Pruning State Space Models with Model Order Reduction for Efficient Raw Audio Classification

Matthias Bittner[a], Daniel Schnöll[a], Dominik Dallinger[a], Matthias Wess[a], Axel Jantsch[a]
[a]Christian Doppler Laboratory for Embedded Machine Learning, TU Wien, Austria, {firstname.lastname}@tuwien.ac.at

***Abstract*—Deep State Space Models (SSMs) have shown good performance on long-sequence classification tasks such as raw audio classification. Targeting edge devices it is crucial to further improve their inference efficiency. However, pruning techniques are not well explored for SSMs. We propose a layer-wise Model Order Reduction (MOR) technique based on balanced truncation combined with an iterative pruning algorithm to increase the efficiency of already trained SSM models, without the need for retraining. Specifically, we focus on S-Edge models, a class of hardware-friendly SSMs. Evaluated on the Google Speech Commands dataset we prune models ranging from 141k–8k in parameters and 94.9%–90.0% in test accuracy. Given an accuracy loss constraint of 0.5pp we are able to find models which reduce parameters by 36.1% for the biggest and 5.8% for the smallest model.**

***Index Terms*—Pruning, Model Order Reduction, Deep State Space Models, Raw Audio Classification**

I. Introduction

The amount of machine learning applications and the demand for deploying them to low-cost edge devices is constantly increasing. This also raises the need for methods that increase the Neural Networks (NNs) efficiency by reducing their memory, storage, and computational footprint.

In the early days of deep learning, Recurrent Neural Networks (RNNs) were the natural choice for modeling sequential data and architectures such as the Gated Recurrent Unit (GRU) [1] and the Long Short-Term Memory (LSTM) [2] have significantly improved the ability to learn long-term dependencies by addressing the vanishing gradient problem. However, scaling traditional RNNs to long sequence tasks is a challenge due to their inherent sequential nature [3]. The Transformer architecture improved the long sequence reasoning tasks with the parallelizable attention mechanism [4]. The attention mechanism mitigates the vanishing gradient problem by modeling interactions between any two time points with direct weights in the network. This comes with the drawback of taking a step back in efficiency. Computational and memory costs scale quadratically $O(T^2)$ with the sequence length T, compared to RNNs, with constant memory and, computational costs that scale linearly $O(T)$.

The recent advent of using deep structured State Space Models (SSMs) motivated by Linear Time Invariant (LTI) continuous-time SSMs with architectures like the S4 model [5], along with its variants (DSS, S4D, S5, LRU etc.) [6]–[9] has shown remarkable results on long-range reasoning tasks such as the Long Range Arena [10] and raw audio classification [11], where vanilla RNNs struggle. Efficiency-wise deep continuous-time SSMs overcome the $O(T^2)$ bottleneck of attention layers by materializing the discrete variant of the Linear Time Invariant (LTI) systems' differential equations, which is equivalent to an RNN at inference time [5]. Motivated by the strong performance of deep SSMs on long-sequence raw audio classification tasks, we explore if the mathematically well-defined LTI state space structure can be used to improve the efficiency of deep SSMs along with the following contributions:

Layerwise Model Order Reduction. We propose using Model Order Reduction (MOR) techniques stemming from linear control theory for increasing the efficiency in terms of parameters and number of operations for already trained complex-valued continuous-time diagonal SSMs. Specifically, we show how to balance, truncate, and rediagonalize continuous-time S-Edge SSM layers, a hardware-friendly SSM version (see Algorithm 1). For truncating the balanced systems we either use Direct Truncation (DT) or Singular Pertubation (SP). The layer-wise approach allows us to analyze the number of states that we are able to eliminate for individual layers. This gives an intuition about the pruning capacity (see Fig. 1 and Tab. II) and can be used to weight per-layer accuracy loss constraints for the model pruning.

Model Pruning Algorithm. Given an accuracy loss constraint, we provide a pruning algorithm, to iteratively reduce the parameters of S-Edge models (see Algorithm 2).

The rest of the paper is structured as follows: Section II reviews related work and background. Section III shows the design of S-Edge layers. Section IV presents the layer-wise MOR and the iterative model pruning algorithm for S-Edge models. Section V shows results for five different S-Edge model configurations based on the Google Speech Commands [11] dataset.

II. Background and Related Work

A. Deep State Space Models

One key difference between SSMs and standard RNNs is the linear recurrence in the hidden state, which allows for efficient and parallelizable computation of the hidden state recurrence based on associative scans [8]. Setting the foundation for good performance on long-range reasoning tasks, where RNNs struggle, Gu et al. proposed the HiPPO framework [12] for structured initialization of SSM layers. The resulting S4

This work is supported by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology, and Development, the Christian Doppler Research Association.

layer is modeled using multiple Single Input Single Output (SISO) systems. The SISO approach allows for training in the frequency domain [5]. To improve training and inference speed, several authors proposed diagonal approximations of the state transition matrix initializations, resulting in S4D [6], DSS [7] and the first diagonal Multi Input Multi Output (MIMO) version S5 [8].

