2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

COM-PACT: COMponent-Aware Pruning for Accelerated Control Tasks in Latent Space Models

Ganesh Sundaram*[1], Jonas Ulmen*[1], Amjad Haider*[1] and Daniel Görges*[1]

Abstract— **The rapid expansion of resource-constrained mobile platforms, such as mobile robots, wearable systems, and Internet-of-Things devices, has heightened the need for computationally efficient neural network controllers (NNCs) that can operate within strict hardware limitations. Although deep neural networks (DNNs) achieve high performance in control applications, their considerable computational complexity and memory demands hinder practical deployment on edge devices. This study presents a comprehensive model compression methodology that employs component-aware structured pruning to determine the optimal pruning magnitude for each group, thereby balancing compression and stability for NNC deployment. The proposed approach is rigorously evaluated on Temporal Difference Model Predictive Control (TD-MPC), a leading model-based reinforcement learning algorithm, with systematic integration of mathematical stability guarantees, specifically Lyapunov criteria. The principal contribution is a principled framework for identifying the theoretical limits of model compression while preserving controller stability. Experimental results indicate that the methodology effectively reduces model complexity while maintaining essential control performance and stability. Additionally, the approach defines a quantitative boundary for safe compression ratios, enabling practitioners to systematically determine the maximum permissible model reduction before compromising critical stability properties, thus supporting the reliable deployment of compressed NNCs in resource-limited environments.**

I. INTRODUCTION

The integration of neural networks into control systems marks a significant departure from classical approaches, providing enhanced adaptability and performance, particularly for complex nonlinear systems. NNCs excel at learning from high-dimensional observational data, such as raw pixels, and can compress this information into effective latent space representations. Despite these advantages, a fundamental challenge emerges when deploying these controllers in real-world settings. The substantial computational and memory requirements of NNCs often conflict with the strict resource limitations of embedded hardware in automotive systems, robotics, and Internet of Things (IoT) devices. This disparity between capability and practical deployability constitutes a significant obstacle to system integration.

Model compression techniques, particularly pruning, are critical for addressing the implementation gap. While these methods have achieved success in fields such as computer vision, their application to neural network controllers introduces distinct and significant challenges. There is a fundamental misalignment between the goals of conventional compression and the requirements of control theory. For predictive models, such as image classifiers, compression primarily threatens accuracy. In contrast, compressing a controller can compromise the stability and safety of a physical system, potentially resulting in severe real-world consequences. This context necessitates a compression approach that prioritizes more than just accuracy. Stability is the foremost requirement for any controller and is typically established through rigorous mathematical frameworks, such as Lyapunov stability theory. Standard pruning techniques, which modify network weights and structure, can undermine these formal guarantees. In closed-loop systems, even minor deviations in controller output caused by compression can accumulate, resulting in oscillatory, unpredictable, or hazardous behavior. Consequently, for effective system integration of NNCs, compression methods must treat control-related properties as primary constraints.

This paper introduces a framework for resource-focused pruning, designed to facilitate the safe and efficient deployment of NNCs on embedded hardware. Building upon previous work in application-specific pruning [1], the methodology is extended to incorporate control-theoretic constraints. Specifically, a novel pruning problem is formulated in which the preservation of Lyapunov stability is treated as a primary constraint, alongside strict requirements for model size and inference time. The proposed approach is validated on a complex NNC architecture, and the effectiveness of post-pruning fine-tuning is examined to recover the performance lost during compression, thereby enhancing the feasibility of integrating real-world systems.

II. RELATED WORKS

Neural network pruning, which involves removing parameters to reduce model complexity, has evolved from early heuristic approaches to advanced, theoretically grounded frameworks. Early methods primarily relied on magnitude-based criteria, eliminating weights with the smallest absolute values. Despite their simplicity, these approaches have proven effective, often matching or surpassing the performance of more sophisticated techniques [2]. However, unstructured pruning, which removes individual weights and produces irregular patterns with limited practical speedup on standard hardware, has led to the emergence of structured pruning methods.

