Mamba ブロックが帰納ヘッドタスクを実行するメカニズム

山本 悠士[1,a)]　松崎 拓也[1,b)]

概要：現在，Transformer は自然言語処理分野において最も人気のアーキテクチャであるが，その主要モジュールである Self-Attention の計算量は系列長に対して二次関数的に増加するため，長系列の推論に大きな計算コストを要する．一方，状態空間モデルに基づく言語モデルである Mamba は，計算量を線形に抑えつつ，Trasnformer に匹敵する性能を発揮したと報告されている．このため，長系列を扱うタスクにおいて，Transformer に代わる新しい選択肢として注目されている．
本研究では，Mamba の主要モジュールである Selective SSM が帰納ヘッドタスクを実行する際の内部状態を分析した．ここで，帰納ヘッドタスクとは，トークンを生成する際に入力系列中の似た文脈から情報を取得する，いわゆる Few-shot のような状況を抽象化した人工タスクである．分析の結果，Selective SSM は入力系列中の bi-gram の表現を内部状態に記憶していて，同じ bi-gram の前半が再び出現した際に，記憶された bi-gram の後半の表現を出力していることが分かった．

キーワード：状態空間モデル, Mamba, 文脈内学習, 深層学習モデルの解釈性向上

The Mechanism by which the Mamba Block Performs the Inductive Head Task

Abstract: Transformer is currently the most popular architecture in the field of natural language processing, but the space complexity of its main module, i.e. Self-Attention, increases quadratically with the length of the sequence, making the inference for long sequences computationally expensive. On the other hand, Mamba, a state-space model-based language model, achieved a performance comparable to Transformer while keeping the computational complexity linear. Therefore, Mamba is expected to be an alternative to Transformer for long-sequence tasks.
In this study, we analyzed the mechanism by which Selective SSM, the main module of Mamba, performs induction head tasks. The induction head task is a synthetic task that abstracts the so-called few-shot scenario, in which information is retrieved from similar contexts in the input sequence when generating tokens. We found that the Selective SSM stores the representations of the bi-grams in the input sequence in its hidden state and outputs the stored representation of the second half of a bi-gram when the first half of the same bi-gram appears again.

Keywords: State space model (SSM), Mamba, In-Context Learning, Mechanistic Interpretability

1. はじめに

現在，Transformer [18] は自然言語処理において最も人気のアーキテクチャだが，その主要な構成要素である Self-Attention の訓練時の計算量は系列長 L に対して $O(L^2)$ であるため，長距離の文脈を捉えるモデルの実現を困難にしている．さらに，Self-Attention は 1 トークン生成する度に $O(L)$ の計算量を要するため高速な推論を妨げる．このような課題を解決するために，Sparse Attention [2], [4] や Linear Attention [9], [12], [19], [21], [22] といったモデルの変更や，ハードウェア最適化による効率化 [5] が提案されている．これらの手法は [17], [20] にまとめられている．

近年，Attention を用いずに時系列を処理する手法として状態空間モデル (State Space Model, SSM) が注目されている．状態空間モデルを用いた深層学習モデルである

[1] 東京理科大学
Tokyo University of Science
[a)] 1423531@ed.tus.ac.jp
[b)] matuzaki@rs.tus.ac.jp

S4 [8] は，推論の効率や長距離依存性能を測る Long Range Arena ベンチマーク [16] に対し初めて完答し，状態空間モデルが長系列の処理能力の獲得に有用であることを示した．しかし，言語タスクにおける性能は依然として S4 よりも Transformer の方が優位だった．Fu ら [6] は，S4 が Attention よりも過去の文脈を参照する能力が低いことに着目して，それを解決する言語モデルである H3 を提案した．H3 は Attention と併用することで Transformer に匹敵する言語推論性能を示した．また，Gu ら [7] は，状態空間モデルのパラメータを入力に依存させる Selective SSM を導入したモデルである Mamba を提案した．Mamba は，言語タスクに対する Zero shot 推論や Perplexity による評価において，Attention と併用しない場合でも Transformer に匹敵するスコアを示した．

言語モデルに長系列を処理させたい背景に文脈内学習（In-Context Learning, ICL）がある．ICL とは，Few Shots や指示といった有用な文脈がプロンプトとして与えられた状況の方が，より適当な出力が生成される現象を指す．従来は，言語モデルにタスクを実行させるためには，Fine Tuning などによりタスクを学習させる必要があったが，ICL の能力を獲得したモデルは，与えられた出力例や指示に従うことで未知のタスクに対応することができる [3]．

このような背景から，ICL の能力の獲得が重要視され，どのようなメカニズムでこの能力が獲得されるのかを探る研究が行われている．Olsson ら [11] は 2 層の Attention 層からなるモデルを分析し，帰納ヘッド（Induction head）が出現したときに，ICL に対する能力も向上することを示した．ここで，帰納ヘッドとは，与えられた文脈に存在する bi-gram w_iw_j の前半トークン w_i が再度出現したとき，すなわち，$w_iw_j\cdots w_i$ のような系列が与えられたときに，その後半トークンである w_j が生成される確率を高める機構と定義される．本稿では，$w_iw_j\cdots w_i$ のような系列が与えられたときに w_j を出力できるかを検証するタスクのことを**帰納ヘッドタスク**と呼ぶ．

帰納ヘッドタスクは，2 層の Attention からなるモデルでは正解率 100% で実行することができるが，パラメータが入力に非依存の SSM では実行不能だった．Fu ら [6] は Shift SSM 層（一般化された畳み込み層）を導入したモデルである H3 を提案し，SSM に基づくモデルでも帰納ヘッドタスクを 100% で実行できることを示した．同じく畳み込み層を持ち，SSM のパラメータが入力に依存するモデルである Mamba も帰納ヘッドタスクを正解率 100% で実行できる．過去に入力されたトークンの内部状態をすべて保持する Attention とは異なり，SSM は固定サイズの隠れ状態を再帰的に更新する．RNN は勾配消失問題により長系列の学習が困難である一方で，SSM は RNN と同じく処理が再帰的であるにもかかわらず過去の情報を参照する能力に長けていることは意外な事実である．

本研究では，単純化した 1 層の Mamba が帰納ヘッドタスクをどのようにして実行しているかを明らかにする．具体的な貢献は以下の通りである．

- 畳み込み層と Selective SSM のみからなるモデルが帰納ヘッドタスクを実行するメカニズムの仮説を示す (§4.1)．
- 学習済みモデルの中間状態の可視化を通じて，モデルが実際にそのメカニズムを実現していることを示す (§4.2)．
- 帰納ヘッドタスクを実行する際に，過去のどの位置を参照しているかを可視化する方法を示す (§5)．

