Safety-Driven DNN Sizing for Vehicular CPS

Tingan Zhu* Mier Li* Bineet Ghosh† Samarjit Chakraborty* Parasara Sridhar Duggirala*
*Department of Computer Science, UNC Chapel Hill †Department of Computer Science, University of Alabama

***Abstract*—Perception processing in cyber-physical systems (CPS) is now almost exclusively done using Deep Neural Networks (DNNs). Here, camera, radar and LiDAR data – in autonomous vehicles or robots – is fed into DNNs that detect surrounding obstacles and distances to them. These results are used by controllers to compute appropriate actuation signals. But a CPS typically has multiple state components, where each of them might be estimated using a different camera, radar or lidar and an associated DNN. Hence, an emerging problem is to implement multiple DNNs on a resource-constrained graphics processing unit (GPU). While many GPUs from NVIDIA and AMD allow them to be split into multiple virtual GPUs, there is little work on *how* to partition them, and therefore size the corresponding DNNs, when they are a part of the same CPS. In contrast to the existing practice of focusing on the inference accuracy of individual DNNs in isolation, we propose a system-level safety-driven DNN sizing (and hence GPU partitioning) scheme for vehicular CPS. Our main technical contribution is a detailed experimental evaluation of this DNN sizing approach and an empirical validation of the formal technique behind it.**

I. Introduction and Related Work

In implementing cyber-physical systems (CPS), we study the problem of sizing multiple Deep Neural Network (DNN) models that are a part of the same CPS. The CPS software implements a feedback controller that gets *state* estimates as input and computes actuation signals based on the estimated state of the *plant* (such as a car or a robot arm) and the desired behavior to be imposed on it. The state estimates are inferences from Deep Neural Networks (DNNs), which are fed by sensors such as cameras, radars and LiDARs. Since the state x of the plant being controlled can have multiple components, i.e., $x = [x_1, ..., x_k]$, each component x_i might be estimated by a different DNN. How should such DNNs be implemented on a shared graphics processing unit (GPU)? While this question arises in many CPSs such as robotics and manufacturing, it is more pronounced in the domain of autonomous vehicles because of their safety critical nature.

Many modern GPUs from NVIDIA (such as the A40 and the L40) and ARM allow themselves to be partitioned into multiple virtual GPUs, which is an attractive feature for the setting outlined above. Hence, how should be the DNN associated with each state component x_i be sized so that all the DNNs fit inside the available GPU? Designing DNNs for resource-constrained platforms has attracted considerable attention [1]–[3]. Similarly, techniques for neural architecture search have also been extensively studied [4]. In parallel, techniques for GPU partitioning to concurrently implement multiple DNNs have also been explored [5], [6]. However, almost all existing studies on DNN sizing have focused on improving the inference accuracy of a DNN in isolation. They do not consider the impact of the sizing decision on the overall safety and performance of the CPS and further that all the DNNs share the same the same computational resources. While control theorists have proposed techniques for accurate state estimation from noisy measurements obtained from DNNs, they do not explicitly study the effect of these uncertainties on the overall system behavior or quantify it.

Our contributions: In this paper we attempt to address this gap by proposing a system-safety driven sizing of the DNNs responsible for estimating the different state components of a CPS, while taking into account the capacity of the shared GPU on which these DNNs are implemented. Our approach is based on the observation that the accuracies of different state components have different impacts on the overall safety of the CPS. This depends on the dynamics of the system. Hence, state components that have a bigger influence on system safety need to be more accurately estimated, and therefore should be assigned proportionately larger DNNs. While our method is oblivious to the notion of safety being used, in this paper we use a very general form of system safety. It is measured by the distance between the trajectory of the closed-loop system in its state space and the trajectory of the *ideal* system in which all the state components are always accurately estimated.

