Controller-aware Perception Neural Network Pruning for CPS Safety Optimization

JINGXIAN CHEN*, University of Pittsburgh, USA
YUKAI SONG*, University of Pittsburgh, USA
SHENGJIE XU*, The University of North Carolina at Chapel Hill, USA
KYLE KESSLER, University of Pittsburgh, USA
PRATEEK GANGULI, The University of North Carolina at Chapel Hill, USA
ARKAPRAVA GUPTA, The University of North Carolina at Chapel Hill, USA
BINEET GHOSH, The University of Alabama at Birmingham, USA
PARASARA SRIDHAR DUGGIRALA, The University of North Carolina at Chapel Hill, USA
TIANLONG CHEN, The University of North Carolina at Chapel Hill, USA
SAMARJIT CHAKRABORTY†, The University of North Carolina at Chapel Hill, USA
JINGTONG HU, University of Pittsburgh, USA

Traditional neural architecture pruning methods primarily focus on trade-offs between inference accuracy and computational cost (*e.g.*, FLOPs), without accounting for system-level behaviors. In the context of cyber-physical systems (CPS), perception models and the downstream controllers that rely on their outputs are typically designed independently. In this paper, we study the joint design of neural architecture pruning strategies and the controllers that utilize the resulting inference outputs. We propose two key innovations: (i) incorporating system-level safety metrics into the pruning optimization loop, and (ii) a regression-based scaling law method that efficiently models the relationship between pruning ratios and system-level safety outcomes. This enables rapid pruning guided by system-level constraints, avoiding the inefficiency of exhaustive search or conventional neural architecture search approaches that focus on local accuracy or latency metrics. We demonstrate our approach on YOLO-based perception models deployed in control systems for distance estimation. To the best of our

*Both authors contributed equally to this research.
†Also with TUM Institute for Advanced Study.

Authors' Contact Information: Jingxian Chen, jic358@pitt.edu, University of Pittsburgh, Department of Electrical and Computer Engineering, Pittsburgh, PA, USA; Yukai Song, yus190@pitt.edu, University of Pittsburgh, Department of Electrical and Computer Engineering, Pittsburgh, PA, USA; Shengjie Xu, sxunique@cs.unc.edu, The University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA; Kyle Kessler, kyk37@pitt.edu, University of Pittsburgh, Department of Electrical and Computer Engineering, Pittsburgh, PA, USA; Prateek Ganguli, pganguli@cs.unc.edu, The University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA; Arkaprava Gupta, arkaprava.gupta@gmail.com, The University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA; Bineet Ghosh, bineet@ua.edu, The University of Alabama at Birmingham, Department of Computer Science, Birmingham, AL, USA; Parasara Sridhar Duggirala, psd@cs.unc.edu, The University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA; Tianlong Chen, tianlong@cs.unc.edu, The University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA; Samarjit Chakraborty, samarjit@cs.unc.edu, The University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA; Jingtong Hu, jthu@pitt.edu, University of Pittsburgh, Department of Electrical and Computer Engineering, Pittsburgh, PA, USA.

ACM 2378-9638/2026/7-ART
https://doi.org/10.1145/3828657

knowledge, this is the first work to introduce a scaling-law-guided pruning method for YOLO-based perception models, offering a principled design automation framework for safety-aware, machine learning–enabled CPS.

CCS Concepts: • **Computer systems organization → Embedded software**.

Additional Key Words and Phrases: Cyber-Physical Systems, Control-Aware Neural Network Pruning, Pruning Scaling Laws, Reachability Analysis, Safety Optimization

1 Introduction

Deep neural networks (DNNs) have become foundational in modern Cyber-Physical Systems (CPSs), and are widely adopted in applications such as smart manufacturing [41], robotics [38], and autonomous driving [10, 27]. As the complexity of neural network architectures continues to grow, numerous pruning techniques have been proposed to reduce computational overhead and memory requirements while preserving model accuracy. These methods compress models by removing less critical components such as parameters, channels, or layers. This compression accelerates inference and reduces the model size, making pruning particularly attractive for resource-constrained edge devices and embedded systems [13, 28, 34].

While most existing pruning methods emphasize the trade-off between model accuracy and computational efficiency—typically quantified by the number of floating-point operations (FLOPs)—few consider the safety implications in real-world control scenarios. Without explicitly incorporating system-level safety metrics into the pruning process, achieving optimal end-to-end performance remains challenging. Current approaches often rely on empirical validation to assess whether a pruned model still performs adequately—a process that is both time-consuming and unable to provide formal safety guarantees. In CPS, particularly in safety-critical applications such as autonomous driving, predictable and verifiable behavior is essential. However, these systems are often constrained by space, energy, and cost. It is therefore challenging to simultaneously achieve both high accuracy and low inference latency, necessitating trade-offs between the two. Moreover, the optimal balance between accuracy and latency depends on the characteristics of the underlying dynamical system. A system requiring high-frequency control inputs is significantly more sensitive to inference latency than one operating at lower control frequency. However, existing neural network pruning research seldom accounts for such system-level safety constraints, making it difficult to identify the most suitable model for a given control system.

Consider YOLO [35], a widely used perception model in autonomous driving. Existing pruning frameworks for YOLO typically require users to specify a pruning rate and then generate the corresponding pruned model. Although metrics such as mean average precision (mAP) and FLOPs are commonly used to evaluate pruning performance, they are far from sufficient in capturing downstream safety implications. For example, a 10% drop in mAP after pruning may lead to a 4% increase in mean relative absolute error (MRAE) during subsequent distance estimation. While the pruned model may exhibit lower inference latency, the combined impact of increased estimation error and reduced latency on system-level safety remains unclear. Assessing this impact often relies on repeated empirical testing, which is both time-consuming and lacks formal safety guarantees. Relying solely on discrete pruning rates and post-hoc empirical evaluation poses significant challenges in building efficient and safe autonomous systems. This challenge is compounded by the conventional separation in CPS design: system-level safety is addressed at the controller design stage, which produces derived timing and accuracy requirements for the implementation stage. Pruning under this paradigm is guided by such proxy metrics—e.g., meeting a predefined inference latency budget—rather than by the end goal of system safety itself.

To bridge this gap, we propose a co-design approach that integrates model pruning with control-theoretic principles to ensure both computational efficiency and safety in real-world CPSs applications, such as autonomous driving. Rather than treating pruning solely as a model compression task constrained by predefined timing budgets, we introduce a **Control Policy-Aware Automatic Pruning Framework** that explicitly incorporates system-level control performance metrics into the pruning process. This ensures that the resulting pruned model

satisfies the safety requirements of the CPS when deployed as part of a closed-loop control system. At the core of our approach is a technique from control theory known as *reachability analysis* [2]—a numerical method that computes the set of states reachable by a dynamical system from all initial conditions under all admissible inputs and parameters. To quantitatively assess the impact of pruning strategies and pruning ratios on control performance, we adopt the Maximum Diameter of Reachable Sets (MDRS) [44, 45] over a finite time horizon as a proxy for system-level safety. A smaller MDRS implies that the system trajectories remain tightly bounded under uncertainty, leading to more predictable and safer behavior. In contrast, a larger MDRS reflects greater sensitivity to estimation errors, increasing the likelihood of unsafe outcomes.

Beyond the need for safety-aware evaluation, another limitation of prevailing pruning methodologies is their reliance on brute-force searches over discrete pruning ratios [13, 28, 34]. Each configuration requires retraining and empirical evaluation, rendering these methods resource-intensive and impractical for deployment in real-time systems. In addition to their inefficiency, such approaches lack interpretability and provide no theoretical insight into how pruning affects critical system-level metrics, such as safety. As a result, they are unsuitable for use in automated design flows in time-sensitive CPSs. This limitation motivates the use of pruning scaling laws, which have recently been explored in large language models to reveal predictable relationships between pruning configurations and post-training performance. However, analogous studies in vision-based models—especially those deployed in control-aware settings such as YOLO—remain limited. To address this gap, we investigate whether similar structured relationships exist in perception-driven control systems and demonstrate that such dependencies can indeed be observed and effectively modeled. We further propose a regression-based pruning scaling law framework that models the relationship between pruning rate and system-level performance. This framework enables efficient identification of pruning configurations that satisfy safety constraints, thus avoiding exhaustive retraining and enabling principled, control-aware model compression. We integrate this predictive scaling law into a broader pruning pipeline, where it serves as a core component for guiding safe model selection. By embedding it into the system architecture, we enable automatic pruning decisions that are directly informed by control-theoretic constraints.

In this paper, we present the first **Control Policy-Aware Automatic Pruning Framework** for YOLO, designed to enable safe and efficient deployment in cyber-physical systems (CPS). Figure 1 illustrates the overall architecture of our system, which consists of two main components: **Piecewise Function Modeling** and the **Control-Aware Neural Network Pruning System**.

The **Piecewise Function Modeling** component starts with a fully trained YOLOv8-based distance estimation model. By applying different pruning ratios, we generate a set of pruned models, which are used to estimate object distances via a lightweight linear regression module. These distance estimates are fed into a closed-loop control system, completing the perception-to-control pipeline and allowing us to evaluate how pruning impacts system-level safety. Specifically, we use the maximum diameter of reachable sets (MDRS) as the safety metric. The system then enters the **Progressive Model Pruning and Data Collection Loop**, in which it iteratively sweeps over multiple pruning ratios, conducts closed-loop simulation for each configuration, and computes the corresponding MDRS. This process constructs the relationship between pruning ratio and system safety performance. We repeat the same procedure for each pruning method, resulting in multiple datasets of pruning rate and MDRS pairs. We fit a piecewise parametric function to each dataset, modeling how the pruning ratio affects the system's MDRS under each specific pruning method. The result is a set of pruning scaling laws.

The **Control-Aware Neural Network Pruning System** component takes a user-defined safety specification—typically a control-theoretic MDRS threshold—and identifies the intersection points between the safety threshold and all pruning scaling curves. It then selects the pruning configuration that both satisfies the safety constraint and minimizes model size. The final output is a compact, control-compliant YOLOv8 model optimized for deployment on resource-constrained embedded hardware platforms.