Structured pruning addresses these limitations by removing entire groups of parameters, such as neurons, filters, or channels, thereby reducing computational cost while maintaining dense operations, which are essential

*Authors have contributed equally
[1]Department of Electrical and Computer Engineering, RPTU University Kaiserslautern-Landau, Germany. E-mail: {ganesh.sundaram, jonas.ulmen, ahaider, daniel.goerges}@rptu.de

for edge deployment [3]. In addition to magnitude-based criteria, contemporary methods incorporate Hessian-based, class-aware [4], [5], and operator-theoretic approaches [6]. To minimize performance degradation, techniques including knowledge distillation [7], sparse optimization [8], gradient-based methods, joint pruning-quantization [9], and energy-constrained optimization [10], [11] are employed. Structured pruning facilitates hardware acceleration through compiler optimizations [12] and platforms such as HALP, which balance accuracy and latency [13]. Notable improvements include a $2\times$ speedup for YOLOv7 on FPGA [14], $19.1\times$ for LSTMs on Jetson [15], $2\times$ for GNNs [16], $11.1\times$ size reduction with compute-in-memory [17], and $4.79\times$ with systolic arrays. Despite these advances, traditional techniques rely on architecture-specific grouping, which limits their generalizability. SPA enables architecture-agnostic pruning [18], while DepGraph [19] employs dependency graphs and norm-based criteria to achieve consistent and flexible pruning across architectures.

While these frameworks promote architectural generalization, a significant gap remains in understanding how pruning affects the functional and structural interdependencies among network components, especially in multi-component architectures with distinct module functions. This limitation is particularly pronounced in control applications, where maintaining system stability and safety properties is crucial; yet current pruning methods do not adequately address these requirements.

III. Neural Network Controllers on Embedded Hardware

The successful deployment of an NNC on resource-constrained hardware presents a dual challenge. The controller must not only operate within the stringent physical limitations of the hardware but also rigorously adhere to the principles of control theory to ensure safe and reliable operation. These two distinct sets of requirements, one driven by the hardware's resource budget and the other by the system's control-theoretic properties, must be jointly satisfied for successful system integration. The following subsections elaborate on both categories of constraints.

A. Resource-Related Constraints

The successful integration of NNCs into physical systems is fundamentally constrained by the hardware limitations inherent in embedded platforms. These devices are designed to prioritize cost, power efficiency, and a small physical footprint over raw computational power. Consequently, any NNC intended for deployment must adhere to a strict set of resource budgets. The primary constraints include:

Memory: Embedded systems operate with stringent memory limitations, both for volatile memory (RAM), which is often in the range of kilobytes to a few megabytes, and for non-volatile storage, which dictates the maximum size of the controller's program code.

Computation: The processing power, often measured in Floating Point Operations Per Second (FLOPs), is significantly lower than that of desktop systems. This directly impacts inference time (latency), which is a critical performance metric.

Real-Time Performance: Many control applications, particularly in safety-critical domains, have hard real-time requirements, meaning they must execute tasks within a guaranteed timeframe. This necessitates careful optimization and often relies on a Real-Time Operating System (RTOS).

Power and Energy: Power consumption is a first-order design constraint. Firmware and hardware are heavily optimized to minimize energy use, often employing low-power sleep modes to extend battery life.

These resource limitations are complex requirements that dictate the feasibility of deploying an NNC. Therefore, they must be treated as primary design constraints in any practical system integration workflow.

B. Control-Theoretic Constraints

Performance evaluation during the pruning of NNCs should not rely exclusively on task-specific metrics such as reward or accuracy. In contrast to general predictive models, NNCs must satisfy stringent control-theoretic requirements that ensure the safety and reliability of physical systems. Among these, stability is paramount, as it guarantees that the system's output remains bounded and predictable. Stability is typically verified through mathematical frameworks such as Lyapunov stability theory or contraction theory. Additionally, sensitivity and robustness are essential for enabling controllers to manage model uncertainties and external disturbances. Consequently, any effective pruning strategy for NNCs must prioritize the preservation of these control-related properties.

C. TD-MPC Inference

To quantitatively assess the computational and deployment challenges posed by NNCs, we selected the TD-MPC agent, which is designed to balance an inverted pendulum directly from raw pixel observations. The model architecture and hyperparameters follow those identified as optimal benchmarks for control performance [20]. The characteristics of the trained TD-MPC model are detailed in Table I, including input representation, model capacity, and achieved control performance.

