Received 1 July 2024, accepted 12 August 2024, date of publication 20 August 2024, date of current version 30 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446642

RESEARCH ARTICLE

Model Compression Method for S4 With Diagonal State Space Layers Using Balanced Truncation

HARUKA EZOE AND KAZUHIRO SATO, (Member, IEEE)
Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo 113-8656, Japan

Corresponding author: Kazuhiro Sato (kazuhiro@mist.i.u-tokyo.ac.jp)

This work was supported by Japan Society for the Promotion of Science KAKENHI under Grant 23H03680.

ABSTRACT To implement deep learning models on edge devices, model compression methods have been widely recognized as useful. However, it remains unclear which model compression methods are effective for Structured State Space Sequence (S4) models incorporating Diagonal State Space (DSS) layers, tailored for processing long-sequence data. In this paper, we propose to use the balanced truncation, a prevalent model reduction technique in control theory, applied specifically to DSS layers in pre-trained S4 model as a novel model compression method. Moreover, we propose using the reduced model parameters obtained by the balanced truncation as initial parameters of S4 models with DSS layers during the main training process. Numerical experiments demonstrate that our trained models combined with the balanced truncation surpass conventionally trained models with Skew-HiPPO initialization in accuracy, even with fewer parameters. Furthermore, our observations reveal a positive correlation: higher accuracy in the original model consistently leads to increased accuracy in models trained using our model compression method, suggesting that our approach effectively leverages the strengths of the original model.

INDEX TERMS Balanced truncation, deep learning, diagonal state space model, model compression.

I. INTRODUCTION

In recent years, deep learning models have garnered substantial attention due to their versatility across a range of applications, including sequence prediction, natural language translation, speech recognition, and audio generation [1], [2], [3]. These models' ability to understand and predict sequential data underpins their success in these domains. A critical aspect of these models' effectiveness is their capacity to capture dependencies between sequential data points, a fundamental requirement for achieving high levels of performance in tasks involving time series or sequential input. For instance, the Transformer [4] is effective in capturing short-range dependencies in sequential data, and achieved a state-of-the-art BLEU score of 41.0 on the WMT 2014 English-to-French translation task. However, the Transformer's ability to capture long-range dependencies in time series data is limited, leading to the loss of temporal information due to its permutation-invariant self-attention mechanism [5].

Contrary to the limitations observed in the Transformer model, the Structured State Space Sequence (S4) model, as introduced in [6], demonstrates exceptional capability in capturing long-range dependencies within sequential data. This effectiveness is largely attributed to the innovative use of HiPPO initialization [7], a technique specifically designed to enhance model performance by leveraging the principles of the state space model (SSM) from control theory. Notably, the S4 model has shown to surpass conventional models, including the Transformer, in Long Range Arena (LRA) tasks [8], signifying a substantial advancement in handling sequential data. Further refinement of the S4 architecture led to the introduction of the Diagonal State Space (DSS) layers [9], offering a simplified yet effective version of the original S4 model, maintaining its high performance with a more streamlined architecture. In addition to the original and simplified S4 models, several deep learning models related to the SSM, such as H3 [10], Hyena [11], S4D [12], S4ND [13], S5 [14], SSSD [15], and Mamba [16],

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Xu.

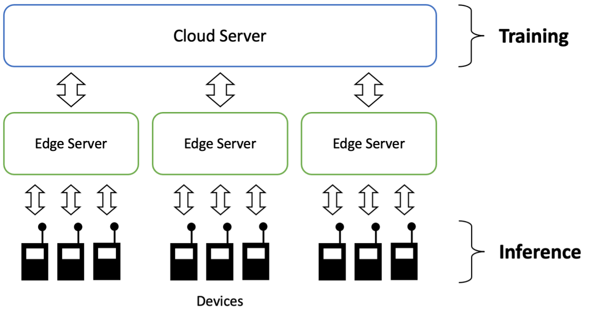

FIGURE 1. Edge Intelligence (EI). In EI, data gathered from various devices is not processed entirely in the cloud but rather locally on each device. These devices, like sensors in industrial settings, face limitations that make it difficult to deploy large-scale deep learning models typically trained in the cloud, due to constraints such as computing resources and power consumption.

have been proposed. Generally, in tasks requiring long-range dependency modeling, these deep learning models tend to perform better with a larger number of parameters.

However, deep learning models, which have a vast number of parameters, demand considerable computational resources for inference, thereby limiting their practical and sustainable use. For example, in Edge Intelligence (EI) [17], [18], data from individual devices are processed both in the cloud and locally on each device (Fig. 1). EI devices, such as sensors in factories, have limited computational resources and power consumption constraints. This limitation poses a challenge for performing inference using deep learning models with numerous parameters. Therefore, it is crucial to achieve optimal performance using models with fewer parameters and reduced computational costs in EI applications.

When considering the application of deep learning models in EI, the implementation of model compression techniques is essential [19], [20], [21], [22]. For example, the techniques include:

- Pruning, which reduces the number of parameters in deep learning models [23], [24].
- Quantization, which reduces the number of bits used to represent the weights and activations in deep learning models [25].
- Knowledge distillation, involving training a smaller student model to replicate a larger teacher model [26], [27], [28], [29].

However, the effectiveness of these model compression techniques in deep models that incorporate SSMs remains unclear.

Thus, our goal is to provide a novel and effective model compression method tailored for S4 models with DSS layers, especially for EI scenario deployment. To achieve this, we leverage SSM's use in DSS layers to enable the use of various well-established reduction methods [30], [31], [32], [33], [34], [35], [36]. In this study, we employ the balanced truncation method [33], a widely used control theory approach. To train the original model, we use Skew-HiPPO initialization [9], [12], consistently outperforming models started with random initialization.

The contributions of this paper are summarized as follows: To reduce computational costs during inference, we introduce a novel model compression method that applies the balanced truncation technique to DSS layers in pre-trained S4 models. Moreover, we propose using the reduced model parameters obtained by the balanced truncation as initial parameters of S4 models with DSS layers during the main training process. As demonstrated in Section VI, our trained models combined with the balanced truncation achieved superior accuracy on LRA tasks compared to conventionally trained models using Skew-HiPPO initialization as described in [9] and [12], even with fewer parameters. While [37] reports minimal impact from dimension reduction in the MultiHyena variant of the Hyena model on performance, our findings with the S4 model underscore a critical distinction with significant performance improvement.

The paper is organized as follows. In Section II, we introduce the balanced truncation method for the reduction of state space models and explain the HiPPO matrix used in Skew-HiPPO initialization. In Section III, we present a deep learning model with DSS layers. Section IV describes existing training methods for this model, and discusses the model's computational cost during inference. To address the issue of computational cost, in Section V, we propose a model compression method using the balanced truncation method for SSMs. In Section VI, the results of numerical experiments are presented. Finally, in Section VII, we discuss the effectiveness of the proposed method based on the results of the numerical experiments, clarify the limitations of our work, and outline future work.

Notation:

- $\mathbb{R}$ and $\mathbb{C}$ denote the sets of real and complex numbers, respectively.
- For $a \in \mathbb{C}$, $|a|$ denotes the absolute value of a.
- For $v = (v_i) \in \mathbb{C}^N$, $\|v\|$ denotes the Euclidean norm of v, i.e., $\|v\| := \sqrt{|v_1|^2 + \cdots + |v_N|^2}$.
- $A^\top$ and A^* denote the transpose and complex conjugate transpose of matrix $A \in \mathbb{C}^{n\times n}$, respectively.
- $f(x)\mid_{x\leq a}$ denotes the function that coincides with $f(x)$, a function defined on $x \geq 0$, on its domain $[0, a]$.
- $\deg(f)$ represents the degree of polynomial $f(x)$.
- i denotes the imaginary unit.
- $\mathrm{diag}(\lambda_1, \cdots, \lambda_N) \in \mathbb{C}^{N\times N}$ is a diagonal matrix with $\lambda_1, \cdots, \lambda_N \in \mathbb{C}$ as diagonal elements.
- $\mathrm{Re}(\alpha)$ and $\mathrm{Im}(\alpha)$ are the real and imaginary parts of $\alpha \in \mathbb{C}$, respectively.
- $a*b$ represents the Hadamard product of vectors a and b.
- $\mathbb{I}_{[a,b]}(x) = \begin{cases} 1 & \text{if} \quad a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$
- $\mathcal{U}(a, b)$ denotes the uniform distribution on (a, b).
- $\mathcal{N}(\mu, \sigma)$ is a normal distribution with mean μ and variance σ.

II. PRELIMINARIES

In this section, we introduce a state space model (SSM), an important component of the DSS layer. We also present the balanced truncation method [33], a reduction method for SSMs employed in our training method. Furthermore, we explain the HiPPO matrix [6] utilized in Skew-HiPPO initialization [9], [12], which enhances the performance of trained models.