2. 背景: Mamba

S4 が長い系列を効率的に処理できることが示されて以降，深層学習モデルに状態空間モデル (SSM) を組み込むアプローチが増えている．入出力をそれぞれ $x, y \in \mathbb{R}$ としたとき，連続時間の SSM は以下の微分方程式として定義される:

$$\boldsymbol{h}'(t) = A(t)\boldsymbol{h}(t) + \boldsymbol{b}(t)x(t) \tag{1}$$

$$y(t) = \boldsymbol{c}(t)^\mathsf{T}\boldsymbol{h}(t). \tag{2}$$

トークン列などの離散時間データを扱う際は，次の離散化された更新式が用いられる:

$$\boldsymbol{h}_t = \overline{A}_t\boldsymbol{h}_{t-1} + \overline{\boldsymbol{b}}_t x_t \tag{3}$$

$$y_t = \boldsymbol{c}_t^\mathsf{T}\boldsymbol{h}_t. \tag{4}$$

$$(\overline{A}_t \in \mathbb{R}^{N\times N}, \overline{\boldsymbol{b}}_t \in \mathbb{R}^N, \boldsymbol{c}_t \in \mathbb{R}^N)$$

ここで，初期状態は $\boldsymbol{h}_{-1} = \boldsymbol{0}$ とする．Mamba では $\overline{A}_t$ は対角行列に制限される．したがって，$\overline{A}_t$ の対角成分を並べたベクトルを $\overline{\boldsymbol{a}}_t$ とおくと，式 (3) は次のように書くこともできる:

$$\boldsymbol{h}_t = \overline{\boldsymbol{a}}_t \odot \boldsymbol{h}_{t-1} + \overline{\boldsymbol{b}}_t x_t. \tag{5}$$

式 (5) を深層学習モデルのモジュールとして用いるためには，SSM の入出力をスカラーから D 次元ベクトルに拡張する必要がある．Mamba では，D 個の SSM を用意して，それらを $\boldsymbol{x}_t$ の各要素に並列に適用することで，入出力を多次元化している．数式で表記すると次のようになる:

$$H_t = \widetilde{A}_t \odot H_{t-1} + \overline{\boldsymbol{b}}_t\boldsymbol{x}_t^\mathsf{T} \tag{6}$$

$$\boldsymbol{y}_t^\mathsf{T} = \boldsymbol{c}_t^\mathsf{T}H_t. \tag{7}$$

ただし，$\widetilde{A}_t = \overline{\boldsymbol{a}}_t\mathbb{1}_D^\mathsf{T}$ である．ここで，$\mathbb{1}_D$ は，要素がすべて 1 の D 次元ベクトルである．**図 1** は式 (6-7) の処理の流れを図示している．本稿では，式 (6) における行列 H_t を**隠れ状態**と呼ぶ．

多次元化してみると，式 (6) は $\overline{\boldsymbol{b}}_t$ で重み付けられた入力 $\boldsymbol{x}_t$ を隠れ状態 H_t の各行に加算する処理であり，式 (7)

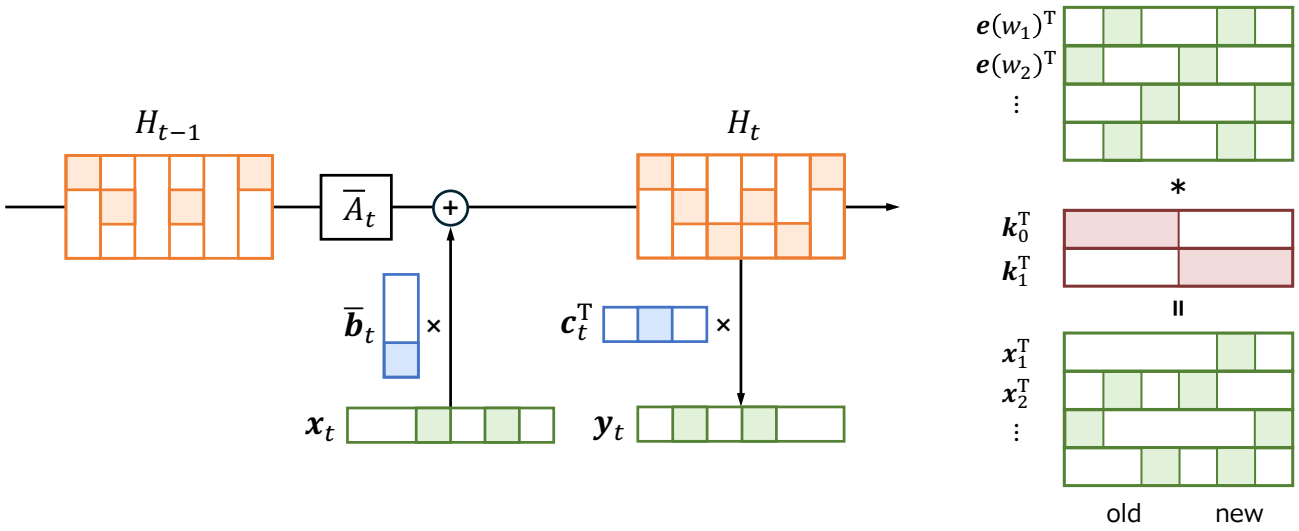


図 1: （左）Selective SSM のフロー図．状態数を $N = 3$，入力埋め込みの次元数 $D = 6$ とした．$\overline{\boldsymbol{b}}_t$ の第 i 成分が大きいとき，入力 $\boldsymbol{x}_t$ は主に隠れ状態 H_{t-1} の第 i 行に加算され，$\boldsymbol{c}_t$ の第 j 成分が大きいとき，隠れ状態 H_t の第 j 行に類似した成分が出力 $\boldsymbol{y}_t$ として得られる．（右）畳み込みのフロー図．畳み込みカーネルの絶対値が大きい成分がずれていれば，畳み込み後の埋め込みは bi-gram の表現になる．詳細は式 (15) を参照されたい．

Fig. 1 (Left) Illustration of the Selective SSM process (Eq. (6-7)). (Right) Illustration of convolution process (Eq. (15)).