B. Diagonalization and Balancing of Linear SSMs

State space transformations are a common practice in linear control theory to transform systems into representations that have the same input-output behavior but are, e.g., structured for easier derivation of the solution of the system (Diagonalization), or structured that states are sorted by equal controllability and observability (Balancing) [13]. Assuming a transformation matrix $\mathbf{T}$ we can apply the change of variables $\mathbf{x} = \mathbf{T}\mathbf{z}$ to the original system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u}$, $\mathbf{y} = \mathbf{C}\mathbf{x}$ in order to get to the transformed system $\dot{\mathbf{z}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{z} + \mathbf{T}^{-1}\mathbf{B}\mathbf{u}$, $\mathbf{y} = \mathbf{C}\mathbf{T}\mathbf{z}$. If the Matrix $\mathbf{A}$ has only distinct eigenvalues we are able to diagonalize the system and $\mathbf{T}_\mathrm{D}$ corresponds to the linear independent eigenvectors resulting from the eigendecomposition $\mathbf{A}\mathbf{T}_\mathrm{D} = \mathbf{T}_\mathrm{D}\mathbf{\Lambda}$, where $\mathbf{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$ is a diagonal matrix with the individual eigenvalues $\boldsymbol{\lambda}$.

If we set the focus on significant coherent structures influencing the system's input-output behavior we can apply a balancing transformation $\mathbf{T}_\mathrm{B}$. We outline the specific steps to get to $\mathbf{T}_\mathrm{B}$ as used in our study. Under the assumption that the system is stable ($\Re(\boldsymbol{\lambda}) < \mathbf{0}$) we first compute the controllability Gramian $\mathbf{P}$ and the observability Gramian $\mathbf{Q}$ of the original system, by solving the continuous Lyapunov equations $\mathbf{A}\mathbf{P}+\mathbf{P}\mathbf{A}^H=-\mathbf{B}\mathbf{B}^H$, and $\mathbf{A}^H\mathbf{Q}+\mathbf{Q}\mathbf{A}=-\mathbf{C}^H\mathbf{C}$. Second step, computes the square root matrix $\mathbf{P}^{1/2}$. Third step, solves the singular value decomposition $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^H = \mathbf{P}^{1/2}\mathbf{Q}\mathbf{P}^{1/2}$. Last step, computes the balancing transform with $\mathbf{T}_\mathrm{B}=\mathbf{P}^{1/2}\mathbf{U}\mathbf{\Sigma}^{-1/4}$. We refer the interested reader to the book of Antoulas [14] for a deeper understanding of MOR and the balancing transformation.

C. Pruning Techniques for Deep State Space Models

Forgione et al. [15] apply MOR for the Linear Recurrent Unit (LRU). Specifically, they apply the Balancing Transformation and they compare Direct Truncation (DT), Singular Pertubation (SP), and modal approximations. They evaluate their approach on an industrial vibration regression task. However, compared to our work, their model parameters are modeled in discrete-time, and they do not consider analyzing the pruning capacity on a per-layer base.

Ezoe et al. [16] propose model compression for diagonal S4 layers using balanced truncation. They propose a three-step approach of first pre-training the model at default size and then reducing the order of the individual SISO systems in the continuous-time parameter space. They keep the sampling rates Δ_i fixed for each individual SISO system, during MOR. As a third step, they perform the main training in the reduced-order parameter space. They evaluate on the Long Range Arena tasks [10]. They do not analyze the model performance without the re-training phase, and they do not analyze how strong one could prune already trained models.

Gwak et al. [17] propose LAST (Layer-Adaptive STate) pruning for deep SSMs. They provide a layer-wise modal-truncation approach by reducing the state size based on normalized H-Infinity scores. They analyze S4 and S5 layer types and perform pruning in the discrete-time domain. They mention that balanced truncation induced a non-diagonal matrix representation, which limits its applicability to the diagonal framework of current SSM structures. However, we show that it is possible to use balanced truncation with rediagonalization for pruning models without the need for retraining.

III. S-Edge a hardware friendly SSM Design

Within this work, we consider optimizing S-Edge [18], a hardware-friendly SSM layer, which extends the S5 model definition by dynamic input/output shapes and input/output bias units. An S-Edge layer is modeled as a complex-valued continuous-time system,

$$\begin{aligned}\dot{\widetilde{\mathbf{x}}}(t) &= \widetilde{\mathbf{A}}\widetilde{\mathbf{x}}(t) + \widetilde{\mathbf{B}}\mathbf{u}(t) + \widetilde{\mathbf{b}}\\ \mathbf{y}(t) &= \Re(\widetilde{\mathbf{C}}\widetilde{\mathbf{x}}(t) + \widetilde{\mathbf{c}}),\end{aligned} \tag{1}$$

with the complex-valued hidden state $\widetilde{\mathbf{x}}(t) \in \mathbb{C}^\mathrm{H}$, a diagonal state matrix $\widetilde{\mathbf{A}} = \mathrm{diag}(\widetilde{\boldsymbol{\lambda}}) \in \mathbb{C}^{\mathrm{H}\times\mathrm{H}}$ with the diagonal elements $\widetilde{\boldsymbol{\lambda}} \in \mathbb{C}^\mathrm{H}$ directly representing the complex eigenvalues. The input matrix $\widetilde{\mathbf{B}} \in \mathbb{C}^{\mathrm{H}\times\mathrm{Y}}$, the input bias $\widetilde{\mathbf{b}} \in \mathbb{C}^\mathrm{H}$, the output matrix $\widetilde{\mathbf{C}} \in \mathbb{C}^{\mathrm{O}\times\mathrm{H}}$, and the output bias $\widetilde{\mathbf{c}} \in \mathbb{C}^\mathrm{O}$ are also complex valued. The inputs $\mathbf{u}$ and outputs $\mathbf{y}$ of an individual SSM layer are considered to be real-valued.