This paper builds on our recent work [7], where we studied different heuristics for determining the sensitivity of different state components on the safety of a closed-loop control system. However, there are several differences. First, the study in [7] used the size of the *reachable set* of the closed-loop system as a measure of safety. In contrast, the distance between the real and the ideal system trajectory – as we use in this paper – provides a notion of safety with a more concrete physical interpretation. Second, and more importantly, the study in this paper deals with nonlinear systems, whereas the one in [7] was restricted to the much simpler case of linear systems. Third, and most importantly, the primary focus of this paper is an experimental evaluation and an empirical validation of our hypothesis particularly in the domain of automotive CPS. Towards this, we have implemented our DNN sizing approach to choose DNNs used for estimating the lane width, vehicle position and orientation of a `F1TENTH` vehicle [8] and validated our hypothesis using the `F1TENTH Gym` environment. Our results show that in contrast to the commonly followed approach of using bigger GPUs, appropriate DNN sizing has an oversized impact on system safety.

Paper organization: Section II outlines the necessary mathematical background and introduces our DNN sizing approach. This is followed by the main results of this paper in Section III: the empirical validation of our hypothesis driving the sizing approach. We conclude the paper by discussing some directions for future work in Section IV.

II. Background and Methodology

A. Background: System Model and Problem Statement

Autonomous systems interacting with the environment are modeled as nonlinear ordinary differential equations (ODEs) with inputs $\dot{x} = f(x, u)$. Here, important aspects of autonomous system behavior such as position and acceleration are modeled as continuous variables and the description of all these components is called as a *state*, denoted as $x \in \mathbb{R}^n$. The system's behavior can be guided by applying appropriate control input $u \in \mathbb{R}^d$. To achieve a desired behavior from the system, a widely used technique is design a feedback control, where the control input u is a function $g(x)$ of the current state x. Under such feedback, the behavior of the system is defined a solution of the nonlinear ODE $\dot{x} = f(x, g(x))$. The solutions of the ODE that describe the state of the system after time t starting from intial state x are called the *trajectories*, denoted as $\xi(x, t)$ or $\xi(t)$.

Traditionally, the state x is obtained by applying Kalman filtering on the data from sensors. In modern autonomous systems, the state estimation x is performed using deep neural networks (DNNs) over high dimensional inputs such as cameras and LiDAR. These DNNs do not return the *exact* state x, but rather produce an estimate $\hat{x} \approx x$. Substituting the feedback from the estimated state $\hat{x}$ instead of the true state would result in the nonlinear ODE $\dot{x} = f(x, g(\hat{x}))$. We denote trajectories with feedback calculated using $\hat{x}$ as $\hat{\xi}(x, t)$ or $\hat{\xi}(t)$.

For estimating every component x_i of the state, an autonomous system designer can deploy a wide range of DNNs. In this paper we assume that each state variable x_i can be estimated using j DNNs $\mathsf{NN_1}, \dots, \mathsf{NN_j}$ where each successive DNN would require more resources than the previous one. Conventional wisdom is that using a larger DNN results in better state estimation, i.e., lower value of $|\hat{x} - x|$. However, the designer is constrained by computational resources and therefore cannot assign the most accurate DNN for sensing all the state components. Furthermore, assigning a lower accuracy DNN for estimating a state component could potentially lead to a large deviation between ξ and $\hat{\xi}$, which would compromise the *safety* of the system. We say that a system is unsafe if the deviation between the ideal and real behavior cross a safety threshold, $|\hat{\xi} - \xi| > d$. The system designer has to assign an appropriate DNN for estimating state components that is safe and computationally feasible.

B. Proposed Technique: Sensitivity Analysis for DNN Sizing

We use sensitivity analysis to determine the appropriate DNN for state estimation of each state component. Sensitivity analysis, broadly speaking, quantifies the effect of an uncertainty on the system trajectory. Since performing sensitivity analysis on nonlinear system is very challenging, we first linearize the system dynamics, i.e., construct a linear approximation of the nonlinear system. We then perform sensitivity analysis on the linearized system and obtain an ordering on the sensitivity of each state component. This ordering on the sensitivity can be used to assign the appropriate DNN for each state component.

