SEO: Safety-Aware Energy Optimization Framework for Multi-Sensor Neural Controllers at the Edge

Mohanad Odema, James Ferlez, Yasser Shoukry, Mohammad Abdullah Al Faruque
Department of Electrical Engineering and Computer Science
University of California, Irvine, CA, USA

2023 60th ACM/IEEE Design Automation Conference (DAC) | 979-8-3503-2348-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/DAC56929.2023.10247751

***Abstract*—Runtime energy management has become quintessential for multi-sensor autonomous systems at the edge for achieving high performance given the platform constraints. Typical for such systems, however, is to have their controllers designed with formal guarantees on safety that precede in priority such optimizations, which in turn limits their application in real settings. In this paper, we propose a novel energy optimization framework that is *aware* of the autonomous system's safety state, and leverages it to *regulate* the application of energy optimization methods so that the system's formal safety properties are preserved. In particular, through the formal characterization of a system's safety state as a dynamic processing deadline, the computing workloads of the underlying models can be adapted accordingly. For our experiments, we model two popular runtime energy optimization methods, *offloading* and *gating*, and simulate an autonomous driving system (ADS) use-case in the CARLA simulation environment with performance characterizations obtained from the standard Nvidia Drive PX2 ADS platform. Our results demonstrate that through a formal awareness of the perceived risks in the test case scenario, energy efficiency gains are still achieved (reaching 89.9%) while maintaining the desired safety properties.**

***Index Terms*—Edge Computing, Formal Methods, Autonomous Systems, Safe Control, Multi-sensor Autonomous Driving Systems**

I. Introduction

Today, autonomous systems are capable of running high complexity neural networks (NNs) on self-sufficient edge platforms with heterogeneous hardware units (e.g., GPUs, ASICs), and integrate a wide variety of sensors (e.g., cameras, LiDAR, and IMUs) to attain a robust control performance [1]. As such, substantial computing power is required at the edge platform to enable such high performance, a requirement that goes against its other desired properties for the edge computing platform (e.g., compactness and reduced battery sizes). Even more so, having a power-hungry computing platform can worsen the performance of other broader system functionalities, as in how an autonomous driving system (ADS) can cause reductions in a vehicle's driving range by a factor reaching 12% [2].

In accordance, recent research efforts have targeted enhancing the energy efficiency of these edge platforms on both the hardware and software levels. For instance, support for processing and hardware reconfiguration has enabled effective resource management through computational workloads adjustments [3], [4]. In a similar vein, advancements in the wireless communication networking infrastructure have led to the emergence of the *remote edge computing* paradigm [5]–[9], which would equip autonomous systems with the flexibility to manage their workloads through offloading task computations to nearby servers existing at the edge of the networking infrastructure in millisecond communication latencies.

§This work was partially supported by the NSF under awards CCF-2140154, CNS-2002405, ECCS-2139781 and by C3.ai Digital Transformation Institute.

Encouraging as it may be, the consequences of adopting such energy optimizations with regards to the safety properties of the system are quite unclear. This is a major challenge for real-world adoption scenarios as autonomous systems are required to constantly react to continuously evolving environments, prioritizing safety above all other aspects. In many cases, this is achievable in autonomous systems through *provably-safe* controllers in which raw control outputs are filtered so as to be confined within the bounds of a *formal* safety function, a function that is evaluated continuously through a complete, precise estimation of the corresponding system state. To give a practical example, a radar processing pipeline in an ADS can support such *safety filtering* functionality, where radars inputs are processed to evaluate the safety state of the system (e.g., distance to closest obstacle), and if certain safety conditions are not satisfied (e.g., imminenet collision), the radar pipeline can override the main control pipeline to enforce safe steering or braking actions [1]. Accordingly, such a processing pipeline with critical *safety* responsibilities must continuously operate at maximum performance to realize as accurate state estimates as possible for maintaining the desired control safety guarantees.

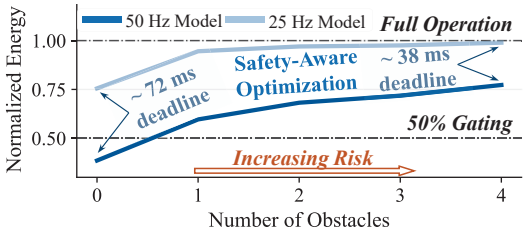

Fig. 1. Safety-aware gating optimization for two detector models across test runs with different number of obstacles simulated in Carla [10].

