HydraFusion: Context-Aware Selective Sensor Fusion for Robust and Efficient Autonomous Vehicle Perception

Arnav Vaibhav Malawade*
Trier Mortlock*
Mohammad Abdullah Al Faruque
University of California, Irvine
Irvine, California, USA

ABSTRACT

Although autonomous vehicles (AVs) are expected to revolutionize transportation, robust perception across a wide range of driving contexts remains a significant challenge. Techniques to fuse sensor data from camera, radar, and lidar sensors have been proposed to improve AV perception. However, existing methods are insufficiently robust in difficult driving contexts (*e.g.*, bad weather, low light, sensor obstruction) due to rigidity in their fusion implementations. These methods fall into two broad categories: (i) early fusion, which fails when sensor data is noisy or obscured, and (ii) late fusion, which cannot leverage features from multiple sensors and thus produces worse estimates. To address these limitations, we propose HydraFusion: a selective sensor fusion framework that learns to identify the current driving context and fuses the best combination of sensors to maximize robustness without compromising efficiency. HydraFusion is the first approach to propose dynamically adjusting between early fusion, late fusion, and combinations in-between, thus varying both *how* and *when* fusion is applied. We show that, on average, HydraFusion outperforms early and late fusion approaches by **13.66%** and **14.54%**, respectively, without increasing computational complexity or energy consumption on the industry-standard Nvidia Drive PX2 AV hardware platform. We also propose and evaluate both static and deep-learning-based context identification strategies. Our open-source code and model implementation are available at https://github.com/AICPS/hydrafusion.

KEYWORDS

Sensor Fusion, Autonomous Vehicles, Object Detection, Robustness, Adaptive Fusion, Context-Aware

1 INTRODUCTION

Autonomous vehicles (AVs) are cyber-physical systems (CPSs) that operate in complex, dynamic environments with many different actors. An AV must be able to perceive the environment accurately and efficiently to ensure safety across driving settings. Most modern AVs are equipped with multiple sensors and use sensor fusion techniques to help handle the uncertainties present in challenging driving scenes. Even with these methods, autonomous driving is a highly complex task, and large deep-learning algorithms are necessary to enable accurate perception.

Despite recent advances, industry-standard AV perception systems still tend to fail in difficult contexts [20, 21]. A naïve solution to the problem is to continue increasing the size and complexity of AV algorithms and incorporate more sensors to cover as many driving contexts as possible. However, AVs are energy-constrained CPSs, so the use of larger algorithms comes at the cost of reduced driving range, increased expense, and increased power and thermal demands on the vehicle [17]. Moreover, as shown in Section 1.1, in some contexts fusing *more* sensors can actually result in a *less* precise result. Thus, robust and accurate AV perception requires algorithms that can *adapt* to dynamically changing driving contexts as they appear without increasing the computation requirements.

Typical AV perception systems implement deep convolutional neural networks (CNNs) [23], in which sensor measurements are fed through a series of convolutional layers to produce spatial features. These features are then used to detect objects in different regions of the visual scene. Sensor performance can vary depending on factors such as weather, lighting, and physical obstructions [24, 26, 27]. Sensor fusion algorithms attempt to combine the benefits from each sensor to produce a more accurate result. However, in dynamic environments, the *context* of the scene is often overlooked or excluded from the fusion method entirely. Most modern multi-sensor approaches typically perform sensor fusion at only one point in the model, whether it be early fusion across the raw sensor measurements or late fusion after detections have been made [22, 26, 30]. Furthermore, most works use static algorithms for fusion that do not depend on the context of the AV's operating environment. Context-aware sensing approaches have proven beneficial for a wide range of CPS applications [10, 14]. Humans intuitively leverage contextual information about the driving scene (*e.g.*, weather, lighting, road type, high-level visual features) to adjust their decisions and focus while driving. Similarly, contextual information can inform AV perception and enable more robust fusion in complex driving contexts.

The scope of this paper addresses the following core research problems: (i) implementing a fusion approach that is robust across diverse contexts, noise sources, and sensor error types; (ii) using the context of a scene to improve sensor fusion performance; and (iii) implementing an efficient multi-sensor fusion approach for energy-constrained AV edge devices.