A. STATE SPACE MODEL (SSM)

A crucial component of a deep learning model discussed in our study, as outlined in Section III, is the hidden layer known as the DSS layer. This layer is defined by using the SSM

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t), \end{cases} \tag{1}$$

where $u(t) \in \mathbb{R}$, $y(t) \in \mathbb{C}$, and $x(t) \in \mathbb{C}^N$ denote the input, output, and state, respectively, and $(A, B, C) \in \mathbb{C}^{N\times N} \times \mathbb{C}^{N\times 1} \times \mathbb{C}^{1\times N}$. The matrix A denotes a state transition matrix, which describes the internal influence on the time evolution of the internal state x. The matrix B is an input matrix, which describes how the external input u affects the internal state x. The matrix C serves as an output matrix, which describes how the internal state x is transformed into the observable output y.

Notably, SSM (1) is a single-input and single-output system, and the matrices A, B, C, state $x(t)$, and output $y(t)$ are all complex-valued. Moreover, the state dimension N is relatively large, unlike the standard setting in control theory [38].

B. BALANCED TRUNCATION METHOD

To address computational issues arising from the large state dimension of SSM (1), we consider using the balanced truncation method [33], a reduction technique. This method focuses on the controllability and observability of SSM (1) to derive another SSM with dimension r ($\leq N$), which gives almost the same output as (1):

$$\begin{cases} \dfrac{\mathrm{d}x_r}{\mathrm{d}t}(t) = A_r x_r(t) + B_r u(t) \\ y_r(t) = C_r x_r(t), \end{cases} \tag{2}$$

where $A_r \in \mathbb{C}^{r\times r}$, $B_r \in \mathbb{C}^{r\times 1}$, $C_r \in \mathbb{C}^{1\times r}$. Below is a brief explanation of the balanced truncation method. For more detailed information, refer to Appendix A.

For SSM (1) with asymptotic stability, where all the real parts of the eigenvalues of matrix A are negative, the controllability Gramian P and observability Gramian Q are defined as

$$P := \int_0^\infty \exp(At)BB^* \exp(A^*t)\mathrm{d}t,$$
$$Q := \int_0^\infty \exp(A^*t)C^*C \exp(At)\mathrm{d}t.$$

These are the unique solutions to the Lyapunov equations

$$AP + PA^* + BB^* = 0, \tag{3}$$
$$A^*Q + QA + C^*C = 0, \tag{4}$$

as shown in [38, Theorem 4.1].

In the balanced truncation method, the subspace spanned by eigenvectors corresponding to small eigenvalues of the controllability Gramian P and the observability Gramian Q is ignored. The minimum input energy required to achieve $\lim_{t\to\infty} x(t) = x_f$ from the initial condition $x(0) = 0$ is expressed using the controllability Gramian P as

$$\int_0^\infty \|u(t)\|^2 \mathrm{d}t = x_f^* P^{-1} x_f.$$

Thus, eigenvectors corresponding to smaller eigenvalues of P correspond to directions in the state space that are less influenced by the input u. On the other hand, when $x(0) = x_0$ and $u(0) = 0$, the output energy can be expressed using the observability Gramian Q as

$$\int_0^\infty \|y(t)\|^2 \mathrm{d}t = x_0^* Q x_0.$$

Thus, eigenvectors corresponding to smaller eigenvalues of Q correspond to directions in the state space that have less impact on the output y.

A coordinate transformation $\bar{x}(t) = Tx(t)$ is applied to SSM (1), obtaining another SSM where the controllability Gramian and observability Gramian coincide as a diagonal matrix Σ:

$$\begin{cases} \dfrac{\mathrm{d}\bar{x}}{\mathrm{d}t}(t) = TAT^{-1}\bar{x}(t) + TBu(t) \\ y(t) = CT^{-1}\bar{x}(t). \end{cases} \tag{5}$$

SSM (5) is referred to as the balanced realization of SSM (1). Here, the diagonal elements of Σ are denoted as $\sigma_1, \cdots, \sigma_N$, which are called the Hankel singular values, satisfying $\sigma_1 \geq \cdots \geq \sigma_N > 0$ under the assumption that SSM (1) is controllable and observable. By partitioning the matrices TAT^{-1}, TB, CT^{-1} in SSM (5) as

$$TAT^{-1} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{6}$$
$$TB = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}, \tag{7}$$
$$CT^{-1} = \begin{pmatrix} C_1 & C_2 \end{pmatrix}, \tag{8}$$

we define the parameters (A_r, B_r, C_r) of reduced model (2) as

$$(A_r, B_r, C_r) = (A_{11}, B_1, C_1). \tag{9}$$

The resulting system (2) with (9) can be interpreted as a reduced SSM of SSM (1), obtained by truncating the state space associated with the smaller Hankel singular values $\sigma_{r+1}, \cdots, \sigma_N$, which correspond to the subspace spanned by eigenvectors that are less influenced by the input or have less influence on the output. Moreover, if SSM (1) is asymptotically stable, the reduced SSM (2) with (9) is also asymptotically stable, as shown in [38, Proposition 4.15].

C. HiPPO MATRIX

For the model explained in Section III, the parameters of the matrices (A, B, C) in the SSM (1) are trained using a suitable optimization algorithm. The initialization of matrix A significantly influences the performance of trained models, as it sets the initial state for the optimization process. The High-order Polynomial Projection Operators (HiPPO) matrix [7] is derived from a method for online compression of continuous signals using projections onto subspaces spanned by polynomial bases. The matrix is an effective choice for the initial A [6], [9], [12].

The derivation of the HiPPO matrix is explained below. With respect to a measure μ on $[0, \infty)$, let

$$L_2(\mu) := \{f : [0,\infty) \to \mathbb{C} \;\Big|\; f \text{ is measurable}, \int_0^\infty |f(\tau)|^2 \mathrm{d}\mu(\tau) < \infty\}.$$

The inner product and norm on $L_2(\mu)$ are defined as

$$\langle f_1, f_2\rangle_\mu := \int_0^\infty f_1^*(\tau) f_2(\tau)\mathrm{d}\mu(\tau),$$
$$\|f\|_{L_2(\mu)} := \langle f, f\rangle_\mu^{1/2},$$

respectively.

For an input signal $u(t)$ defined on $t \geq 0$, the history $u_{\leq t} := u(\tau)\mid_{\tau \leq t}$ at each time $t > 0$ is approximated by projecting it onto a subspace spanned by polynomial bases, and the corresponding coefficient vector $x(t)$ represents the history of input signal. This compression is useful because storing $u_{\leq t}$ requires a significant amount of memory. Thus, at each time t, $x(t)$ contains sufficient information to reconstruct $u_{\leq t}$, even though it requires less memory compared to directly storing $u_{\leq t}$.

The vector $x(t)$ is expressed as the optimal solution to a convex optimization problem, which is defined by a measure $\mu^{(t)}$ on $(-\infty, t]$ and orthogonal polynomial basis of the subspace of $L_2(\mu^{(t)})$ denoted as $\{g_n^{(t)}\}_{1\leq n\leq N}$ (i.e. $\deg(g_i^{(t)}) = i$ $(i = 1, \cdots, N)$, $\langle g_i^{(t)}, g_j^{(t)}\rangle_{\mu^{(t)}} = 0$ $(i \neq j)$). The optimization problem is

$$\min_{x(t)\in\mathbb{R}^N} \quad \|u_{\leq t} - g\|^2_{L_2(\mu^{(t)})}$$
$$\text{subject to } g(\tau) = \sum_{k=1}^N x_k(t) g_k^{(t)}(\tau).$$

If $\{g_n^{(t)}\}_{1\leq n\leq N}$ is a normalized orthogonal basis, i.e. $\|g_i^{(t)}\|_{L_2(\mu^{(t)})} = 1$ $(i = 1, \cdots, N)$, the optimal solution is given by

$$x_k(t) = \langle u_{\leq t}, g_k^{(t)}\rangle_{\mu^{(t)}}. \tag{10}$$

The vector $x(t)$ defines the approximation $g = \sum_{k=1}^N x_k(t) g_k^{(t)}$ of $u_{\leq t}$, thus it retains information necessary for reconstructing the history $u_{\leq t}$ of the input $u(t)$ at time t. This property of memorizing the input history in the state vector will be useful for modeling long sequential data, as capturing dependencies in sequential data requires referencing information from previous inputs to compute each output at every time step. Moreover, by Equation (10), the measure $\mu^{(t)}$ represents the importance of each time step when compressing the history $u_{\leq t}$. This $x(t)$ satisfies the differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = A(t)x(t) + B(t)u(t),$$

where $A(t) \in \mathbb{R}^{N\times N}, B(t) \in \mathbb{R}^{N\times 1}$ depend on the polynomial basis $\{g_n^{(t)}\}_{1\leq n\leq N}$ and the measure $\mu^{(t)}$. Unlike SSM (1) introduced in Subsection II-A, $A(t)$ and $B(t)$ are time-dependent.