A. Motivational Example

In fact, using the *precise* state estimates provided through the critical safety-preserving pipelines (e.g., the Radar pipeline) and the corresponding evaluation of the safety function, we can further *regulate* the application of energy optimizations onto the remaining subset of less-critical processing models in a safety-aware fashion. We showcase this in Figure 1 through a premature example from our experiments that illustrates how this can be achieved in a *formal* manner, where the test case scenario – implemented in Carla [10] – involves a simulated autonomous vehicle with a pair of object detector models that support *gating* of their processing models at specific time intervals for energy optimization. The detectors operate on different processing frequencies (e.g., 50 Hz and 25 Hz) to reflect heterogeneous sensors of diverse specifications and sampling frequencies [11]. In the Figure, the horizontal axis

reflects the *risk* in the simulated driving scenario, represented by the number of obstacles along the vehicle's route, whereas the vertical axis represents the normalized energy consumption of the ADS under gating optimizations. As shown, the key idea is that gating optimizations are tuned based on the perceived risk on the road, i.e., safety state, through a *formally-derived* safe dynamic deadline, which evaluates to *lower* values at higher perceived risks (i.e., increasing number of obstacles) to prioritize robust processing over energy gains.

B. Novel Contributions

From here, we can summarize our novel contributions:

- We present SEO, a novel safety-aware energy optimization framework for multi-sensor autonomous controllers at the edge designed with specific safety properties
- Given the formal safety properties of an autonomous system, SEO proposes to divide the set of sensory processing models within the system into two subsets: a *critical* subset that contributes to the preservation of safety guarantees, and a *normal* subset leveraging energy optimizations.
- SEO regulates the application of energy optimizations to the models in the *normal* subset through a safety dynamic deadline that is estimated based on formal evaluations on the outputs from the *critical* subset.
- We characterize the performance of the *normal* processing models given dynamic safety deadlines for two popular energy optimization methods: *task offloading* and *gating*
- Our experiments for an autonomous driving use case simulated through Carla [10] across a variety of sensors and risk scenarios show that energy efficiency gains up to 89.9% can be achieved under formal guarantees on safety.

II. Related Works

Energy Optimizations. Numerous methods have been proposed to effectively manage energy consumption of edge autonomous systems at runtime, most notably through: (*i*) *Gating* [4], [12] in which components of the NN pipelines, if not all, can be scaled/gated based on the corresponding system state and runtime context. (*ii*) Task offloading [6], [8], [13] in which compute-intensive kernels can be offloaded to be processed at the nearby edge computing infrastructure, enabling an effective management of the local compute resources. To date, the matter of how adopting such optimizations can affect the *formal* safety properties of the autonomous system is highly understudied, especially considering the modular multi-sensory pipeline structure of today's autonomous system platforms.

Formal Methods for NN controllers. One research direction has been to apply formal verification techniques to asses the formal safety properties of neural network controllers [14], [15]. Whereas another leverages control theory concepts to augment NN controllers with formal safety guarantees on their outputs, filtering them and applying necessary corrections if needed [16], [17]. The scope of this work aligns with the latter. Specifically, our analysis focus is on the prominent 'controller-shielding' technique from that category [18], [19].

III. System Model

In this section, we provide the system model to formally regulate the application of energy optimizations for a controller while satisfying specific safety properties.

A. Safety Guarantees for Closed-loop Controllers

Let $\dot{x} = f(x, u)$ be a control system in a closed-loop with a state feedback $\pi : x \mapsto u$, where an input state, x, can be mapped into a control action, u. Let $h(x, u)$ be a real-valued function that characterizes the safety of f through a binary variable, $\mathbb{S}$, based on the x and u estimates as follows:

$$\mathbb{S} = \begin{cases} 1, & \text{if } h(x,u) \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $\mathbb{S} = 1$ indicates that the system is in a *safe* state whenever h evaluates to a *non-negative* value, and an *unsafe* state otherwise. In order to enforce a safe state, control outputs, u, are to be *filtered* through a *safety filter*, Ψ, that applies necessary corrections to u if needed in order to prevent function $h(\cdot)$ from evaluating to a negative value. (i.e., u remains within the bounds of the formal safety function). Thus, the *filtered* control output, u' can be described as:

$$u' = \Psi(x, u) = \begin{cases} u, & \text{if } \mathbb{S} = 1 \\ \psi(x; U), & \text{otherwise} \end{cases} \tag{2}$$

where ψ represents a function for applying corrective behavior whenever the system is deemed to enter an unsafe state. U represents the set of admissible control actions that the safety filter can apply. When a solvable function is derived for ψ capturing the underlying *dynamics of motion* of the physical system (e.g., the physical dynamics of rotating a steering wheel when changing steering angles) and exhibits a strong sense of uniform continuity on the control outputs, then $\dot{x}$ can be characterized as a *safe* control system.