In this paper, we propose HydraFusion — a context-aware sensor fusion approach that actively identifies the driving context and uses it to selectively fuse sensor data from different modalities at varying depths in the model. By using a selective sensor fusion approach, HydraFusion can improve the robustness of AV perception without increasing the computational demands on the energy-constrained AV edge platform. Our work is the first to study a context-aware selective sensor fusion approach that can dynamically adjust both *how* and *when* fusion is applied. We specifically study the problem of object detection in the AV perception system;

*Both authors contributed equally to this research.

978-1-6654-0967-4/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCPS54341.2022.00013

however, we posit that our proposed approach can be applied to a variety of cyber-physical sensor fusion applications, including tracking, localization, and mapping [6, 11, 18, 25]. The key contributions of this work are as follows:

(1) We propose a novel multi-branch sensor fusion architecture that enables early fusion, late fusion, as well as intermediate combinations.
(2) We propose intelligent, context-aware gating strategies that maximize robustness by dynamically selecting the fusion methodology depending on the current context.
(3) We demonstrate that our approach outperforms existing methods on a challenging real-world dataset containing a wide range of driving contexts, including bad weather, poor lighting, and various location types.
(4) We implement our approach on an industry-standard AV hardware platform, the Nvidia Drive PX2, to demonstrate that our approach can be practically deployed in a real AV with comparable energy consumption, latency, and memory usage to state-of-the-art methods.
(5) We open-source our algorithmic implementation and architecture[1] to benefit the research community and enable further study of selective sensor fusion approaches for CPS problems.

In the remainder of this section, we provide a motivating example for our work. In Section 2, we discuss related work. Section 3 presents our problem formulation. In Section 4, we discuss our proposed approach, HydraFusion, and in Section 5, we provide numerical results on the performance of our approach. Concluding remarks are given in Section 6.

1.1 Motivational Example

In this section, we outline a motivating example where a standard sensor fusion approach would produce non-ideal results in the problem setting of AV object detection. We provide some mathematical formulations coupled with qualitative analysis across diverse conditions an AV may encounter. This section serves to illustrate shortcomings with current approaches and makes some assumptions on linearity and error statistics, however we support similar analysis could be expanded to more complex settings.

Theoretical Analysis. In this example, we provide mathematical motivation for the direct affect a faulty sensor measurement could have on the estimation process. For performing late fusion in the problem of object detection, we can assume we want to predict an object's location $\mathbf{y}$ and that we have access to n multiple object detectors, each which produce their own predictions which we use as measurements $\mathbf{x}$. Examining a single measurement, we aim to model the relationship between the parameter we want to estimate and that measurement. This can be achieved through a commonplace parameter estimation formulation [4] as follows:

$$\mathbf{x} = h(\mathbf{y}) + \mathbf{e} \tag{1}$$

where $\mathbf{y}$ is the true parameter we are attempting to estimate of dimension n_y, $\mathbf{x}$ is the measurement to be fused of dimension n_x, h is a function mapping the state of the system to the measurements, and $\mathbf{e}$ is the estimation error. In this case, h can be linearized into $\mathbf{H}$, an $n_x \times n_y$ measurement matrix. We additionally can model the error, $\mathbf{e}$, such that it is zero mean, $\mathbb{E}(\mathbf{e}) = \mathbf{0}$ with $\mathbb{E}(\mathbf{e}\mathbf{e}^T) = \mathbf{R}$ where $\mathbf{R}$ is the measurement error covariance. This yields a weighted linear least-squares solution such that the minimum variance unbiased estimator can be derived in a batch manner as:

$$\hat{\mathbf{y}}_n = [\sum_{i=1}^{n} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i]^{-1} \cdot \sum_{i=1}^{n} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{x}_i \tag{2}$$

where $\hat{\mathbf{y}}_n$ is the fused estimate of the detected object. Typically, the more information a fusion filter has, the better the performance will be. However, this argument breaks apart in cases where there are discrepancies in the measurement models, among other sources of estimation errors. These conditions commonly occur in autonomous driving [9] and can negatively impact the quality of sensor measurements. Given incorrect $\mathbf{R}$ values in Eq. 2, this can lead to known problems of model divergence and/or filter inconsistencies [4]. For example, suppose a camera sensor on an AV has raindrops obscuring the lens (like in the first column of Figure 1). In that case, it may generate overconfident estimates where the $\mathbf{R}$ value does not reflect the true amount of noise in the measurement.