TABLE I: Model characteristics and training specifications

Parameter	Value	Parameter	Value
Input Size	28×28 pixels	Latent Dim.	50
Components	Encoder, Dynamics, Reward, Value, Policy	Episode Reward	858.6
		Episode Length	125 steps
		Total Parameters	1,548,360
Layers/Comp.	3	Total FLOPs	4.12×10^8
Hidden Units	512	Model Size	12.4 MB

To evaluate the real-world deployability of the agent, its inference performance was benchmarked across a range of hardware platforms, from high-performance workstations to resource-constrained embedded systems, including the Jetson

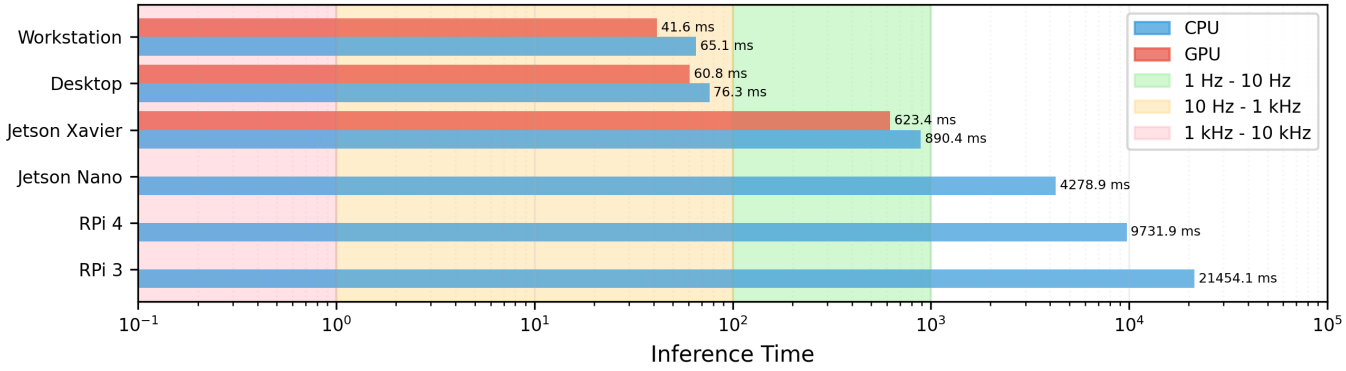

Fig. 1: Inference time benchmark of the uncompressed TD-MPC model on various hardware. The colored regions represent typical real-time operating frequencies for control systems: slow dynamics (green), mid-range applications like robotics (yellow), and high-speed systems (red).

Nano and Raspberry Pi series. For control systems like the inverted pendulum, where controller frequencies must range between 10 Hz and 1 kHz (yellow region in Figure 1), the results reveal a substantial performance gap between platform categories. The inference times on all tested embedded systems are orders of magnitude higher than those on high-end desktop hardware. This significant computational bottleneck demonstrates that deploying the uncompressed model directly is infeasible, firmly establishing model compression as a prerequisite for successful system integration.

IV. Resource-Aware Structured Pruning

Given the sensitivity of multi-component NNCs and the need to preserve control-theoretic properties, a targeted compression strategy is essential. Unlike global techniques such as quantization, which can unpredictably alter the entire model, pruning carefully eliminates redundant elements, making it the preferred method for this application. Specifically, structured pruning is chosen over unstructured pruning, which creates inefficient sparse matrices. This technique eliminates entire functional blocks [21], preserving the architectural integrity necessary for stable control while producing a dense and compact model that operates efficiently on embedded hardware.

A. Component-Aware Pruning Groups

A prerequisite for effective structured pruning is the partitioning of a model's parameters into groups of structurally and functionally coupled elements. Given the multi-component nature of the TD-MPC architecture, we employ the Component-Aware DepGraph method from [22] to perform this segmentation.

This approach analyzes the model's computational graph to identify both component-specific groups (parameters exclusive to a single component) and coupling groups (parameters shared across components). The analysis yields a total of ten distinct pruning groups: eight component-specific groups and two coupling groups, which form the basis of our pruning strategy. Table II provides a detailed breakdown of these groups, enumerating their constituent modules and parameter counts.