B. Safe Time Intervals Characterization

With the above safety characterization, We want to determine for a system $\dot{x}$ at $\mathbb{S} = 1$ the following: *Given a state (x_t, u_t) at time t, denoted as x_t and u_t, what is the maximum allowable time that $\dot{x}$ can tolerate under the same applied control action, u_t, before $\dot{x}$ transitions to an unsafe state ($\mathbb{S} \rightarrow 0$)?*

From equation (1), let $\dot{x} = f(x, u)$ be a controller in a safe state $\mathbb{S} = 1$ at (x, u). Under the application of the same control u for a certain time period, the system is expected to enter an unsafe state $\mathbb{S} = 0$ at (x', u). Formally, if we consider $\dot{x} = f(x, u)$ and Ψ enforce a strong form of uniform continuity on control outputs, that is, changes from (x_t, u_t) to the immediate next state ($x_{t+\Delta}$, $u_{t+\Delta}$) are bounded by a small constant (i.e., *Lipshitz constant* in control theory). Then, we can express the maximum allowable safe time interval as such:

$$\Delta_{max} = \varphi(x, x', u) \tag{3}$$

where under the application of same control value u, the differentiation from x to x' through their encompassing continuous function can be characterized in time units. At this stage, we provide the following practical example for elaboration: Let x and x' characterize the respective *position*, *velocity*, and *orientation* for both an autonomous vehicle and an obstacle along its path. Then, given the vehicle's applied control values, u, (e.g., steering angle and throttle), we can compute the time, Δ_{max}, as the *time-to-collision* through numerical evaluations of φ under the assumption that the uniform continuity property holds. In truth, we also emphasize that x' is not necessarily

the exact state description of the obstacle per say, but rather a characterization of its safety bound coordinates (e.g., the minimum distance to a safety sphere around the obstacle).

C. Safe time Intervals as Dynamic Deadlines

Let Δ_{max} be a real-time value representing the *safety expiration time* given state (x, u) at a time t. Let $\dot{x}$ be a control system whose inputs are produced through N multi-sensory processing models (e.g., N neural networks) constituting the model set, Λ, contributing to the down-stream control task. Define subset $\Lambda' \subset \Lambda$ as an N' subset of models in the pipeline that the safety filter, Ψ, does not rely on for its state estimation, x. This means that every $\mathcal{N}_i \in \Lambda'$ does not influence the *formal* control safety guarantees. Then, Λ' can be designated as the set of models that can benefit from incorporated runtime optimization methods whose processing workloads can be adjusted in a *safety-aware* manner in accordance with the Δ_{max} values formally generated through the remainder subset of models $\Lambda'' = \Lambda - \Lambda'$.

Let each model $\mathcal{N}_i \in \Lambda'$ be associated with a single sensor, where the processing period of $\mathcal{N}_i \in \Lambda'$ is synchronized to its sensor's sampling period, denoted as p_i. In order to unify the time scale $\forall \mathcal{N}_i \in \Lambda'$, we define a period, τ, as the base time window, and discretize the sampling periods as multiples of τ:

$$\forall\, \mathcal{N}_i \in \Lambda', \quad \delta_i = \begin{cases} \frac{p_i}{\tau}, & \text{if } (p_i \,\%\, \tau) == 0 \\ \lfloor \frac{p_i}{\tau} \rfloor + 1, & \text{otherwise} \end{cases} \tag{4}$$

Similarly, Δ_{max} can be discretized to its following multiplier:

$$\delta_{max} = \lfloor \frac{\Delta_{max}}{\tau} \rfloor \tag{5}$$

From here, we can regulate the application of energy optimizations for every model $\mathcal{N}_i \in \Lambda'$ to obtain its safety-aware optimized model version, $\hat{\mathcal{N}}_i$:

$$\hat{\mathcal{N}}_{i[0:\delta_{max}-\delta_i]} = \begin{cases} \Omega_{i[0:\delta_{max}-2\delta_i]} + \mathcal{N}_{i(\delta_{max}-\delta_i)} & \text{if } \delta_i < \delta_{max} \\ \mathcal{N}_{i[0:\delta_{max}-\delta_i]} & \text{otherwise} \end{cases} \tag{6}$$

in which Ω represents the processing model under the applied energy optimization. Thus, given a sequence of discrete time intervals indexed by $[0 : \delta_{max} - \delta_i]$, Ω can be instantiated until the last period preceding $\delta_{max} - \delta_i$ as long as $\delta_i < \delta_{max}$. After that, the original $\mathcal{N}_i$ needs to be instantiated at $\delta_{max} - \delta_i$ to meet the safety deadline at δ_{max}. Otherwise, if $\delta_i \geq \delta_{max}$ (i.e., no viable optimization periods under the current deadline), $\hat{\mathcal{N}}_i$ proceeds to evaluate as the original $\mathcal{N}_i$ to maximize downstream control performance in the lesser safe states

IV. SEO Optimization Framework

In this section, we present our safety-aware energy optimization framework (SEO) for an autonomous system with guarantees on safe control. Figure 2 provides an illustration of how an abstract modular pipeline of a multi-sensor autonomous system would look with the supported safety-aware optimizations. As safety properties vary from one autonomous system to the other due to varying *dynamics of motion* and *control actions*, we will breakdown the different framework components below with a specific emphasis on autonomous driving systems considering how the existing literature derived and proposed methods to maintain formal safety guarantees for such systems.