For HiPPO-LegS that is a variant of HiPPO [7], the measure $\mu^{(t)}$ is defined as the scaled Legendre (LegS) measure

$$\mu^{(t)} = \frac{1}{t}\mathbb{I}_{[0,t]}.$$

This assigns uniform importance to the entire history $[0, t]$ at each time t. Furthermore, the polynomial basis $g_n^{(t)}(\tau)$ is the normalized orthogonal basis

$$g_n^{(t)}(\tau) = g_n(t,\tau) = (2n+1)^{1/2} P_n\left(\frac{2\tau}{t} - 1\right),$$

where $P_n(\alpha)$ is the Legendre polynomial

$$P_n(\alpha) = \frac{1}{2^n \cdot n!}\frac{\mathrm{d}^n}{\mathrm{d}\alpha^n}(\alpha^2 - 1)^n.$$

In this case, as shown in [7, Appendix D.3], $x(t)$ satisfies

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t}(t) &= \frac{1}{t}Ax(t) + \frac{1}{t}Bu(t) \\ A_{nk} &= -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if} \quad n > k \\ n+1 & \text{if} \quad n = k \\ 0 & \text{if} \quad n < k \end{cases} \\ B_n &= (2n+1)^{1/2}. \end{aligned} \tag{11}$$

This matrix A of Equation (11) is called the HiPPO matrix. For the SSM incorporating the HiPPO matrix, the state vector $x(t)$ retains information about the history $u_{\leq t}$ of the input u at each time t [7].

III. DEEP LEARNING MODEL EMPLOYED IN THIS STUDY

The deep learning model employed in this study, proposed in [9], has the structure illustrated in Fig. 2. In this paper, the model is referred to as S4 with DSS layers, despite being named DSS by the authors of [9]. The input layer receives sequential data and outputs H features as 1-dimensional sequential data. This conversion adapts data from various formats to the DSS layer's input format, as detailed in Subsection III-A. The term H denotes the hidden size, representing the count of features processed by the DSS layer. Finally, the output layer converts the H features of 1-dimensional sequential data into the model's final output format. For further details, refer to Appendix B.

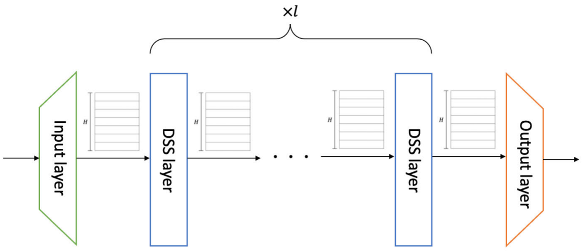

FIGURE 2. Deep learning model with DSS layers. This represents the overall architecture of the deep learning model used in this study, with the intermediate DSS layer being the most critical component.

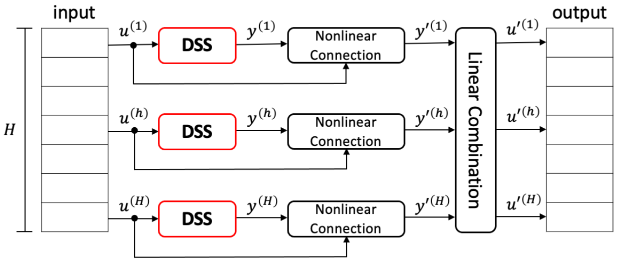

FIGURE 3. DSS layer, which consists of *H* DSS models, nonlinear connection blocks, and a linear combination block.

A. DIAGONAL STATE SPACE LAYER

The most important component of the deep learning model illustrated in Fig. 2 is the DSS layer. As shown in Fig. 3, the DSS layer consists of:

- Independent H DSS models.
- Nonlinear connection blocks.
- Linear combination block.

The details of each of these components are explained below.

By restricting the matrix $A \in \mathbb{C}^{N\times N}$ in (1) to be diagonal, assuming that the diagonal elements do not lie on the imaginary axis, the DSS model is defined by the following discretization for a sample time $\Delta \in \mathbb{R}_{>0}$:

$$\begin{cases} x_{k+1} = \bar{A}x_k + \bar{B}u_k \\ y_k = \bar{C}x_k, \end{cases} \tag{12}$$

where $\bar{A} = e^{A\Delta}$, $\bar{B} = (\bar{A} - I)A^{-1}B$, and $\bar{C} = C$. The diagonal elements of the matrix $\bar{A}$ do not lie on the unit circle in the complex plane, due to the assumption on the matrix A.

The nonlinear connection block receives the input u_k and output y_k from the DSS model and outputs 1-dimensional sequential data

$$y'_k = \mathrm{GELU}(\mathrm{Re}(y_k) + Du_k), \tag{13}$$

where $D \in \mathbb{R}$, and GELU [39] is a nonlinear activation function expressed as

$$\mathrm{GELU}(\alpha) = \alpha\Phi(\alpha).$$

Here, $\Phi(\alpha)$ is the cumulative distribution function of the standard normal distribution. This approach is expected to enhance the performance of the model. Note that $\mathrm{Re}(y_k)$ is used in Equation (13) since y_k can be a complex number.

Finally, the H 1-dimensional sequential data $(y_k'^{(h)})_{1\le h\le H}$ outputted from each nonlinear connection block is mixed to obtain the final output of the DSS layer, resulting in H 1-dimensional sequential data $(u_k'^{(h)})_{1\le h\le H}$. With parameters of weight $W_{out} \in \mathbb{R}^{H\times H}$ and bias $b \in \mathbb{R}^H$, the output is expressed as

$$u_k'^{(h)} = \sum_{h'=1}^{H} W_{out,hh'}y_k'^{(h')} + b_h \cdot \mathbf{1}, \tag{14}$$

where $\mathbf{1}$ is a 1-dimensional sequential data of the same length as $y_k'^{(h)}$ with all elements 1.

B. DSS$_{EXP}$ AND DSS$_{SOFTMAX}$

The output of DSS (12) can be calculated as

$$y_k = \sum_{i=0}^{k} h_i \cdot u_{k-i} \tag{15}$$

with $h_i := \bar{C}\bar{A}^i\bar{B}$, which is referred to as the impulse response of DSS (12). Given a sample time Δ, h_i is determined by (A, B, C), and different sets of parameters (A, B, C) may result in the same sequence of impulse responses. In fact,

$$\bar{K}_{\Delta,L}(A, B, C) := (h_0, h_1, \cdots, h_{L-1})$$

are the same for different (A, B, C), as described below [9].

Proposition 1: Suppose that the parameters $A = \mathrm{diag}(\lambda_1, \cdots, \lambda_N), B, C, \Delta$ of DSS (12) are given, and define $K := \bar{K}_{\Delta,L}(A, B, C) \in \mathbb{C}^L$. Then, there exist $w, \tilde{w} \in \mathbb{C}^{1\times N}$ satisfying the following equations:

(a) $K = \bar{K}_{\Delta,L}(A, (1)_{1\le i\le N}, \tilde{w})$
$= \tilde{w} \cdot A^{-1}(e^{A\Delta} - I) \cdot \text{elementwise-exp}(P)$

(b) $K = \bar{K}_{\Delta,L}(A, ((e^{L\lambda_i\Delta} - 1)^{-1})_{1\le i\le N}, w)$
$= w \cdot A^{-1} \cdot \text{row-softmax}(P)$

where

$$\text{elementwise-exp}(P) = (\exp(P_{ik}))_{1\le i\le N, 0\le k<L},$$
$$\text{row-softmax}(P) = \left(\frac{\exp(P_{ik})}{\Sigma_{r=0}^{L-1}\exp(P_{ir})}\right)_{1\le i\le N, 0\le k<L},$$
$$P_{i,k} = \lambda_i k\Delta.$$

Proposition 1 implies that, under weak assumptions, the impulse responses $h_0, h_1, \cdots, h_{L-1}$ of DSS (12) can be achieved with special structure of (A, B, C). DSS (12) with $(B, C) = ((1)_{1\le i\le N}, w)$, as stated in Proposition 1(a), is referred to as DSS$_{\text{EXP}}$, and DSS (12) with $(B, C) = (((e^{L\lambda_i\Delta} - 1)^{-1})_{1\le i\le N}, w)$, as stated in Proposition 1(b), is referred to as DSS$_{\text{SOFTMAX}}$ [9]. DSS$_{\text{EXP}}$ and DSS$_{\text{SOFTMAX}}$ offer different approaches to modeling the impulse responses, with potential implications for the performance and interpretability of the DSS model. In the following sections, we utilize DSS$_{\text{EXP}}$ or DSS$_{\text{SOFTMAX}}$ as DSS (12).