Our contributions are as follows:

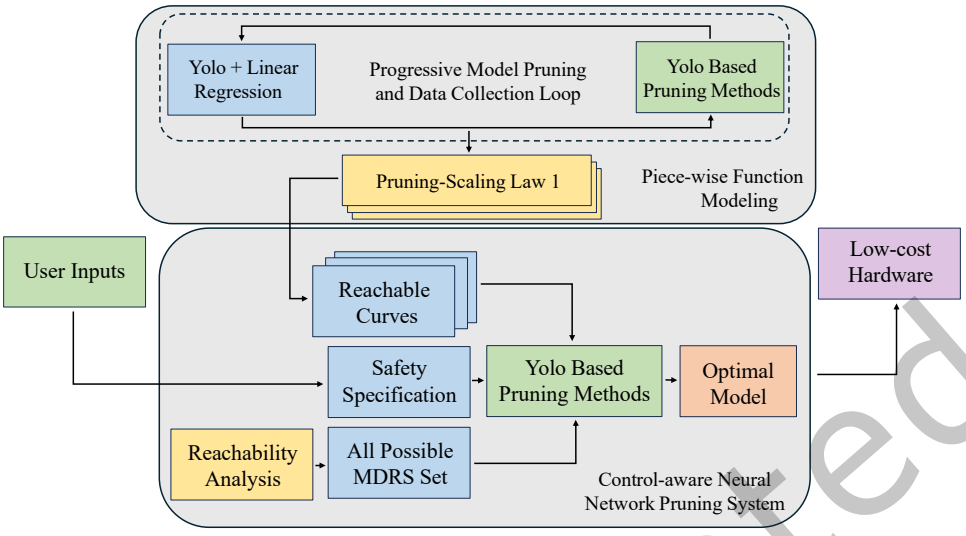

Fig. 1. Control Policy-Aware Automatic Pruning Framework

- **System-Safety-Driven Pruning of Neural Networks:** We propose a pruning framework that explicitly integrates system-level safety metrics MDRS into the pruning loop, ensuring safety compliance throughout and significantly reducing the reliance on post-hoc validation.
- **YOLOv8 Scaling Law for Structured Pruning:** We construct an analytical mapping between the pruning ratio, model size, and system-level safety measured by MDRS for YOLOv8 models. This transforms the traditionally discrete, trial-and-error pruning process into a continuous, tractable optimization space.
- **Control-Aware Co-Design for Efficient Model Selection:** Leveraging control-theoretic principles, we enable automatic selection of the smallest pruned model that satisfies task-specific safety constraints. Our approach achieves up to **148.33× speedup** in pruning search on distance estimation tasks, while satisfying both hardware constraints and safety objectives.
- **Fully Automated and Generalizable Framework:** The proposed pruning framework is end-to-end automated and adaptable to various control formulations, allowing generalization across diverse system-level safety specifications without manual tuning.

2 Related Work

2.1 Traditional Structured Pruning

Traditional structured pruning methods aim to reduce the computational cost and model size of deep neural networks by removing redundant architectural components such as channels, filters, or layers. Most existing approaches primarily focus on balancing accuracy and efficiency—typically measured in FLOPs or inference latency—without accounting for the downstream impact on control system behavior. This omission is particularly critical in CPS, where perception models are tightly integrated with feedback controllers, and any degradation in perception performance may directly compromise overall system safety.

2.1.1 Torch Structured Pruning. PyTorch provides native support for structured pruning via the torch.nn.utils.prune module [34]. This framework enables developers to remove entire channels or filters based on predefined criteria, such as the L_n norm of weight tensors. These operations effectively reduce model size without requiring architectural redesign. However, these pruning utilities are primarily designed for perception-level optimization, focusing on metrics such as FLOPs, inference latency, or detection accuracy while overlooking the downstream effects on control system behavior. When applied in CPS, such isolated pruning strategies may unintentionally degrade the safety and reliability of control decisions.

2.1.2 DepGraph. DepGraph [13] introduces a novel dependency graph representation to enable flexible and fine-grained structured pruning across a broad range of network architectures, including YOLOv8 [23]. It models inter-parameter dependencies to identify pruning candidates while ensuring the resulting architecture remains valid. However, similar to previous methods, DepGraph does not consider system-level performance metrics. It assumes that preserving perception-level accuracy—such as mAP or classification accuracy—is sufficient, which may not hold in real-world control scenarios where small perception errors can propagate into unsafe control behaviors.

2.1.3 Network Slimming. Network Slimming [28] is a channel-level pruning technique that leverages scaling factors learned through batch normalization layers to identify and eliminate less important channels. It achieves impressive compression ratios while maintaining high accuracy by introducing sparsity regularization during training. Although this method improves inference efficiency and model compactness, it does not consider how pruning-induced changes in perception quality may impact downstream control safety. Consequently, the resulting models may be unsuitable for CPS applications, where strict safety guarantees are required.

In summary, while traditional pruning methods such as Torch Pruning, DepGraph, and Network Slimming [13, 28, 34] are effective in reducing model size and computational cost, they operate under a perception-centric view of optimization. In contrast, our proposed method adopts a system-centric perspective by embedding control-aware safety metrics directly into the pruning process, enabling safe and efficient deployment in CPS environments.

2.2 Scaling Law

Recent work on large language models has shown that model performance follows predictable scaling trends with respect to key parameters such as model size and pruning ratio [9, 22, 24]. These empirical scaling laws enable performance forecasting without exhaustive retraining or hyperparameter search. However, existing scaling laws are primarily accuracy-centric, focusing on training loss or test error [22, 40], and do not capture how scaling affects system-level behavior in closed-loop CPS. In safety-critical applications such as autonomous driving, even minor perception errors introduced by pruning may significantly alter control trajectories and safety margins. Moreover, current scaling analyses have largely been developed for transformer-based NLP models [24], with limited validation on real-time vision models used in control systems, such as YOLO [35].

To address these limitations, we propose the first pruning scaling law for YOLOv8-based distance estimation in CPS. Instead of linking pruning to perception-level metrics, we establish a direct relationship between pruning ratio and a system-level safety measure, namely the closed-loop MDRS [45]. This formulation enables direct computation of the pruning ratio required to satisfy a target safety threshold, eliminating exhaustive configuration search. As shown in Section 5.3, this significantly reduces search time and provides a foundation for safe and scalable model compression in resource-constrained CPS deployments.

2.3 YOLO in Perception-Centric CPS

The YOLO family of models [35] has become a foundational tool for real-time object detection, particularly in latency-sensitive applications such as autonomous driving. Its one-stage architecture enables efficient visual perception, making it well-suited for cyber-physical systems (CPS) where timely and reliable decisions are essential. Among recent variants, YOLOv8 [23] is widely adopted due to its balance of accuracy, efficiency, and stability. Although not the newest version, its maturity and strong community support make it a practical choice for real-world CPS deployment. YOLOv8 has been applied to various perception tasks, including traffic light recognition [47], vehicle and pedestrian detection [12], and traffic sign classification [46], establishing it as a reliable foundation for control-oriented perception systems.

However, increasing model complexity challenges deployment on resource-constrained platforms. Prior work [7, 29] has explored compression and optimization to improve runtime efficiency or reduce model size for edge devices, yet largely overlooks system-level safety implications—an essential concern in CPS. To address this gap, we introduce a control-aware pruning framework that integrates control-theoretic safety metrics into the pruning of YOLOv8-based distance estimation models, enabling lightweight deployment while preserving safety-critical performance in closed-loop CPS environments.

2.4 CPS Safety Optimization with Non-Ideal Components

The safety analysis of control systems—and by extension, CPSs—often relies on idealized assumptions regarding implementation details such as numerical precision, communication delays, and model fidelity. However, when these assumptions do not hold in real-world deployments, the safety guarantees derived from the formal analysis may no longer be valid [14, 26]. As the hardware and software complexity of CPSs continues to increase, optimizing safety under realistic implementation constraints, rather than idealized abstractions, is becoming an increasingly critical research challenge.

For example, timing uncertainty in the underlying hardware/software implementation has been shown to significantly impact CPS safety [15, 20]. To address this, schedule synthesis techniques for CPS implementations have been proposed [21, 42, 43] to mitigate the effects of timing variability in embedded systems. In addition, resource contention—including network [36], memory, and computation [8]—has been studied for its implications on CPS safety [18, 37]. Our work aligns with these efforts in its focus on system-level safety rather than isolated component performance. However, instead of targeting timing or resource constraints applicable to embedded software in general, we focus on the unique characteristics of neural networks and develop specialized pruning strategies to optimize safety in control-aware CPS deployments.

Some prior work has explored the connection between neural network-based components and CPS safety. For example, the resource allocation problem of deploying multiple neural networks on shared hardware was studied in [45], with the objective of maximizing system-level safety. However, that work assumes a fixed set of pre-configured networks and focuses solely on allocating resources among them. In contrast, our work aims to optimize the neural networks themselves through pruning, rather than selecting from a predetermined pool of static architectures.

3 Preliminaries

3.1 Control System Safety

CPSs can be modeled as dynamical systems governed by one or more feedback control loops. A general time-invariant continuous-time system is described by the differential equation:

$$\dot{x} = f(x, u), \tag{1}$$

where $x \in \mathbb{R}^p$ is the *state vector*, $u \in \mathbb{R}^q$ is the *input vector*, and $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^p$ defines the system's continuous dynamics. Since f does not explicitly depend on time t, the system is considered time-invariant.

To implement feedback control in practice, controllers are mathematically designed and deployed as periodic real-time tasks on embedded processors. This requires discretization of the continuous dynamics using a fixed sampling period. Assuming a constant sampling period h, such that $t_{k+1} - t_k = h$, the system's discrete-time dynamics are described by the following difference equation:

$$x_{t_{k+1}} = f_h(x_{t_k}, u_{t_k}) ,$$

where $f_h : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^p$ is the discretized version of f with period h.