TABLE II: Pruning group decomposition of the model showing component-specific and coupling groups with their constituent modules and parameter counts

Group Type	Component	Group	Modules	Parameters
Comp.-specific	Encoder	1	1	25,632
		2	1	9248
		3	1	9248
	Dynamics	1	1	262,656
	Reward	1	1	262,656
	Policy (Pi)	1	1	262,656
	Q1	1	1	262,656
	Q2	1	1	262,656
Coupling	Encoder-Pi	1	2	26,112
	EncoderPi-Dyn.Rew.Q1Q2	1	5	369,152

B. Pruning Coefficient Search

To precisely control the level of sparsity applied to each of the ten identified groups, we introduce a set of corresponding pruning coefficients. Each group i is assigned a coefficient $c_i \in [0, 1]$, which dictates the fraction of parameters to be removed from that specific group. The complete pruning configuration for the controller is defined by the coefficient vector $\mathbf{c} = [c_1, c_2, \ldots, c_{10}]^T$.

The major challenge is to find an optimal vector $\mathbf{c}$ that compresses the TD-MPC model sufficiently to meet the resource constraints of the target embedded system (Section III-A) while simultaneously preserving critical control-theoretic properties such as stability (Section III-B). To solve this complex, multi-objective challenge, we formulate the search for these coefficients as a formal optimization problem.

C. Performance Evaluation Function

The stability of both the original and pruned TD-MPC agents in the inverted pendulum environment was systematically evaluated using a neural Lyapunov function [23], [24] defined on the latent states of the environment. The neural Lyapunov function is trained independently from the TD-

MPC agents. Training data are generated by exciting the system with random perturbations or exploration policies, resulting in diverse state trajectories that cover the region around the upright equilibrium. In the pendulum balancing task, the Lyapunov function V is constructed to be positive definite except at the upright equilibrium, where the pendulum is perfectly vertical ($\theta = 0$ and $\dot{\theta} = 0$). This formulation enforces the following conditions:

$$V(x) > 0 \quad \forall\, x \neq x^*, \qquad V(x^*) = 0 \tag{1a}$$

$$\Delta V = V(x_{t+1}) - V(x_t) < 0 \quad \forall\, x_t \neq x^* \tag{1b}$$

Here, x^* represents the upright equilibrium state; for example, $x^* = [\theta = 0, \dot{\theta} = 0]$. After successful training, the neural Lyapunov function is used to assess the closed-loop behavior of the parent TD-MPC agent. During multiple pendulum-balancing episodes, the output of the Lyapunov function is monitored.

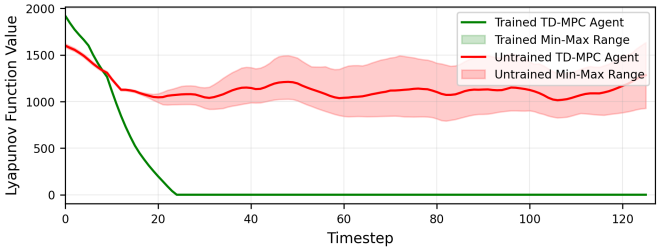

Fig. 2: Mean Lyapunov function values over five episodes for untrained (bold red) and trained (bold green) TD-MPC agents. Shaded areas show the observed minimum and maximum value ranges. The trained agent demonstrates consistent performance, with nearly identical trajectories and a narrow green region. The untrained agent displays greater variability, as shown by the wider red region.

Figure 2 illustrates that the value V initially starts at a high positive value and decreases monotonically until it reaches zero. The convergence of V to zero closely aligns with the pendulum's observed settling time in the simulation, thereby verifying the accuracy of the trained Lyapunov network. Once the system stabilizes, V remains at zero for the remainder of the episode, indicating that the TD-MPC agent maintains the pendulum in its upright position and that the Lyapunov function consistently represents this stable state. With the neural Lyapunov function established as the stability verification framework, the optimization objective for the resource-aware pruning strategy can be defined. In contrast to traditional application-specific pruning approaches that optimize for performance metrics such as reward [1], the proposed optimization-based coefficient search prioritizes preserving the monotonic decrease in the Lyapunov function as the system approaches equilibrium. This approach ensures that stability serves as the primary constraint guiding the pruning process.

V. Experiments and Discussion

To evaluate whether the proposed resource-aware pruning strategy can facilitate the integration of NNCs into embedded platforms, we designed a two-stage experimental framework. The first stage targets the practical challenge of reducing model size to fit available hardware resources, while the second stage explores the fundamental limits of compression.

Experiment I: *Achieving a Sparsity Target:* The initial goal is to validate that our method can achieve a precise, modest level of compression without violating stability. The optimization is constrained to a target model sparsity $\rho = 10\%$ with a tolerance of $\varepsilon = +1\%$, meaning the final sparsity must fall within the range of $[10\%, 11\%]$.