1) Linearization of System Dynamics with Uncertainties: We first substitute the DNN state estimate $\hat{x} = x + \delta x$ in the nonlinear ODE, which gives us $\dot{x} = f(x, g(x + \delta x))$. Here, δx represents the difference between the estimated state $\hat{x}$ and real state x, that can be modeled as accuracy of the neural network. While $f(x, g(x + \delta x))$ is a nonlinear function, we expand it and collect all the terms without δx as $\tilde{f}$ and and consider only first order terms in δx as $h(x)\delta x$ resulting in the following equation.

$$\dot{x} = \tilde{f}(x) + h(x)\delta x. \tag{1}$$

We linearize the function $\tilde{f}$ and h around the equilibrium $\mathbf{0}$, resulting in the following linear approximation.

$$\dot{x} = \frac{\partial}{\partial x}\tilde{f}\Big|_{\mathbf{0}} x + \tilde{f}(\mathbf{0}) + \frac{\partial}{\partial \mathbf{x}}\mathbf{h}\Big|_{\mathbf{0}} \mathbf{x}\delta\mathbf{x} + \mathbf{h}(\mathbf{0})\delta\mathbf{x} \tag{2}$$

The system is stable around the operating point; that is, the closed loop dynamics $\tilde{f}(\mathbf{0})$ and $h(\mathbf{0})$ evaluate to 0. Therefore, the system behavior is often dominated by the the Jacobian $\frac{\partial}{\partial x}\tilde{f}$ of the nonlinear function $\tilde{f}$ denoted as A and the uncertainty denoted as $\frac{\partial}{\partial x}h\delta x$ denoted as $B\delta x$. The linear approximation of the dynamics and the effect of incorrect state estimate δx is given as:

$$\dot{x} = (A + B\delta x)x. \tag{3}$$

2) Sensitivity Analysis of Linear Dynamical Systems: While typical linear dynamics are of the form $\dot{x} = Ax$, the effect of estimation uncertainties on the dynamics is modeled as $B\delta x$ term in Equation 3. Therefore, the distance between the ideal and real trajectories $|\hat{\xi}(t) - \xi(t)|$ can be approximated as the distance between the trajectories of system in Equation 3 and the dynamical system $\dot{x} = Ax$. Using the closed form expression for the solution of linear dynamical systems the deviation $|\hat{\xi} - \xi| \approx |(e^{(A+B\delta x)t} - e^{At})x|$, where $e^M = I + \frac{M}{1!} + \frac{M^2}{2!} + \dots$ is called matrix exponential.

Since e^{At} remains unchanged, the deviation is said to be large if the matrix norm of $e^{A+B\delta x}$ is large. Therefore, uncertainties in state component δx_i that increase the matrix norm of $A + B\delta x$ by a larger amount are considered more sensitive. Since the matrix exponential is a nonlinear function that is challenging to analyze, instead of calculating the exact matrix norm, we observe the change in singular values of $A + B\delta x$ for different uncertainties δx. That is, given an uncertainty in state component δx_i, we calculate the change in singular values for $A + B\delta x_i$ and the state component that results in the maximum change in largest singular value will be ranked as the most sensitive. Therefore the order of sensitivity of state components x_i is the order of the change in largest singular value of $A + B\delta x_i$. *As a result of this analysis, we hypothesize that DNNs that are more accurate should be used to estimate the state components that are more sensitive.*

3) Ordering State Components: The total ordering of the sensitivity for state components is computed as follows:

1) Expand the term $f(x, g(x + \delta x))$ as $\tilde{f}(x) + h(x)\delta x$ and ignore the higher order terms in δx.
2) Compute the Jacobian of functions $\tilde{f}$ and h at $\mathbf{0}$ and denote them A and B respectively.
3) Compute the change in the singular values of $A + B\delta x_i$ for each state component x_i and order them according to the change in the largest singular value.
4) Return the state components ordered according to the change of singular values as $\mathsf{SensitivityOrdering}$.

Assigning resources in $\mathsf{SensitivityOrdering}$ would minimize the effect of uncertainty on the trajectory and hence minimize the deviation from the ideal behavior.

