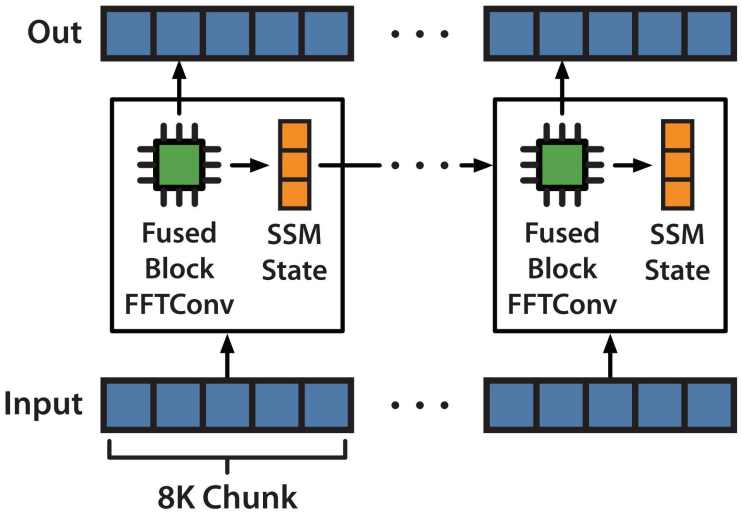
Attention
[Vaswani+19]



FlashConv：SSMの計算の高速化

SSMの計算をGPUに寄り添った計算になるように工夫 → SSM全体に利用可

- Fused Block FFTConv：2つの方法で計算を高速化
  - Kernel Fusion：計算順を工夫してメモリへのread/writeを減らす
    - AttentionでいうFlashAttention [Dao+22]
  - Block FFT：計算途中に出現するブロック対角行列をTensorコアで計算
- State Passing：系列がメモリに乗らない場合に分けて計算
  - RNNのような再帰的なモデルなので，計算済みの状態を渡せば任意のサイズのチャンクに分割可能
    ⇒ メモリに載るようにチャンクに分割すれば長い系列も扱える



FlashConvの評価

計算速度を既存のS4で比較

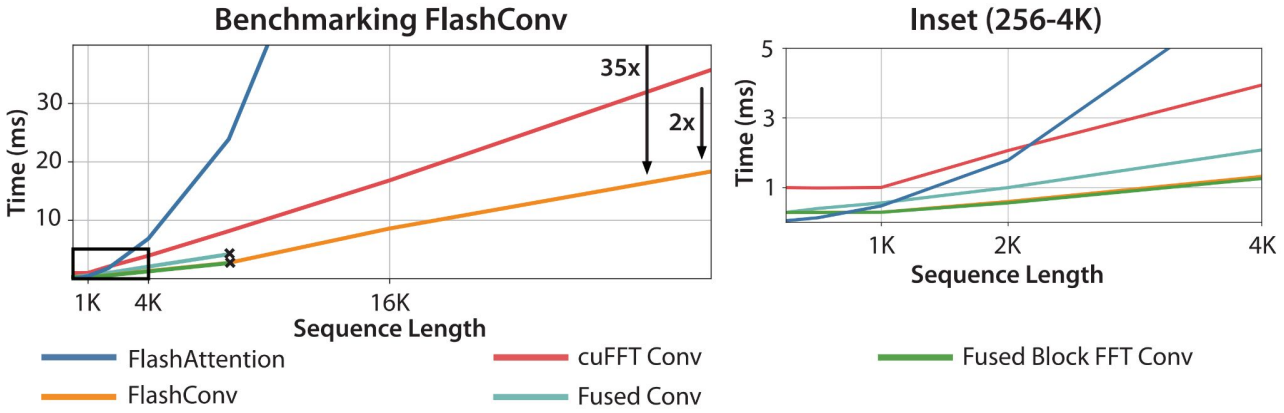
- Long-Range Arena ベンチマークでの速度評価
  - 既存のSoTAのS4を2倍の速度で動作させられるようになった
- Forward+Backwardの実行時間
  - 系列長に対してほぼ線形にしか実行時間は増加しない
  - CUDA比で2倍，Attention比で35倍の速さで実行可能

LRAの実行速度比較

Models	Speedup
Transformer	1×
FlashAttention (Dao et al., 2022b)	2.4×
Block-sparse FlashAttention (Dao et al., 2022b)	2.8×
S4 (Gu et al., 2022c)	2.9×
S4 with FLASHCONV	5.8×

LRAの性能評価

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
S4D (Gu et al., 2022b)	**58.3**	87.3	90.7	**87.5**	**93.6**	**92.3**	**85.0**
H3	57.5	**88.2**	**91.0**	87.3	93.0	91.8	84.8