は隠れ状態 H_t を $\boldsymbol{c}_t$ に射影することで出力 $\boldsymbol{y}_t$ を得ていると見ることができる．§4 では，これらの処理がそれぞれ記憶と想起に対応していることを示す．

Mamba はパラメータ $\overline{A}_t, \overline{\boldsymbol{b}}_t, \boldsymbol{c}_t$ が入力 $\boldsymbol{x}_t$ に依存する．このような SSM を *Selective SSM* と呼び，パラメータが時刻 t について不変な SSM と区別する．Selective SSM のパラメータは以下のように定義される: [*1] [*2]

$$\Delta_t = \log(1 + \exp(\boldsymbol{\alpha}_\Delta^{\mathrm{T}} \boldsymbol{x}_t + \beta_\Delta)) \tag{8}$$

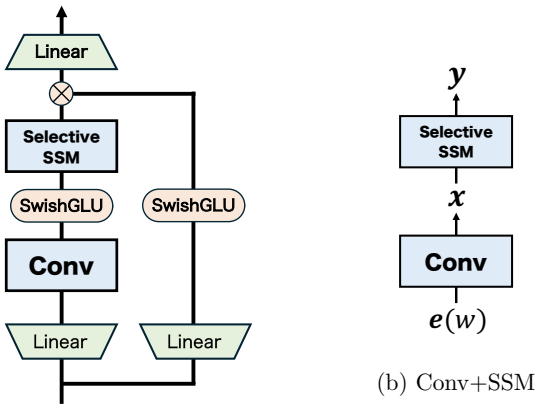
$$\overline{A}_t = \exp(\Delta_t A), \tag{9}$$

$$\overline{\boldsymbol{b}}_t = \Delta_t \boldsymbol{b}_t = \Delta_t W_{\boldsymbol{b}} \boldsymbol{x}_t, \tag{10}$$

$$\boldsymbol{c}_t = W_{\boldsymbol{c}} \boldsymbol{x}_t. \tag{11}$$

ここで，$\boldsymbol{\alpha}_\Delta, \beta_\Delta, A, W_{\boldsymbol{b}}, W_{\boldsymbol{c}}$ は学習可能なパラメータである．

この Selective SSM と 1 次元畳み込み層 (Conv)，Swish Gated Linear Unit (SwiGLU) [13] を**図 2a** のように組み合わせたものを Mamba ブロックと呼ぶ．



(a) Mamba block　(b) Conv+SSM

図 2: アーキテクチャ．

Fig. 2 Architecture.

3. Mamba による帰納ヘッドタスクの追試

本節では，Gu ら [7] より一般的な設定の下で，Mamba による帰納ヘッドタスクの実験を行う．さらに，帰納ヘッドタスクにおける Selective SSM モジュールの寄与に興味があるため，畳み込み層と Selective SSM のみからなる簡素化されたモデルの検証も行う．

3.1 帰納ヘッドタスクの定義

帰納ヘッドタスクとは，文脈内にとある bi-gram が存在して，その bi-gram の前半のトークンが出現したときに，

[*1] 本稿では，式 (8) により Δ_t をスカラーとして定義するが，$\boldsymbol{\alpha}_\Delta$ を行列に，β_Δ をベクトルに拡張することで Δ_t をベクトルとして定義することもできる．ステップ幅 Δ_t が各次元で異なる場合，式 (1-2) を離散化した結果も各次元で異なるため，表現力が強まることが期待される．

[*2] Mamba[7] の Eq. (4) では，$\overline{\boldsymbol{b}}_t = (\Delta_t A)^{-1}(\exp(\Delta_t A) - I) \cdot \Delta_t \boldsymbol{b}_t$ と定義されているが，著者実装では $\overline{\boldsymbol{b}}_t = \Delta_t \boldsymbol{b}_t$ に簡略化されている．本稿では実装に等価な後者を記述する．詳細は `https://github.com/state-spaces/mamba/issues/19` .

表 1: 帰納ヘッドタスクの入出力の例.
Table 1 Example inputs/outputs of the induction head task.

特殊トークン	入力系列	正解
あり	a b c [key] d e f ... x y z [key]	d
なし	例 1) a b c d e f ... x y z c	d
	例 2) a b c d e f ... x y z d	e

後半のトークンをモデルが出力できるかを検証するタスクである．例えば**表 1** のように，“a b c d e f ... x y z c” という系列が与えられた場合は，末尾のトークン “c” は文脈内に存在する bi-gram “c d” の前半トークンであるため，トークン “d” を出力すれば正解となる．

より具体的には，トークン v を語彙からランダムに復元抽出して，入力トークン列 $S = [v_1, v_2, \ldots, v_{T-1}, v_T]$ を作成する．ただし，v_T は $[v_1, v_2, \ldots, v_{T-1}]$ のいずれかと等しいとする．ここで，$v_s = v_T$ となる最大の s に対し，モデルの出力が v_{s+1} と一致すれば正解となる．本研究では，モデルの学習には末尾のトークンのみに対する Cross-Entropy 損失を用いた．また，入力トークン列 S の長さは 255 に固定して，バッチサイズを 8 として最大 204,800 ステップ学習した．

Gu ら [7] の定義では，入力系列の末尾（すなわち，bi-gram の前半トークン）を，入力系列中で 2 回のみ出現する特殊トークン `[key]` としていたが[*3]，本研究では特殊トークンを用いない場合についても考慮する．特殊トークン `[key]` を用いた場合，モデルは「`[key]` が出現したら次のトークンを記憶して，再び `[key]` が出現したら記憶したトークンを出力する」というメカニズムを獲得する．しかし，自然言語のテキストには，記憶と想起のタイミングを明示的にモデルに教えるような特殊トークンは存在しないため，特殊トークンを用いずに定義した方が，より言語タスクに近い状況の下でモデルの文脈内学習能力およびモデルが獲得したメカニズムを分析できると考えられる．

3.2 結果

表 2 に示した 5 通りの構成のモデルで帰納ヘッドタスクの学習と評価を行った．2 層の Mamba ブロックからなるモデルで特殊トークンなしの帰納ヘッドタスクを学習すると正解率は 94.8% となった．また，Mamba ブロックを 1 層に減らしても正解率は大きく悪化せず 92.9% だった．さらに，**図 2b** のように Mamba ブロックを単純化した場合でも正解率は 87.5% であり，予測するトークンを語彙からランダムに選んだ場合の正解率 $1/16 = 6.25\%$ とは顕著な差がある．したがって，畳み込み層 (Conv) と Selective SSM のみからなるモデルでも帰納ヘッドタスクを実行する能力を獲得できると考えられる．

[*3] 文献 [7] では，帰納ヘッドタスクが詳細に定義されていないが，その先行研究である文献 [6] に特殊トークンを用いた定義が記述されている．

表 2: 帰納ヘッドタスクにおける各モデルの設定と正解率．D は入力埋め込みの次元数．N は状態数．W は畳み込み層のカーネル幅．V は語彙サイズ．正解率の左列 w/（右列 w/o）は，特殊トークンあり（なし）としたときの正解率である．
Table 2 Configurations and accuracies for each model in the induction head task.

