Poster Abstract: Neural Architecture Sizing for Autonomous Systems

Shengjie Xu[1], Clara Hobbs[1], Yukai Song[2], Bineet Ghosh[3], Sharmin Aktar[1], Lei Yang[4], Yi Sheng[4], Weiwen Jiang[4], Jingtong Hu[2], Parasara Sridhar Duggirala[1], Samarjit Chakraborty[1]

***Abstract*—Neural networks (NNs) are now widely used for perception processing in autonomous systems. Data from sensors like cameras and lidars, after being processed by NNs, feed control algorithms that form the core of autonomy-related functions. Such NNs are implemented on graphics processing units (GPUs) and modern GPUs can be partitioned into multiple virtual machines, each implementing a separate NN. Given an autonomous system with multiple NNs, how should each NN be sized and the GPU implementing them be optimally partitioned? In this work, we study multiple GPU partitioning techniques with the goal of optimal and safe system-level control performance.**

I. Introduction

The advancement in deep learning technologies has led to the widespread deployment of neural networks (NNs) in autonomous systems. Due to their mission criticality, autonomous driving systems usually require high accuracy from NN components. However, achieving state-of-the-art accuracy often results in increased computational and memory demands. Despite efforts to compress NNs for efficiency (*e.g.*, [1]), the challenge of meeting accuracy requirements within an autonomous systems' intrinsic space, energy, and cost constraints remains significant. Furthermore, the overall performance of such systems, encompassing sensing, decision-making, and actuation, is impacted by not only the accuracy of its NN components, but also the control system's sensitivity to uncertainties of NN outputs. Since the impact of NN estimation errors varies across the system, optimizing for ***overall system performance*** necessitates a nuanced approach to resource allocation among NNs, prioritizing critical functions while allocating adequate resources for others.

***Contributions of this work*:** We address resource allocation of NNs in autonomous systems, to optimize safety and control performance. Specifically, we focus on NNs used for state estimation (*e.g.*, depth estimation). Since NNs are implemented on GPUs and modern GPUs can be partitioned, the allocation problem reduces to NN sizing and GPU partitioning. We present three heuristics for NN sizing and demonstrate near-optimal system performance with significantly less computational effort than an exhaustive search. To the best of our knowledge, no prior work relates control performance of autonomous systems to NNs' sizing or GPU allocation.

Related Work*:** Extensive literature exists on the optimization of memory, computation, and energy demands of embedded NNs. Notable strategies include developing smaller, more efficient NNs [1], [2] and implementing early exit strategies [3]. However, little work exists on *how* to choose NNs that collectively contribute to ***overall system performance and safety. This work is also inspired by recent trends in GPU partitioning: whereas GPUs have been traditionally treated as exclusive-access resources in scheduling, efforts have been made [4]–[6] to explore various approaches to *GPU partitioning*, *viz.*, the mechanism to share a single GPU among multiple applications, such as multiple neural networks.

[1]The University of North Carolina at Chapel Hill, USA
[2]University of Pittsburgh, USA
[3]The University of Alabama, USA
[4]George Mason University, USA

II. Preliminaries

We focus on linear time-invariant (LTI) systems modeled in discrete time, captured by $x[t+1] = Ax[t] + Bu[t]$, where $x[t] \in \mathbb{R}^p$ represents the state and $u[t] \in \mathbb{R}^q$ *control inputs* at time t, and $A \in \mathbb{R}^{p\times p}$ and $B \in \mathbb{R}^{p\times q}$ define the system's *dynamics*. Feedback control is implemented through the computation of control inputs $u[t] = K\hat{x}[t]$, where $K \in \mathbb{R}^{q\times p}$ is the *feedback gain* and $\hat{x}[t]$ is the NN-estimated state. We assume that there is uncertainty in state estimation, so that $\hat{x}[t] \in x[t] \oplus E$, where $\oplus$ is the Minkowski sum operator, and E is the uncertainty bounding zonotope, obtained from uncertainties ϵ_i of individual states.

Given system dynamics A, B, an initial set of states $x[0]$, and the uncertainty bound E, we can compute the reachable sets $R[t]$ of the system. We denote the maximum reachable set diameter across the time horizon as $D = \max_t \operatorname{diam}(R[t])$. This lets us compare the effects of varying uncertainty, as more measurement uncertainty gives larger maximum diameters.

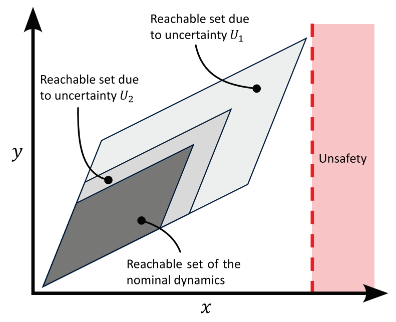
III. Methods

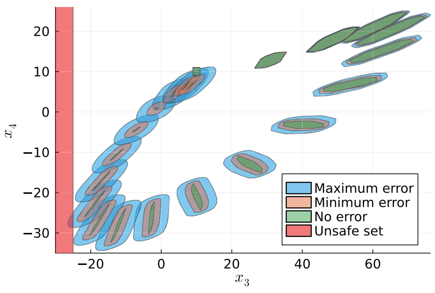
Problem 1 (Optimal neural architecture sizing problem)**.** *Given a p-dimensional system with dynamics A, B, feedback gain K, initial states $x[0]$, and n available NNs for state estimation with associated uncertainty bounds ϵ_j and costs c_j, we want to find an optimal selection of NNs $J = (j_1, \dots, j_p) \in \{1, \dots, n\}^p$ such that the maximum diameter of the reachable sets D is minimized, while the overall cost of all NNs stays within the budget. i.e., $\sum_{i=1}^{p} c_{j_i} \leq C$.*

We propose three heuristics to solve the optimal neural architecture sizing problem.