For simplicity, we denote x_{t_k} as $x[k]$ and u_{t_k} as $u[k]$ and obtain the following:

$$x[k+1] = f_h(x[k], u[k]) . \tag{2}$$

In the context of CPS safety, the sampling period h determines the response time of the control system to external disturbances.

The control input $u[k]$ is computed as:

$$u[k] = g_h(\hat{x}[k]) , \tag{3}$$

where $g_h : \mathbb{R}^p \to \mathbb{R}^q$ is an arbitrary function that determines the suitable control input for a given system state, taking into account the sampling period h. $\hat{x}$ is a measurement of the ground truth system state. Errors in state estimation $\hat{x}$ can result in incorrect control inputs, which in turn compromise system performance and safety. This effect is particularly pronounced in neural network-based perception tasks, such as distance estimation in autonomous driving. There are many quantitative metrics of system performance and safety—settling time, distance to an unsafe set of states, or state space tracking error, to name a few. In this work, we use one such metric, introduced in the next subsection.

3.2 Maximum Diameter of Reachable Sets (MDRS)

To evaluate the effects of sampling period and state estimation errors on system-level safety, we use the quantitative metric Maximum Diameter of Reachable Sets (MDRS), introduced in [45]. The MDRS is computed by propagating reachable sets forward in time. For a discrete Linear Time-Invariant (LTI) system expressed as $x[k+1] = \Phi\, x[k] + \Gamma\, u[k]$, the standard recursive formulation of set-based reachability analysis [16] yields the successor set

$$X[k+1] = \Phi\, X[k] \oplus \Gamma\, U[k], \tag{4}$$

where $\Phi\, X[k] = \{\Phi\, x \mid x \in X[k]\}$, $\Gamma U[k] = \{\Gamma\, u \mid u \in U[k]\}$, and $\oplus$ denotes the Minkowski sum $A \oplus B = \{a + b \mid a \in A,\ b \in B\}$. By repeatedly applying eq. (4) from an initial set $X[0]$, we can compute the reachable sets $X[1], \ldots, X[\frac{H}{h}]$ over an arbitrary time horizon H. The MDRS is then calculated as

$$D = \max_{k \in [1, \frac{H}{h}]} \max_{x_1, x_2 \in X[k]} \|x_1 - x_2\|. \tag{5}$$

It is worth noting that although the recursion in eq. (4) is conceptually straightforward, exact computation is often intractable because the geometric complexity of the set representation (e.g., the number of polytope vertices) can grow at each time step. Various techniques have been proposed to maintain tractability, including set propagation techniques using *zonotopes* [16, 17]. We use the JuliaReach toolbox [6] to handle the practical computation of reachable sets in this work.

While MDRS can be used with both linear and nonlinear systems, we focus on linear systems in this work to leverage closed-form reachable set computation. In practice, controllers for many nonlinear systems are designed via Jacobian linearization around operating points [25]. Similarly, reachability analysis for nonlinear systems commonly employs conservative linearization: the dynamics are linearized around the center of the current

reachable set, and set propagation is computed as a linear dynamics propagation augmented by an error bound from the linearization remainder [2, 3]. An alternative approach, Hamilton–Jacobi (HJ) reachability [30], solves a partial differential equation over a gridded state space. However, this grid-based computation scales exponentially with the number of state dimensions—the so-called "curse of dimensionality"—rendering it intractable for most realistic systems beyond a few dimensions [5]. Since our primary contribution concerns the integration of reachability-based safety metrics into the pruning loop rather than advances in reachability computation itself, and since the dominant flowpipe-construction methods for nonlinear systems are fundamentally built on linearization, using linear systems as case studies is well-justified without loss of methodological generality.

MDRS provides a physically interpretable measure of system-level safety. Each reachable set $X[k]$ characterizes the possible states of the system at step k, and its diameter represents the largest "spread" of possible system positions in the state space. For instance, in the F1/10 lane-keeping system used in our experiments (section 5), the MDRS bounds the worst-case spread of possible vehicle lateral positions: if the MDRS exceeds the lane width, no controller can guarantee the vehicle remains in its lane under worst-case perception errors. More generally, while the average system behavior may be acceptable, a large MDRS indicates that the system's trajectories are not tightly bounded, leaving it vulnerable to worst-case deviations. This makes MDRS particularly suitable for safety-critical applications where worst-case guarantees matter.

An alternative safety formulation commonly used in reachability-based verification defines safety as the reachable set not intersecting a designated unsafe region [1]. While intuitive, this formulation in its standard form yields a binary safe/unsafe verdict and does not quantitatively distinguish between a system that barely avoids the unsafe set and one that stays far from it. Continuous extensions exist—notably control barrier functions [4], which provide a scalar safety margin, and signal temporal logic robustness [11], which quantifies how far a trajectory is from violating a specification—but all such formulations require defining a scenario-specific unsafe region or specification (e.g., obstacle positions, lane boundaries at particular locations), making the resulting metric sensitive to these configuration choices. MDRS avoids this dependency by measuring the intrinsic behavioral uncertainty of the closed-loop system, independent of any particular obstacle or scenario configuration.

We note that our pruning framework is not specific to MDRS; it can accommodate any quantitative system-level metric as the optimization objective. Alternative metrics include maximum state deviation from ideal trajectories [20, 43], as well as classical control performance measures such as settling time, overshoot, and steady-state error [31]. We adopt MDRS as a principled default because it captures safety-relevant worst-case behavioral spread without requiring scenario-specific assumptions, but the framework's scaling-law-based approach generalizes to any metric that can be computed as a function of pruning ratio.

4 Methods

4.1 Problem Background: Perception-Induced Safety Constraints

We consider a closed-loop autonomous driving system where perception is provided by a YOLOv8-based distance estimation module, requiring timely and accurate environment understanding for safe operation. Given a camera image, YOLOv8 detects objects such as vehicles and pedestrians, and a lightweight regression module estimates their distances $\hat{Z}$. We use a modified version of the *Mean Relative Absolute Error* (MRAE) as the evaluation metric, which quantifies how close the predicted distance $\hat{Z}$ is to the ground-truth Z for detected objects. We further use MDRS to quantify the system's dynamic uncertainty under perception degradation. State estimation error is measured using the MRAE from the distance estimation task, while the sampling period is approximated by the model's inference latency. Together, these two factors critically influence overall control performance.

In our problem setting, the objective is to predict the distance to objects in front of the camera, providing critical signals for subsequent decision-making. Therefore, we restrict our evaluation to objects that are fully visible to the camera. Occluded or truncated objects are typically blocked by other closer objects. Furthermore, this partial

visibility can lead to unreliable bounding boxes, introducing noise into the distance estimation process. From both a practical and data quality standpoint, such objects are excluded from evaluation to avoid confounding the analysis.

A key limitation of the standard MRAE metric is that it is computed only over correctly detected objects, thereby ignoring missed detections where the detector fails to identify ground-truth objects. As a result, models with poor detection accuracy may paradoxically achieve lower MRAE values, since undetected objects are excluded from the error computation. In extreme cases, a model that detects only one object and happens to predict its distance accurately could appear to perform exceptionally well under the MRAE metric. This can be misleading, as it fails to reflect the model's overall performance in joint detection and distance estimation.

To address this, we propose an adjusted MRAE formulation that incorporates a penalty for missed detections. Specifically, for each ground-truth object that is not detected, we assign a relative error of 100% (i.e., 1.0), reflecting the assumption that a missed detection incurs the maximum relative error under this metric. This modification ensures that models with low object detection accuracy are properly penalized and are not mistakenly perceived as more accurate than they actually are. The final metric is:

$$\text{MRAE}_{\text{final}} = \frac{1}{N}\left(\sum_{i=1}^{N'} \frac{|\hat{Z}_i - Z_i|}{|Z_i|} + (N - N') \times 1\right) \times 100\% \tag{6}$$

Here, N denotes the total number of ground-truth object bounding boxes, and N' is the number of these objects that are successfully detected by the model. The first term computes the average relative error over detected objects, where $\hat{Z}_i$ is the predicted distance and Z_i is the ground-truth distance for the i-th object. The second term assigns a penalty of 100% error to each missed detection, ensuring that models with poor detection performance are properly penalized.

4.2 Problem Formulation

We aim to automatically select the optimal pruned YOLO model that not only minimizes hardware resource usage but also strictly satisfies a safety constraint derived from control theory. In our setting, system safety is quantified by MDRS. As introduced in section 3.1, a smaller MDRS implies better control performance. Hence, any model selected must operate within a predefined safety threshold for MDRS, which is determined based on control-theoretic simulation and task-specific requirements.

Formally, the problem can be formulated as:

$$\min_{\theta} \quad \mathit{Params}(\theta) \quad \text{subject to} \quad D(\theta) \leq D_{\text{safe}} \tag{7}$$

where:

- θ denotes the pruning configuration (e.g., pruning rate, pruning method),
- $\mathit{Params}(\theta)$ represents the hardware cost (model parameters),
- $D(\theta)$ denotes the **MDRS** as a function of the pruned model,
- D_{safe} is the maximum safety threshold, determined by the user and the specific application scenario.

Solving the constrained optimization problem in Eq. 7 presents a significant challenge due to the large and continuous search space, which spans pruning rates, methods, and layer selections. Identifying the optimal configuration θ^* that minimizes hardware cost while satisfying the safety constraint $D(\theta) \leq D_{\text{safe}}$ is computationally intractable via exhaustive search. This necessitates an efficient and predictive evaluation strategy to guide the search process. In the following sections, we introduce our approach to addressing this challenge.