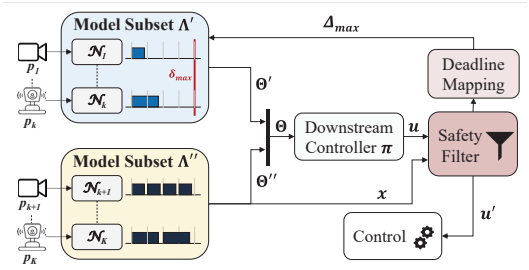

Fig. 2. Overview of a multi-sensor autonomous system pipeline supporting safety-aware optimizations as obtained provided through our SEO framework

A. Optimization and State Estimation Subsets

To realize energy efficiency gains while preserving the desired safety guarantees, the set of processing models deployed on an autonomous computing system are to be divided into Λ' and Λ'' subsets according to their criticality (as defined in Section III-C) where critical models in Λ'' are the ones tasked with providing state estimates, x, to the safety controller in order to uphold the formal safety guarantees. Therefore, models in Λ'' need to be constantly operating at full processing capacity to ensure that updated state estimates, x, are constantly fed to the safety filter. As for Λ', its models' evaluations are not used for safety state estimation, and thus can benefit from supported runtime performance optimizations to adapt their computational workloads. Still, these models are crucial to realize a smooth and robust end-to-end control performance along the main control pipeline, which involves the controller π processing aggregate predictions Θ from either subset of models (see Figure 2). In other words, proposed optimizations should be applied in a context-aware, adaptive manner to limit the need for the *overriding control* procedures by the safety controller.

B. The Safety Filter

A safety filter ensures that raw control predictions are confined within the boundaries of a safety function while accounting for the system dynamics of motion. As shown in Figure 2, the filter evaluates its safety boundaries on the corresponding state estimates generated from the model subset Λ'', and accordingly filters control predictions u as u' to be fed to the control unit. An example of such a filter is the *controller shield* proposed in [19] which was designed to filter steering angle outputs for autonomous driving control. This filter modeled the vehicle's dynamics relative to a fixed point in the plane (i.e., an obstacle), and extracted the relative distance and orientation angle as the x inputs to the filter. These x values are then used to evaluate the safety function h with respect to the obstacle, i.e., specifying the set of safe states and control with respect to the obstacle. With that characterization, the controller shield is able to receive vehicle steering angles, and apply the necessary corrections if needed.

C. Characterization of Safe Interval Times

Given the strong sense of continuity exhibited by an autonomous system with regards to its dynamics of motion, an expression for the vehicle's progression as a function of time can be derived. Where based on the system state with respect to a reference point in the plane (e.g., an obstacle), safety expiration times, Δ_{max}, can be obtained. In [20], such a

mapping function has been formally derived for the autonomous driving controller shield from the previous subsection, where the autonomous vehicle's relative states with respect to an obstacle can be mapped to a corresponding safety expiration times. Specifically, computed Δ_{max} values based on the corresponding state (distance to obstacle and its relative orientation angle) can be leveraged as dynamic execution deadlines for the models in Λ'. For instance, a vehicle driving head on towards an obstacle within a short distance would lead to low Δ_{max} values, which in turn would cause the models in Λ' to process inputs at near-full capacity due to the higher perceived risk. Lastly, through enough evaluations of the safety expiration function, a low-cost proxy lookup table, denoted as $\boldsymbol{T}(x, u)$, is constructed to enable real-time sampling of Δ_{max} values at runtime.

D. Runtime Control and Safety-Aware Optimization

In Algorithm 1, we describe the overall runtime control loop experienced by the autonomous system with support for safety-aware optimizations. An additional notation is y_i representing the input to the i^{th} sensory model. **Line 3** shows the estimation of a new state, x, and features, Θ'', from the Λ'' models to be fed to the safety component and the main controller, respectively. **Lines 4-6**, show the main control execution path in which generated controls u are filtered through Ψ to attain safe control actions. **Lines 7-11** indicate the start of a new safe optimization interval in which a new Δ_{max} value is sampled from **T** and discretized to δ_{max} based on the unified timing axis, whereas all Δ_{max} expiration flags are reset for the Λ' models. The **Lines 13-21** presents our safety-aware model optimization for each involved pipeline $\mathcal{N}_i \in \Lambda'$ based on its discretized operational period, δ_i, following equation (6). As detailed, the full model version, $\mathcal{N}_i$, will be invoked either when $p_i > \delta_{max}$ (no surplus optimization periods), or when δ_{max} expires. Otherwise, energy optimizations are applicable in that time step through Ω_n. Prediction outputs are constantly added from each model to Θ' for π's control outputs predictions in the following control loop. Lastly, **Lines 22-23** show that once the optimization interval has expired for all deadlines, new_Δ flag is set to sample new Δ_{max} value in the next time step.