Experiment II: *Determining the Stability Boundary:* The second stage aims to identify the maximum possible compression that the TD-MPC agent can withstand before its stability is compromised. In this scenario, the sparsity constraint is removed from the optimization problem. The sole objective is to maximize the pruning ratio while ensuring the Lyapunov stability condition (i.e., the monotonic decrease of V) is never violated.

Figure 3 presents the results from both experiments, providing critical insights into the relationship between pruning aggressiveness and system stability. The analysis identifies three distinct behavioral regimes, thereby validating the proposed approach. The unpruned baseline model (blue line) demonstrates ideal Lyapunov behavior, with V values beginning near 1900 and decreasing monotonically to zero by timestep 22. Notably, this fundamental stability characteristic is preserved across all compression levels up to 20.1% sparsity. The pruned models at 10.5%, 15.7%, and 20.1% maintain the essential monotonic decrease property and converge to zero, although with modest increases in settling time. These findings indicate that substantial model compression, reducing the original model by one-fifth, can be achieved while maintaining rigorous stability guarantees. Furthermore, the 10% pruned model reduced inference time on all tested hardware, achieving $39.82\,\mathrm{ms}$ in Workstation, $56.60\,\mathrm{ms}$ in Desktop, $843.41\,\mathrm{ms}$ in Jetson Xavier, $3239.61\,\mathrm{ms}$ in Jetson Nano, $8713.44\,\mathrm{ms}$ in RPi 4, and $14\,436.83\,\mathrm{ms}$ in RPi 3, demonstrating practical speedups across all target embedded platforms.

At 24.4% sparsity, the system demonstrates initial indications of stability degradation, characterized by a non-monotonic reduction in V values and energy, as well as an extended settling time. This level of sparsity represents a critical threshold at which the controller may begin to lose its stability margin. Beyond 25% sparsity, the system experiences catastrophic stability failure. Both the 29.6% and 34.4% pruned models exhibit highly erratic, oscillatory Lyapunov function trajectories that fail to converge to zero, indicating persistent energy oscillations that violate the fundamental Lyapunov stability condition. The severity of oscillations increases with the degree of pruning aggressiveness, confirming the essential instability of the controller. These results establish a safe compression sweet spot of approximately 22% sparsity, where substantial model reduction is achieved without compromising stability. The transition from stable to unstable behavior occurs rapidly around the 24-25% threshold, emphasizing the importance of

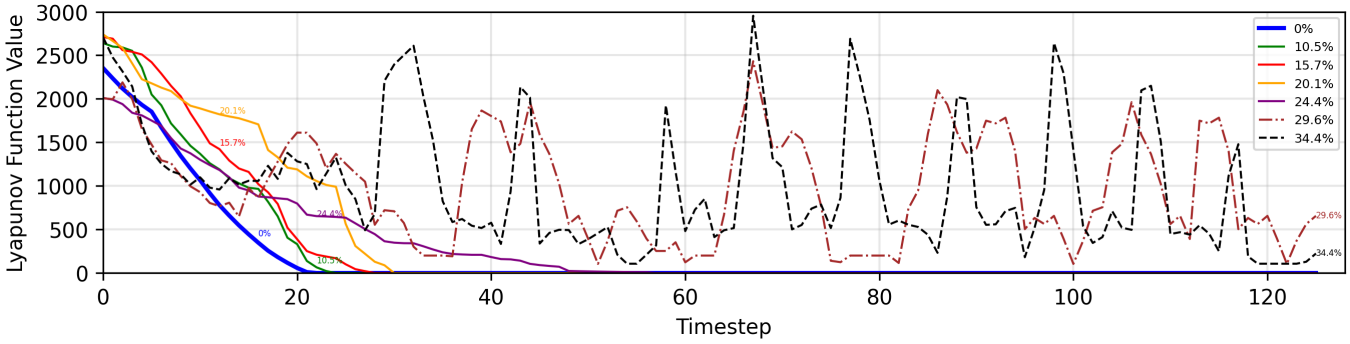

Fig. 3: Lyapunov function evolution across different sparsity levels of the TD-MPC agent. The unpruned baseline (blue) demonstrates ideal stability with monotonic convergence to zero. Progressive pruning from 10.5% to 20.1% maintains stability as settling times gradually increase. A critical transition occurs around 24.4% sparsity, beyond which the system exhibits unstable oscillatory behavior (29.6% and 34.4%), clearly violating the Lyapunov stability condition.

staying within safe compression limits. Table III details the final pruning coefficient configurations for both component-specific and coupling groups that achieved the specified sparsity levels, along with their corresponding episode rewards. Notably, coupling groups were deliberately excluded from pruning (coefficients = 0) due to their inherent sensitivity arising from their complex inter-component dependencies. This optional strategic decision not only preserves critical architectural connections but also reduces the complexity of the optimization search space, allowing the algorithm to focus on component-specific parameter allocation.