During training and inference, it is necessary to map the continuous system to a discrete version (e.g., via Zero Order Hold (ZOH)). SSM layers do this with a learnable time-scale parameter $\Delta \in \mathbb{R}^\mathrm{H}$ per hidden dimension and a global sampling rate $\mathrm{T_s}$. By adjusting $\mathrm{T_s}$, the discrete SSM layers can operate at different effective sampling rates. Due to the diagonal implementation of the state transition matrix discretization can be performed efficiently on an elementwise basis,

$$\begin{aligned}\begin{bmatrix}\widetilde{\mathbf{B}}_\mathrm{d} & \widetilde{\mathbf{b}}_\mathrm{d}\end{bmatrix} &= \mathrm{diag}\left(\widetilde{\boldsymbol{\lambda}}^{-1} \circ (\widetilde{\boldsymbol{\lambda}}_\mathrm{d} - 1)\right)\begin{bmatrix}\widetilde{\mathbf{B}} & \widetilde{\mathbf{b}}\end{bmatrix},\\ \widetilde{\boldsymbol{\lambda}}_\mathrm{d} = e^{\boldsymbol{\Delta}\circ\widetilde{\boldsymbol{\lambda}}\mathrm{T_s}},&\quad \widetilde{\mathbf{C}}_\mathrm{d} = \widetilde{\mathbf{C}},\quad \widetilde{\mathbf{c}}_\mathrm{d} = \widetilde{\mathbf{c}}.\end{aligned} \tag{2}$$

The discrete recurrence of the S-Edge models forward path is then defined by,

$$\begin{aligned}\widetilde{\mathbf{x}}_k &= \widetilde{\boldsymbol{\lambda}}_d \circ \widetilde{\mathbf{x}}_{k-1} + \widetilde{\mathbf{B}}_\mathrm{d}\mathbf{u}_k + \widetilde{\mathbf{b}}_\mathrm{d},\\ \mathbf{y}_k &= \Re(\widetilde{\mathbf{C}}_\mathrm{d}\widetilde{\mathbf{x}}_k + \widetilde{\mathbf{c}}_\mathrm{d})\\ \text{Output} &= \mathrm{Skip}(\mathbf{u}_k) + \mathrm{LeakyReLU}(\mathbf{y}_k),\end{aligned} \tag{3}$$

where Skip $\in \mathbb{R}^{\mathrm{O}\times\mathrm{Y}}$ is a trainable real-valued matrix. Based on the discrete representation we get the following number of parameters for an S-Edge layer,

$$\text{Total Parameters} = \underbrace{2\mathrm{YH}}_{\widetilde{\mathbf{B}}} + \underbrace{2\mathrm{OH}}_{\widetilde{\mathbf{C}}} + \underbrace{\mathrm{YO}}_{\text{Skip}} + \underbrace{4\mathrm{H}+\mathrm{O}}_{\widetilde{\boldsymbol{\lambda}}_d,\ \text{biases}}. \tag{4}$$

This work focuses on decreasing the effective state dimension H in order to decrease parameters and operations.

Algorithm 1: MOR for an individual S-Edge Layer

Input: $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{b}}, \widetilde{\mathbf{C}}, \widetilde{\mathbf{c}}, \boldsymbol{\Delta})$ // Default layer param.
R // Reduced order
Γ // Approximation method (DT or SP)
Output: $(\widetilde{\boldsymbol{\lambda}}_{\mathrm{R}}, \widetilde{\mathbf{B}}_{\mathrm{R}}, \widetilde{\mathbf{b}}_{\mathrm{R}}, \widetilde{\mathbf{C}}_{\mathrm{R}}, \widetilde{\mathbf{c}}_{\mathrm{R}}, \boldsymbol{\Delta}_{\mathrm{R}})$ // Red. param.
// Compute *true* continuous-time input repres.
$\widetilde{\boldsymbol{\lambda}}_{\mathrm{tr}} = \Delta \circ \widetilde{\boldsymbol{\lambda}}$, $\widetilde{\mathbf{B}}_{\mathrm{tr}} = \Delta \circ \widetilde{\mathbf{B}}$, $\widetilde{\mathbf{b}}_{\mathrm{tr}} = \boldsymbol{\Delta} \circ \widetilde{\mathbf{b}}$
$\boldsymbol{\Delta}_{\mathrm{R}} = \mathbf{1}$ // Reset time scale parameter.
$(\widetilde{\mathbf{A}}_{\mathrm{B}}, \widetilde{\mathbf{B}}_{\mathrm{B}}, \widetilde{\mathbf{b}}_{\mathrm{B}}, \widetilde{\mathbf{C}}_{\mathrm{B}}, \widetilde{\mathbf{c}}_{\mathrm{B}})$ = Balance(diag($\widetilde{\boldsymbol{\lambda}}_{\mathrm{tr}}$), $\widetilde{\mathbf{B}}_{\mathrm{tr}}$, $\widetilde{\mathbf{b}}_{\mathrm{tr}}$, $\widetilde{\mathbf{C}}$, $\widetilde{\mathbf{c}}$)
$(\widetilde{\mathbf{A}}_{\mathrm{T}}, \widetilde{\mathbf{B}}_{\mathrm{T}}, \widetilde{\mathbf{b}}_{\mathrm{T}}, \widetilde{\mathbf{C}}_{\mathrm{T}}, \widetilde{\mathbf{c}}_{\mathrm{T}})$ = Ord.-Red.(R, Γ $\widetilde{\mathbf{A}}_{\mathrm{B}}, \widetilde{\mathbf{B}}_{\mathrm{B}}, \widetilde{\mathbf{b}}_{\mathrm{B}}, \widetilde{\mathbf{C}}_{\mathrm{B}}, \widetilde{\mathbf{c}}_{\mathrm{B}}$)
$(\widetilde{\boldsymbol{\lambda}}_{\mathrm{R}}, \widetilde{\mathbf{B}}_{\mathrm{R}}, \widetilde{\mathbf{b}}_{\mathrm{R}}, \widetilde{\mathbf{C}}_{\mathrm{R}}, \widetilde{\mathbf{c}}_{\mathrm{R}})$ = Rediagonalize($\widetilde{\mathbf{A}}_{\mathrm{T}}, \widetilde{\mathbf{B}}_{\mathrm{T}}, \widetilde{\mathbf{b}}_{\mathrm{T}}, \widetilde{\mathbf{C}}_{\mathrm{T}}, \widetilde{\mathbf{c}}_{\mathrm{T}}$)