モデル名	D	N	W	V	正解率 w/	正解率 w/o
Mamba block ×2	64	16	4	16	100%	94.8%
Mamba block ×1	64	16	4	16	100%	92.9%
Conv+SSM	64	16	2	16	100%	90.3%
Conv only	64	-	2	-	12.7%	21.8%
SSM only	64	16	-	16	8.50%	20.8%

特殊トークンありの帰納ヘッドタスクに対して，畳み込み層 (Conv) と Selective SSM の両方を持つモデルの正解率は 100% となった．これは，Gu ら [7] による 2 層 Mamba ブロックの結果と一致している．この正解率の高さの要因は，特殊トークンがあることでモデルは記憶や想起をするタイミングを知ることができるためであると考えられる．しかし，そのような補助的な入力が与えられなくても帰納ヘッドタスクは実行可能であることを本節の結果は示唆している．

4. Selective SSM が帰納ヘッドタスクを実行するメカニズム

§3 より，Conv+SSM でも帰納ヘッドタスクを実行できる能力があることが分かった．本節では，§4.1 で Selective SSM が特定のパラメータを獲得すれば帰納ヘッドタスクを実行できることを説明し，§4.2 で実際に学習済みモデルが §4.1 で述べたメカニズムを実現していることを示す．

4.1 メカニズムの仮説

本項では，Conv+SSM が帰納ヘッドタスクを実行できるモジュールであることを説明する．ただし，埋め込みや隠れ状態，パラメータは次のような疎な表現であると仮定する．また，本稿では，標準基底（one-hot ベクトル）を $\boldsymbol{e}_i$，トークン w に対応する埋め込みを $\boldsymbol{e}(w)$ と表記する．

次元数 語彙サイズを V とし，語彙は $\{w_0, w_1, \ldots, w_{V-1}\}$ とする．また，入力埋め込みの次元数は $D = 2V$，隠れ状態の次元数は $N = V$ とする．

入力埋め込み モデルに入力されるトークン w_i の埋め込みは，one-hot ベクトル $\boldsymbol{e}_i \in \mathbb{R}^V$ を 2 つ結合したベクトル $\boldsymbol{e}(w_i)^\mathsf{T} = [\boldsymbol{e}_i^\mathsf{T}, \boldsymbol{e}_i^\mathsf{T}] \in \mathbb{R}^{2V}$ とする．

畳み込みカーネル Selective SSM の直前に適応される畳み込み層のカーネルを次のように定める:

$$\boldsymbol{k}_0 = \begin{bmatrix} \mathbb{1}_V \\ \mathbf{0}_V \end{bmatrix},\ \boldsymbol{k}_1 = \begin{bmatrix} \mathbf{0}_V \\ \mathbb{1}_V \end{bmatrix}. \tag{12}$$

Selective SSM のパラメータ 式 (10-11) で用いられるパラメータを次のように定める:

$$W_{\boldsymbol{b}} = [I_V, O_V],\ W_{\boldsymbol{c}} = [O_V, I_V],\ \Delta_t = 1. \tag{13}$$

まず，bi-gram のペアを記憶するメカニズムを説明し，続いて bi-gram の前半が出現したときに後半を想起するメカニズムを説明する．

bi-gram の記憶

$w_i w_j$ という bi-gram が入力されたときを考える．帰納ヘッドタスクを実行するためには，bi-gram の後半 w_j が入力されたときに式 (3) によって w_i に続いて w_j が出現したことを隠れ状態に記憶する必要がある．以下にこれを行うメカニズムを述べる．

(1) 入力された埋め込み列に畳み込みを適用し，SSM への入力 $\boldsymbol{x}_t$ を bi-gram $w_i w_j$ の表現にする:

$$\boldsymbol{x}_t = \boldsymbol{k}_0 \odot \boldsymbol{e}(w_i) + \boldsymbol{k}_1 \odot \boldsymbol{e}(w_j) \tag{14}$$

$$= \begin{bmatrix} \mathbb{1}_V \\ \mathbf{0} \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{e}_i \\ \boldsymbol{e}_i \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbb{1}_V \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{e}_j \\ \boldsymbol{e}_j \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_i \\ \boldsymbol{e}_j \end{bmatrix} \tag{15}$$

(2) パラメータ $W_{\boldsymbol{b}}$ を用いて bi-gram の前半の表現のみを抽出することで，$\bar{\boldsymbol{b}}_t$ を w_i の表現にする:

$$\bar{\boldsymbol{b}}_t = \Delta_t W_{\boldsymbol{b}} \boldsymbol{x}_t = [I_V, O_V] \begin{bmatrix} \boldsymbol{e}_i \\ \boldsymbol{e}_j \end{bmatrix} = \boldsymbol{e}_i\ . \tag{16}$$

(3) 式 (6) より，入力 $\boldsymbol{x}_t = [\boldsymbol{e}_i^\mathsf{T},\ \boldsymbol{e}_j^\mathsf{T}]^\mathsf{T}$ は $\bar{\boldsymbol{b}}_t = \boldsymbol{e}_i$ に重みづけられることで，隠れ状態 H_t の第 i 行に加算される．これにより，*bi-gram* $w_i w_j$ **が出現したことが隠れ状態** H_t **の第** i **行に記憶される．**

bi-gram の想起

次に，$w_i w_j \cdots w_i$ のように再び w_i が出現した文脈を考える．最後の w_i が入力されたとき，過去に w_i の直後に w_j が出現したことが式 (4) によって想起される．以下にこれを行うメカニズムを述べる．

(1) パラメータ $W_{\boldsymbol{c}}$ を用いて bi-gram の後半（すなわち，現在のトークン）の表現のみを抽出することで，$\boldsymbol{c}_t$ を w_i の表現にする:

$$\boldsymbol{c}_t = W_{\boldsymbol{c}} \boldsymbol{x}_t = [O_V, I_V] \begin{bmatrix} * \\ \boldsymbol{e}_i \end{bmatrix} = \boldsymbol{e}_i\ . \tag{17}$$

(2) 式 (7) より，隠れ状態 H_t が $\boldsymbol{c}_t = \boldsymbol{e}_i$ に射影されることで，隠れ状態 H_t の第 i 行目が出力 $\boldsymbol{y}_t$ として得られる．

もし，入力系列中の bi-gram $w_i w_j$ に対して「bi-gram の記憶」のステップ (3) が行われていた場合，隠れ状態 H_t の第 i 行の成分は，その bi-gram 表現 $[\boldsymbol{e}_i^\mathsf{T},\ \boldsymbol{e}_j^\mathsf{T}]^\mathsf{T}$ に類似している．したがって，Conv+SSM をもつモデルにトークン w_i を入力すると，系列に存在する w_i から始まる bi-gram の表現に類似した成分が出力されると考えられる．

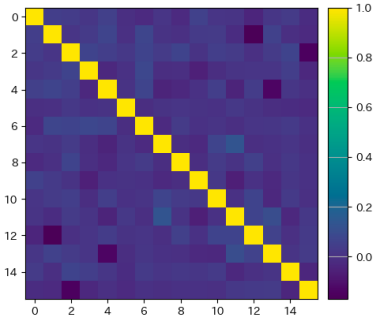


図 3: 学習後の入力埋め込み間のコサイン類似度
Fig. 3 Cosine similarity between trained input embeddings.