Qualitative Analysis. To illustrate these points, Figure 1 visualizes the object detection results for a variety of contexts from a public driving dataset [27]. Measurements from three sensing modalities (camera, radar, lidar) are shown from left to right across three different driving contexts: (a) *sunny*, (b) *rainy*, (c) *snowy*. The ground truth objects in the scenes are shown in dotted yellow boxes. A deep-learning-based object detection pipeline featuring Faster R-CNN [23] with a ResNet-18 [13] backbone was used to generate the detections for each sensor input. The fusion method, shown in purple in the last column for each scene, represents the standard approach to fuse detections from all sensors. In the same column, we also show our approach, HydraFusion, that selectively fuses sensors based on the context derived from a scene. For clarity, only the highest scoring predictions for each configuration are shown in the figure.

Some clear trends emerge in the results: (i) cameras predict fewer false positives but struggle in severe weather, as shown with the rainy and snowy images where the camera lens is obscured; (ii) radars can struggle in scenes with many objects blocking or deflecting measurements as shown in the urban setting, but remains robust in adverse weather conditions of rain and snow; and (iii) lidar can experience high levels of noise in a densely populated scene, can miss objects that are behind other objects, and can degrade in performance due to weather like the snowflakes shown in the figure. A summary of each modality's qualitative performance in different contexts of the dataset is shown in Table 1.

Scene	Camera	Radar	Lidar	Fusion
Urban	✓	✗	✗	✓
Rainy	✗	✓	✓	✓
Foggy	✗	✓	✓	✓
Snowy	✗	✓	✗	✓
Night	✗	✓	✓	✓

Table 1: Qualitative object detection performance of each sensing modality in different driving contexts.

[1] https://github.com/AICPS/hydrafusion

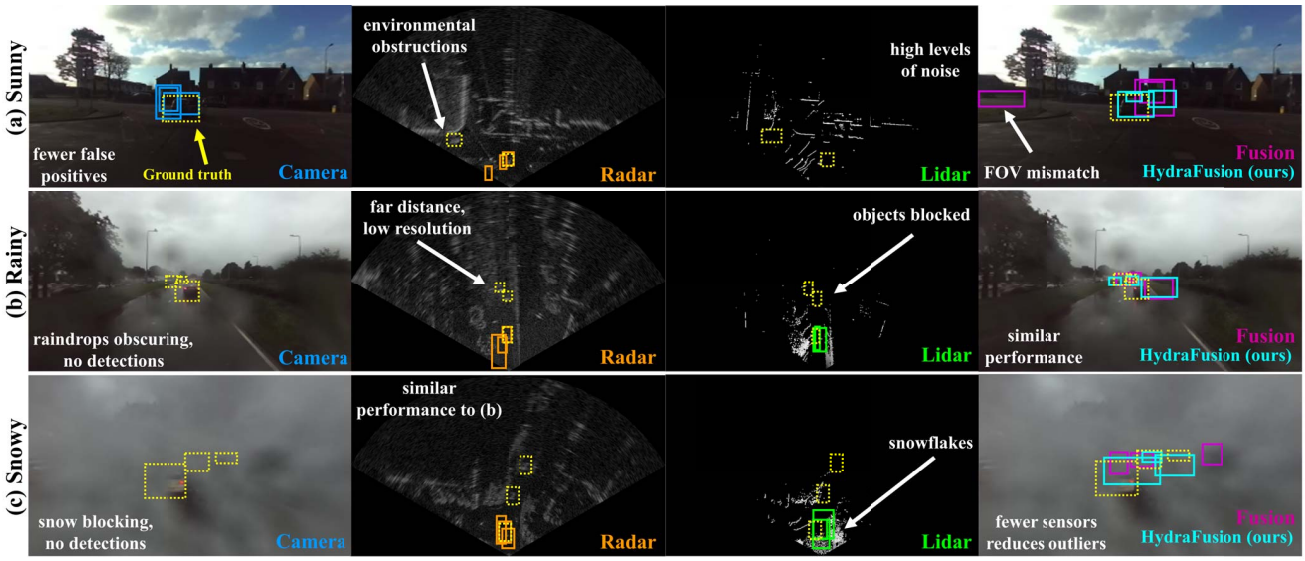

Figure 1: Qualitative analysis of object detection with different sensors and their fusion across three contexts. Ground truth detections are shown in yellow, while sensor-specific and fusion detections are shown in their respective colors. HydraFusion achieves the most accurate predictions across contexts.