Algorithm 1: Safe Runtime Control and Optimization

Input: Controller: π, Safety filter: Ψ, Lookup Table: $\boldsymbol{T}$, Base Period: τ, Optimization Subset: Λ', State Estimation Subset: Λ''

1 Initialize: n=0, Δ_{max}=0, Θ'={}, new_Δ=*True*
2 **while** *True* **do**
  `// state estimation and safe control`
3 $x, \Theta'' \leftarrow \mathcal{N}_l(y_l, x, u) \forall \mathcal{N}_l \in \Lambda''$ `// state estimation`
4 $\Theta \leftarrow aggregate(\Theta', \Theta'')$
5 $u \leftarrow \pi(\Theta)$ `// main control`
6 $u' \leftarrow \Psi(x, u)$ `// safe control`
  `// sample new safety deadline`
7 **if** new_Δ == *True* **then**
8 $\Delta_{max} \leftarrow \boldsymbol{T}(x, u)$ `// probe lookup table`
9 $\delta_{max} = \lfloor \frac{\Delta_{max}}{\tau} \rfloor$
10 n = 0, new_Δ = *False* `// new interval`
11 $done_i$ == *False* $\forall \mathcal{N}_i \in \Lambda'$ `// reset done flags`
  `// optimized safe processing`
12 Θ'={}
13 **for** $\mathcal{N}_i \in \Lambda'$ **do**
14 **if** $\delta_i \geq \delta_{max}$ *or* n == $(\delta_{max} - \delta_i)$ **then**
15 $\hat{\mathcal{N}}_{i(n)} = \mathcal{N}_{i(n)}$ `// invoke processing`
16 $\theta_i \leftarrow \hat{\mathcal{N}}_{i(n)}(y_i)$
17 $\Theta' \cup \{\theta_i\}$ `// update aggregates`
18 **if** n == $(\delta_{max} - \delta_i)$ **then**
19 $done_i$ = *True*
20 **else**
21 $\hat{\mathcal{N}}_{i(n)} = \Omega_n$ `// invoke optimization`
22 **if** $done_i$ == *True* $\forall \mathcal{N}_i \in \Lambda'$ **then**
  `// safe interval ended for all`
23 new_Δ=*True*
24 $n = n + 1$

V. Safe Energy Optimization Methods

In this section, we describe two common methods for Ω and how they influence the operation of $\hat{\mathcal{N}}$ in equation (6).

A. Task Offloading

Through wirelessly offloading compute-intensive tasks to be processed at compute-capable servers at the edge, task offloading can offer considerable energy efficiency gains for the local computing systems [5], [6], [8]. To conduct task offloading for critical workloads (such as perception kernels affecting downstream control decisions of an autonomous vehicle), there are two aspects to be incorporated:

- Server response times ($\hat{\delta}$) should be estimated to avoid offloads that are not expected to meet processing deadlines
- a safety fall back mechanism to re-invoke the local model if server responses after an offloading decision were delayed beyond $\hat{\delta}$ due to wireless uncertainty, and are projected to miss the critical deadline (e.g., δ_{max})

Accordingly, we demonstrate how this offloading logic can be incorporated within our primary optimization function in (6). Figure 3 provides examples of the potential experienced operational outcomes through this logic detailed below. At the start of every time interval, every model that meets the global safety deadline ($\delta_i < \delta_{max}$), proceeds to compare its δ_i against $\hat{\delta}$. If $\delta_i \leq \hat{\delta}$, then offloading is not feasible as there exists no fallback periods, and the model proceeds to evaluate locally. Otherwise, offloading is chosen with two potential outcomes: (*i*) if responses are received before $(\delta_{max} - \delta_i)$, then they can be applied directly as processing outputs, and thus, local compute was avoided and energy gains were realized (*ii*) if $(\delta_{max} - \delta_i)$ expired before receiving server responses, then the local model is instantiated to compute in the last period for safety.