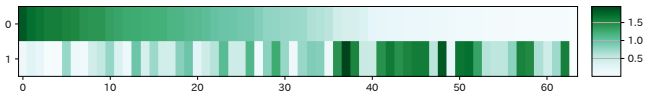


図 4: 畳み込みカーネル $[\boldsymbol{k}_0, \boldsymbol{k}_1]^\mathsf{T}$ の絶対値．$\boldsymbol{k}_0$ の絶対値を基準にソートした．
Fig. 4 Absolute values of the convolution kernels, sorted by those of $\boldsymbol{k}_0$.

4.2 学習済みモデルの分析による仮説検証

本項では，前項で示した仮説が実現されていると見なせることを確認したうえで，隠れ状態 H_t を線形変換することで記憶の様子を解釈可能にできることを示す．

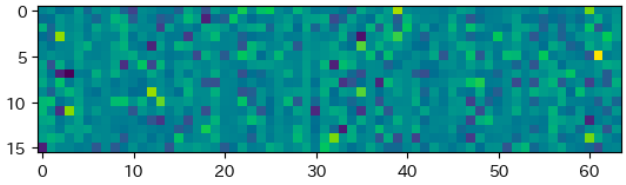
4.2.1 入力埋め込みは two-hot ベクトルとみなせる

図 3 に示した入力埋め込み間のコサイン類似度によると，入力埋め込みは互いに直交している．したがって，入力埋め込み行列を適当に線形変換（回転・拡大縮小）することで，各埋め込みを two-hot ベクトル化することができる．よって，§4.1 のようにトークン w_i の埋め込みは $\boldsymbol{e}(w_i)^\mathsf{T} = [\boldsymbol{e}_i^\mathsf{T},\ \boldsymbol{e}_i^\mathsf{T}]^\mathsf{T}$ とみなしてよい．

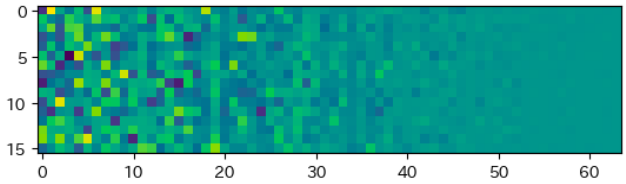
自然言語における単語とは異なり，帰納ヘッドタスクでは各トークンがそれぞれ独立に生起するため，各トークン間に類似性が生じないことは直感的な結果である．

4.2.2 畳み込み層は bi-gram の表現を作っている

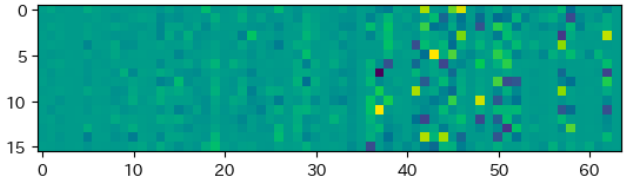
学習後の畳み込みカーネルと畳み込みカーネル適用前後の入力埋め込みをそれぞれ**図 4**, **図 5** に示す．図 4 によると，$\boldsymbol{k}_0$ の成分の絶対値が大きいときは $\boldsymbol{k}_1$ の成分の絶対値が小さい傾向がある．絶対値をとった $\boldsymbol{k}_0$ と $\boldsymbol{k}_1$ のピアソンの積率相関係数は -0.55 であり，スピアマンの順位相関係数は -0.48 だった．したがって，式 (15) のように，学習済みモデルでも畳み込み層の出力 $\boldsymbol{x}_t$ の各次元の成分は，bi-gram のうち片方のトークンの表現のみが保持される傾向が強いことが示唆される．



(a) 入力埋め込み．第 i 行は $\boldsymbol{e}(w_i)$ に対応する．



(b) 畳み込みカーネル $\boldsymbol{k}_0$ を掛けた後の入力埋め込み．第 i 行は $\boldsymbol{k}_0 \odot \boldsymbol{e}(w_i)$ に対応する．



(c) 畳み込みカーネル $\boldsymbol{k}_1$ を掛けた後の入力埋め込み．第 i 行は $\boldsymbol{k}_1 \odot \boldsymbol{e}(w_i)$ に対応する．

図 5: 畳み込みカーネル適用前後の入力埋め込み

Fig. 5 Input embedding before and after applying the convolution kernels.

4.2.3 隠れ状態に bi-gram の表現が記憶されている

本節では，状態数 N と語彙数 V が等しい場合を考える．これは表 2 に示した実験設定と同じである．式 (6) より，隠れ状態 $H_t \in \mathbb{R}^{N\times D}$ の各行の成分は過去の入力 $\boldsymbol{x}_t$ の重み付き和であるため，隠れ状態 H_t を入力トークンの埋め込みの方向に射影すれば成分の解釈性が向上することが期待される．学習済みモデルが §4.1 で述べたメカニズムを実現していれば，隠れ状態 H_t には *bi-gram* **の後半トークンの表現が含まれている**と考えられる．これを確認するために，後半トークンの表現を抽出する畳み込みカーネル $\boldsymbol{k}_1$ を適用した入力埋め込み $\boldsymbol{k}_1 \odot \boldsymbol{e}(w_j)$ に対して隠れ状態 H_t を射影する:

$$\widehat{\boldsymbol{h}}_t^{(j)} = H_t(\boldsymbol{k}_1 \odot \boldsymbol{e}(w_j)). \tag{18}$$

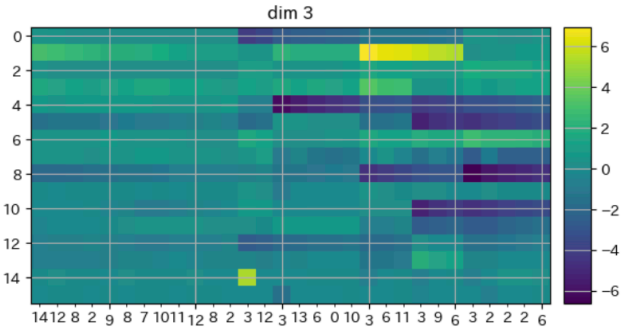
これを並べた配列 $[\widehat{\boldsymbol{h}}_t^{(3)}]_t$ を**図 6a** に示す．図より，隠れ状態をトークン w_3 に対応する埋め込みに射影すると，その成分はトークン w_3 が出現したときに不連続に変化するようになる．しかし，隠れ状態を埋め込みに射影しただけでは，時刻 t における隠れ状態にどのトークンに対応する成分が含まれているかまでは不明瞭である．

この原因は，学習されたパラメータ $W_{\boldsymbol{b}}$ は，一般に §4.1 で定めた成分 $W_{\boldsymbol{b}} = [I_V, O_V]$ とはならないからである．そのため，式 (16) のように $\overline{\boldsymbol{b}}_t = \boldsymbol{e}_i$ とはならない．すなわち，学習済みモデルにおいては，トークン w_j の成分は第 j 行だけで記憶されるのではなく，隠れ状態の複数の行にまたがって記憶される．そこで，トークン w_j に対応する成分が隠れ状態の第 j 行に集約されるように，隠れ状態 H_t の基底を取り替える:

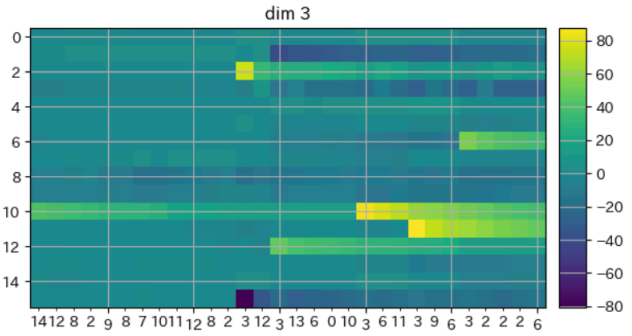
$$\boldsymbol{h}_t^{(j)} = S^{\mathsf{T}} H_t(\boldsymbol{k}_1 \odot \boldsymbol{e}(w_j)). \tag{19}$$

ただし，S は $S = W_{\boldsymbol{b}}[\boldsymbol{k}_0 \odot \boldsymbol{e}(w_0), \ldots, \boldsymbol{k}_0 \odot \boldsymbol{e}(w_{V-1})] \in \mathbb{R}^{N\times V}$ で与えられ，導出については付録 A.1 で詳説する．

式 (19) を各時刻 t について計算した結果を並べた配列 $[\boldsymbol{h}_t^{(j)}]_t$ を**図 6b** に示す．図 6b より，トークン w_3 が出現したときに，その直前に出現したトークン番号に対応する行の成分は不連続に更新され，その他の行の成分は減衰していることが分かる．この減衰の原因は，学習済みモデルに



(a) トークン w_3 の埋め込み $\boldsymbol{e}(w_3)$ に射影された隠れ状態 $[\widehat{\boldsymbol{h}}_t^{(3)}]_t$.



(b) 射影後の隠れ状態 $[\widehat{\boldsymbol{h}}_t^{(3)}]_t$ を S^{T} で線形変換した結果 $[\boldsymbol{h}_t^{(3)}]_t$.

図 6: 線形写像を施した隠れ状態．横軸ラベルは入力トークン列．(a) トークン w_3 の埋め込みに射影された隠れ状態は，トークン w_3 が出現したときに成分が不連続に変化する．(b) さらに，行列 S^{T} で線形変換すると，トークン w_3 の直前に出現したトークン番号に対応する行の成分が変化する．

Fig. 6 The hidden states after applying a linear projection. The horizontal axis labels represent the input token sequence. (a) The hidden state projected onto the embedding of token w_3 reveals that its component changes discontinuously when token w_3 appears. (b) Additionally, the linear transformation by matrix S^{T} allows the hidden state to change in the rows corresponding to the token IDs that appear just before token w_3.

より計算される行列 $\overline{A}_t$（式 (9)）の要素の絶対値がすべて 1 未満であるためである．これにより，位置が近いトークンを重要視するという xPos [15] の指数的減衰に類似した効果が生じる．以上のように，Selective SSM は帰納ヘッドタスクを実行するために，各トークンに対応する次元の成分を更新することで，与えられた文脈を記憶していると考えられる．

5. Hidden Attention Map

更新式 (3) は，入力 $\boldsymbol{x}_t$ に関して線形だから，展開することで入力の線形結合に書き直せる:

$$\boldsymbol{h}_t = \overline{A}_t \boldsymbol{h}_{t-1} + x_t \Delta_t W_{\boldsymbol{b}} \boldsymbol{x}_t = \sum_{s=1}^{t} W_s^K \boldsymbol{x}_s. \tag{20}$$

ただし，

$$W_s^K = \begin{cases} \left(\prod_{r=s+1}^{t} \overline{A_r} \right) (x_s \Delta_s W_{\boldsymbol{b}}) & (1 \le s < t) \\ x_s \Delta_s W_{\boldsymbol{b}} & (s = t) \\ 0 & (t < s \le T) \end{cases} \tag{21}$$

とした．導出は付録 A.2 を参照されたい．すると，出力 y_t は:

$$y_t = \boldsymbol{h}_t^{\mathsf{T}} (W_{\boldsymbol{c}} \boldsymbol{x}_t) = \sum_{s=1}^{t} (W_s^K \boldsymbol{x}_s)^{\mathsf{T}} (W_{\boldsymbol{c}} \boldsymbol{x}_t) \tag{22}$$

と変形できる．ここで，

$$\begin{aligned} Q &= [W_{\boldsymbol{c}} \boldsymbol{x}_s]_s^{\mathsf{T}}, \ K = \left[W_s^K \boldsymbol{x}_s \right]_s^{\mathsf{T}}, \\ V &= [\mathbb{1}_D, \ldots, \mathbb{1}_D]^{\mathsf{T}} \quad (s = 1, \ldots, T) \end{aligned} \tag{23}$$

とすれば Selective SSM の出力は $Y = [\boldsymbol{y}_s]_s^{\mathsf{T}} = (QK^{\mathsf{T}})V$ となり，Self-Attention のように表記することができる．このような式変形は，Mamba の推論過程を Attention のように分析することを可能にするため，Attention の解釈手法を Mamba に適応する研究 [1] や Attention と SSM を比較する研究 [14] でも用いられている．

帰納ヘッドタスクの推論をしている Conv+SSM モデルの Hidden Attention Map [1] を可視化する．具体的には，式 (23) で定義した Q, K の積 QK^{T} を計算して**図 7** に示す．t 行目は，t 番目に入力されたトークンが過去 (すなわち，$t-1$ 列目まで) のどの入力の記憶を参照しているかを色の濃さで表している．例えば最下行を見ると，トークン w_{11} が入力されたときは，過去にトークン w_{11} が出現した直後のトークン（$\ldots, 9, 11, \underline{9}, 4, \ldots, 15, 11, \underline{3}, 0, 11, \underline{1}, 9, \ldots$ の下線部）が参照される．式 (22) より，これは $s-1$ 番目のトークンが w_{11} だったときに $(W_s^K \boldsymbol{x}_s)^{\mathsf{T}} (W_{\boldsymbol{c}} \boldsymbol{x}_s)$ の値が大きくなることに対応する．

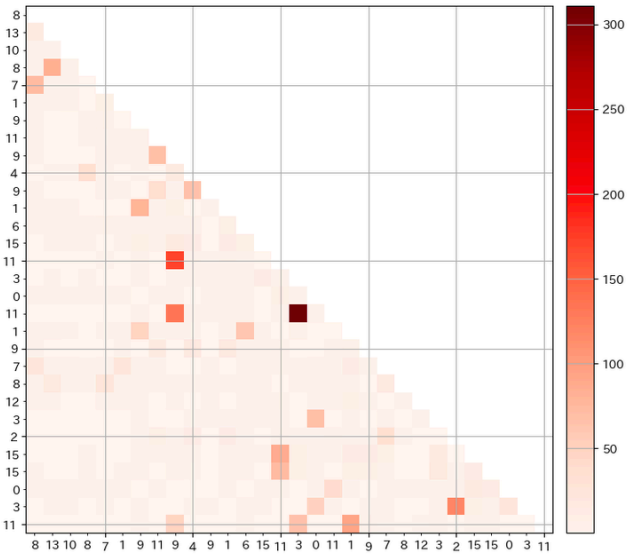


図 7: Hidden Attention Map.