IV. EXISTING TRAINING METHODS AND LIMITATIONS

In the training of deep learning models, the goal is to minimize the loss function $E(W)$ with respect to the training dataset $\{(\chi_i, d_i)\}$. Here, (χ_i, d_i) represents a pair of input χ_i and its desired output d_i, and W denotes the parameters of the model. The model's output for an input χ with parameters W is denoted as $\zeta(\chi; W)$. For each input χ_i, a loss function $E_i(W)$ is defined to measure the difference between the desired output d_i and the model's output $\zeta(\chi_i; W)$. The loss function for the entire training dataset is expressed as $E(W) = \Sigma_i E_i(W)$, where the summation is over all training examples. As an algorithm for minimizing the loss function $E(W)$, we can consider using AdamW [40].

A. TRAINING PARAMETERS WITHIN THE DIAGONAL STATE SPACE LAYER

Among the parameters W trained in our deep learning model, those in the DSS layer include:

- (A, B, C, Δ) for each of the H DSS models, as defined in (12).
- D for each of the H nonlinear connection blocks, as defined in (13).
- Weight $W_{out} \in \mathbb{R}^{H\times H}$ and bias $b \in \mathbb{R}^H$ for the linear combination block, as defined in (14).

For DSS$_{\text{EXP}}$ defined in Subsection III-B, the parameters (A, B, C) are defined as

$$\begin{aligned} A &= \text{diag}(\lambda_1, \cdots, \lambda_N), \\ B &= (1)_{1\le i\le N}, \\ C &= w, \end{aligned} \tag{16}$$

where

$$\lambda_i = -\exp(\Lambda_{re,i}) + \mathrm{i}\cdot\Lambda_{im,i}. \tag{17}$$

For DSS$_{\text{EXP}}$, the parameters $\Lambda_{re}, \Lambda_{im} \in \mathbb{R}^N$ and $w \in \mathbb{C}^N$ are trained to determine (A, B, C).

For DSS$_{\text{SOFTMAX}}$ defined in Subsection III-B, the parameters (A, B, C) are defined as:

$$\begin{aligned} A &= \text{diag}(\lambda_1, \cdots, \lambda_N), \\ B &= ((e^{L\lambda_i\Delta} - 1)^{-1})_{1\le i\le N}, \\ C &= w, \end{aligned} \tag{18}$$

where

$$\lambda_i = \Lambda_{re,i} + \mathrm{i}\cdot\Lambda_{im,i}. \tag{19}$$

Similarly to DSS$_{\text{EXP}}$, the parameters $\Lambda_{re}, \Lambda_{im} \in \mathbb{R}^N$ and $w \in \mathbb{C}^N$ are trained to determine (A, B, C) for DSS$_{\text{SOFTMAX}}$, with a different expression for A and B.

B. INITIALIZATION OF THE DSS LAYER

The performance of S4 with DSS layers is sensitive to initialization of the state matrix A. To obtain an effective initial value for A, the HiPPO matrix is decomposed into a normal matrix and a low-rank matrix. This decomposition allows for a more structured and interpretable initialization of the state matrix A, which can improve the performance of the model. The eigenvalues of the normal matrix are employed to initialize the diagonal elements of A.

In more detail, the HiPPO matrix $\mathcal{H} \in \mathbb{R}^{N'\times N'}$ is defined as explained in Subsection II-C:

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

For SSM (1) incorporating this HiPPO matrix, the state $x(t)$ retains information about the history of the input $u(t)$ [7]. The HiPPO matrix $\mathcal{H}$ can be decomposed into a normal matrix and a low-rank matrix as

$$\mathcal{H} = \mathcal{H}' - \frac{1}{2}PQ^\top,$$

where $\mathcal{H}' \in \mathbb{R}^{N'\times N'}$, $P \in \mathbb{R}^{N'}$, $Q \in \mathbb{R}^{N'}$ are defined as

$$\mathcal{H}'_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2}/2 & \text{if} \quad n > k \\ 1/2 & \text{if} \quad n = k \\ -(2n+1)^{1/2}(2k+1)^{1/2}/2 & \text{if} \quad n < k \end{cases}$$
$$P_n = (2n+1)^{1/2}, \quad Q_k = (2k+1)^{1/2}.$$

This $\mathcal{H}'$ is a normal matrix. Under the assumption that $N' = 2N$ and $\mu_1, \ldots, \mu_N$ are the eigenvalues of $\mathcal{H}' \in \mathbb{R}^{2N\times 2N}$ with positive imaginary parts, $\Lambda_{re}, \Lambda_{im} \in \mathbb{R}^N$ are defined as follows:

- For DSS$_{\text{EXP}}$,
$$\Lambda_{re,i} = \log(-\text{Re}(\mu_i)), \quad \Lambda_{im,i} = \text{Im}(\mu_i).$$
- For DSS$_{\text{SOFTMAX}}$,
$$\Lambda_{re,i} = \text{Re}(\mu_i), \quad \Lambda_{im,i} = \text{Im}(\mu_i).$$

Using Λ_{re} and Λ_{im}, the matrix A is initialized as $A := \text{diag}(\lambda_1, \ldots, \lambda_N)$, where each λ_i is derived from Equation (17) for DSS$_{\text{EXP}}$ or Equation (19) for DSS$_{\text{SOFTMAX}}$. This process is known as the Skew-HiPPO initialization [9], [12]. Other parameters within the DSS are randomly sampled, as detailed in Section VI. According to [9], models utilizing Skew-HiPPO initialization demonstrate superior prediction accuracy compared to those initialized with values from $\mathcal{N}(0, 1)$.

C. COMPUTATIONAL COST OF THE DSS LAYER OUTPUT

When the input is entered one by one or all at once, reducing H (the hidden size) and N (the state dimension) facilitates a reduction in the computational cost of the DSS layer output. In fact, the time and space complexities of the DSS layer output are as illustrated in Table 1 and Table 2, respectively, as discussed below. Here, the input length of the 1-dimensional sequence is denoted as L.

In the case where input u_k at each time step is entered one by one into DSS (12), the output y_k can be computed using the previous state vector x_{k-1} according to (12). The time and space complexities per step are both $O(N)$. The time complexity for processing the entire input of length L is $O(NL)$, and the space complexity involves overwriting at each step, thus remaining $O(N)$. For the nonlinear connection

TABLE 1. Computational complexities of the DSS layer output when input of length L is entered one by one.

	Time	Space
DSS (1 unit)	$O(NL)$	$O(N)$
Nonlinear connection(1 unit)	$O(L)$	$O(1)$
Linear combination	$O(H^2L)$	$O(H)$
Total	$O(HNL) + O(H^2L)$	$O(HN)$

TABLE 2. Computational complexities of the DSS layer output when input of length L is entered at once.

	Time	Space
DSS (1 unit)	$O(L\log(L))$	$O(L)$
Nonlinear connection(1 unit)	$O(L)$	$O(L)$
Linear combination	$O(H^2L)$	$O(HL)$
Total	$O(HL\log(L)) + O(H^2L)$	$O(HL)$

block, both the time and space complexities per step are $O(1)$. The time complexity for processing the entire input of length L is $O(L)$, and the space complexity involves overwriting at each step, thus remaining $O(1)$. Regarding the following linear combination block, the time complexity per step is $O(H^2)$, and the space complexity is $O(H)$. The time complexity for processing the entire input of length L is $O(H^2L)$, and the space complexity involves overwriting at each step, thus remaining $O(H)$. Therefore, adding up H DSS, H nonlinear connection blocks, and one linear combination block, the time complexity of the DSS layer output when input is entered one by one is $O(HNL) + O(H^2L)$, and the space complexity is $O(HN)$ (Table 1).

In the case where whole input $(u_k)_{0\leq k<L}$ is entered all at once, the output

$$y_k = \sum_{i=0}^{L-1} h_i \cdot u_{k-i} \tag{20}$$

can be efficiently computed. In fact, leveraging the fast Fourier transform [41] implies that the time complexity is $O(L\log(L))$ and the space complexity is $O(L)$. Furthermore, the computation is parallelizable. Besides, the impulse response h_i in Equation (20) can be easily computed. In fact, for a diagonal matrix $A \in \mathbb{C}^{N\times N}$ with diagonal elements λ_i, h_i can be calculated as

$$\begin{aligned} h_i &= \bar{C}\bar{A}^i\bar{B} \\ &= Ce^{A\cdot i\Delta}(e^{A\Delta} - I)A^{-1}B \\ &= \sum_{j=1}^{N} C_j \cdot e^{\lambda_j \cdot i\Delta}(e^{\lambda_j\Delta} - 1)\lambda_j^{-1} \cdot B_j. \end{aligned}$$

The computation time for the sequence of impulse responses $(h_0, h_1, \cdots, h_{L-1})$ is $O(NL)$. As for the nonlinear connection block, the time and space complexities are both $O(L)$ Regarding the following linear combination block, the time complexity is $O(H^2L)$, and the space complexity is $O(HL)$. Therefore, adding up H DSS, H nonlinear connection blocks, and one linear combination block, the time complexity of the DSS layer output when input is entered all at once is $O(HL\log(L)) + O(H^2L)$, and the space complexity is $O(HL)$ (Table 2).