4.3 System-Level Safety-Aware Performance Modeling

Given either a pre-trained or an untrained YOLO model and a specified control system, our framework automatically generates an optimally pruned model that satisfies control-theoretic safety constraints while minimizing hardware resource usage. The output includes both the selected pruning method and its corresponding pruning rate. If the model is untrained, our system can optionally perform the training stage on behalf of the user before proceeding with pruning and selection. The entire pipeline—from model training to control-safe pruning and efficient deployment—is fully automated. The proposed framework operates through the following core components: Control-Theoretic Simulation for Safety Characterization, Control-Aware Scaling Law Construction, Scaling Law-Based Single-Model Optimization under Safety and Hardware Constraints, and Scaling Law-Based Multi-Model Optimization under Joint Safety and Hardware Constraints.

4.3.1 Control-Theoretic Simulation for Safety Characterization. We adopt the MDRS measure [45] to evaluate the system-level safety of pruned models, following a control-aware estimation process. Given a pruning configuration θ, we first evaluate its corresponding MRAE ϵ and inference latency h. We then discretize the continuous dynamics function f using h to obtain the discrete-time dynamics f_h. The controller is also re-synthesized using h, resulting in a discrete-time control policy g_h. Finally, we compute the MDRS based on f_h, g_h, ϵ, and a fixed initial state set $X[0]$ using reachability analysis, following the approach in [45]. We denote the resulting system-level safety metric as $D(\theta) = D(\epsilon, h)$, indicating its dependence on both perception error and latency.

Since computing the MDRS for a given (ϵ, h) pair is significantly more efficient than pruning and fine-tuning the corresponding configuration θ, we precompute $D(\epsilon, h)$ values over a dense grid of (ϵ, h) pairs, using the reachability-based procedure described in Section 3.1. In our implementation, ϵ is sampled uniformly from 0 to 1 in increments of 0.01, and h is varied from 1 to 500 milliseconds with a step size of 1 ms. We then fit a continuous surface over the resulting (ϵ, h, D) tuples using nonlinear regression, and apply interpolation to estimate D values for use in later pruning configuration evaluation.

4.3.2 Control-Aware Scaling Law Construction. Our methodology is centered on efficient model selection via control-theoretic co-design, where pruning decisions are not solely guided by accuracy or compression ratio, but are instead jointly optimized under explicit control-theoretic safety constraints. To support this, we construct a direct mapping between the pruning rate and the system's MDRS, enabling efficient safety assessment of any pruning configuration without resorting to exhaustive empirical search, which is typically required by traditional pruning approaches.

We initially explored estimating system-level safety through a multi-stage mapping pipeline: from pruning rate to detection accuracy, from accuracy to distance estimation error, and finally from estimation error to control-level safety measured by MDRS. However, this approach introduces cumulative approximation errors across stages, particularly under aggressive pruning, which compromises the reliability of safety assessment. To address this, we instead adopt a direct mapping from pruning rate to MDRS, bypassing all intermediate variables. This design serves as a core constraint in our framework and provides a more robust and interpretable mechanism for pruning under safety guarantees. Empirical results further demonstrate that this direct mapping achieves more accurate and stable predictions of system safety compared to the multi-stage approximation strategy. The specific mathematical formulations of these mappings are provided in the following subsection 4.4.

4.3.3 Scaling Law-Based Single-Model Optimization under Safety and Hardware Constraints. Our framework integrates ten diverse pruning strategies to enhance generality and adaptability. For each method, we fit a corresponding scaling law capturing the relationship between pruning rate and system-level safety, quantified by MDRS. We then identify the safety-compliant pruning range for each method by locating the intersection points between the fitted curves and the predefined control-theoretic maximum safety threshold (D_{safe}). Under a fixed hardware platform and consistent inference settings—including input resolution, batch size, and data

type—we observe that higher pruning rates are generally associated with smaller model parameter sizes and faster inference times. This trend is further supported by the reduced size of the saved model checkpoints, which consistently shrink as pruning increases.

Based on this analysis, we select the optimal pruned model as the configuration that both satisfies the control-theoretic safety constraint—by lying within the safety-compliant pruning range—and minimizes hardware resource usage, such as model size or inference latency. This approach enables the selection of a single pruned model that optimally balances control safety and computational efficiency across multiple pruning methods.

4.3.4 Scaling Law-Based Multi-Model Optimization under Joint Safety and Hardware Constraints. Beyond this, our method also generalizes to more complex scenarios where multiple models must be deployed simultaneously on a shared hardware platform under a global resource constraint. After applying our single-model pruning strategy—ensuring that each individual model satisfies its own maximum allowable MDRS threshold while minimizing parameter count—we extend the approach to a multi-model deployment setting. The key challenge in this scenario is to select a combination of pruned models that, when deployed together, remain within a global hardware resource budget (e.g., a shared memory of 512 MB), while still meeting the safety requirements of each individual model. The system first prunes each model under its respective maximum safety threshold to identify the architecture with the minimal parameter count that satisfies the requirement. The resulting pruned models constitute the candidate pool for multi-model deployment.

For each candidate pair of models, the system computes the total parameter size. If the combined size exceeds the hardware budget, the pair is deemed infeasible and discarded. Otherwise, the system enters a model recovery phase, which aims to leverage the remaining parameter budget to further improve overall model quality.

In the recovery phase, the controller selectively increases the parameter count of one or more models, thereby reducing their respective MDRS to enhance system-level safety. By default, the model with the largest current MDRS is prioritized, as it represents the most vulnerable component in terms of safety. Alternatively, users may configure the system to prioritize models with tighter MDRS thresholds, which are typically more critical in safety-sensitive applications. This recovery process proceeds iteratively until the total parameter usage approaches the hardware budget or no further improvements are feasible. By incorporating this strategy, our method effectively balances strict safety and resource constraints while maximizing deployment performance, enabling efficient multi-model deployment in resource-constrained environments.

4.4 Scaling Law Based Automatic Pruning

To address the key challenge described in Eq. (7), we propose a pruning scaling law framework that characterizes the relationship between the pruning ratio and system-level safety. This scaling law takes the pruning ratio as input and predicts the corresponding system-level safety metric, providing a basis for safety-constrained pruning configuration selection. Once the scaling law is constructed, the model selection process can avoid performing reachability-based safety evaluation at every pruning ratio, thereby significantly reducing the computational cost of pruning decisions. This framework is inspired by prior observations that neural model performance often follows stable empirical scaling behaviors with respect to model capacity. We define a compact regression formulation that models the impact of the pruning ratio p on system-level safety, quantified by the maximum diameter of reachable sets (MDRS), denoted as D. Specifically, this formulation captures the mapping $p \longrightarrow D(p)$, where $D(p)$ represents the corresponding system-level safety metric.

4.4.1 Piecewise Scaling Law Modeling. To model the scaling law, we adopt a flexible piecewise regression formulation. Specifically, we partition the pruning ratio domain $[0, 1]$ into K intervals $\{[p_{i-1}, p_i]\}_{i=1}^{K}$, based on observed variations in system-level safety sensitivity. Within each interval, we fit a parametric function $f_i(p)$ to

capture the local relationship between the pruning ratio p and the diameter D. The overall scaling law is then defined as:

$$D(p) = \begin{cases} f_1(p) & \text{if } p \in [p_0, p_1] \\ f_2(p) & \text{if } p \in (p_1, p_2] \\ \quad\vdots & \\ f_K(p) & \text{if } p \in (p_{K-1}, p_K] \end{cases} \tag{8}$$

Each $f_i(p)$ can be instantiated using different regression strategies depending on the construction method. In this work, the uniform sampling baseline employs nonlinear least-squares fitting within each segment, whereas the proposed adaptive strategy instantiates the scaling law as a monotonic piecewise-linear function via isotonic regression.

4.4.2 Scaling Law Construction Strategies. While the piecewise formulation defines the functional form of the scaling law, its practical construction depends on how pruning ratios are sampled and how segment boundaries are determined. We consider two construction strategies: uniform sampling as a baseline and an adaptive sampling strategy proposed in this work.

Uniform Sampling A simple baseline strategy is to uniformly sample the pruning ratio p over $[0, 1]$ and fit the piecewise scaling law using nonlinear least squares. This approach is easy to implement and requires no prior knowledge of the scaling law shape, and can produce accurate fits when the sampling resolution is sufficiently dense.

However, uniform sampling is sample-inefficient due to its inability to adapt to non-uniform safety variation. In practice, system-level safety does not change uniformly with the pruning ratio: some pruning ranges are relatively flat, while others exhibit more rapid variation. As a result, uniform sampling tends to waste samples in flat regions while providing insufficient resolution in regions with steeper safety changes, leading to unnecessary reachability evaluation costs for a given level of fitting accuracy.

Adaptive Sampling We propose an adaptive sampling strategy for constructing the pruning scaling law, with the goal of significantly improving sample efficiency without sacrificing fitting accuracy. The procedure starts from a small set of initial pruning ratios selected to provide stratified coverage over the pruning domain $p \in [0, 1]$, ensuring adequate initial coverage. We impose a monotonicity constraint motivated by the empirical observation that safety degradation generally increases with pruning intensity. The pruning–safety scaling law is modeled as a monotonic piecewise-linear function constructed via isotonic regression followed by adaptive knot refinement.

We then perform **interval-level scoring** over adjacent pruning intervals to determine the next sampling location. Let the currently selected points, sorted by pruning ratio, be $\{(p_i, D_i)\}_{i=1}^{m}$. For each adjacent interval $[p_i, p_{i+1}]$, we define the normalized safety variation and pruning gap as

$$\Delta D_i = \frac{|D_{i+1} - D_i|}{\max_j |D_{j+1} - D_j|}, \qquad \Delta p_i = \frac{|p_{i+1} - p_i|}{\max_j |p_{j+1} - p_j|}, \tag{9}$$

Here, D_i denotes the system-level safety metric (MDRS) evaluated at pruning ratio p_i, and the index j ranges over all currently sampled adjacent intervals. The normalization by $\max_j(\cdot)$ is applied to place the safety variation and pruning gap on comparable scales, preventing either term from dominating the interval score due to magnitude differences. We then assign the interval score

$$s_i = \alpha \Delta D_i + (1 - \alpha) \Delta p_i, \tag{10}$$

where $\alpha \in [0, 1]$ balances *local sensitivity* (rapid safety variation) and *sampling sparsity* (wide uncovered gaps). In practice, we set $\alpha = 0.7$ empirically, which provides a stable balance across all evaluated systems without

additional tuning. At each iteration, a greedy refinement strategy is applied by selecting $i^\star = \arg\max_i s_i$ and adding a new pruning ratio near the interval midpoint

$$p_{\text{mid}} = \tfrac{1}{2}\big(p_{i^\star} + p_{i^\star+1}\big),$$

choosing the closest available point in a discrete candidate set. This strategy prioritizes regions exhibiting either large safety variation or insufficient sampling coverage.