III. Evaluation of DNN Allocation and Discussion

We evaluate the proposed technique for allocating DNN resources for state estimation on a miniature autonomous racing platform and present our findings.

A. Vehicle Dynamics

We conduct experiments in the `F1TENTH Gym` simulation platform, designed for autonomous systems research with deterministic and reproducible vehicle dynamics suitable for reinforcement learning and multi-agent scenarios. For our experiments, we use the widely used bicycle model for the vehicle dynamics with pure-pursuit controller for navigation [9]. It is a 5 dimensional (x, y, δ, v, ψ) nonlinear system with two inputs $(\delta u, a)$. Here, x and y denotes vehicle's position in Cartesian coordinates, δ is the steering angle, v is the velocity, ψ is the vehicle's orientation relative to the x-axis, u_δ is the change in steering angle, and a is the acceleration respectively.

B. State Estimation Using DNN

The `F1TENTH` gets a 2-Dimensional LiDAR scan as an input. Each Scan returns the nearest obstacle from the LiDAR mount from $-135°$ to $+135°$ with a resolution of $0.1°$ — a total of 2700 distance values. Here, $0°$ orientation of LiDAR is aligned with the longitudinal axis of the vehicle. We implement a simple algorithm to navigate a 2-dimensional map, that is, to stay in the middle of the track. For implementing this algorithm, we need three key parameters: width of the track, lateral displacement (distance from the end of the track), and the vehicle orientation. We navigate the `F1TENTH` vehicle in a given racing lap and collect the input-output data of LiDAR scan and the width of the track, lateral displacement, and vehicle orientation. These estimates will be used to infer the state estimates for the position $\hat{x}, \hat{y}$, and orientation $\hat{\psi}$.

We implement three specialized DNNs for state estimation: `DNN_wid` for track width, `DNN_dis` for lateral displacement, and `DNN_ang` for vehicle orientation respectively. Each network processes 1080-dimensional LiDAR inputs (subsampled from the 2700 distance values) through optimized 1D convolutional architectures. The 1D convolution approach was selected over 2D alternatives due to its efficiency in processing sequential LiDAR data while preserving angular relationships.

All networks employ: (1) Exponential Linear Unit (ELU) activation $\text{ELU}(x) = \max(0, x) + \min(0, \alpha(e^x - 1))$ with $\alpha = 1.0$ for smoother gradient flow in deep layers, (2) L2 weight regularization ($\lambda = 0.0001$) to prevent overfitting, and (3) Adam optimization ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) for efficient training convergence.

The architectural progression **(a)** - **(c)** (Table I) reflects three design considerations: (a) smaller network prioritize parameter efficiency through aggressive pooling, (b) medium architectures balance feature extraction and computational cost using convolutions, and (c) larger models employ expanded filters and hidden layers for complex pattern matching. The possible configurations of DNNs for estimating state components are provided in Table II; – indicates configuration not considered because of limited resources on the GPU. We have five DNN architectures (labeled (a)-(e), 24–596 kB) for width estimation and three architectures ((a)-(c), 24–192 kB) each for lateral displacement and orientation estimation, resulting in a total of 45 different configurations. Each configuration is denoted by a three-letter code that specifies the model size used for each parameter, where, for example, **(c)(c)(a)** represents size (c) (192 kB) for width and lateral displacement, and size (a) (24 kB) for orientation.

TABLE I
DNN Architectures

Label	Size	Params	Key Architecture Features
(a)	24kB	3,241	Single conv (1 filter), aggressive pooling (8×)
(b)	66kB	16,842	Triple conv blocks, progressive pooling (4,2,4)
(c)	192kB	48,112	Wide conv (8 filters), 32-unit hidden layer
(d)	326kB	92,368	Expanded filters (16), deeper feature extraction
(e)	596kB	152,912	High-capacity (8-16 filters), 64-unit hidden

TABLE II
Mean Squared Error (MSE) of DNNs for estimating width, distance from right wall, and orientation respectively by size.