6. おわりに

本研究では，単純化した 1 層の Mamba ブロックが帰納ヘッドタスクをどのようにして実行しているかを分析した．その結果，畳み込み層と Selective SSM のみからなる単純なモデルが帰納ヘッドタスクを実行できることを確認した．さらに，内部状態の分析を通じて，Selective SSM による帰納ヘッドタスクの推論メカニズムを明らかにした．

本研究の限界と今後の課題: 本研究の分析対象は，畳み込み層と Selective SSM のみからなる単純なモデルであり，タスクも帰納ヘッドタスクという単純な人工タスクに限られる．より現実的な推論メカニズムを明らかにするためには，多層の Mamba ブロックからなる言語モデルに対して，自然言語を入力にしたときのモデルの推論過程を分析しなければならない．

本研究では，語彙サイズと状態数が等しかったが (表 2)，自然言語により学習された Mamba の語彙サイズは状態数よりもはるかに大きく設定される．[*4]そのため，自然言語を推論する Mamba は，本研究で示したものよりも一般化されたメカニズムに基づいていると考えられる．例えば，隠れ状態の各次元にはトークン自身ではなくカテゴリのような抽象的な表現が記憶されることが予想される．また，トークンの出現頻度の差により生じる意味の強さの違いを Mamba がどのように捉えているかを分析するためには，本研究では注目しなかったパラメータ Δ_t（式 (8)）などを考慮する必要がある．

[*4] HuggingFace に公開されている `state-spaces/mamba-2.8b-hf` [7] と `TRI-ML/mamba-7b-rw` [10] の語彙サイズは，それぞれ 50280 と 50432 であるのに対して，状態数はどちらも 16 に設定されている．

参考文献

[1] Ali, A., Zimerman, I. and Wolf, L.: The Hidden Attention of Mamba Models (2024).

[2] Beltagy, I., Peters, M. E. and Cohan, A.: Longformer: The Long-Document Transformer (2020).

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D.: Language Models are Few-Shot Learners, *Advances in Neural Information Processing Systems* (Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. and Lin, H., eds.), Vol. 33, Curran Associates, Inc., pp. 1877–1901 (online), available from ⟨https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf⟩ (2020).

[4] Child, R., Gray, S., Radford, A. and Sutskever, I.: Generating Long Sequences with Sparse Transformers (2019).

[5] Dao, T.: FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning, *The Twelfth International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=mZn2Xyh9Ec⟩ (2024).

[6] Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A. and Re, C.: Hungry Hungry Hippos: Towards Language Modeling with State Space Models, *The Eleventh International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=COZDy0WYGg⟩ (2023).

[7] Gu, A. and Dao, T.: Mamba: Linear-Time Sequence Modeling with Selective State Spaces (2023).

[8] Gu, A., Goel, K. and Re, C.: Efficiently Modeling Long Sequences with Structured State Spaces, *International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=uYLFoz1vlAC⟩ (2022).

[9] Katharopoulos, A., Vyas, A., Pappas, N. and Fleuret, F.: Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention, *Proceedings of the 37th International Conference on Machine Learning* (III, H. D. and Singh, A., eds.), Proceedings of Machine Learning Research, Vol. 119, PMLR, pp. 5156–5165 (online), available from ⟨https://proceedings.mlr.press/v119/katharopoulos20a.html⟩ (2020).

[10] Mercat, J., Vasiljevic, I., Keh, S., Arora, K., Dave, A., Gaidon, A. and Kollar, T.: Linearizing Large Language Models (2024).

[11] Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S. and Olah, C.: In-context Learning and Induction Heads, *Transformer Circuits Thread* (2022). https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[12] Peng, B., Goldstein, D., Anthony, Q., Albalak, A., Alcaide, E., Biderman, S., Cheah, E., Ferdinan, T., Hou, H., Kazienko, P., GV, K. K., Kocoń, J., Koptyra, B., Krishna, S., au2, R. M. J., Muennighoff, N., Obeid, F., Saito, A., Song, G., Tu, H., Woźniak, S., Zhang, R., Zhao, B., Zhao, Q., Zhou, P., Zhu, J. and Zhu, R.-J.: Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence (2024).

[13] Shazeer, N.: GLU Variants Improve Transformer (2020).

[14] Sieber, J., Alonso, C. A., Didier, A., Zeilinger, M. N. and Orvieto, A.: Understanding the differences in Foundation Models: Attention, State Space Models, and Recurrent Neural Networks (2024).

[15] Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X. and Wei, F.: A Length-Extrapolatable Transformer, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Rogers, A., Boyd-Graber, J. and Okazaki, N., eds.), Toronto, Canada, Association for Computational Linguistics, pp. 14590–14604 (online), DOI: 10.18653/v1/2023.acl-long.816 (2023).

[16] Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S. and Metzler, D.: Long Range Arena : A Benchmark for Efficient Transformers, *International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=qVyeW-grC2k⟩ (2021).

[17] Tay, Y., Dehghani, M., Bahri, D. and Metzler, D.: Efficient Transformers: A Survey, *ACM Comput. Surv.*, Vol. 55, No. 6 (online), DOI: 10.1145/3530811 (2022).

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u. and Polosukhin, I.: Attention is All you Need, *Advances in Neural Information Processing Systems* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R., eds.), Vol. 30, Curran Associates, Inc., (online), available from ⟨https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf⟩ (2017).

[19] Wang, S., Li, B. Z., Khabsa, M., Fang, H. and Ma, H.: Linformer: Self-Attention with Linear Complexity (2020).

[20] Wang, X., Salmani, M., Omidi, P., Ren, X., Rezagholizadeh, M. and Eshaghi, A.: Beyond the Limits: A Survey of Techniques to Extend the Context Length in Large Language Models (2024).

[21] Yang, S. and Zhang, Y.: FLA: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism (2024).

[22] Zhai, S., Talbott, W., Srivastava, N., Huang, C., Goh, H., Zhang, R. and Susskind, J.: An Attention Free Transformer (2021).