IV. Model Order Reduction for Deep Diagonal State Space Models

Within this work, we investigate how to apply MOR techniques, stemming from control theory, in order to decrease the number of parameters and operations, for already trained SSM networks. Specifically, we focus on S-Edge models.

A. Model Order Reduction for Diagonal S-Edge Layers

An important step for successful and correct MOR of continuous time MIMO SSM variants, such as S5 [8] and S-Edge [18], which utilize an additional time scale parameter vector $\boldsymbol{\Delta} \in \mathbb{R}^{\mathrm{H}}$ is to correctly represent the *true* continuous time dynamics. The time scale vector is originally motivated to be a learnable sampling-rate scaling parameter for each state dimension and is normally evaluated during the discretization step. However, when applying the balancing transformation for MOR, it is crucial to view $\boldsymbol{\Delta}$ as a scaling in the continuous domain and to balance the *true* continuous-time dynamical system. Otherwise, the continuous and discrete systems simply do not represent the same dynamic behavior (also discussed by Bonassi et al. [19]). Now viewing $\boldsymbol{\Delta}$ as a scaling in continuous time for the S-Edge layer results in the *true* continuous diagonal eigenvalues $\widetilde{\boldsymbol{\lambda}}_{\mathrm{tr}} = \boldsymbol{\Delta} \circ \widetilde{\boldsymbol{\lambda}}$, the *true* continuous input matrix $\widetilde{\mathbf{B}}_{\mathrm{tr}} = \boldsymbol{\Delta} \circ \widetilde{\mathbf{B}}$ and the *true* input bias $\widetilde{\mathbf{b}}_{\mathrm{tr}} = \boldsymbol{\Delta} \circ \widetilde{\mathbf{b}}$.

Applying the change of variables $\widetilde{\mathbf{x}} = \mathbf{T}\widetilde{\mathbf{z}}$ to the S-Edge continuous-time representation (Eq. 1) given a transformation $\mathbf{T}$ (either balancing $\mathbf{T}_{\mathrm{B}}$ or diagonalization $\mathbf{T}_{\mathrm{D}}$ in our approach) we get the following transformed representation

$$\begin{aligned} \dot{\widetilde{\mathbf{z}}}(t) &= \mathbf{T}^{-1}\widetilde{\mathbf{A}}\mathbf{T}\widetilde{\mathbf{z}}(t) + \mathbf{T}^{-1}\widetilde{\mathbf{B}}\mathbf{u}(t) + \mathbf{T}^{-1}\widetilde{\mathbf{b}} \\ \mathbf{y}(t) &= \Re(\widetilde{\mathbf{C}}\mathbf{T}\widetilde{\mathbf{z}}(t) + \widetilde{\mathbf{c}}), \end{aligned} \tag{5}$$

where we see how to transform the state matrix, the input matrix, the input bias and the output matrix.

Algorithm 1 highlights our proposed MOR reduction pipeline for getting from the default diagonal S-Edge layer parameters with order H to the reduced system with order R. Balancing the *true* continuous-time parameters with $\mathbf{T}_{\mathrm{B}}$ (how to derive $\mathbf{T}_{\mathrm{B}}$ see Sec. II-B), we observe the balanced system matrices, $\widetilde{\mathbf{A}}_{\mathrm{B}} = \mathbf{T}_{\mathrm{B}}^{-1}\mathrm{diag}(\widetilde{\boldsymbol{\lambda}}_{tr})\mathbf{T}_{\mathrm{B}}$, $\widetilde{\mathbf{B}}_{\mathrm{B}} = \mathbf{T}_{\mathrm{B}}^{-1}\widetilde{\mathbf{B}}_{tr}$, $\widetilde{\mathbf{b}}_{\mathrm{B}} = \mathbf{T}_{\mathrm{B}}^{-1}\widetilde{\mathbf{b}}_{tr}$, $\widetilde{\mathbf{C}}_{\mathrm{B}} = \widetilde{\mathbf{C}}\mathbf{T}_{\mathrm{B}}$, $\widetilde{\mathbf{c}}_{\mathrm{B}} = \widetilde{\mathbf{c}}$. This balanced representation is already sorted (in terms of controllability and observability) in the state dimension and allows to partition and truncate based on a given reduced order R. If we now partition the balanced S-Edge layer parameters ($\widetilde{\mathbf{A}}_{\mathrm{B}}$, $\widetilde{\mathbf{B}}_{\mathrm{B}}$, $\widetilde{\mathbf{b}}_{\mathrm{B}}$, $\widetilde{\mathbf{C}}_{\mathrm{B}}$, $\widetilde{\mathbf{c}}_{\mathrm{B}}$), formally we get,