DNN Size	DNN_wid MSE	DNN_dis MSE	DNN_ori MSE
(a) (24 kB)	0.8125	0.2323	0.0930
(b) (66 kB)	0.2270	0.0603	0.0760
(c) (192 kB)	0.0247	0.0059	0.0205
(d) (326 kB)	0.0057	–	–
(e) (596 kB)	0.0029	–	–

C. Sensitivity Analysis on System Parameter

We perform the sensitivity analysis of the dynamics by 1) expanding the nonlinear feedback system and ignoring the higher order δx terms, 2) linearizing the dynamics around $\mathbf{0}$, and 3) compute the change in the maximum singular value of the linearized dynamics for each state component. *The sensitivity values of the width and the lateral displacement is 5.62 and the sensitivity of the orientation is 5.28.* This implies that introducing uncertainty in the width or the lateral displacement estimates would change the trajectory by a large amount than introducing uncertainty in the orientation. *Therefore, a principled way to assign computational resources for DNNs would be to assign a larger DNN to estimate the width or lateral displacement when compared to estimation of the orientation.* The sensitivity values do not exactly quantify the effect of uncertainty in a given state estimate, but rather help us in deriving an ordering among the state components. Sensitivity analysis could discover Pareto-optimal DNNs sizing 30% of times and close to Pareto-optimal allocation in all cases for linear dynamics [7]. Therefore, we believe our allocation would be nearly Pareto-optimal, provided the linear approximation closely matches the actual dynamics. The exact resources to be assigned to each state component would have to be determined empirically.

D. DNN Allocation Results and Analysis

We conducted our experiment on a 10-meter track with variable width w ranging from 1 to 8 meters. The initial lateral position for the vehicle is selected from the set $\{0.15w, 0.25w, 0.32w, 0.5w, 0.67w, 0.75w, 0.85w\}$ and the initial orientation is selected from the set $\{-\pi/3, 0, \pi/3\}$ radians. The control algorithm is designed to bring the vehicle to the center of the track with orientation parallel to the track. We observe the trajectory of every DNN configuration for all possible initial states (21 of them) for duration of 12 seconds and record its deviation from the ideal trajectory. We evaluate each DNN configuration over two metrics: (1) *Average Deviation*, defined as the mean Euclidean distance between actual and ideal trajectories from different initial conditions; (2) *Average Maximum Deviation*, representing the mean of maximum deviations observed across different trajectories.

We do not provide the metrics for all 45 DNN configurations owing to space limitations. Instead, we divide the DNN configurations into groups where the total computational resources for DNN in each group would remain same. The table documenting the evaluation metrics for each group are presented in Table III. While these are a subsample of the various DNN configurations, the trends in Table III extend to all the other configurations that we have considered.

TABLE III
SUMMARY OF NETWORK COMBINATIONS AND PERFORMANCE

DNN_wid	**DNN_dis**	**DNN_ang**	**Cost**	**Avg. Dev.**	**Max Dev.**
Group 1: Cost = 408kB					
(c)	(c)	(a)	408	0.0518	0.0864
(a)	(c)	(c)	408	0.0686	0.1107
(c)	(a)	(c)	408	0.0736	0.1113
Group 2: Cost = 324kB					
(c)	(b)	(b)	324	0.0582	0.0998
(b)	(c)	(b)	324	0.0839	0.1241
(b)	(b)	(c)	324	0.0863	0.1168
Group 3: Cost = 240kB					
(c)	(a)	(a)	240	0.0625	0.0981
(a)	(c)	(a)	240	0.0716	0.1097
(a)	(a)	(c)	240	0.0746	0.1113
Group 4: Cost = 114kB					
(b)	(a)	(a)	114	0.0531	0.0906
(a)	(b)	(a)	114	0.0919	0.1435
(a)	(a)	(b)	114	0.1494	0.2118

We highlight two primary observations from Table III. First, Table III validates our hypothesis that allocating more computational resources for more sensitive state components improves the safety metrics. For example, in Group 1, configuration **(c)(c)(a)** — which uses larger models for *width* and *lateral displacement* estimation and a smaller model for *orientation* — achieves the lowest average deviations and average maximum deviations. In contrast, configuration **(a)(c)(c)**, which allocates the largest model to *orientation* and minimal resources to *width*, performs substantially worse. A third combination, **(c)(a)(c)**, further highlights this discrepancy. Other configuration groups in Table III exhibit similar trends which reinforce our observation that allocating larger models to *width* and *lateral displacement* leads to improved system safety when compared to emphasizing *orientation* for a given computational budget.