After introducing each new sampling point, the fitting error of the current scaling law (e.g., measured by MRAE) is evaluated on a validation set. The adaptive process terminates once the validation error falls below a predefined threshold, at which point the scaling law is fixed as the final model. The test set is used only for final generalization evaluation and does not participate in sampling decisions. Extensive experiments show that the proposed strategy achieves comparable fitting accuracy while requiring significantly fewer reachability evaluations.

This piecewise formulation offers two key advantages: (1) it remains interpretable and analytically invertible, enabling constrained optimization over the pruning ratio p; and (2) its segment-wise structure allows for easy adaptation to more complex safety patterns or hardware-specific variants, without requiring retraining of the entire model.

Our system can automatically fit a scaling law for different hardware platforms—regardless of architectural differences such as latency profiles or execution characteristics. This hardware-adaptive design enables the framework to dynamically adjust segment boundaries and adapt functional forms based on platform-specific observations, ensuring consistent modeling accuracy across diverse deployment environments.

Moreover, although our primary objective is to model the relationship between pruning rate and system-level safety (quantified by MDRS), the proposed methodology in principle readily extends to other critical metrics, including mAP, inference latency, and memory usage. This flexibility enables a unified and automated framework for multi-dimensional analysis of pruning effects. Experimental validation on NVIDIA RTX 4090 and 4060 GPUs, with consistently low regression errors, demonstrates the robustness and generality of our framework in capturing architecture-invariant safety degradation patterns. These results further confirm the scalability and hardware adaptability of the proposed scaling-law-based pruning strategy.

5 Experiments

5.1 Objective

The primary goal of our experiments is to demonstrate that the proposed control-aware pruning system outperforms baseline search methods by achieving faster search speed and identifying pruned models with the smallest number of parameters while satisfying the target D_{safe} constraint. Our experiments focus on YOLOv8-based distance estimation in autonomous driving scenarios, where the objective is to prune the YOLOv8 model to minimize hardware deployment cost while preserving system-level safety, as quantified by the MDRS. We compare the effectiveness and efficiency of our system against several baseline search methods.

To evaluate hardware deployment cost, we use the number of model parameters as the primary metric. For the same model architecture deployed on an identical hardware platform, a smaller number of parameters typically results in lower memory usage, reduced computational load, and more efficient resource utilization. Consequently, minimizing the number of parameters directly correlates with reduced hardware demands, making the pruned model more suitable for deployment in resource-constrained environments such as embedded systems or edge devices.

5.2 Experimental Setup

Our experiments are conducted on the KITTI dataset [39], a widely used benchmark in autonomous driving research that provides synchronized and calibrated multi-sensor data, including stereo cameras and LiDAR, for perception tasks such as object detection and distance estimation. For distance supervision, we use the provided Z-axis distance, which measures the object's depth along the optical axis and serves as ground truth.

The YOLOv8 model is trained and pruned directly on the raw KITTI dataset without additional data augmentation or preprocessing. After training, bounding box features extracted from YOLOv8 outputs are used to train a linear regression model for distance estimation. Specifically, the model learns to map bounding box characteristics, such as width and height, to the corresponding Z-axis distance. On the validation set, the regression model achieves a mean relative absolute error (MRAE) of approximately 6%, demonstrating reliable distance prediction for downstream control tasks. The MDRS is computed using the YOLOv8 inference latency and the MRAE derived from the regression-based distance estimation.

5.2.1 Baseline Methods for Experiments 1 and 3.

- **Brute-Force Search:** Exhaustively evaluates all pruning rates. Records the configuration with minimum relative error under the MDRS threshold.
- **Depth-First Search (DFS):** Starts from the middle of pruning rates. Explores both directions, prioritizing solutions where MDRS is below the target and error is minimized. Terminates early upon reaching the step or depth limit, and returns the best pruning rate and MDRS found.
- **Breadth-First Search (BFS):** Starts from the middle of pruning rates. Explores adjacent configurations using a first-in, first-out (FIFO) queue, comparing MDRS at each step. Updates the best match found. Stops when a valid solution is reached or search step limit is reached.
- **Binary Search:** Starts from the middle of the sorted pruning rates. Compares MDRS with the target and narrows the search range accordingly. Continues until the best solution is found or the range is exhausted. Much faster than linear search methods.

5.2.2 Evaluation Metrics.

$$\text{MRAE}_{\text{Diameter}} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{D}_i - D_i|}{|D_i|} \times 100\% \tag{11}$$

where D_i denotes the ground truth diameter for the i-th sample, and $\hat{D}_i$ denotes the predicted diameter.

All experiments were carried out on a single NVIDIA RTX 4090 GPU, and all performance metrics—including inference latency and pruning-related evaluations—were measured under an identical hardware setting to ensure fair comparison.

5.2.3 Control System Models. We conducted the control part experiments using the following two dynamical systems:

F1/10 Car (F1). Our first model captures the motion of an F1/10 car [33] and is adapted from [20]. The controller aims to keep the car driving straight in the positive x direction in a plane. The underlying vehicle dynamics follow a bicycle model—a standard nonlinear model for lateral vehicle motion—which is linearized around the straight-line operating point, a common practice in automotive control design [25]. The system dynamics are:

$$\dot{x}(t) = \begin{bmatrix} 0 & 6.5 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 19.685 \end{bmatrix} u(t).$$

Cruise Control (CC). Our second model is a cruise control system adapted from [19, 32] that regulates the vehicle speed to follow the driver's command. The system dynamics are:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6.0476 & -5.2856 & -0.238 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 2.4767 \end{bmatrix} u(t).$$

In both cases, we discretize the continuous dynamics with the inference latency h, and design a controller using discrete Linear-Quadratic Regulator (LQR) control with state cost matrix Q and control cost matrix R both being identity matrices.

5.3 Experiment 1: Ultra-Fast Control-Aware Neural Network Pruning Using Scaling Laws

This experiment evaluates the search efficiency of our control-aware pruning system in comparison to several baseline methods. We use search time as the sole evaluation metric and decompose the overall cost of the proposed approach into two components: (i) an *offline* cost for constructing the pruning scaling law, which is incurred once per pruning method and control system, and (ii) an *online* cost for identifying a pruning configuration that satisfies a given MDRS constraint. In this experiment, we primarily evaluate the *online search time*, which dominates deployment-time decision-making and is the relevant cost for real-time or iterative pruning scenarios.

For baseline methods, identifying the model with the smallest parameter count that also satisfies the MDRS constraint would require exhaustively evaluating all pruning rates, due to the absence of a scaling law for guidance. Such an exhaustive search leads to the worst-case search time for all baselines. To enable a fair and practical comparison, this constraint was relaxed: baselines were only required to return a model that satisfies the MDRS constraint, without necessarily minimizing the parameter count. In contrast, our control-aware pruning system is designed to satisfy a stricter objective: identifying the model with the smallest number of parameters that still satisfies the target MDRS constraint. This ensures structural compliance while maximizing model compactness for efficient deployment.

To ensure a consistent comparison, all baseline methods are initialized with a pruning rate of 0.001 and terminate at 0.4, with a fixed increment of 0.001 per step. This schedule standardizes the search space across methods. In contrast, the search path in our control-aware pruning algorithm is dynamically determined through a control feedback mechanism guided by a scaling law. To minimize the impact of hardware variability, we perform multiple runs to measure the time required for a single pruning operation and MDRS computation. The average time across runs is used as the standardized time cost per search step, ensuring that all methods are evaluated under consistent computational assumptions. For each baseline method, we recorded the MDRS corresponding to each pruning rate in a structured table. Based on this pre-computed table, we simulated the behavior of each search algorithm and counted the number of steps required to identify the pruning rate that minimizes the relative error with respect to the target MDRS. This approach ensures fair comparison since all methods share identical evaluation cost per pruning configuration. The final reported search time is calculated as the product of the average time cost per step and the total number of search steps for each method.

Table 1. Search Methods and Online Search Times at D_{safe}=20 (F1)

Search Method	Search Time (s)
Scaling Laws	**841.35**
Brute Force Search	124800.00
Binary Search	2683.20
Depth-First Search (DFS)	20498.40
Breadth-First Search (BFS)	124800.00

Table 2. Search Methods and Online Search Times at D_{safe}=60 (CC)

Search Method	Search Time (s)
Scaling Laws	**841.35**
Brute Force Search	124800.00
Binary Search	2652.00
Depth-First Search (DFS)	17659.20
Breadth-First Search (BFS)	124800.00

As shown in Table 1, where the target MDRS is set to 20, the Scaling Law method achieves the highest search efficiency, requiring only 841.35 seconds to identify a pruned model that satisfies the constraint. In contrast, Brute-Force Search and Breadth-First Search (BFS) are the least efficient, each taking 124,800 seconds due to the exhaustive exploration of all possible pruning rates. Binary Search and Depth-First Search (DFS) perform better, with search times of 2,683.2 seconds and 20,498.4 seconds, respectively. The significant advantage of the Scaling Law method lies in its ability to model the relationship between model structure and MDRS, enabling rapid localization of the search space and efficient adjustment of pruning rates. In contrast, other methods lack such guidance and must rely on exhaustive or heuristic search strategies.