$$\begin{aligned} \begin{bmatrix} \dot{\widetilde{\mathbf{z}}}_1 \\ \dot{\widetilde{\mathbf{z}}}_2 \end{bmatrix} &= \begin{bmatrix} \widetilde{\mathbf{A}}_{11} & \widetilde{\mathbf{A}}_{12} \\ \widetilde{\mathbf{A}}_{21} & \widetilde{\mathbf{A}}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{z}}_1 \\ \widetilde{\mathbf{z}}_2 \end{bmatrix} + \begin{bmatrix} \widetilde{\mathbf{B}}_1 \\ \widetilde{\mathbf{B}}_2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} \widetilde{\mathbf{b}}_1 \\ \widetilde{\mathbf{b}}_2 \end{bmatrix} \\ \mathbf{y} &= \Re\left(\begin{bmatrix} \widetilde{\mathbf{C}}_1 & \widetilde{\mathbf{C}}_2 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{z}}_1 \\ \widetilde{\mathbf{z}}_2 \end{bmatrix} + \widetilde{\mathbf{c}} \right), \end{aligned} \tag{6}$$

MOR literature from control theory suggests two different methods for approximating the reduced order system based on the balanced realization, Direct Truncation (DT) [20] and Singular Pertubation (SP) [21].

Using DT we approximate the system by $\widetilde{\mathbf{A}}_{\mathrm{T}} = \widetilde{\mathbf{A}}_{11} \in \mathbb{C}^{\mathrm{R}\times\mathrm{R}}$, $\widetilde{\mathbf{B}}_{\mathrm{T}} = \widetilde{\mathbf{B}}_1 \in \mathbb{C}^{\mathrm{R}\times\mathrm{Y}}$, $\widetilde{\mathbf{b}}_{\mathrm{T}} = \widetilde{\mathbf{b}}_1 \in \mathbb{C}^{\mathrm{R}}$ $\widetilde{\mathbf{C}}_{\mathrm{T}} = \widetilde{\mathbf{C}}_1 \in \mathbb{C}^{\mathrm{O}\times\mathrm{R}}$ and $\widetilde{\mathbf{c}}_{\mathrm{T}} = \widetilde{\mathbf{c}} \in \mathbb{C}^{\mathrm{O}}$. While DT is a good approximation for the balanced system in the high-frequency domain, it may suffer in correctly reflecting the steady state behavior. To better match the low-frequency domain we can apply the SP approximation [21]. For SP we consider the dynamics of $\widetilde{\mathbf{z}}_2$ fast and stable, which can be approximated by setting $\dot{\widetilde{\mathbf{z}}}_2 = 0$. Calculating the resulting $\widetilde{\mathbf{z}}_2$, and reinserting into the equation of $\dot{\widetilde{\mathbf{z}}}_1$, allows to reformulate the SP based reduced order S-Edge layers dynamics by,

$$\begin{aligned} \dot{\widetilde{\mathbf{z}}}_1 &= \widetilde{\mathbf{A}}_{\mathrm{T}}\widetilde{\mathbf{z}}_1 + \widetilde{\mathbf{B}}_{\mathrm{T}}\mathbf{u} + \widetilde{\mathbf{b}}_{\mathrm{T}} \\ \mathbf{y} &= \Re(\widetilde{\mathbf{C}}_{\mathrm{T}}\widetilde{\mathbf{z}}_1 + \widetilde{\mathbf{D}}_{\mathrm{T}}\mathbf{u} + \widetilde{\mathbf{c}}_{\mathrm{T}}), \end{aligned} \tag{7}$$

where

$$\begin{aligned} \widetilde{\mathbf{A}}_{\mathrm{T}} &= \widetilde{\mathbf{A}}_{11} - \widetilde{\mathbf{A}}_{12}\widetilde{\mathbf{A}}_{22}^{-1}\widetilde{\mathbf{A}}_{21},\ \widetilde{\mathbf{B}}_{\mathrm{T}} = \widetilde{\mathbf{B}}_1 - \widetilde{\mathbf{A}}_{12}\widetilde{\mathbf{A}}_{22}^{-1}\widetilde{\mathbf{B}}_2 \\ \widetilde{\mathbf{b}}_{\mathrm{T}} &= \widetilde{\mathbf{b}}_1 - \widetilde{\mathbf{A}}_{12}\widetilde{\mathbf{A}}_{22}^{-1}\widetilde{\mathbf{b}}_2,\ \widetilde{\mathbf{C}}_{\mathrm{T}} = \widetilde{\mathbf{C}}_1 - \widetilde{\mathbf{C}}_2\widetilde{\mathbf{A}}_{22}^{-1}\widetilde{\mathbf{A}}_{21} \\ \widetilde{\mathbf{c}}_{\mathrm{T}} &= \widetilde{\mathbf{c}} - \widetilde{\mathbf{C}}_2\widetilde{\mathbf{A}}_{22}^{-1}\widetilde{\mathbf{b}}_2,\ \widetilde{\mathbf{D}}_{\mathrm{T}} = -\widetilde{\mathbf{C}}_2\widetilde{\mathbf{A}}_{22}^{-1}\widetilde{\mathbf{B}}_2. \end{aligned} \tag{8}$$