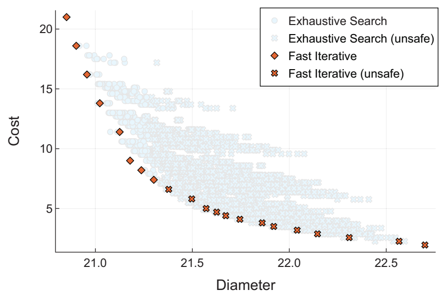
A. Sensivity Analysis-Based Approach

Our first heuristic is adapted from an analysis that measures a system's dynamic sensitivity—the degree to which uncertainty in a system's dynamic matrix A impacts the system's trajectory [7]. We adapt this method for system states rather than dynamics by introducing augmented system dynamics


Reachable set due to uncertainty U_1
Reachable set due to uncertainty U_2
Reachable set of the nominal dynamics
Unsafety
y
x
(a) Impact of uncertainties on a system's reachable set.



(b) Reachable sets projected into dimensions 3 and 4, up to time $t = 20$.



(c) Fast iterative heuristic results (solid) compared to exhaustive search

Algorithm 1: Dynamic programming heuristic

```
$J \leftarrow$ vector of $n$ ones;
$D \leftarrow$ calculate diameter using $J$;
Add $(J, D)$ to solution set;
while there are further solutions to explore do
    foreach solution from the last iter of outer loop do
        $J \leftarrow$ solution indices;
        foreach $i \in 1 \dots n$ do
            $J' \leftarrow J$ incremented in position $i$;
            if $J' \in$ tried solutions then
                continue;
            $D \leftarrow$ calculate diameter using $J$;
            Add $(J', D)$ to solution set;
    Prune dominated solutions from set;
```

that incorporate state uncertainties. This allows computing sensitivity values V_i for individual system states x_i by considering the impact of uncertainties on the system's evolution. A state with a higher sensitivity value V is more likely to cause system unsafety, as illustrated in Figure 1a.

We formulate the NN selection with sensitivity values as an integer programming problem: $\min_{j_1,\dots,j_p} \sum_{i=1}^{p} \epsilon_{j_i} \cdot V_i$, subject to $\sum_{i=1}^{p} c_{j_i} \leq C$ where $j_i \in \{1, \dots, n\}$ $\forall i$. Intuitively, we minimize the sum of products of the sensitivity value and uncertainty bound for each state, and use reachability analysis to estimate the impact on the system's trajectory.

B. Dynamic Programming Approach

Our next heuristic is a dynamic programming-based approach outlined in Algorithm 1. This method sorts NNs j_i by increasing cost and initially assigns the lowest-cost network to each state. It iteratively explores higher-cost network assignments by increasing allocated resources per state and evaluates these via reachability analysis to determine their reachable sets' maximum diameter. For example, the allocation $(2, 3, 3)$, would result in $(3, 3, 3)$, $(2, 4, 3)$, and $(2, 3, 4)$. The process iteratively refines the resource allocation, eliminating dominated solutions, until it either considers the highest-cost network for all states or when no new non-dominated solutions emerge.

C. Fast Iterative Approach

Our third heuristic exploits the diminishing returns in NN accuracy improvements with increased resources, sequentially

Algorithm 2: Fast iterative heuristic

```
$J \leftarrow$ vector of $n$ ones;
while $J \leq$ vector of $n$ $k$'s do
    $D \leftarrow$ calculate diameter using $J$;
    Add $(J, D)$ to solution set;
    Increment next element of $J$ following $order$;
```

incrementing the NN indices one at a time in some pre-determined order while maintaining that no two states' indices differ by more than 1. We outline it in Algorithm 2. For instance, if the pre-determined order is $1, 3, 2$, then we will try the assignments $(1, 1, 1)$, $(2, 1, 1)$, $(2, 1, 2)$, $(2, 2, 2)$, and so on until the most expensive option is used for all networks. Any order may be used for the dimensions; in our experiments, we order them by sensitivity values as described in Section III-A.

IV. Evaluation

We plan to evaluate the three classes of heuristics using a numerical five-dimensional state-space model from the ReachabilityModels.jl package. The neural network uncertainty ϵ and cost c values are adapted from the accuracy and FLOPS values of EfficientNet [1]. Figure 1b visualize the reachable sets of dimensions 3 and 4 of the five-dimensional model. Preliminary results from the fast iterative heuristic are highlighted in Figure 1c and compared to the exhaustive search values. They are promising, as the fast iterative heuristic found near-optimal solutions while exploring only a fraction of the solution space of the exhaustive search.

Acknowledgments: This work was supported by the NSF grant# 2038960.

References

[1] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 2020, arXiv:1905.11946 [cs, stat].
[2] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, arXiv:1704.04861 [cs].
[3] Y. Wu *et al.*, "Intermittent inference with nonuniformly compressed multi-exit NNs for energy harvesting powered devices," in *DAC*, 2020.
[4] J. Bakita and J. H. Anderson, "Hardware Compute Partitioning on NVIDIA GPUs," in *29th RTAS*, 2023.
[5] S. Jain *et al.*, "Fractional GPUs: Software-Based Compute and Memory Bandwidth Reservation for GPUs," in *IEEE RTAS*, 2019.
[6] T. Yandrofski *et al.*, "Making Powerful Enemies on NVIDIA GPUs," in *IEEE RTSS*, 2022.
[7] B. Ghosh and P. S. Duggirala, "Robustness of Safety for Linear Dynamical Systems: Symbolic and Numerical Approaches," 10.48550/arXiv.2109.07632, 2021.