The last column of Figure 1 allows us to examine fusion across the three different scenes. For most objects, the fusion method performs better than a single sensing modality. However, there are specific drawbacks to this approach. In (a), a field-of-view (FOV) mismatch arises between fusing detections across different modalities. Furthermore, the original camera detections for the center object were skewed far to the right by the other sensors' predictions. In (c), it is clear that the fusion method predicts more outliers that deviate from the ground truth. Overall, a more optimal estimate across all the images is achieved by fusing only a subset of sensors, as done in our approach, HydraFusion. This result motivates the need for a selective sensor fusion approach that can dynamically adjust to different contexts. The experimental results shown in Section 5 of this paper further validate the theoretical and qualitative analysis provided here.

2 RELATED WORK

This section discusses related works on sensor fusion, object detection, and multi-branch deep learning. We elaborate on their scope and limitations and compare them with our proposed approach.

2.1 Sensor Fusion

In traditional sensor fusion approaches that have known dynamics, noise, and measurement models, more sensors can help achieve better results [4]. Fusion across multiple homogeneous sensors can help reduce uncertainties by increasing *confidence* or providing measurements over a wider observation area to increase *coverage*. Fusing heterogeneous sensors can also reduce sensing uncertainties by providing information across a different feature set for the same task. However, the fusion of all sensors does not always guarantee better estimates, especially with highly nonlinear and dynamic systems such as AV perception systems. Hence, there are potential benefits to selectively fusing information obtained from sensors, as shown in some recent works. In [7], a selective sensor fusion scheme is developed for a visual-inertial odometry system to provide robustness against data corruption. The authors implement feature selection using data-driven models that consider measurement reliability and vehicle-environment dynamics. This work is extended to a generic framework for selective sensor fusion in deep pose estimation in [6]. However, these works only implement late-fusion over the outputs of sensor-specific deep learning models, limiting their performance and efficiency. Authors in [15] propose a strategy to alter the power levels and operating state of an AV lidar sensor depending on the vehicle's speed and environment. Similarly, [11] proposes adjusting the sensing frequency for indoor robot localization according to environmental dynamics. These approaches primarily focus on improving sensor efficiency. In contrast to these related works, our approach is the first to propose selective fusion for AVs with a dynamic gating component. By selecting between multiple modalities and fusion locations, HydraFusion maximizes robustness by selecting both *how* and *when* fusion takes place in the model.

In a similar vein, several works have studied the use of contextual information from the environment within an information fusion framework. Authors in [28] survey context-based information fusion and discuss how different types of contextual information interact with state variables and traditional fusion approaches. Both [25] and [18] show that context-aided sensor fusion frameworks for navigation improve robustness over standard methods. Distinct from these works, our approach utilizes deep learning models to learn contextual representations of scenes instead of static fusion rules to provide more robust results. Authors in [12] extract contextual information using specialized feature mining within a CNN for

object detection in very-high-resolution imagery. However, their approach is focused on obtaining contextual information from regions of interest in images, whereas our approach extracts the context of a *scene* using multiple heterogeneous sensory inputs.

2.2 Fusion in Object Detection Methods

Traditional object detection methods use CNNs to extract spatial features from inputs to identify objects in the scene [23]. Object detection in AVs is more challenging as the physical aspects of the environment affect performance. Both [9] and [3] survey object detection in AVs; [9] focuses on probabilistic methods, while [3] studies 3D detection methods. Both papers identify gaps in modeling sensor uncertainty. As detailed in the previous subsection, sensor fusion methods can help offset some measurement inaccuracies.