付　　録

A.1　トークン番号と隠れ状態の行番号の整合

図 6b と 図 6a を比較すると，トークン w_3 が出現したときに隠れ状態が変化する傾向は共通しているが，トークン w_3 の直前に出現したトークンが対応する次元に記憶される様子が図 6a では見られない．この問題は，学習されたパラメータ $W_{\boldsymbol{b}} \in \mathbb{R}^{N\times D}$ の第 i 行が各トークン w_i に対応していないため，トークン w_i に対応する成分が隠れ状態 H_t の第 i 行以外の行にも加算されることで生じている．

そこで，トークン w_i の出現が隠れ状態 H_t の第 i 行で記憶されると解釈できるようにする方法を考える．§4.1 で説明したメカニズムにおける bi-gram の記憶 (3) によると，Selective SSM は ***bi-gram* の前半がトークン w_i であることを，隠れ状態 H_t の第 i 行に入力 $\boldsymbol{x}_t$ の成分を加算する**ことで記憶する．これは，式 (16): $\boldsymbol{b}_t = W_{\boldsymbol{b}}\boldsymbol{x}_t = \boldsymbol{e}_i$ を満たすことで実現する．すなわち，トークン w_i が入力系列の $t-1$ 番目に出現したとき，ベクトル $\boldsymbol{x}_t$ がパラメータ $W_{\boldsymbol{b}}$ の第 i 行と類似して，その他の行と直交していればよい．

ここでは，bi-gram の前半トークンの表現の影響のみに興味があるため，パラメータ $W_{\boldsymbol{b}}$ の各行と畳み込みにより抽出された前半トークンの表現 $\boldsymbol{k}_0 \odot \boldsymbol{e}(w_i)$ との内積を計算することで類似度を調べる．これをすべての語彙 $(i = 0 \ldots V-1)$ について計算した結果を並べた行列を $S \in \mathbb{R}^{N\times V}$ とおく:

$$S = W_{\boldsymbol{b}}[\boldsymbol{k}_0 \odot \boldsymbol{e}(w_0), \ldots, \boldsymbol{k}_0 \odot \boldsymbol{e}(w_{V-1})]. \tag{A.1}$$

図 A·1 に示した行列 S の成分を見ると，行列 S は単位行列のようになっていない．これは，学習済みモデルにおける隠れ状態 H_t の第 i 行は，トークン w_i 以外のトークンの記憶に対応していることを意味している．例えば，図 6a において，19 番目のトークン w_{10} に続いて 20 番目にトークン w_3 が出現したとき，1 行目の成分が大きくなり 8 行目の成分が小さくなっている．このトークン番号と隠れ状態 H_t の行番号の不一致は，行列 S の (1, 10) 成分と (8, 10) 成分の絶対値が大きいため，トークン w_{10} が出現したことが H_t の第 10 行ではなく 第 1 行と第 8 行に記憶されることが原因である．このような不一致は，H_t に S^{T} を左から掛けることで解消される．そうすることで，今回のケースでは第 1 行と第 8 行の成分が第 10 行に加算され，図 6a を図 6b に変換することができる．

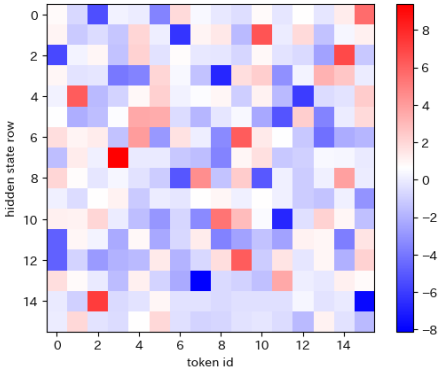


図 A·1: 行列 S の成分．第 i 列は $W_{\boldsymbol{b}}(\boldsymbol{k}_0 \odot \boldsymbol{e}(w_i))$ である．

A.2　Selection Map の式変形

更新式 (3) を展開する:

$$\begin{aligned}
\boldsymbol{h}_t &= \overline{A}_t\boldsymbol{h}_{t-1} + \overline{\boldsymbol{b}}_t\boldsymbol{x}_t \\
&= \overline{A}_t\overline{A}_{t-1}\boldsymbol{h}_{t-2} + \overline{A}_t\overline{\boldsymbol{b}}_{t-1}\boldsymbol{x}_{t-1} + \overline{\boldsymbol{b}}_t\boldsymbol{x}_t \\
&= \overline{A}_t\overline{A}_{t-1}\overline{A}_{t-2}\boldsymbol{h}_{t-3} + \overline{A}_t\overline{A}_{t-1}\overline{\boldsymbol{b}}_{t-2}\boldsymbol{x}_{t-2} \\
&\quad + \overline{A}_t\overline{\boldsymbol{b}}_{t-1}\boldsymbol{x}_{t-1} + \overline{\boldsymbol{b}}_t\boldsymbol{x}_t \\
= \cdots &= \prod_{r=1}^{t}\overline{A_r\boldsymbol{b}}_0\boldsymbol{x}_0 + \prod_{r=2}^{t}\overline{A}_r\overline{\boldsymbol{b}}_1\boldsymbol{x}_1 \\
&\quad + \cdots + \prod_{r=t}^{t}\overline{A}_r\overline{\boldsymbol{b}}_{t-1}\boldsymbol{x}_{t-1} + \overline{\boldsymbol{b}}_t\boldsymbol{x}_t \\
&= \sum_{s=0}^{t-1}\prod_{r=s+1}^{t}\overline{A}_r\overline{\boldsymbol{b}}_s x_s + \overline{\boldsymbol{b}}_t\boldsymbol{x}_t \\
&= \sum_{s=0}^{t-1}\prod_{r=s+1}^{t}\overline{A}_r(\boldsymbol{x}_s\Delta_s W_{\boldsymbol{b}})\boldsymbol{x}_s + (\boldsymbol{x}_t\Delta_t W_{\boldsymbol{b}})\boldsymbol{x}_t.
\end{aligned}$$

ここで，

$$W_s^K = \begin{cases} \left(\prod_{r=s+1}^{t}\overline{A}_r\right)(\boldsymbol{x}_s\Delta_s W_{\boldsymbol{b}}) & (1 \le s < t) \\ \boldsymbol{x}_t\Delta_t W_{\boldsymbol{b}} & (s = t) \end{cases} \tag{A.2}$$

とすれば，

$$\boldsymbol{h}_t = \sum_{s=1}^{t} W_s^K\boldsymbol{x}_s \tag{A.3}$$

と書ける．したがって，これを式 (4) に代入すると:

$$y_t = \boldsymbol{h}_t^{\mathsf{T}}(W_{\boldsymbol{c}}\boldsymbol{x}_t) \tag{A.4}$$

$$= (W_{\boldsymbol{c}}\boldsymbol{x}_t)^{\mathsf{T}}\sum_{s=1}^{t} W_s^K\boldsymbol{x}_s \tag{A.5}$$

$$= \sum_{s=1}^{t}(\boldsymbol{x}_t W_{\boldsymbol{c}})^{\mathsf{T}}(W_s^K\boldsymbol{x}_s). \tag{A.6}$$

が得られる．