We see that the SP approximation, adds a Feed-Trough term $\widetilde{\mathbf{D}}_{\mathrm{T}}$ to the reduced-order parametrization. S-Edge by default neglects this term for parameter efficiency. We either have to add this additional Feed-Trough Matrix to the representation, which adds additional parameters or we neglect it.

Since the balancing transform results in a non-diagonal form, we have to rediagonalize the truncated system ($\widetilde{\mathbf{A}}_{\mathrm{T}}$, $\widetilde{\mathbf{B}}_{\mathrm{T}}$, $\widetilde{\mathbf{b}}_{\mathrm{T}}$, $\widetilde{\mathbf{C}}_{\mathrm{T}}$, $\widetilde{\mathbf{c}}_{\mathrm{T}}$), with its diagonalization transform $\mathbf{T}_{\mathrm{D}}$ (see Sec. II-B how to derive) to get the final diagonal reduced order model parameters ($\widetilde{\boldsymbol{\lambda}}_{\mathrm{R}}, \widetilde{\mathbf{B}}_{\mathrm{R}}, \widetilde{\mathbf{b}}_{\mathrm{R}}, \widetilde{\mathbf{C}}_{\mathrm{R}}, \widetilde{\mathbf{c}}_{\mathrm{R}}$) fitting the diagonal SSM framework. The time-scale parameter is set to $\boldsymbol{\Delta}_{\mathrm{R}} = \mathbf{1}$ to match the original discrete dynamics.

B. Model Pruning Algorithm

Given a trained S-Edge model with L layers, we ask ourselves the following question: *How strong are we able to reduce the number of parameters and operations given a global accuracy loss constraint δ?*

The proposed iterative pruning strategy is depicted by Algorithm 2. The per-layer accuracy loss constraint ϵ_l can either be set in a *naive* way by assigning $\epsilon_l = \delta/L$, or we *weight* the constraints by incorporating the information about the number of states which may be eliminated based on a layer-wise analysis (see Sec. V-A). Considering n_l as

Algorithm 2: Pruning S-Edge Models given an accuracy loss constraint

Input: δ // Accuracy loss constraint
N // Number of iterations
Γ // Approximation method (DT or SP)
$(\widetilde{\mathbf{A}}_{l:L}, \widetilde{\mathbf{B}}_{l:L}, \widetilde{\mathbf{b}}_{l:L}, \widetilde{\mathbf{C}}_{l:L}, \widetilde{\mathbf{c}}_{l:L})$ // S-Edge model parameters
Output: // Reduced model parameters
$(\widetilde{\lambda}_{\mathrm{R},l:L}, \widetilde{\mathbf{B}}_{\mathrm{R},l:L}, \widetilde{\mathbf{b}}_{\mathrm{R},l:L}, \widetilde{\mathbf{C}}_{\mathrm{R},l:L}, \widetilde{\mathbf{c}}_{\mathrm{R},l:L})$
$\mathrm{R}_{l:L} = \mathrm{H}_{l:L}$ // Init. red. orders with default values
Acc_d // Evaluate default accuracy
$\Delta\mathrm{A}_{\mathrm{last}} = 0$ // Init. last valid Acc. loss
for $i = 1$ *to* N **do**
  $\delta_i = i\delta/N$ // Update acc. constraint for iteration
  $\epsilon_{l:L} = f(\delta_i, L)$ // Update per layer acc. loss constr.
  for $l = 1$ *to* L **do**
    $\Delta\mathrm{A} = \Delta\mathrm{A}_{\mathrm{last}}$ // Init with last valid acc. loss
    $\mathrm{A}_{\mathrm{thresh.}} = \sum_{i=1}^{l} \epsilon_i$ // Set new threshold
    while $\Delta\mathrm{A} < \mathrm{A}_{\mathrm{thresh.}}$ **do**
      $\Delta\mathrm{A}_{\mathrm{last}} = \Delta\mathrm{A}$ // Remember last valid acc.
      $R_l = R_l - 1$ // Reduce order
      $\mathrm{MOR}(\mathrm{R}_{l:L}, \Gamma)$ // Perform MOR reduction
      Acc // Evaluate model accuracy
      $\Delta\mathrm{A} = \mathrm{Acc}_d - \mathrm{Acc}$ // Compute accuracy loss
    end
    $R_l = R_l + 1$ // Restore last valid order
  end
end

the number of prunable states not hurting the accuracy loss constraint (see Tab. II for specific results) for an individual layer l we perform a weighting $\epsilon_l = \frac{n_l}{\sum_{l=1}^{L} n_l}\delta$. If n_l is equal for the individual layers, the layerwise accuracy constraint ϵ_l defaults to the naive way.