For a target MDRS of 60, the search times follow a trend similar to that in Table 2. The Scaling Law method remains the most efficient, completing the search in 841.35 seconds. Binary Search also performs well, requiring 2,652.0 seconds, while DFS takes 17,659.2 seconds. As expected, Brute-Force Search and Breadth-First Search (BFS) are the least efficient, each taking 124,800 seconds due to exhaustive exploration. These results demonstrate that Scaling Laws can directly focus on the most promising regions of the search space, substantially reducing both search path length and total time, thereby improving overall search efficiency.

Table 3. Search Methods and Corresponding Search Times at D_{safe}=1 under F1 or CC

Search Method	Search Time (s)
Scaling Laws	**0.09**
Brute Force Search	124800.00
Binary Search	124800.00
Depth-First Search (DFS)	124800.00
Breadth-First Search (BFS)	124800.00

As shown in Table 3, when the target MDRS is set to 1—smaller than the model's best achievable MDRS—all baseline methods, except the Scaling Law method, must exhaustively explore the entire pruning space to confirm that the target is infeasible, resulting in a search time of 124,800 seconds. In contrast, the Scaling Law method leverages structural relationships to promptly determine the infeasibility of the target, completing the search in only 0.09 seconds. This capability significantly reduces unnecessary computational overhead and demonstrates the advantage of our approach in handling unreachable constraints.

Table 4. Offline Scaling Law Construction Cost

System	Number of Pruning Methods	Offline Time (s)
F1/10 Car (F1)	10	5576.92
Cruise Control (CC)	10	5785.26

Offline cost interpretation. The offline time reported in Table 4 corresponds to the *one-time* cost of constructing scaling laws for all pruning methods under a given control system. Importantly, this offline cost is incurred only once per (model, control system) pair and is independent of the number of subsequent pruning queries. Once the scaling laws are constructed, all online pruning decisions—including feasibility checking and optimal pruning-rate selection under a specified D_{safe}—are performed without additional system-level safety evaluations, as shown in Tables 1–3. As a result, the offline construction cost is amortized across all future pruning tasks, making the proposed approach particularly efficient in repeated or real-time deployment scenarios. In contrast, baseline search methods require a substantially larger number of system-level safety evaluations over the pruning space for each new target constraint, resulting in much higher cumulative search cost.

The proposed system employs a scaling law to efficiently predict whether a pruned model can satisfy a given MDRS constraint. For a specified target MDRS, the system rapidly determines the feasibility of achieving the constraint and identifies the pruned model with the smallest number of parameters that satisfies it, thereby minimizing hardware deployment costs. By avoiding exhaustive searches, the approach substantially reduces computational overhead and offers a practical solution for control-aware real-time applications.

5.4 Experiment 2: Optimal Pruning Method and Model Selection

Our primary contribution is the development of a system that automatically selects the optimal pruning method and corresponding model under given constraints. The system dynamically evaluates multiple pruning strategies, comparing model parameters and MDRS to ensure that the selected configuration satisfies the MDRS constraint while minimizing the remaining parameter count. While Experiment 1 validates the efficiency of the proposed approach by demonstrating its ability to identify the optimal pruning rate significantly faster than baseline methods, this experiment evaluates the quality of the selected model and confirms that it satisfies the dual objective of meeting the MDRS constraint while minimizing the number of parameters.

In this setup, all methods were constrained to a similar parameter budget to ensure a fair comparison. By measuring the resulting MDRS while enforcing the MDRS constraint, we verified that no alternative method could achieve a smaller parameter count without violating the constraint. These results confirm that our system consistently selects the optimal pruning configuration under the given safety requirements.

Table 5. Comparison of Pruning Methods under a Safety Constraint of $D_{\text{safe}} = 9$ for the F1 System

Params (M)	Pruning Method	MDRS
49.14	prune_anything	490.18
49.13	**network_slimming**	**8.54**
49.18	structured_1_0	11455.29
49.18	structured_1_1	207647.05
49.18	structured_2_0	18512.33
49.18	structured_2_1	258083.00
49.18	structured_inf_0	258083.00
49.18	structured_inf_1	212547.86
49.18	structured_-inf_0	258083.00
49.18	structured_-inf_1	235943.31

Note: Results are obtained using the proposed scaling-law-guided pruning framework. The highlighted configuration satisfies the safety constraint while achieving the minimum parameter count.

Table 6. Comparison of Pruning Methods under a Safety Constraint of $D_{\text{safe}} = 49$ for the CC System

Params (M)	Pruning Method	MDRS
45.59	prune_anything	227.86
45.34	**network_slimming**	**46.46**
45.64	structured_1_0	227.91
45.64	structured_1_1	224.16
45.64	structured_2_0	228.55
45.64	structured_2_1	228.55
45.64	structured_inf_0	228.55
45.64	structured_inf_1	225.75
45.64	structured_-inf_0	228.55
45.64	structured_-inf_1	222.52

Note: All pruning methods, except the first two, follow the format `structured_<norm>_<dim>`, where `<norm>` specifies the norm used for pruning (ℓ_1, ℓ_2, ℓ_∞, or $\ell_{-\infty}$), and `<dim>` indicates the pruning dimension.

Theoretically, a higher parameter count suggests that more layers are preserved in the pruned model, which generally leads to better retention of the original structure. However, due to differences in pruning strategies, it is not possible to achieve identical parameter counts across methods. To ensure fairness, we intentionally selected slightly higher parameter budgets for the alternative methods, allowing them to retain more of the original model structure during comparison. The model selected by our system is highlighted in bold in the table. Tables 5 and 6 present a comparison of different pruning methods in terms of parameter count and MDRS under two experimental settings. The target MDRS thresholds are set to 9 and 49, corresponding to control objectives F1 and CC, respectively. The results illustrate that `network_slimming` not only satisfies the MDRS constraints but also achieves a smaller MDRS compared to alternative pruning methods.

Table 7. Best Pruning Rate and MDRS Results under 950 s Time Budget

Method	F1 System ($D_{\text{safe}} = 60$)			CC System ($D_{\text{safe}} = 83$)		
	Prune	**MDRS**	**Params**	**Prune**	**MDRS**	**Params**
Brute Force	0.001	3.18	68.13	0.002	8.82	67.99
Binary	0.099	52.31	61.38	0.149	80.98	57.98
DFS	0.201	15.12	47.47	0.203	29.37	47.27
BFS	0.200	14.48	47.53	0.202	29.24	47.37
Scaling Laws	**0.226**	**53.67**	**45.26**	**0.248**	**82.61**	**43.44**

Table 8. Best Pruning Rate and MDRS Results under 4500 s Time Budget

Method	F1 System ($D_{\text{safe}} = 27$)			CC System ($D_{\text{safe}} = 42$)		
	Prune	**MDRS**	**Params**	**Prune**	**MDRS**	**Params**
Brute Force	0.015	3.38	67.12	0.015	9.24	67.12
Binary	0.223	26.98	53.00	0.226	41.68	52.81
DFS	0.213	26.07	46.40	0.214	40.82	46.33
BFS	0.206	21.05	47.00	0.208	37.13	46.82
Scaling Laws	**0.213**	**26.07**	**46.40**	**0.221**	**41.80**	**45.73**

Specifically, in the F1 system under $D_{\text{safe}} = 9$, `network_slimming` achieves a MDRS of 8.54 with approximately 49.13 million parameters, significantly outperforming all other methods, whose MDRS range from 490.18 to 258,083. Similarly, in the CC scenario, `network_slimming` attains a MDRS of 46.46 with around 45.34 million parameters, again surpassing competing methods with MDRS between 224.16 and 228.55. These results demonstrate that our system effectively selects the optimal pruning method, ensuring that the resulting model achieves the smallest parameter count while satisfying both system-level safety and efficiency constraints.

5.5 Experiment 3: Optimal Model Identification Under Time Constraints

This experiment evaluates the efficiency of our control-aware pruning system relative to other search methods under a fixed time constraint. Specifically, we compare the models returned by each method within the same search duration, assessing their parameter counts and corresponding MDRS. We adopt the same baseline methods as in Experiment 1 for comparison. Four experiments are conducted under two different time constraints and two dynamical systems (CC and F1). Across all scenarios, our system consistently produces models with the smallest parameter counts while maintaining MDRS below the specified thresholds.

From Tables 7 to 8, we systematically compare various pruning strategies under different target objectives and search time constraints. Overall, the Scaling Laws method exhibits consistently strong performance across all settings. It not only satisfies the system-level safety requirements in terms of MDRS, but also achieves the smallest parameter counts among all methods, demonstrating superior model compression capability. These results collectively validate that our system achieves an effective balance among system-level safety, model compactness, and search efficiency, illustrating strong generalization capability and practical value across diverse objectives and time constraints.

5.6 Experiment 4: Controller-Guided Model Selection Strategy for Multi-Model Deployment

This experiment aims to validate the effectiveness of the proposed controller-guided model selection strategy in a multi-model deployment scenario. The objective is to assess whether the strategy can efficiently identify

competitive deployment combinations that simultaneously satisfy model safety requirements—measured by MDRS—and hardware resource constraints—measured by parameter count.

A candidate pool of 180 pruned models was constructed, each corresponding to a distinct pruning rate and covering a broad range of MDRS and parameter footprint combinations. The deployment task involves selecting models for two dynamical systems, F1 and CC, under the following constraints: the MDRS of the CC model must not exceed 17, the MDRS of the F1 model must not exceed 8, and the total model size must remain under 400 MB. The objective is to determine a model pair that satisfies all constraints while maximizing overall safety performance.

Table 9. Search Time Comparison Between Methods

Search Method	Search Time (s)
Multi-methods	**872.55**
Brute Force	252,720.00

Table 10. MDRS and Parameter Sum of Selected Model Combinations

MDRS Sum	Parameter Sum (M)
16.33	**104.66**
13.76	104.51
17.33	101.64
19.84	104.66
20.47	101.64
24.04	98.77

Table 9 compares the search times of different model selection strategies. The proposed multi-method controller identified a valid model combination in 872.55 seconds, whereas the brute-force method required over 252,720.00 seconds, representing a substantial improvement in search efficiency by several orders of magnitude.