From here, given an optimizable model $\hat{\mathcal{N}}$ (see equation 6), we can characterize its energy consumption *when offloading* (case 1 in equation 6) at discrete period, n, as follows:

$$E_{\hat{\mathcal{N}}} = \underbrace{T_{tx} \cdot P_{tx}}_{E_\Omega} + \underbrace{\mathbb{I}[n==(\delta_{max} - \delta_i)] \cdot T_{\mathcal{N}} \cdot P_{\mathcal{N}}}_{E_{\mathcal{N}}} \quad (7)$$

where T_{tx} and P_{tx} are the respective transmission latency and power; $\mathbb{I}[\cdot]$ is an indicator function to invoke local processing if the guarantee on safety expires. In this case, the system incurs additional energy consumption equal to the product of $\mathcal{N}$'s local processing overheads in terms of latency, $T_{\mathcal{N}}$ and power consumption, $P_{\mathcal{N}}$. We remark that although we omitted subscript, n, for notational simplicity, T_{Tx} and P_{Tx} evaluations are dependent on it since some offloading overheads may traverse multiple windows.

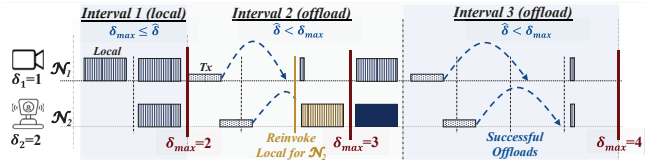

Fig. 3. Demonstration of task offloading under safety guarantees

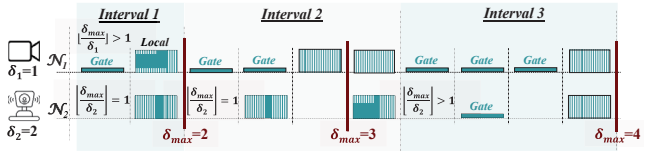

Fig. 4. Demonstration of gating optimizations under safety guarantees

B. Gating Mechanisms

Gating (Figure 4) is another scheme for energy efficiency that benefits from the determinism offered by *on-device* computing. The mechanism is straightforward in the sense given $\delta_i < \delta_{max}$, we can *gate* the processing model until the final interval period for energy efficiency. Even more so, we can also gate the sensor measurements themselves when the timeline is synchronized to their sampling periods, τ. In such case, we can model energy consumption for both *gating* and *computing* periods as:

$$E_\Omega = \tau \cdot P_{mech}, \quad E_\mathcal{N} = \tau \cdot (P_{mech} + P_{meas}) + T_\mathcal{N} \cdot P_\mathcal{N} \quad (8)$$

in which P_{mech} and $P_{measure}$ are the power drawn by the sensor due to its mechanical and measurement operations. This separation is because gating cannot be directly applied to the mechanical aspects of the sensor, such as a rotating motor, due to inertia considerations. For instance, a LiDaR sensor motor needs to keep on rotating even if sensor measurement is gated.

VI. Experiments and Results

A. Experimental Setup

We use Carla simulation environment to implement an experimental scenario similar to the one proposed in [19] in which we have a Reinforcement Learning (RL) agent trained as an autonomous vehicle controller to travel along a 100m road that is populated with obstacles in the final third. We train the agent using the same reward function for 2000 episodes to output steering and throttle control actions. To reflect the Λ'' and Λ' components that feed inputs into the agent, we first reuse the Variational Autoencoder in [19] for Λ'', and deploy two pretrained ResNet-152 object detectors for Λ', where they operate at respective periods $p = \tau$ and $p = 2\tau$ to imitate sensor operational diversity [11]. Unless otherwise stated, we set τ to 20 ms based on practical numbers from the literature and benchmark datasets [11].

Our forthcoming analysis for energy optimizations is conducted under both cases for when the safety component tasked with filtering steering angle outputs (recall Subsection IV-B) is active and inactive, referred to by respective *filtered* and *unfiltered*. Our main results are the average from 25 test runs in which the agent successfully completed the route without any collisions in either of the above cases. We retrieve the state estimates (i.e., distance and relative orientation) needed by the safety component directly from Carla for simplicity.

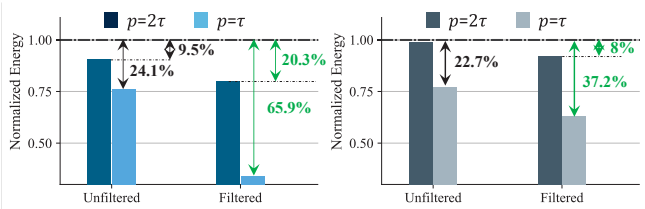

Fig. 5. Energy gains relative to local execution for the ResNet-152 detectors with different p when **offloading (*left*)** and **model gating (*right*) at** $\tau = 20ms$