V. Experimental Results

We evaluate the proposed MOR and pruning approach using the Google Speech Command [11] dataset, sampled with 16kHz and consisting of 35 classes. Specifically, we focus on the raw autoregressive audio-waveform classification task. We analyze five S-Edge model configurations spanning a default range in parameters 141k–8k and accuracy 95%–90% (see Tab. I–IV).

A. Layer-wise Model Order Reduction Effects

First, we analyze how strongly we are able to reduce the order of individual layers and compare the effect of the two proposed model order reduction approaches DT and SP. We investigate each layer separately, by iteratively reducing the order of a single layer while fixing the other layers to their default order. Fig. 1 highlights the evolution of the test accuracy over the per layer pruning ratios for three S-Edge model configurations (Full, M, Tiny) when approximating with

TABLE I
S-Edge Model Configurations.

	Input dim. (Y)	Output dim. (O)	State dim. (H)	Param.	Flops
Full	[1, 96, 96, 96, 96, 96]	[96, 96, 96, 96, 96, 96]	[64, 64, 64, 64, 64, 64]	141k	4400M
L	[1, 24, 32, 40, 48, 56]	[24, 32, 40, 48, 56, 64]	[96, 80, 64, 48, 32, 16]	56k	1715M
M	[1, 24, 44]	[24, 44, 64]	[96, 64, 32]	28k	805M
S	[1, 8, 16, 32, 42, 54]	[8, 16, 32, 42, 54, 64]	[32, 24, 16, 14, 12, 8]	20k	567M
Tiny	[1, 8, 32]	[8, 32, 64]	[32,16,8]	8k	192M

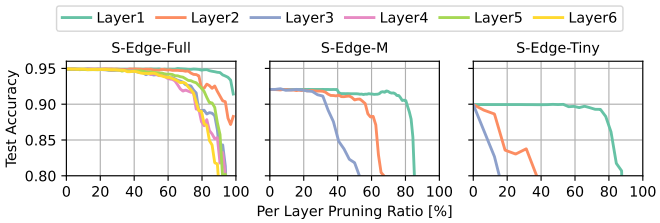

Fig. 1. Layerwise pruning effects when iteratively reducing the model order of a single layer and fixing the other layers to their default order for S-Edge models (based on SP approximation).

TABLE II
Layer-wise Pruning Results for Accuracy Loss Constraints δ.

	δ [pp]	Number of eliminated states n_l per Layer l using either DT/SP						
		l=1	l=2	l=3	l=4	l=5	l=6	Sum
Full	1	53/60	52/50	36/37	27/37	42/42	39/36	249/**262**
Full	2	59/62	53/50	45/44	35/41	47/49	43/45	282/**291**
L	1	82/80	63/64	38/35	22/24	12/12	2/3	**219**/218
L	2	84/86	69/68	43/41	24/24	14/15	3/4	237/**238**
M	1	37/73	30/28	10/10	-	-	-	77/**111**
M	2	50/77	35/36	12/10	-	-	-	97/**123**
S	1	17/25	8/10	4/6	2/2	1/1	0/0	32/**44**
S	2	21/29	10/12	7/13	3/4	1/1	0/0	42/**59**
Tiny	1	14/24	0/1	0/0	-	-	-	14/**25**
Tiny	2	14/25	1/2	0/0	-	-	-	15/**27**

SP. This illustration shows that individual layers can be pruned more or less. Tab. II provides additional results evaluating the number of states that we were able to eliminate given two different accuracy loss constraints. Comparing the number of prunable states based on the two approximation approaches (DT or SP) shows that we are able to eliminate more states when approximating with SP (indicated by the fictive sum over all layers). Even though we decided to neglect the Feed-Trough term $\widetilde{\mathbf{D}}_{\mathrm{T}}$ for keeping parameter reduction as high as possible.

B. Model Pruning Results

We evaluate the proposed pruning algorithm for five S-Edge model configurations (Full, L, M, S, Tiny) and analyze the reached reduction in parameters and operations with an accuracy loss constraint of $\delta = 2.0$pp. Setting the number of iterations to $N = 4$ allows a drop in accuracy per iteration of 0.5pp. Our approach allows to choose between the two approximation methods (either DT or SP), and the way we set the per-layer accuracy loss thresholds ϵ_l (either *naive*

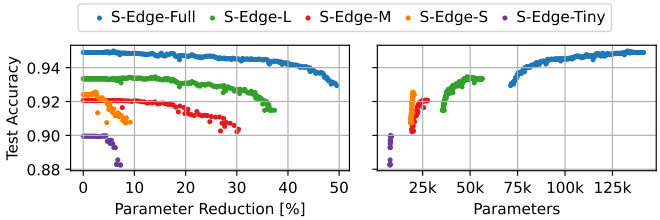

Fig. 2. Pruning evolution given a accuracy loss constraint of $\delta = 2$pp and splitting the search into four iterations $N = 4$. Results are based on using SP approximation and *weighted* per-layer accuracy loss thresholds.

TABLE III
MODEL PRUNING RESULTS FOR S-EDGE MODELS WHILE MAINTAINING AN ACCURACY LOSS LOWER 0.5pp.