Table 10 presents several candidate combinations of two models, each satisfying the MDRS and parameter size constraints while exhibiting varying levels of safety and resource usage. Among them, the final deployment configuration has a total MDRS of 16.33 (CC: 8.95, F1: 7.38). Although the summation of MDRS values across models lacks direct physical interpretability, it serves as a practical aggregate metric to facilitate quantitative comparison among candidate configurations. Assuming full precision (float32) in YOLOv8, the 104.66 million parameters result in a total model size of approximately 399.24 MB, calculated using Eq. 12, which remains within the 400 MB parameter budget. The selected model achieves a desirable balance between system safety and resource efficiency. This configuration satisfies all deployment constraints while effectively utilizing the available parameter budget.

$$\text{Model Size (MB)} = \frac{\text{Parameters} \times \text{Bytes per parameter}}{1024^2} \tag{12}$$

Although this solution is not globally optimal, it represents a high-quality suboptimal result obtained through an efficient search process, underscoring the controller's practicality under deployment constraints. Collectively, these results demonstrate the controller's ability to rapidly identify feasible sub-optimal configurations and further refine them through model recovery, making it well-suited for real-world multi-model deployment scenarios with stringent hardware limitations.

5.7 Accuracy and Efficiency of Scaling Law Construction

This section evaluates the accuracy and efficiency of scaling law construction for modeling the pruning–safety relationship. We first establish a dense-sampling baseline to verify that the relationship can be accurately captured under exhaustive evaluations, and then investigate whether comparable accuracy can be recovered using adaptive sampling with significantly fewer reachability analyses.

5.7.1 Dense Baseline: Scaling Law Accuracy under Exhaustive Sampling. We first evaluate the accuracy of the proposed scaling law under a fully observed setting with densely evaluated pruning configurations. We collect a dense dataset using ten pruning methods on two dynamical systems (CC and F1), resulting in 20 (method, system) combinations. For each configuration, pruning ratios are densely sampled over $[0, 0.4]$ to capture safety degradation trends. Each data point includes the pruning ratio, the corresponding reachable set diameter, and auxiliary performance metrics such as mAP.

Following Section 4.4, we fit a piecewise parametric function $D(p;\text{method})$ for each (pruning method, system) pair to model the mapping from pruning ratio p to the predicted reachable set diameter D. Under this fully observed setting, the dataset is randomly split into 80% for training and 20% for testing, and is used to train and evaluate the scaling law by assessing its accuracy in predicting the reachable set diameter for each configuration. Prediction accuracy is assessed by comparing the predicted diameter D_{pred} with the diameter D_{true} obtained from reachability-based safety evaluation, using mean relative absolute error (MRAE) as the evaluation metric.

Table 11. Summary of CC Pruning Methods and MRAE

Pruning Method	Average Diameter MRAE
Structured_1_0	0.0359
Structured_1_1	0.0040
Structured_2_0	0.0309
Structured_2_1	0.0040
Pruning_Anything	0.0540
Structured_inf_0	0.0355
Structured_inf_1	0.0131
Structured_negative_inf_0	0.0805
Structured_negative_inf_1	0.0860
Network_Slimming	0.0951

Table 12. Summary of F1 Pruning Methods and MRAE

Pruning Method	Average Diameter MRAE
Structured_1_0	0.0324
Structured_1_1	0.0222
Structured_2_0	0.0756
Structured_2_1	0.0215
Pruning_Anything	0.0590
Structured_inf_0	0.0569
Structured_inf_1	0.0131
Structured_negative_inf_0	0.0805
Structured_negative_inf_1	0.0829
Network_Slimming	0.0334

As summarized in Tables 11 and 12, the fitted scaling laws achieve MRAE below 10% across all pruning methods and both control systems. These results indicate that, when the pruning–safety relationship is sufficiently observed, it can be accurately captured by the proposed piecewise scaling law formulation, providing a reliable reference for subsequent experiments.

5.7.2 Dynamic Scaling Law Construction with Adaptive Sampling. While dense sampling provides a reliable reference for the pruning–safety relationship, exhaustively evaluating all pruning configurations is computationally prohibitive due to the high cost of reachability-based safety evaluation. We therefore investigate whether comparable accuracy can be achieved using adaptive sampling with significantly fewer evaluations, while maintaining a strict separation between model construction and evaluation to ensure fairness and unbiased assessment.

Pruning configurations are partitioned into disjoint training, validation, and test sets. Adaptive sampling draws exclusively from the training set; the validation set (10%) controls termination, and the test set is used solely for final generalization evaluation. We additionally evaluate extreme regimes where up to 50% of configurations are held out to assess robustness under limited observability.

Adaptive sampling starts from a small set of uniformly distributed pruning ratios to ensure coarse coverage of the pruning domain. New sampling points are selected using the interval-level scoring mechanism introduced earlier, balancing safety variation and sampling sparsity ($\alpha = 0.7$). After each update, a provisional scaling law is fitted, and validation MRAE determines whether further sampling is required. The process terminates once the validation error falls below the predefined threshold. All experiments are repeated five times with randomized initialization orders, and results are reported as mean and standard deviation.

Table 13. Dynamic Scaling Law on CC

Pruning Method	Diameter MRAE (mean ± std)
Structured_1_0	0.0351 ± 0.0059
Structured_1_1	0.0801 ± 0.0179
Structured_2_0	0.0346 ± 0.0046
Structured_2_1	0.0274 ± 0.0145
Pruning_Anything	0.0379 ± 0.0052
Structured_inf_0	0.0557 ± 0.0244
Structured_inf_1	0.0361 ± 0.0165
Structured_negative_inf_0	0.0727 ± 0.0286
Structured_negative_inf_1	0.0487 ± 0.0181
Network_Slimming	0.0299 ± 0.0079

Table 14. Dynamic Scaling Law on F1

Pruning Method	Diameter MRAE (mean ± std)
Structured_1_0	0.0459 ± 0.0121
Structured_1_1	0.0573 ± 0.0349
Structured_2_0	0.0445 ± 0.0068
Structured_2_1	0.0309 ± 0.0116
Pruning_Anything	0.0593 ± 0.0234
Structured_inf_0	0.0472 ± 0.0116
Structured_inf_1	0.0595 ± 0.0364
Structured_negative_inf_0	0.0641 ± 0.0130
Structured_negative_inf_1	0.0427 ± 0.0092
Network_Slimming	0.0477 ± 0.0153

For accuracy comparison, we summarize the dense baseline results obtained under the most informative sampling condition in Tables 11 and 12, while the corresponding dynamic scaling law results are reported in Tables 13 and 14. Across all pruning methods and control systems, adaptive sampling achieves test-set MRAE comparable to the dense baseline while requiring only a small fraction of pruning evaluations. Reported MRAE values correspond to the 50% unseen generalization split, while the remaining 10% split is used solely for early stopping during adaptive sampling. Although moderate performance variations are observed in a few configurations, the overall generalization error remains stable and consistent with the dense-sampling reference. Notably, this accuracy is achieved with significantly fewer safety evaluations, demonstrating a favorable trade-off between predictive accuracy and computational cost.

Table 15. Average Time Cost Comparison for Scaling Law Construction

Method	F1 Time (s)	CC Time (s)
Dense Baseline (Exhaustive Sampling)	25,801.28	30,384.62
Dynamic (Adaptive Sampling)	5,576.92	5,785.26

5.7.3 Accuracy–Cost Trade-off. In addition to prediction accuracy, we compare the control simulation cost required to construct the scaling law. For both the dense baseline and the adaptive setting, pruning configurations with extremely large diameters are excluded during construction, as such points are far beyond the safety-relevant range and would otherwise dominate simulation time. As summarized in Table 15, adaptive scaling law construction reduces the total control simulation time by a factor of approximately 4–6× compared to the dense baseline, while achieving comparable prediction accuracy. Both settings apply the same practical filtering strategy during construction, ensuring a fair comparison of simulation cost. Together with the accuracy results, this demonstrates that the proposed adaptive scaling law construction achieves a favorable trade-off between prediction accuracy and simulation cost, enabling practical safety-aware pruning under realistic computational budgets.

6 Conclusion and Future Work

In this work, we proposed a control-aware neural network pruning framework that integrates system-level safety metrics into the model compression process. By incorporating a control-theoretic diameter constraint MDRS and constructing pruning scaling laws for YOLOv8-based distance estimation, our method enables efficient, safe, and fully automated deployment of compressed models in real-world cyber-physical systems (CPS). Empirical evaluations on autonomous driving tasks demonstrate that our approach not only achieves substantial pruning

speedup (up to 148.33×), but also consistently identifies the most efficient models that satisfy strict safety requirements under both single-model and multi-model deployment scenarios.

In the future, we plan to extend our framework to a broader range of perception tasks, such as semantic segmentation and lane detection. Additionally, we aim to explore joint pruning in multi-controller coordination scenarios, in which multiple safety-critical subsystems operate concurrently and pruning decisions must be jointly optimized to satisfy global system constraints under limited computational budgets. We also believe that incorporating inter-model dependency strategies may further enhance the applicability of our framework to time-sensitive, resource-constrained CPS environments. Finally, while our current case studies use linear dynamical systems, extending the framework to nonlinear system models (e.g., bicycle models for autonomous driving) is a natural next step, as the reachability analysis techniques we build upon generalize to nonlinear dynamics through conservative linearization [3].

Acknowledgments

This work was supported in part by the National Science Foundation under Grant Nos. CNS-2328972, CCF-2324937, and CNS-2122320. Samarjit Chakraborty is a Fellow of the TUM Institute for Advanced Study in Germany and is partially supported by a Dieter Schwarz Courageous Research Grant.