Additionally, Equation (20) is an approximation that holds when DSS (12) is asymptotically stable and L is sufficiently large. The exact output of DSS (12) is given by (15). However, when (12) is asymptotically stable and L is sufficiently large, $h_i \approx 0$ for $i \geq L$, making the approximation in (20) valid.

D. ISSUES FOR PRACTICAL APPLICATIONS

Let us consider the issues that hinder the application of S4 with DSS layers to EI [17], [18], as explained in Section I. These issues include memory constraints, computational complexity, and the trade-offs between model performance and resource efficiency.

The following can be concluded from the arguments of Subsection IV-C:

- When processing the entire input of large length L at once, the application of S4 with DSS layers to EI is challenging, even if H and N of the trained model are sufficiently small. This is because the space complexity is $O(HL)$ (Table 2), making it difficult to conduct inference in devices with small capacity memory (e.g. sensors set in factories).
- When processing the input one-by-one, which corresponds to $L = 1$, S4 with DSS layers can be applied to EI if H and N of the trained model are sufficiently small. Specifically, the time and space complexities are those shown in Table 1. That is, when $L = 1$, the time complexity is $O(HN)+O(H^2)$, and the space complexity is $O(HN)$. This implies that even for very large input sequences, inference can be conducted in devices with small capacity memory.

In summary, to apply S4 with DSS layers to EI, the input needs to be processed one-by-one, and it is desirable to keep the values of H and N as small as possible. However, excessively small values of H and N may limit the model's capacity to capture complex patterns in the data, leading to a deterioration in performance.

V. PROPOSED MODEL COMPRESSION METHOD

In this section, to address the issues discussed in Subsection IV-D, we propose an effective model compression method for S4 with DSS layers aiming to reduce computational costs during inference by one-by-one processing. Specifically, this method enables the acquisition of parameter values that achieve higher accuracy compared to existing methods when training models with DSS layers of the same H and N.

The following procedure is our proposed model compression method.

1) Apply the balanced truncation method, as explained in Subsection II-B, to a large-scale DSS that is part of a trained model.
2) Retrain the model using the reduced DSS obtained in step 1) for improved initialization.

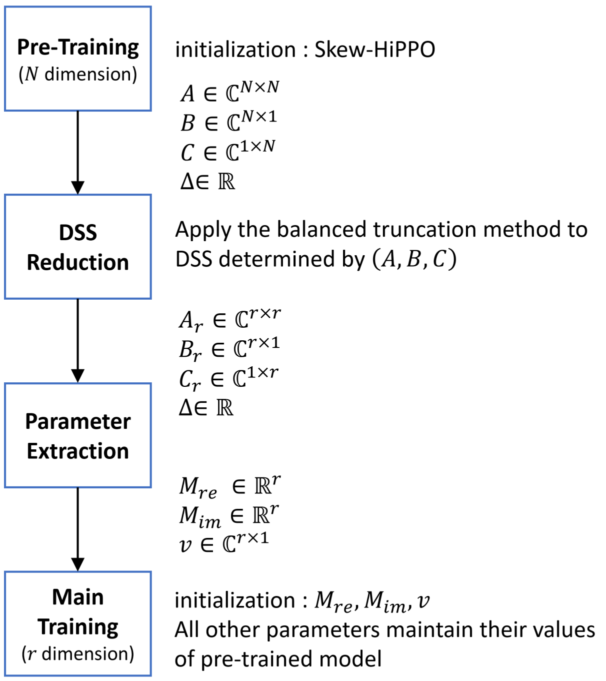
Pre-Training
(N dimension)
initialization : Skew-HiPPO
$A \in \mathbb{C}^{N\times N}$
$B \in \mathbb{C}^{N\times 1}$
$C \in \mathbb{C}^{1\times N}$
$\Delta \in \mathbb{R}$
DSS Reduction
Apply the balanced truncation method to DSS determined by (A, B, C)
$A_r \in \mathbb{C}^{r\times r}$
$B_r \in \mathbb{C}^{r\times 1}$
$C_r \in \mathbb{C}^{1\times r}$
$\Delta \in \mathbb{R}$
Parameter Extraction
$M_{re} \in \mathbb{R}^r$
$M_{im} \in \mathbb{R}^r$
$v \in \mathbb{C}^{r\times 1}$
Main Training
(r dimension)
initialization : M_{re}, M_{im}, v
All other parameters maintain their values of pre-trained model

FIGURE 4. Proposed method, which consists of Pre-Training, DSS Reduction, Parameter Extraction, and Main Training. At DSS Reduction step, the balanced truncation method is applied.

In more detail, our proposed method consists of the following Pre-Training, DSS Reduction, Parameter Extraction, and Main Training, illustrated in Fig. 4.

1) PRE-TRAINING

The parameters (A, B, C) and Δ of DSS (12) are determined by training the model with the Skew-HiPPO initialization [9], as detailed in Section IV-B. For $\mathrm{DSS_{EXP}}$ and $\mathrm{DSS_{SOFTMAX}}$, calculations use (16) and (18), respectively.

2) DSS REDUCTION

The DSS model (12), determined by parameters (A, B, C) and Δ obtained through Pre-Training, is reduced using the balanced truncation method described in Subsection II-B and Appendix A. The state dimension is reduced to $r\,(\leq N)$, resulting in the reduced SSM (2). Here, the sample time Δ remains unchanged.

3) PARAMETER EXTRACTION

Assuming A_r is diagonalizable, we can transform the reduced SSM (2) into the DSS (12), as detailed below. There exist a diagonal matrix $M = \mathrm{diag}(\mu_1, \cdots, \mu_r)$ and an invertible matrix $V \in \mathbb{C}^{r\times r}$ satisfying $A_r = VMV^{-1}$. Using a coordinate transformation $\hat{x} = V^{-1}x_r$, we obtain the new SSM

$$\begin{cases} \dfrac{\mathrm{d}\hat{x}}{\mathrm{d}t}(t) = M\hat{x}(t) + V^{-1}B_r u(t) \\ y_r(t) = C_r V\hat{x}(t), \end{cases} \tag{21}$$

TABLE 3. Text classification (negative).

	Train	Test
Number of examples	12,500	12,500
Max text length	8,969	6,385
Min text length	52	32
Avg text length	1302.97904	1285.14968

TABLE 4. Text classification (positive).

	Train	Test
Number of examples	12,500	12,500
Max text length	13,704	12,988
Min text length	70	65
Avg text length	1347.16024	1302.43512

which is equivalent to the reduced SSM (2). Consequently, the transformation allows expressing the reduced SSM (2) in the explicitly diagonal form of DSS, as shown in (21).

From Proposition 1, the impulse response of DSS (21) with the state dimension r can also be derived from $\mathrm{DSS_{EXP}}$ or $\mathrm{DSS_{SOFTMAX}}$.

- For $\mathrm{DSS_{EXP}}$, the parameters are determined as

$$\begin{aligned} M_{re,i} &= \log(-\mathrm{Re}(\mu_i)), \\ M_{im,i} &= \mathrm{Im}(\mu_i), \\ v &= (C_r V)^\top * (V^{-1}B_r). \end{aligned} \tag{22}$$

- For $\mathrm{DSS_{SOFTMAX}}$, the parameters are determined as

$$\begin{aligned} M_{re,i} &= \mathrm{Re}(\mu_i), \\ M_{im,i} &= \mathrm{Im}(\mu_i), \\ v &= (C_r V)^\top * (V^{-1}B_r) * (e^{L\mu_i\Delta} - 1)_{1\leq i\leq r}. \end{aligned} \tag{23}$$

For the vectors of (22) and (23), refer to the proof of Proposition in [9, Appendix A.1].

4) MAIN TRAINING

As detailed in Subsection IV-A, $\Lambda_{re}, \Lambda_{im} \in \mathbb{R}^r$ and $w \in \mathbb{C}^r$ are training parameters within $\mathrm{DSS_{EXP}}$ and $\mathrm{DSS_{SOFTMAX}}$. Here, $(\Lambda_{re}, \Lambda_{im})$ and w are initialized with (M_{re}, M_{im}) and v obtained by Parameter Extraction, respectively. It is important to note that the dimension, previously denoted as N, is adjusted to r for the context of this initialization. All other parameters maintain their values as obtained from the Pre-Training phase, ensuring consistency in the model's initialization process.

VI. NUMERICAL EXPERIMENTS

To evaluate the proposed method, we employed tasks of LRA [8], which is available at https://github.com/google-research/long-range-arena. The benchmark includes sequence data ranging from 1,000 to 16,000 in length and evaluates the model's ability to capture long-range dependencies required for learning long sequences. In the text classification task, we classify movie reviews in the Internet Movie Database (IMDb) review dataset [42] as negative or positive. Tables 3 and 4 summarize the statistics of the text classification dataset, including the counts and lengths of the

TABLE 5. Accuracy of models through various training methods (Text classification, $\mathrm{DSS}_{\mathrm{EXP}}$, $H = 16$).