Parameter red. ΔP[%], Flops red. ΔF[%]

S-Edge	Default Acc.[%]	MOR	Per layer Acc. thresholds Weighted ΔP	Weighted ΔF	Naive ΔP	Naive ΔF
Full	94.9	SP	**36.1**	**37.1**	29.8	30.6
L	93.3	SP	**28.2**	**29.6**	28.0	29.4
M	92.0	SP	19.7	21.6	**22.9**	**25.1**
S	92.4	SP	4.5	5.2	**5.4**	**6.2**
Tiny	90.0	SP	5.8	8.1	5.8	8.1
Full	94.9	DT	32.0	32.9	32.3	33.3
L	93.3	DT	23.5	24.7	27.1	28.5
M	92.0	DT	21.9	24.0	22.4	24.6
S	92.4	DT	3.4	3.9	4.8	5.5
Tiny	90.0	DT	3.1	4.4	3.1	4.4

TABLE IV
MODEL PRUNING RESULTS FOR S-EDGE MODELS WHILE MAINTAINING AN ACCURACY LOSS LOWER 2pp.

Parameter red. ΔP[%], Flops red. ΔF[%]

S-Edge	Default Acc.[%]	MOR	Per layer Acc. thresholds Weighted ΔP	Weighted ΔF	Naive ΔP	Naive ΔF
Full	94.9	SP	**49.5**	**50.9**	45.1	46.6
L	93.3	SP	**37.4**	**39.3**	33.0	34.7
M	92.1	SP	30.3	33.3	26.5	29.2
S	92.4	SP	**9.2**	**10.5**	7.1	8.1
Tiny	90.0	SP	6.5	9.2	**7.0**	**9.8**
Full	94.9	DT	44.9	46.1	43.5	44.7
L	93.3	DT	36.6	38.4	31.8	33.4
M	92.1	DT	**30.9**	**34.0**	27.1	29.8
S	92.4	DT	6.3	7.3	6.5	7.4
Tiny	90.0	DT	4.4	6.2	4.4	6.2

or *weighted*). Fig. 2 shows the evolution of the parameter reduction during the iterative pruning algorithm for all S-Edge configurations when applying SP approximation and *weighted* per layer accuracy loss thresholds.

Tab. III and IV shows the reached reduction in parameters and operations not hurting a 0.5pp and 2pp accuracy drop. We find that using SP during model order reduction performs better in most cases. Combined with weighted per-layer accuracy loss constraints we achieve the highest reduction in parameters for the two largest models (Full and L).

VI. CONCLUSION

We show that pruning complex-valued diagonal MIMO SSM layers in the continuous-time domain works through balancing, truncation, and rediagonalization. Balancing the system sorts states by input/output importance, without a loss in accuracy or computational overhead. Motivated by control theory, we removed weakly influential dynamics and validated the assumption that iterative state truncation gradually decreases prediction accuracy. Using weighted per-layer accuracy loss thresholds improved the iterative pruning strategy going from 29.8% (naive) to 36.1% (weighted) parameter reduction for the largest model with an accuracy drop smaller 0.5pp. While we encourage further research in model pruning algorithms, we demonstrate its applicability to SSM configurations ranging from 141k to 8k parameters and reach parameter reductions of 49.5% for the largest model and 7.0% for the smallest model not hurting a 2pp accuracy loss constraint.

REFERENCES

[1] K. Cho, B. v. Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: encoder–decoder approaches," *Proceedings Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[3] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *CoRR*, vol. abs/1610.10099, 2016.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[5] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *The International Conference on Learning Representations*, 2022.

[6] A. Gu, A. Gupta, K. Goel, and C. Ré, "On the parameterization and initialization of diagonal state space models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[7] A. Gupta, A. Gu, and J. Berant, "Diagonal state spaces are as effective as structured state spaces," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[8] J. T. Smith, A. Warrington, and S. Linderman, "Simplified state space layers for sequence modeling," in *The Eleventh International Conference on Learning Representations*, 2023.

[9] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De, "Resurrecting recurrent neural networks for long sequences," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, 2023.

[10] 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 *International Conference on Learning Representations*, 2021.

[11] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[12] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Re, "How to train your HIPPO: State space models with generalized orthogonal basis projections," in *International Conference on Learning Representations*, 2023.

[13] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[14] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*. USA: Society for Industrial and Applied Mathematics, 2005.

[15] M. Forgione, M. Mejari, and D. Piga, "Model order reduction of deep structured state-space models: A system-theoretic approach," *CoRR*, vol. abs/2403.14833, 2024.

[16] H. Ezoe and K. Sato, "Model compression method for S4 with diagonal state space layers using balanced truncation," *IEEE Access*, vol. 12, pp. 116 415–116 427, 2024.

[17] M. Gwak, S. Moon, J. Ko, and P. Park, "Layer-adaptive state pruning for deep state space models," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[18] M. Bittner, D. Schnöll, M. Wess, and A. Jantsch, "Efficient and interpretable raw audio classification with diagonal state space models," *Machine Learning*, vol. 114, no. 8, p. 175, Jun 2025. [Online]. Available: https://doi.org/10.1007/s10994-025-06807-z

[19] F. Bonassi, C. Andersson, P. Mattsson, and T. B. Schön, "Structured state-space models are deep wiener models," *IFAC-PapersOnLine*, vol. 58, no. 15, pp. 247–252, 2024, iFAC Symposium on System Identification SYSID 2024.

[20] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, vol. 26, no. 1, pp. 17–32, 1981.

[21] Y. Liu and B. Anderson, "Singular perturbation approximation of balanced systems," in *Proceedings of the 28th IEEE Conference on Decision and Control,*, 1989, pp. 1355–1360 vol.2.