TABLE I
Offloading and Gating Energy Gains over Local at τ=25 ms

Mode	Control	($p = \tau$) gains	($p = 2\tau$) gains	Average gains
Offload	*unfiltered*	15.3%	7.5%	11.8%
	filtered	27.1%	14.1%	21.1%
Gating	*unfiltered*	13.4%	0%	6.6%
	filtered	23.8%	4.3%	14.5%

For performance comparisons, we follow the scheme proposed in [13] for both local and offloaded performance characterizations in terms of latency and energy consumption. Due to space considerations, we only provide a high-level overview where for the former, we deploy the ResNet-152 models on an Nvidia Drive PX2 ADS platform, and benchmark their local execution overheads using TensorRT in terms of latency and energy (17 ms latency and 7 Watts execution power consumption). For offloading, we assume a Wi-Fi link in which effective data rate values are sampled from a Rayleigh channel distribution model with scale 20 Mbps.

B. Energy Gains under Safety Guarantees

To analyze the extent of energy gains under the dynamic safety execution deadlines, δ_{max}, we illustrate in Figure 5 the extent of energy gains that can be realized across our two ResNet-152 detectors using *offloading* and *model gating* optimization methods in both the *unfiltered* and *filtered* cases. Based on the results, two key observations can be made: 1) models synchronized to sensors with higher sampling frequencies are naturally more likely to benefit more from proposed optimizations, as in the 65.9% energy gains experienced by the detector at $p = \tau$ compared to the 20.3% gains experienced by its $p = 2\tau$ counterpart in the filtered offloading case, which is attributed to the former's higher prospect of optimizations under lower values of δ_{max}. 2) Energy gains in the *filtered* case are more than *unfiltered* (e.g., 65.9% vs 24.1% at p=τ for offloading). This is mainly because the safety component forces the RL agent to maintain a healthy distance from the obstacles through effective maneuvering, which in turn causes higher values of δ_{max} being sampled and more optimizations for both models. We repeat our experiments in Table I when varying the base period τ as a case of more limited hardware settings. As shown, considerable energy gains, are still be attainable, 21.1% and 14.5% on average for respective offloading and gating.

C. Energy Efficiency gains under varying risk levels

To assess our approach under varying degrees of risk, we vary the number of obstacles on the vehicle's trajectory, and analyze how performance efficiency would change. Figure 6 illustrates this for the *unfiltered* case through a histogram of the sampled δ_{max} values for each variation of number of

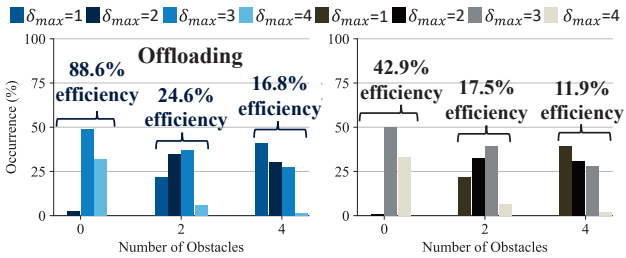

Fig. 6. Average δ_{max} experienced in the *unfiltered* control case when varying number of obstacles for **offloading (*left*)** and **model gating (*right*)**

TABLE II
AVERAGE ENERGY GAINS AND δ_{max} AT τ=20 MS UNDER OBSTACLE VARIATION FOR TWO COMBINED (p=τ) AND (p=2τ) MODELS

Control	#Obst.	Offloading Gains	Gating Gains	δ_{max}
unfiltered	0	88.58%	42.92%	3.67
	2	24.6%	17.47%	2.29
	4	**16.82%**	**11.89%**	**1.92**
filtered	0	89.89%	43.82%	3.7
	2	39.49%	24.26%	2.61
	4	**43.1%**	**22.57%**	**2.53**

obstacles, coupled with the average energy efficiency gain over the two detectors. Across both potential optimization cases, the histogram shows that lesser values of δ_{max} are sampled more frequently as the number of obstacles increase. For instance, δ_{max}=4 occurrence frequency decreases from 33.3% to 6.48% to 2.3% in the model gating approach as the number of obstacles increase from 0 to 2 to 4. That, of course, influences energy efficiency gains accordingly as indicated by the progressive drop in the average energy efficiency numbers. In Table II, we also provide the results for the *filtered* case. Interestingly, we find that the average energy gains and experienced δ_{max} values start to saturate when the number of obstacles $\geq$2. This is again attributed to minimum safety distance imposed by the safety filter leading to more evaluations of $\delta_{max} > 1$.