Our second observation is that allocating more computational resources to state estimation (total resources) results in better safety. Furthermore, appropriate allocation of fewer computational resources can result in better safety when compared to improper allocation of more resources. For example, **(c)(a)(a)** and **(c)(b)(b)** have lower average deviation and lower average maximum deviations when compared to **(b)(b)(c)** and **(a)(c)(c)** respectively.

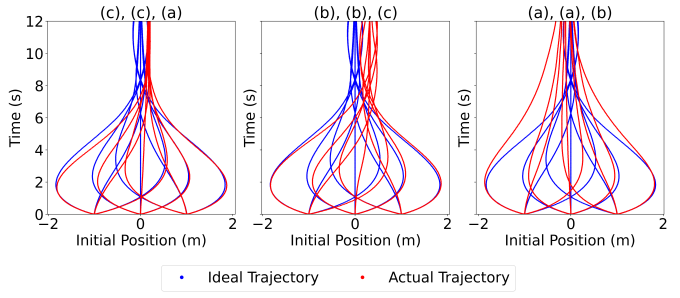

Fig. 1. Comparison of various DNN configurations over trajectories starting from 9 different initial positions with the corresponding ideal trajectory. The DNN configuration is provided with the image.

To help visualize the effect of imprecise state estimation and the deviation from ideal behavior, we present sample trajectories for three DNN configurations namely **(c)(c)(a)**, **(b)(b)(c)**, and **(a)(a)(b)** (at most one from each group in Table III) in Figure 1. The ideal trajectories are given in blue and the trajectories where the state estimation is performed using DNNs are given in red. The difference between ideal and real trajectories are consistent with the findings in Table III.

IV. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a sensitivity analysis based framework for allocating DNN resources for state estimation. We hypothesized that allocating more resources for estimating states that have higher sensitivity would result in better system safety. We have validated our hypothesis over a bicycle model of a miniature autonomous racing vehicle. Our sensitivity analysis revealed that estimation of the track width and lateral displacement is more important than orientation; which is confirmed by empirical measurements. Our analysis shows that sensitivity informed DNN sizing with fewer resources outperforms improper DNN sizing with more resources.

As a part of future work, we would like to validate our hypothesis over a wide variety of nonlinear systems and explore the latency and accuracy tradeoff for DNN state estimation. We also intend to perform evaluation on the physical `F1TENTH` platform.

REFERENCES

[1] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Intl. Conference on Machine Learning*, 2019.
[2] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, arXiv:1704.04861.
[3] T.-J. Yang *et al.*, “NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications,” 2018, coRR: arXiv:1804.03230v2.
[4] B. Lyu *et al.*, “Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing,” *IEEE Transactions on Cybernetics*, vol. 53, no. 2, pp. 1158–1169, 2021.
[5] J. Bakita and J. H. Anderson, “Hardware Compute Partitioning on NVIDIA GPUs,” in *29th RTAS*, 2023.
[6] S. Jain *et al.*, “Fractional GPUs: Software-Based Compute and Memory Bandwidth Reservation for GPUs,” in *IEEE RTAS*, 2019.
[7] S. Xu *et al.*, “GPU partitioning & neural architecture sizing for safety-driven sensing in autonomous systems,” in *International Conference on Assured Autonomy (ICAA)*, 2024.
[8] M. O’Kelly *et al.*, “F1TENTH: an open-source evaluation environment for continuous control and reinforcement learning,” in *NeurIPS Competition and Demonstration Track*, 2019.
[9] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” CMU-RI-TR-92-01, Carnegie Mellon Univ., Tech. Rep., 1992.