N		HiPPO	Random	Proposed Method
128	before	0.5000	0.4998	
	after	0.7997	0.7731	
64	before	0.4967	0.4982	
	after	0.8012	0.7968	
32	before	0.5008	0.4978	
	after	0.7990	0.8100	
16	before	0.4936	0.4970	0.8310
	after	**0.8310**	0.8071	0.8396
8	before	0.4996	0.5028	0.5011
	after	0.8182	0.8115	0.8376
4	before	0.4960	0.4964	0.5002
	after	0.8042	0.8155	**0.8418**

TABLE 6. Accuracy of models through various training methods (Text classification, $\mathrm{DSS}_{\mathrm{SOFTMAX}}$, $H = 16$).

N		HiPPO	Random	Proposed Method
128	before	0.4984	0.5000	0.8216
	after	**0.8216**	0.7540	0.8359
64	before	0.4993	0.5000	0.5013
	after	0.8158	0.7544	0.8382
32	before	0.4994	0.4999	0.5000
	after	0.8026	0.7496	0.8370
16	before	0.5011	0.4994	0.5005
	after	0.8190	0.7461	0.8402
8	before	0.5000	0.5072	0.5036
	after	0.8071	0.7722	**0.8412**
4	before	0.4940	0.5000	0.5012
	after	0.7948	0.7819	0.8377

raw data sequences. These sequences are truncated or padded as necessary to ensure consistent input lengths.

The experiments were conducted on a machine running Windows 10, equipped with 64 GB of memory and an 11th Gen Intel Core i9-11980HK CPU. The model training and evaluation code was implemented in Python using PyTorch 1.11.0 and TensorFlow 2.12.0, and executed with the NVIDIA RTX A3000 Laptop GPU.

A. COMPARISON WITH EXISTING TRAINING METHODS

Table 5 and Table 6 show the accuracy of models obtained through various training methods, using $\mathrm{DSS}_{\mathrm{EXP}}$ and $\mathrm{DSS}_{\mathrm{SOFTMAX}}$ respectively, where N denotes the dimension of the state vector of DSS. The number of DSS layers is 4 and the hidden size H is 16. The columns labeled "before" and "after" denote the accuracy of the model with the initial parameter values and the accuracy of the model after training from that initial state, respectively.

In the "HiPPO" column on the left, the Skew-HiPPO initialization explained in Subsection IV-B was used to initialize the state matrix A of each DSS. In the middle column "Random", the initial values of A were randomly sampled. For $\mathrm{DSS}_{\mathrm{EXP}}$, the real and imaginary parts of the diagonal elements of matrix A were sampled from $\mathcal{U}(-1, 0)$ and $\mathcal{N}(0, 1)$, respectively. For $\mathrm{DSS}_{\mathrm{SOFTMAX}}$, the real and imaginary parts of the diagonal elements were sampled from $\mathcal{N}(0, 1)$. Other parameters within DSS were randomly sampled for both "HiPPO" and "Random". The real and imaginary parts of each element in $w \in \mathbb{C}^N$ were sampled from $\mathcal{N}(0, 1)$.

For $\mathrm{DSS}_{\mathrm{SOFTMAX}}$, it has been reported in [9] that "HiPPO" using Skew-HiPPO initialization achieves higher accuracy after training compared to "Random" using randomly sampled initial values. The results in Table 6 are cosistent, where "HiPPO" achieves higher accuracy after training compared to "Random" for each N, while each accuracy before training is near. For $\mathrm{DSS}_{\mathrm{EXP}}$, the same trend was observed for almost all N as shown in Table 5.

The "Proposed Method" column on the right describes our approach, which uses a reduced SSM from the Pre-Trained models with $N = 16$ ($\mathrm{DSS}_{\mathrm{EXP}}$) and $N = 128$ ($\mathrm{DSS}_{\mathrm{SOFTMAX}}$), obtained through the balanced truncation method, to initialize Main Training. In Table 5, the "Proposed Method" entries for $N = 32$, $N = 64$, and $N = 128$ are blank, because the balanced truncation does not permit expanding the state dimension beyond the original size of $N = 16$.

Before Main Training, the accuracy of models using the "Proposed Method" is comparable to those using "Random" and "HiPPO" for each N excluding $N = 16$ for $\mathrm{DSS}_{\mathrm{EXP}}$ and $N = 128$ for $\mathrm{DSS}_{\mathrm{SOFTMAX}}$. However, after Main Training, the accuracy of "Proposed Method" exceeded that of "HiPPO" for each N. This result is noteworthy because the "Proposed Method" tends to outperform "HiPPO" after Main Training, despite having similar accuracies before the training.

The following points are particularly noteworthy.

- For $\mathrm{DSS}_{\mathrm{EXP}}$ shown in Table 5, the highest accuracy after Main Training with "Proposed Method" was 0.8418 at $N = 4$. Notably, this exceeded the accuracy after training with "HiPPO" at $N = 16$, which was 0.8310, despite having a smaller N while maintaining the same hidden size H.
- For $\mathrm{DSS}_{\mathrm{SOFTMAX}}$ shown in Table 6, the highest accuracy after training with "Proposed Method" was 0.8412 at $N = 8$. Notably, this exceeded the accuracy after training with "HiPPO" at $N = 128$, which was 0.8216, despite having a smaller N while maintaining the same hidden size H.

In summary, the initial parameters obtained by reducing Pre-Trained DSS of $(H, N) = (16, 16)$ for $\mathrm{DSS}_{\mathrm{EXP}}$ and $(H, N) = (16, 128)$ for $\mathrm{DSS}_{\mathrm{SOFTMAX}}$ appear to be effective in enhancing accuracy of the trained model compared to the initial parameters by the Skew-HiPPO initialization. Similar trends are observed in the ListOps task and text retrieval task of LRA, where our method enhanced the accuracy of the trained model. For detailed results, refer to Appendix C.

B. RELATIONSHIP BETWEEN ACCURACY OF PRE-TRAINED MODELS AND MODELS AFTER MAIN TRAINING

Table 7 shows the accuracy of models after Main Training when initialized with different Pre-Trained models. We obtained Pre-Trained models with DSS of $N = 128$

TABLE 7. Accuracy of models initialized using different Pre-Trained models ($\text{DSS}_{\text{SOFTMAX}}$, $H = 16$).

N	HiPPO	Proposed Method ($N = 128$)	Proposed Method ($N = 80$)
128	**0.8216**	0.8359	
80	0.8098	0.8390	**0.8343**
64	0.8158	0.8382	0.8224
32	0.8026	0.8370	0.8255
16	0.8190	0.8402	0.8294
8	0.8071	**0.8412**	0.8334
4	0.7948	0.8377	0.8317

and $N = 80$, and utilized the reduced models for improved initialization of Main Training. The accuracy of the models after main training is in columns "Proposed Method ($N = 128$)" and "Proposed Method ($N = 80$)". Both "Proposed Method ($N = 128$)" and "Proposed Method ($N = 80$)" followed the trend observed in Subsection VI-A, where "Proposed Method" achieved higher accuracy than "HiPPO" for each N.

The accuracy of the Pre-Trained models for $N = 128$ is 0.8216, which is higher than 0.8098 at $N = 80$. As for models after Main Training, the accuracy of "Proposed Method ($N = 128$)" surpasses that of "Proposed Method ($N = 80$)" for each N. This suggests that higher accuracy of the Pre-Trained model leads to higher accuracy of the model obtained through Main Training.

C. NON-TRIVIALITY OF THE OBTAINED RESULTS

The Hankel singular values illustrated in Fig. 5 highlight the non-triviality of the results presented in Tables 5 and 6 from a system-theoretic perspective. These values were derived from the SSM parameters (A, B, C) of the Pre-Trained model using $\text{DSS}_{\text{SOFTMAX}}$ with $N = 128$. Specifically, the Hankel singular values were computed for each SSM in every DSS layer. The detailed computational method is described in Appendix A.

As explained in Section II-B and Appendix A, the Hankel singular values can reveal the important directions in the state space from the controllability and observability perspective. That is, if the Hankel singular values are relatively large, the corresponding directions are relatively important. Notably, Fig. 5 shows that almost all directions in the $N = 128$ dimensional state space are important, because there are few significantly small Hankel sigular values. Therefore, reducing the dimensionality of the Pre-Trained model from $N = 128$ is expected to significantly deteriorate its performance. This expectation is consistent with the results shown in Table 6. In fact, the accuracy of the Pre-Trained model with $N = 128$ was 0.8216, but after reducing N to 8, the accuracy dropped to 0.5036. Nevertheless, after Main-Training, the accuracy of the reduced model improved to 0.8412. This improvement in accuracy is not predicted by the theoretical analysis of balanced truncation introduced in Appendix A and is a non-trivial result.