D. Sensor Gating

In this experiment, we extend our gating model analysis to encompass a broader energy consumption model of both the neural network processing model and the sensor itself (equation 8). Firstly, we retrieve the measurement power specifications for industry-grade sensors commonly used in autonomous systems: ZED Stereo Camera [21], a Navtech CTS350-X Radar [22], and a Velodyne HDL-32e LiDAR [23]. We also specify P_{meas}=2.4 W for LiDAR's rotation power consumption based on common LiDAR motors [4]. The numbers are provided in Table III, where we compare energy gains experienced by each sensor model, both on average during the test run and and when δ_{max} was sampled equivalent to 4τ. As shown, energy gains for the camera pipeline achieves the best scores (37.5% and 8.2% on average) compared to the other sensory pipelines, this is because the absence of any residual energy consumption due to P_{mech} enhances gating efficiency considerably. Between the Radar and LiDAR, we find that the RADAR is more efficient (e.g., 34.84% vs. 32.72% on average at $p = \tau$) as a result of the higher P_{meas} (21.6 W) rating which means that it is more susceptible to benefit from sensor gating optimizations.

TABLE III
SENSOR GATING AT τ=20MS FOR FILTERED CONTROL CASE

Sensor	P_{meas}	P_{mech}	Avg. Gains	4τ Gains
ZED Camera (p=τ)	1.9 W	0	**37.5%**	**75%**
ZED Camera (p=2τ)			**8.2%**	**50%**
Navtech Radar (p=τ)	21.6 W	2.4 W	**34.84%**	**68.93%**
Navtech Radar (p=2τ)			**7.57%**	**45.53%**
Velod. LiDAR (p=τ)	9.6 W	2.4 W	**32.72%**	**64.82%**
Velod. LiDAR (p=2τ)			**6.9%**	**41.91%**

VII. CONCLUSION

We proposed SEO a novel safety-aware energy optimization framework for multi-sensor autonomous systems at the edge that regulates how runtime energy optimizations are applied onto the involved processing pipelines. Our experiments using two common energy optimization techniques for a simulated multi-sensor autonomous vehicle in Carla environment has shown that substantial energy gains, reaching 89.9%, can be achieved while preserving the desired safety properties.

REFERENCES

[1] S. Liu *et al.*, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.
[2] S.-C. Lin *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *ASPLOS'18*.
[3] S. Yi *et al.*, "Energy-efficient adaptive system reconfiguration for dynamic deadlines in autonomous driving," in *ISORC'21*, 2021.
[4] A. V. Malawade *et al.*, "Ecofusion: Energy-aware adaptive sensor fusion for efficient autonomous vehicle perception," in *DAC'22*, 2022.
[5] S. Liu *et al.*, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, 2019.
[6] S. Baidya *et al.*, "Vehicular and edge computing for emerging connected and autonomous vehicle applications," in *DAC'20*, 2020.
[7] M. Cui *et al.*, "Offloading autonomous driving services via edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 10, 2020.
[8] A. Malawade *et al.*, "Sage: A split-architecture methodology for efficient end-to-end autonomous vehicle control," *ACM TECS'21*, vol. 20, no. 5s, 2021.
[9] B. Zamirai *et al.*, "Sieve: Speculative inference on the edge with versatile exportation," in *DAC'20*, 2020.
[10] A. Dosovitskiy *et al.*, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
[11] I. Gog *et al.*, "Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles," in *ICRA'21*, 2021.
[12] S. Lee *et al.*, "Accuracy–power controllable lidar sensor system with 3d object recognition for autonomous vehicle," *Sensors*, vol. 20, no. 19.
[13] M. Odema *et al.*, "Testudo: Collaborative intelligence for latency-critical autonomous systems," *IEEE TCAD'22*, 2022.
[14] X. Sun *et al.*, "Formal verification of neural network controlled autonomous systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019.
[15] W. Xiang *et al.*, "Reachable set estimation and verification for neural network models of nonlinear dynamic systems," in *Safe, Autonomous and Intelligent Vehicles*, 2019.
[16] C. Dawson, S. Gao, and C. Fan, "Safe Control with Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction methods," 2022.
[17] R. Cheng *et al.*, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," 2019.
[18] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe Reinforcement Learning via Shielding," 2017.
[19] J. Ferlez *et al.*, "Shieldnn: A provably safe nn filter for unsafe nn controllers," *arXiv preprint arXiv:2006.09564*, 2020.
[20] M. Odema *et al.*, "EnergyShield: Provably-Safe Offloading of Neural Network Controllers for Energy Efficiency," in *ICCPS'23*, 2023.
[21] Stereolabs, "ZED Camera and SDK Overview." [Online]. Available: https://cdn.stereolabs.com/assets/datasheets/zed-camera-datasheet.pdf
[22] "Navtech CTS Series." [Online]. Available: https://navtechradar.com/clearway-technical-specifications/compact-sensors
[23] "Velodyne HDL-32e Datasheet," May 2021. [Online]. Available: https://velodynelidar.com/products/hdl-32e/