TABLE III: Pruning coefficient configurations and episode rewards across sparsity levels. Component-specific group coefficients are displayed for eight groups, while coupling group coefficients are set to zero to preserve critical architectural connections.

Sparsity	Component-Specific Group Coefficients	Coupling	Reward
0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	[0, 0]	858.6
10.5	[0.0, 0.0, 0.0, 0.0, 0.190, 0.196, 0.102, 0.112]	[0, 0]	851.4
15.7	[0.105, 0.095, 0.0, 0.275, 0.198, 0.142, 0.131, 0.159]	[0, 0]	804.3
20.1	[0.0, 0.115, 0.114, 0.386, 0.453, 0.106, 0.124, 0.159]	[0, 0]	793.7
24.4	[0.124, 0.0, 0.113, 0.584, 0.465, 0.104, 0.159, 0.114]	[0, 0]	689.3
29.6	[0.097, 0.051, 0.020, 0.641, 0.623, 0.149, 0.099, 0.116]	[0, 0]	633.1
34.4	[0.114, 0.116, 0.0, 0.726, 0.708, 0.276, 0.124, 0.159]	[0, 0]	602.1

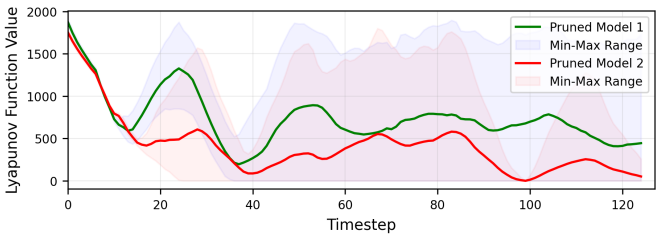

Fig. 4: The sensitivity of component-specific pruning on Lyapunov behavior, making it unstable when targeting critical components.

Although comprehensive pruning experiments established a stability boundary at less than 24%, a critical finding is that achievable sparsity is highly dependent on which specific components are pruned, rather than solely on the total pruning percentage. Figure 4 illustrates this through targeted component experiments. Selective pruning of encoder groups (Model 1) results in catastrophic instability, even at less than 8% total sparsity. Similarly, pruning the group from the dynamics model (Model 2) destabilizes the controller at only 15% total sparsity, which is well below the established threshold. In both cases, the Lyapunov function trajectories become oscillatory and fail to converge, clearly violating stability conditions. These results indicate that not all components contribute equally to system stability. The encoder and dynamics components demonstrate greater sensitivity to pruning, as even small reductions in their capacity can propagate instability throughout the control system.

VI. CONCLUSION

This study addresses a significant challenge in deploying NNCs on embedded hardware by maintaining essential stability guarantees while achieving the aggressive model compression required by resource constraints. Unlike conventional pruning approaches that prioritize accuracy metrics, this work introduces a principled framework that treats Lyapunov stability as the primary optimization constraint, fundamentally shifting the pruning paradigm for control applications. The proposed contribution extends component-aware structured pruning by introducing a novel optimization-based coefficient search methodology that systematically determines optimal pruning ratios for each component group. Experimental validation using a TD-MPC agent demonstrates that the framework achieves precise compression targets while preserving stability guarantees and accurately identifies sharp stability transition boundaries. The results indicate that sensitivity to component-specific pruning varies substantially. Critical components, such as the encoder and dynamics model, induce system instability at significantly lower compression levels, underscoring the need for structured approaches over uniform pruning strategies. This insight supports the safe integration of NNCs into embedded systems, providing practitioners with essential knowledge of stability-critical architectural components. Future research will prioritize the early identification of sensitive component groups and explore the multi-objective optimization of additional control-theoretic properties. These efforts will

strengthen the framework for deploying neural network controllers in safety-critical embedded systems.