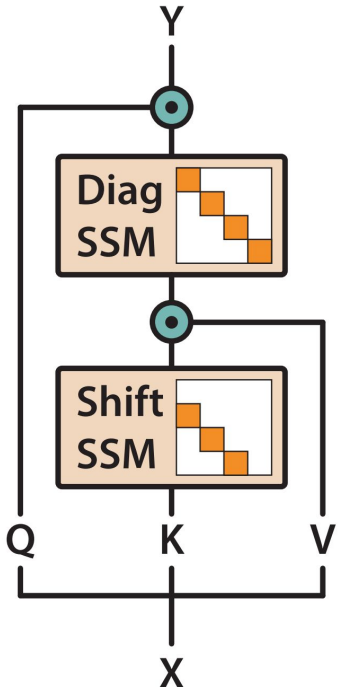


まとめ

- SSMがTransformerに言語面だけ劣っている ⇒ 原因の調査とその改善をした
  - 調査方法：人工言語を使ったタスクを解いて不足した能力を割り出す
  - 調査結果：入力をコピーして出力する機構と入力トークン同士を比較する能力が不十分
  - 改善方法：二種類の行列を使ったSSMでLinear Attentionを真似る
  - 評価結果：人工言語を使ったタスクだけでなく自然言語処理タスクにおいても Transformerと同等以上の性能を示した
- GPUに寄り添った計算方法でSSMの計算コストを下げた

今後の課題

- 1.3Bモデルまではパラメタをうまく使えた ⇒ 更に大きく
- TransformerとH3の組み合わせがよかった ⇒ いい組合せを探す



$$\mathbf{Q} \odot \mathrm{SSM}_{\mathrm{diag}}(\mathrm{SSM}_{\mathrm{shift}}(\mathbf{K}) \odot \mathbf{V})$$

参考文献

[Brogan+74] Brogan. 1974. Modern control theory.
[Fu+23] Fu et al. 2023. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In Proc. of ICLR.
[Gu+22] Gu et al. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In Proc. of ICLR.
[Gu+20] Gu et al. 2020. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In Proc. of NeurIPS.
[Vaswani+19] Vaswani et al. 2019. Attention is All you Need. In Proc. of NeurIPS.
[Katharopoulos+20] Katharopoulos et al. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In Proc. of ICML.
[Gu+21] Gu et al. 2021. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers. In Proc. of ICLR.
[Olson+22] Olson et al. 2022. In-context Learning and Induction Heads. Transformer Circuits Thread.
[Ba+16] Ba et al. 2016. Using Fast Weights to Attend to the Recent Past. In Proc. of NIPS.
[Radford+19] Radford et al. 2019. Language Models are Unsupervised Multitask Learners. In OpenAI blog.
[Gao+20] Gao et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. In ArXiv.
[Wang+19] Wang et al. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In Proc. of NeurIPS.
[Brown+20] Brown et al. 2020. Language Models are Few-Shot Learners. In Proc. of NeurIPS.
[Dao+22] Dao et al. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In Proc. of NeurIPS.

付録

付録：SSMの畳み込みカーネルの計算効率化

行列のクラスを計算しやすいものに限定して高速な計算を計算を可能にする

1. A を正規行列+低rank行列 (Normal Plus Low Rank; NPLR)　　$\Lambda - PQ^*$ で表現
   - Λ：対角行列　　$P, Q \in \mathbb{C}^{N\times 1}$ ベクトル　　$\left(PQ^* \in \mathbb{C}^{N\times N}\right)$
   - 行列のクラスを表現力が十分かつ計算しやすい形に限定する
2. カーネルの母関数をべき級数（離散フーリエ変換, DFT）　　$\sum_{j=0}^{L-1} \overline{\boldsymbol{K}}_j \zeta^j$ で計算
   - Woodbury identity（Sherman–Morrison–Woodburyの公式）によって累乗（L回の行列積）が逆行列を一つ計算すればよくなる
3. 逆フーリエ変換（iFFT）して計算したいカーネルを獲得
   - カーネルを対角行列に限定した場合，高速に計算が可能な問題に帰着（Cauthy Kernel）

付録：SSMの計算の高速化—Fused Block FFTConv

- Kernel Fusion：計算順を工夫してメモリへのread/writeを減らす
  - FFT畳み込み（FFTConv）の通常の実装のボトルネックは畳み込みカーネルのread/write
    ⇒ 畳み込みカーネルを計算してしまって，メモリに載せてしまう
    ＝ AttentionでいうFlashAttention [Dao+22]
- Block FFT：計算途中に出現するブロック対角行列をTensorコアで計算
  - ブロック対角行列をブロックで分けて，小さい行列で計算するようにして，計算量を削減
  - Tensorコアには16×16行列の積を高速に計算可能 ⇒ 小さい行列の積を計算するときに特