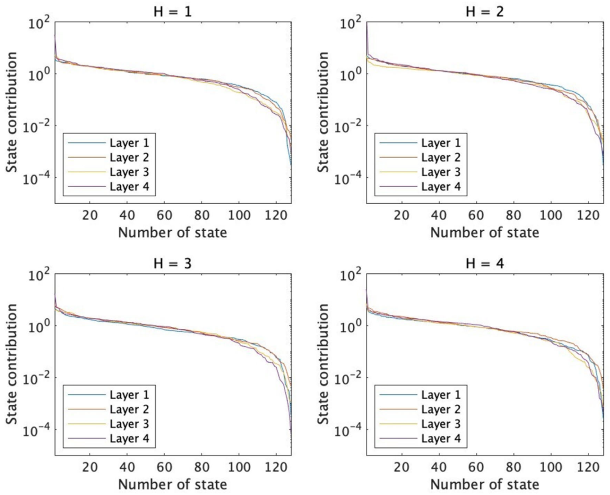

FIGURE 5. Hankel singular values obtained from each SSM for $\text{DSS}_{\text{SOFTMAX}}$ with $N = 128$. These SSMs were part of the Pre-Trained model initialized using the Skew-HiPPO with $N = 128$, as shown in Table 6. Although the hidden size was set to 16, we only presented the cases for $H = 1, 2, 3$, and 4 because the results for $H = 5, 6, \ldots, 16$ were almost identical.

VII. CONCLUSION

We developed a new model compression method specifically for S4 models with DSS layers, using the balanced truncation method [33]. This approach not only reduces the number of parameters but also enhances model performance. We proposed using the reduced model parameters obtained by the balanced truncation as initial parameters for the main training process. Our experiments demonstrated that the proposed method achieves superior accuracy on Long Range Arena (LRA) tasks compared to conventionally trained models using the Skew-HiPPO initialization, even with fewer parameters. Moreover, we observed a positive correlation between the accuracy of Pre-Trained models and their accuracy after Main Training.

The primary limitation of this study lies in the scope of tasks and datasets used for evaluation. While the LRA tasks provide a robust benchmark for long-range dependency modeling, further validation on diverse datasets and real-world applications is necessary. Additionally, the underlying principles of the proposed method remain unclear, which limits the understanding of why this approach is effective.

The following are interesting future directions:

- Future research should investigate the underlying principles of the proposed method, aiming to enhance the development of more effective training methods for deep learning models with DSS layers.
- Reference [23] has shown that combining various model compression methods can yield better results. Investigating whether combining our proposed model compression method based on the balanced truncation with other compression techniques can improve performance is an interesting and promising direction for future work.

- Expanding the scope of evaluation to include real-time deployment scenarios in EI applications will provide more comprehensive insights into the method's practical viability. This can help demonstrate how the reduced models can be effectively used in resource-constrained environments.
- As an extension of our study, applying the proposed compression techniques to Physics-Informed Neural Networks (PINNs) can be explored [43], [44]. This could help to improve the efficiency and performance of PINNs in modeling physical systems with limited computational resources.

APPENDIX A
DETAILS OF THE BALANCED TRUNCATION METHOD

As mentioned in Section II-B, the eigenvectors of the controllability and observability Gramians P and Q of SSM (1) provide important directions in the state space $\mathbb{C}^N$ from the perspectives of controllability and observability. Thus, we can adopt an approach that reduces dimensions along directions that are not significant. However, the eigenvectors do not coincide in general. This means that, in general, it is impossible to uniquely determine the directions to be ignored based solely on the information from the original controllability Gramian P and observability Gramian Q.

To overcome this problem, we apply a coordinate transformation $\bar{x}(t) = Tx(t)$ to SSM (1) to obtain new SSM (5). Then, the corresponding controllability and observability Gramians of SSM (5) become TPT^* and $(T^{-1})^*QT^{-1}$, respectively. Thus, if we can find $T \in \mathbb{C}^{N\times N}$ satisfying

$$TPT^* = (T^{-1})^*QT^{-1}, \tag{24}$$

the controllability and observability Gramians of the transformed SSM (5) will coincide, even if the original Gramians of SSM (1) do not.

To find T satisfying (24), we perform the eigenvalue decomposition of the symmetric positive definite matrix $P^{1/2}QP^{1/2}$ to obtain

$$P^{1/2}QP^{1/2} = U\Lambda U^*, \tag{25}$$

where $P^{1/2}$ is the square root matrix of P, U is a unitary matrix, and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix with positive diagonal elements satisfying $\lambda_1 \geq \cdots \geq \lambda_N$. Defining

$$T := \Lambda^{1/4}U^*P^{-1/2}, \tag{26}$$

we get

$$TPT^* = (T^{-1})^*QT^{-1} = \Lambda^{1/2} = \mathrm{diag}(\sigma_1, \ldots, \sigma_N).$$

We call $\sigma_i = \sqrt{\lambda_i}$ $(i = 1, \ldots, N)$ the Hankel singular values of SSM (1).

Thus, the balanced truncation method consists of the following procedure:

1) Compute the controllability Gramian P and the observability Gramian Q of SSM (1) by solving the Lyapunov equations (3) and (4), respectively.
2) Compute the square root matrix $P^{1/2}$.
3) Perform the eigenvalue decomposition using (25).
4) Determine the transformation matrix T using (26).
5) Compute (6), (7), and (8) using T, and define (A_r, B_r, C_r) of reduced SSM (2) as (9).

The reduced SSM (2) is preferable when it closely approximates the original large-scale SSM (1) in terms of the H^∞ norm of the difference in their transfer functions. In fact, the transfer functions G and G_r of original SSM (1) and reduced SSM (2) are defined by

$$G(s) := C(sI_N - A)^{-1}B, \quad G_r(s) := C_r(sI_r - A_r)^{-1}B_r,$$

respectively. The energy of the difference between original SSM (1) output y and reduced SSM (2) output y_r can be evaluated by

$$\int_0^\infty \|y(t) - y_r(t)\|^2 \mathrm{d}t \leq \|G - G_r\|_{H^\infty}^2 \int_0^\infty \|u(t)\|^2 \mathrm{d}t,$$

where $\|\cdot\|_{H^\infty}$ denotes the H^∞ norm. That is, if the input energy $\int_0^\infty \|u(t)\|^2 \mathrm{d}t$ and $\|G - G_r\|_{H^\infty}$ are sufficiently small, the output error energy $\int_0^\infty \|y(t) - y_r(t)\|^2 \mathrm{d}t$ is also sufficiently small. Moreover, if we use the balanced truncation method, $\|G - G_r\|_{H^\infty}$ is bounded by

$$\sigma_{r+1} \leq \|G - G_r\|_{H^\infty} \leq 2(\sigma_{r+1} + \cdots + \sigma_N),$$

assuming that $\sigma_{r+1} > \cdots > \sigma_N$. Thus, if the Hankel singular values $\sigma_{r+1}, \ldots, \sigma_N$ are small, then $\|G - G_r\|_{H^\infty}$ will also be small. The proofs for the above claims can be found in [38, Chapter 4].

APPENDIX B
DETAILS OF THE DEEP LEARNING MODEL

In the deep learning model employed in this study, residual connections [45] and normalization layers are positioned before and after the DSS layer. In residual connections, a path bypassing one or more layers is created, as illustrated in Fig. 6 and Fig. 7, and the output of the bypassed layer is added to it. The normalization layer can be placed before the DSS layer (Prenorm, Fig. 6) or after the residual connection (Postnorm, Fig. 7). In the case of prenorm, a normalization layer is also placed before the output layer. Normalization layers such as batch normalization [46] or layer normalization [47]

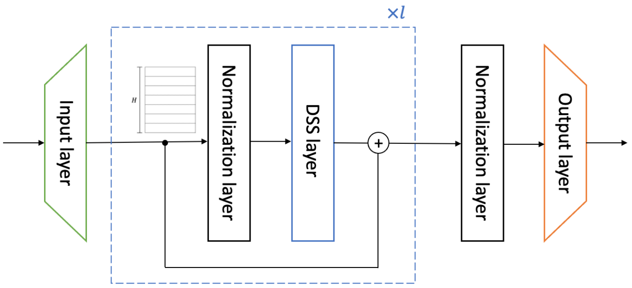

FIGURE 6. Deep learning model with DSS layers (Prenorm). The normalization layer is placed before the DSS layer and output layer.

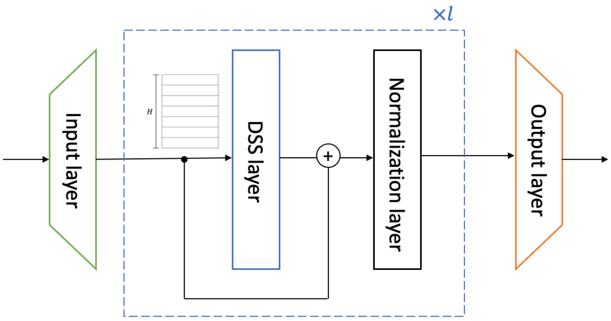

FIGURE 7. Deep learning model with DSS layers (Postnorm). The normalization layer is placed after the residual connection.

are used, which contributes to the stability and acceleration of training. Additionally, residual connections prevent the gradient vanishing and exploding problems.