References

[1] G. Sundaram, J. Ulmen, A. Haider, and D. Görges, "Application-specific component-aware structured pruning of deep neural networks in control via soft coefficient optimization," *arXiv preprint arXiv:2507.14882*, 2025.

[2] M. Gupta, E. Camci, V. R. Keneta, *et al.*, "Is complexity required for neural network pruning? a case study on global magnitude pruning," in *Proc. of the 2024 IEEE Conference on Artificial Intelligence*, 2024, pp. 747–754.

[3] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2900–2919, 2023.

[4] J. Chong, M. Gupta, and L. Chen, "Resource efficient neural networks using hessian based pruning," *arXiv e-prints*, arXiv–2306, 2023.

[5] M. Jiang, J. Wang, A. Eldebiky, *et al.*, "Class-aware pruning for efficient neural networks," in *Proc. of the 2024 Design, Automation & Test in Europe Conference & Exhibition*, 2024.

[6] W. T. Redman, M. Fonoberova, R. Mohr, Y. Kevrekidis, and I. Mezic, "An operator theoretic view on pruning deep neural networks," in *International Conference on Learning Representations*, 2022.

[7] Y. Qian, X. Li, J. Cao, J. Zhang, H. Li, and J. Chen, "Boosting pruned networks with linear overparameterization," in *Proc. of the 2024 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2024, pp. 5070–5074.

[8] Y. Shi, A. Tang, L. Niu, and R. Zhou, "Sparse optimization guided pruning for neural networks," *Neurocomputing*, vol. 574, p. 127 280, 2024.

[9] Y. Wang, Y. Lu, and T. Blankevoort, "Differentiable joint pruning and quantization for hardware efficiency," in *Proceedings 13th of the European Conference on Computer Vision*, 2020, pp. 259–277.

[10] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10 558–10 578, 2024.

[11] S. Liu, Q. Li, B. Peng, Y. Qiao, M. Lin, and Q. Zeng, "An energy-constrained optimization-based structured pruning method for deep neural network compression," in *Proc. of the 2022 IEEE 22nd International Conference on Communication Technology*, 2022, pp. 1903–1907.

[12] Y. Gong, G. Yuan, Z. Zhan, *et al.*, "Automatic mapping of the best-suited DNN pruning schemes for real-time mobile acceleration," *ACM Transactions on Design Automation of Electronic Systems*, vol. 27, no. 5, pp. 1–26, 2022.

[13] M. Shen, H. Yin, P. Molchanov, L. Mao, J. Liu, and J. M. Alvarez, "Structural pruning via latency-saliency knapsack," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 894–12 908, 2022.

[14] S. Pavlitska, O. Bagge, F. Peccia, T. Mammadov, and J. M. Zöllner, "Iterative filter pruning for concatenation-based cnn architectures," in *Proc. of the 2024 International Joint Conference on Neural Networks*, 2024.

[15] J. H. Lindmar, C. Gao, and S.-C. Liu, "Intrinsic sparse LSTM using structured targeted dropout for efficient hardware inference," in *Proc. of the 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems*, 2022, pp. 126–129.

[16] D. Gurevin, M. Shan, S. Huang, M. A. Hasan, C. Ding, and O. Khan, "PruneGNN: Algorithm-architecture pruning framework for graph neural network acceleration," in *Proc. of the 2024 IEEE International Symposium on High-Performance Computer Architecture*, 2024, pp. 108–123.

[17] F.-H. Meng, X. Wang, Z. Wang, E. Y.-J. Lee, and W. D. Lu, "Exploring compute-in-memory architecture granularity for structured pruning of neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, pp. 858–866, 2022.

[18] X. Wang, J. Rachwan, S. Günnemann, and B. Charpentier, "Structurally prune anything: Any architecture, any framework, any time," *arXiv preprint arXiv:2403.18955*, 2024.

[19] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, "DepGraph: Towards Any Structural Pruning," in *Proc. of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 091–16 101.

[20] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," in *International Conference on Machine Learning*, PMLR, 2022, pp. 8387–8406.

[21] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. of the 7th International Conference on Learning Representations*, 2019.

[22] G. Sundaram, J. Ulmen, and D. Görges, "Enhanced pruning strategy for multi-component neural architectures using component-aware graph analysis," in *Proc. of the IFAC Joint Conference on Computers, Cognition and Communication*, 2025.

[23] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1749–1767, 2023.

[24] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.