References

[1] Matthias Althoff and John M. Dolan. 2014. Online Verification of Automated Road Vehicles Using Reachability Analysis. *IEEE Transactions on Robotics* 30, 4 (2014), 903–918. doi:10.1109/TRO.2014.2312453

[2] Matthias Althoff, Goran Frehse, and Antoine Girard. 2021. Set Propagation Techniques for Reachability Analysis. *Annual Review of Control, Robotics, and Autonomous Systems* 4, Volume 4, 2021 (May 2021), 369–395. doi:10.1146/annurev-control-071420-081941

[3] Matthias Althoff, Olaf Stursberg, and Martin Buss. 2008. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *2008 47th IEEE Conference on Decision and Control (CDC)*. 4042–4048. doi:10.1109/CDC.2008.4738704

[4] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. 2017. Control Barrier Function Based Quadratic Programs for Safety Critical Systems. *IEEE Trans. Automat. Control* 62, 8 (2017), 3861–3876. doi:10.1109/TAC.2016.2638961

[5] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J. Tomlin. 2017. Hamilton-Jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2242–2253. doi:10.1109/CDC.2017.8263977

[6] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. 2019. JuliaReach: A Toolbox for Set-Based Reachability. In *Hybrid Systems: Computation and Control (HSCC)*. 39–44. doi:10.1145/3302504.3311804

[7] Zibin Cai, Rongrong Chen, Ziyi Wu, and Wuyang Xue. 2024. YOLOv8n-FAWL: Object Detection for Autonomous Driving Using YOLOv8 Network on Edge Devices. *IEEE Access* 12 (2024), 158376–158387. doi:10.1109/ACCESS.2024.3480976

[8] Wanli Chang and Samarjit Chakraborty. 2016. Resource-aware Automotive Control Systems Design: A Cyber-Physical Systems Approach. *Found. Trends Electron. Des. Autom.* 10, 4 (2016), 249–369.

[9] Xiaodong Chen, Yuxuan Hu, Xiaokang Zhang, Yanling Wang, Cuiping Li, Hong Chen, and Jing Zhang. 2024. P^2 Law: Scaling Law for Post-Training After Model Pruning. *arXiv preprint arXiv:2411.10272* (2024).

[10] Pranav Singh Chib and Pravendra Singh. 2024. Recent Advancements in End-to-End Autonomous Driving Using Deep Learning: A Survey. *IEEE Transactions on Intelligent Vehicles* 9, 1 (2024), 103–118. doi:10.1109/TIV.2023.3318070

[11] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Formal Modeling and Analysis of Timed Systems (FORMATS) (Lecture Notes in Computer Science, Vol. 6246)*. 92–106. doi:10.1007/978-3-642-15297-9_9

[12] Danfeng Du and Yuchen Xie. 2025. Vehicle and Pedestrian Detection Algorithm in an Autonomous Driving Scene Based on Improved YOLOv8. *Journal of Transportation Engineering, Part A: Systems* 151, 1 (2025), 04024095. doi:10.1061/JTEPBS.TEENG-8446

[13] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. 2023. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16091–16101.

[14] Georg Georgakos et al. 2013. Reliability challenges for electric vehicles: from devices to architecture and systems software. In *50th Annual Design Automation Conference (DAC)*.

[15] Bineet Ghosh, Clara Hobbs, Shengjie Xu, Parasara Sridhar Duggirala, James H. Anderson, P. S. Thiagarajan, and Samarjit Chakraborty. 2022. Statistical Hypothesis Testing of Controller Implementations Under Timing Uncertainties. In *2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 11–20. doi:10.1109/RTCSA55878.2022.00008

[16] Antoine Girard. 2005. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control (HSCC)*. 291–305. doi:10.1007/978-3-540-31954-2_19

[17] Antoine Girard, Colas Le Guernic, and Oded Maler. 2006. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In *Hybrid Systems: Computation and Control (HSCC)*. 257–271. doi:10.1007/11730637_21

[18] Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. 2011. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *16th Asia South Pacific Design Automation Conference (ASP-DAC)*.

[19] Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. 2014. Relaxing Signal Delay Constraints in Distributed Embedded Controllers. *IEEE Transactions on Control Systems Technology* 22, 6 (Nov. 2014), 2337–2345. Conference Name: IEEE Transactions on Control Systems Technology. doi:10.1109/TCST.2014.2301795

[20] Clara Hobbs, Bineet Ghosh, Shengjie Xu, Parasara Sridhar Duggirala, and Samarjit Chakraborty. 2022. Safety Analysis of Embedded Controllers Under Implementation Platform Timing Uncertainties. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (Nov. 2022), 4016–4027. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. doi:10.1109/TCAD.2022.3198905

[21] Clara Hobbs, Shengjie Xu, Bineet Ghosh, Enrico Fraccaroli, Parasara Sridhar Duggirala, and Samarjit Chakraborty. 2024. Quantitative Safety-Driven Co-Synthesis of Cyber-Physical System Implementations. In *2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*. Hong Kong, Hong Kong, 99–110. doi:10.1109/ICCPS61052.2024.00016

[22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556* (2022).

[23] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. *Ultralytics YOLOv8*. GitHub repository. https://github.com/ultralytics/ultralytics

[24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[25] Hassan K. Khalil. 2002. *Nonlinear Systems* (3rd ed.). Prentice-Hall, Upper Saddle River, NJ.

[26] Pratyush Kumar et al. 2012. A hybrid approach to cyber-physical systems verification. In *49th Annual Design Automation Conference (DAC)*.

[27] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. 2024. Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[28] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*. 2736–2744.

[29] Yuechen Luo, Yusheng Ci, Shixin Jiang, and Xiaoli Wei. 2024. A novel lightweight real-time traffic sign detection method based on an embedded device and YOLOv8. *J. Real-Time Image Process.* 21, 2 (Jan. 2024), 10 pages. doi:10.1007/s11554-023-01403-7

[30] Ian M. Mitchell, Alexandre M. Bayen, and Claire J. Tomlin. 2005. A time-dependent Hamilton–Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Control* 50, 7 (2005), 947–957. doi:10.1109/TAC.2005.851439

[31] Katsuhiko Ogata. 2010. *Modern Control Engineering* (5th ed.). Prentice-Hall, Upper Saddle River, NJ.

[32] Khairuddin Osman, Mohd. Fuaad Rahmat, and Mohd Ashraf Ahmad. 2009. Modelling and controller design for a cruise control system. In *2009 5th International Colloquium on Signal Processing & Its Applications*. 254–258. doi:10.1109/CSPA.2009.5069228

[33] Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. 2020. F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning. In *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*. 77–89. https://proceedings.mlr.press/v123/o-kelly20a.html

[34] PyTorch Contributors. 2020. Pruning Tutorial. https://pytorch.org/tutorials/intermediate/pruning_tutorial.html. Accessed: 2025-04-07.

[35] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). arXiv:1506.02640 http://arxiv.org/abs/1506.02640

[36] Debayan Roy et al. 2016. Multi-Objective Co-Optimization of FlexRay-Based Distributed Control Systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*.

[37] Debayan Roy et al. 2018. Semantics-Preserving Cosynthesis of Cyber-Physical Systems. *Proc. IEEE* (2018).

[38] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. 2023. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cognitive Robotics* 3 (2023), 54–70.

[39] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. 2017. Sparsity Invariant CNNs. In *International Conference on 3D Vision (3DV)*.

[40] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724* (2024).

[41] Jiawen Xu, Matthias Kovatsch, Denny Mattern, Filippo Mazza, Marko Harasic, Adrian Paschke, and Sergio Lucia. 2022. A review on AI for smart manufacturing: Deep learning challenges and solutions. *Applied Sciences* 12, 16 (2022), 8239.

[42] Shengjie Xu, Bineet Ghosh, Clara Hobbs, Enrico Fraccaroli, Parasara Sridhar Duggirala, and Samarjit Chakraborty. 2023. Statistical Approach to Efficient and Deterministic Schedule Synthesis for Cyber-Physical Systems. In *Automated Technology for Verification and Analysis (Lecture Notes in Computer Science)*, Étienne André and Jun Sun (Eds.). Cham, 312–333. doi:10.1007/978-3-031-45329-8_15

[43] Shengjie Xu, Bineet Ghosh, Clara Hobbs, P. S. Thiagarajan, and Samarjit Chakraborty. 2023. Safety-Aware Flexible Schedule Synthesis for Cyber-Physical Systems Using Weakly-Hard Constraints. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASPDAC '23)*. New York, NY, USA, 46–51. doi:10.1145/3566097.3567848

[44] Shengjie Xu, Clara Hobbs, Yukai Song, Bineet Ghosh, Sharmin Aktar, Lei Yang, Yi Sheng, Weiwen Jiang, Jingtong Hu, Parasara Sridhar Duggirala, and Samarjit Chakraborty. 2024. Neural Architecture Sizing for Autonomous Systems. In *2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*. 289–290. doi:10.1109/ICCPS61052.2024.00040

[45] Shengjie Xu, Clara Hobbs, Yukai Song, Bineet Ghosh, Tingan Zhu, Sharmin Aktar, Lei Yang, Yi Sheng, Weiwen Jiang, Jingtong Hu, Parasara Sridhar Duggirala, and Samarjit Chakraborty. 2024. GPU Partitioning & Neural Architecture Sizing for Safety-Driven Sensing in Autonomous Systems. In *2024 International Conference on Assured Autonomy (ICAA)*. 67–76. doi:10.1109/ICAA64256.2024.00018

[46] Hong Zhang, Mingyin Liang, and Yufeng Wang. 2025. YOLO-BS: a traffic sign detection algorithm based on YOLOv8. *Scientific Reports* 15, 1 (2025), 7558. Published online: 2025-03-04. doi:10.1038/s41598-025-88184-0

[47] Yuwei Zhang, Yan Wu, Junqiao Zhao, Yinghao Hu, Yufei He, and Jijun Wang. 2023. Robust Traffic Light Recognition Pipeline Based on YOLOv8 for Autonomous Driving Systems. In *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*. 1278–1285. doi:10.1109/ICPADS60453.2023.00184