APPENDIX C
OTHER RESULTS

In addition to the text classification task, explained in Section VI, we confirmed that our proposed method improves performance in the ListOps task and the text retrieval task of LRA [8], as shown in Table 8 and Table 9, respectively. Here, the number of DSS layers is 6 and the hidden size H is 16. In the ListOps task, a numerical expression structured with operators MAX, MEAN, MEDIAN, SUM_MOD and parentheses is the input, and its value is the output. For instance,

```
INPUT : [ MAX 1 2 MIN [ 3 4 ] MEDIAN [ 1 5 9 ] ]
OUTPUT : 5
```

The maximum length of input is 2000, and the output values range from 0 to 9. In the text retrieval task, we estimate the similarity between two papers and determine if there is a citation link. The length of each paper is 4000, and the total input length is 8000.

For $\text{DSS}_{\text{SOFTMAX}}$, the left column "HiPPO" using the Skew-HiPPO initialization achieved higher accuracy after training compared to the middle column "Random" using randomly sampled initial values. For DSS_{EXP}, the same trend was observed for almost all N as shown in Table 8 and Table 9.

TABLE 8. Accuracy of models through various training methods (Listops, DSS_{EXP}, $H = 16$).

N		HiPPO	Random	Proposed Method
64	before	0.0715	0.0810	0.5180
	after	**0.5180**	0.4020	0.5300
32	before	0.1705	0.1195	0.3610
	after	0.4920	0.4035	0.5205
16	before	0.0760	0.1780	0.1290
	after	0.4745	0.4001	**0.5390**
8	before	0.0715	0.0960	0.1825
	after	0.4025	0.4300	0.5250
4	before	0.0825	0.1210	0.1725
	after	0.4250	0.4440	0.5175

TABLE 9. Accuracy of models through various training methods (Text retrieval, DSS_{EXP}, $H = 16$).

N		HiPPO	Random	Proposed Method
64	before	0.4943	0.5068	0.8189
	after	0.8189	0.7830	0.8345
32	before	0.4937	0.4976	0.5807
	after	**0.8217**	0.7812	0.8302
16	before	0.5055	0.4932	0.5399
	after	0.8071	0.7907	0.8251
8	before	0.4939	0.4939	0.5064
	after	0.8212	0.7944	0.8270
4	before	0.4939	0.4939	0.5062
	after	0.8136	0.7893	**0.8313**

In the "Proposed Method" column on the right, Main Training was initialized using a reduced model obtained from Pre-Trained models of $(H, N) = (16, 64)$. The accuracy of models before Main Training with "Proposed Method" was comparable to that of "Random" and "HiPPO" for each N excluding $N = 64$. However, after the training, the accuracy of "Proposed Method" exceeded that of "HiPPO" for each N.

The following points are particularly noteworthy.

- For ListOps task, the highest accuracy after Main Training with "Proposed Method" was 0.5390 at $N = 16$. Notably, this exceeded the accuracy after training with "HiPPO" at $N = 64$, which was 0.5180, despite the smaller N and the same hidden size H.
- For text retrieval task, the accuracy after Main Training with "Proposed Method" was 0.8313 at $N = 4$, which exceeded the accuracy after training with "HiPPO" at $N = 64$ and $N = 32$, despite the smaller N and the same hidden size H.

Consequently, the initial parameters obtained by reducing Pre-Trained DSS of $(H, N) = (16, 64)$ appear to be effective in enhancing accuracy of the trained model compared to the initial parameters by the Skew-HiPPO initialization.

REFERENCES

[1] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philos. Trans. Roy. Soc. A*, vol. 379, no. 2194, 2021, Art. no. 20200209.

[2] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria, "A review of deep learning techniques for speech processing," *Inf. Fusion*, vol. 99, Nov. 2023, Art. no. 101869.

[3] A. K. Pandey and S. S. Roy, "Natural language generation using sequential models: A survey," *Neural Process. Lett.*, vol. 55, no. 6, pp. 7709–7742, Dec. 2023.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[5] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 9, pp. 11121–11128.

[6] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–32.

[7] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1474–1487.

[8] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena: A benchmark for efficient transformers," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–16.

[9] A. Gupta, A. Gu, and J. Berant, "Diagonal state spaces are as effective as structured state spaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 22982–22994.

[10] D. Y. Fu, T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Re, "Hungry hungry hippos: Towards language modeling with state space models," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023, pp. 1–27.

[11] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré, "Hyena hierarchy: Towards larger convolutional language models," 2023, *arXiv:2302.10866*.

[12] A. Gu, K. Goel, A. Gupta, and C. Ré, "On the parameterization and initialization of diagonal state space models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 35971–35983.

[13] E. Nguyen, K. Goel, A. Gu, G. Downs, P. Shah, T. Dao, S. Baccus, and C. Ré, "S4ND: Modeling images and videos as multidimensional signals with state spaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 2846–2861.

[14] J. T. H. Smith, A. Warrington, and S. Linderman, "Simplified state space layers for sequence modeling," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023, pp. 1–35.

[15] J. M. L. Alcaraz and N. Strodthoff, "Diffusion-based time series imputation and forecasting with structured state space models," *Trans. Mach. Learn. Res.*, pp. 1–36, Aug. 2023.

[16] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," 2023, *arXiv:2312.00752*.

[17] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[18] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[19] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artif. Intell. Rev.*, vol. 53, no. 7, pp. 5113–5155, Oct. 2020.

[20] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2449–2494, 4th Quart., 2022.

[21] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–37, Oct. 2021.

[22] N. Tekin, A. Aris, A. Acar, S. Uluagac, and V. C. Gungor, "A review of on-device machine learning for IoT: An energy perspective," *Ad Hoc Netw.*, vol. 153, Feb. 2024, Art. no. 103348.

[23] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[24] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 1–8.

[25] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Boca Raton, FL, USA: CRC Press, 2022, pp. 291–326.

[26] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.

[27] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[28] Z. Hao, J. Guo, K. Han, H. Hu, C. Xu, and Y. Wang, "Revisit the power of vanilla knowledge distillation: From small scale to large scale," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 1–14.

[29] T. Huang, Y. Zhang, M. Zheng, S. You, F. Wang, C. Qian, and C. Xu, "Knowledge diffusion for distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 1–18.

[30] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*. Philadelphia, PA, USA: SIAM, 2005.

[31] A. Astolfi, "Model reduction by moment matching for linear and nonlinear systems," *IEEE Trans. Autom. Control*, vol. 55, no. 10, pp. 2321–2336, Oct. 2010.

[32] S. Gugercin, A. C. Antoulas, and C. Beattie, "$\mathcal{H}_2$ model reduction for large-scale linear dynamical systems," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 609–638, 2008.

[33] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Trans. Autom. Control*, vol. AC-26, no. 1, pp. 17–32, Feb. 1981.

[34] K. Sato, "Riemannian optimal model reduction of linear port-Hamiltonian systems," *Automatica*, vol. 93, pp. 428–434, Jul. 2018.

[35] K. Sato, "Riemannian optimal model reduction of stable linear systems," *IEEE Access*, vol. 7, pp. 14689–14698, 2019.

[36] K. Sato, "Reduced model reconstruction method for stable positive network systems," *IEEE Trans. Autom. Control*, vol. 68, no. 9, pp. 5616–5623, Sep. 2023.

[37] S. Massaroli, M. Poli, D. Fu, H. Kumbong, R. Parnichkun, D. Romero, A. Timalsina, Q. McIntyre, B. Chen, and A. Rudra, "Laughing hyena distillery: Extracting compact recurrences from convolutions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 1–45.

[38] G. E. Dullerud and F. Paganini, *A Course in Robust Control Theory: A Convex Approach*. New York, NY, USA: Springer, 2000.

[39] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[40] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed., Cambridge, MA, USA: MIT Press, 2009.

[42] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Human Lang. Technol.*, 2011, pp. 142–150.

[43] J. Donnelly, A. Daneshkhah, and S. Abolfathi, "Physics-informed neural networks as surrogate models of hydrodynamic simulators," *Sci. Total Environ.*, vol. 912, Feb. 2024, Art. no. 168814.

[44] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[47] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

HARUKA EZOE received the bachelor's degree in engineering from The University of Tokyo, in 2024, where she is currently pursuing the master's degree. Her research interest includes data mining.

KAZUHIRO SATO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Kyoto University, Japan, in 2009, 2011, and 2014, respectively. From 2014 to 2017, he was a Postdoctoral Fellow at Kyoto University. From 2017 to 2019, he was an Assistant Professor at Kitami Institute of Technology. From 2019 to 2023, he was a Lecturer at the Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo. He is currently working as an Associate Professor with the Department of Mathematical Informatics. His research interests include mathematical control theory and optimization. He is a member of SICE.