Fusion methods in object detection largely fall into two main categories: feature-level (or *early*) fusion and decision-level (or *late*) fusion. Early fusion approaches can extract many multi-modal features from the input but can be sensitive to noise and outliers from the sensors, reducing their robustness [22, 26]. Late fusion methods are more robust to sensor noise but cannot combine intermediate features across sensors, limiting their performance [30]. HydraFusion remains unique in its approach of combining early and late fusion approaches. To the best of our knowledge, this is the first work to propose a multi-layered fusion approach for object detection in AV perception systems.

2.3 Multi-Branch Deep Learning Architectures

HydraFusion maintains computational efficiency when evaluating multiple object detection pipelines simultaneously by utilizing a gating strategy, which limits the number of detection pipelines, or *branches*, that are run. Several types of multi-branch deep learning approaches have been proposed for image processing tasks. In [1], a network of experts approach to image categorization is proposed. Each branch is a CNN that only discriminates between the subset of classes it is assigned to learn, as this approach lacks an intelligent gating module. Similarly, [2] uses specific expert branches but focuses on life-long learning and the generation of new tasks and experts.

[19] explores efficient methods for single image classification, where the authors use branches developed to compute features on visually similar classes. During training, the authors employ an adaptive form of dropout where entire branches are dropped when they are not chosen by the gating function. Similarly, TridentNet [16] is a network that addresses the problem of scale variation in object detection. Its three-branch architecture shares parameters and structure between branches, resulting in faster training and inference and the enforcement of similar operations across feature maps, but this requires similarly structured branches. Our approach fundamentally differs from these works in that HydraFusion takes in multiple heterogeneous sensor modalities as inputs, incorporates context into an intelligent branch selection method, and targets dynamic sensor fusion for robust object detection via a multi-branch approach. Our approach is also unique because it enables the specialization of branches to individual sensors or subsets of sensors to improve robustness across varying driving contexts.

3 PROBLEM FORMULATION

This section provides a formulation for the key sensor fusion problem targeted in this paper: object detection in AVs. We assume that the AV uses a variety of sensing modalities to take measurements of the driving scene. At discrete time steps, samples are generated, which consist of input measurements, $\mathbf{X}$, from the sensors. The objective is to accurately detect objects, $\mathbf{Y}$, within each scene using the sensor measurements:

$$\mathbf{Y} = \phi(\mathbf{X}), \tag{3}$$

$$\mathbf{Y} = \{\mathbf{Y}^i_{class}, \mathbf{Y}^i_{reg}\}_{i=1\ldots d} \tag{4}$$

where ϕ represents the function for performing object detection, $\mathbf{Y}$ is composed of classification and regression components, and d represents the maximum number of objects to detect in a sample. ϕ can take the form of conventional fusion algorithms, a machine learning model, or an ensemble of machine learning models. Classification refers to the identification of each detected object's class. The classification target for each object can be defined as:

$$\mathbf{Y}^i_{class} \in \{1, 2, 3, \ldots, k\} \tag{5}$$

where k represents the number of classes considered in the problem. These indices represent a pre-defined mapping to object classes (*e.g.*, 1:*car*, 2:*van*, 3:*truck*, and so forth). Regression refers to the estimation of an object's location within the sample. These targets can be represented by:

$$\mathbf{Y}^i_{reg} = [\mu_1, \nu_1, \mu_2, \nu_2] \in \mathbb{R}^2 \tag{6}$$

where μ and ν denote the object's 2D bounding box coordinates in reference to a generic coordinate frame. [2]

The measurements from s sensors can be fused by a variety of means to improve detection results. An early fusion approach involves fusing the raw sensor measurements before passing them to ϕ:

$$\mathbf{Y} = \phi(\psi(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_s)) \tag{7}$$

with ψ representing the function used to fuse the measurements. In the case of late fusion, $\hat{\phi}$ represents a function fusing the separate output detections:

$$\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_s = \phi_1(\mathbf{X}_1), \phi_2(\mathbf{X}_2), \ldots, \phi_s(\mathbf{X}_s) \tag{8}$$

$$\mathbf{Y} = \hat{\phi}(\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_s) \tag{9}$$

The context of scenes in AV driving can vary dramatically: from different lighting conditions, to different road types and locations, to weather conditions that can severely degrade specific sensors. This variance calls for the use of an adaptive ϕ that is not only determined by a set of static scene conditions, but is instead learned within the model. In this case, ϕ represents an ensemble of object detection models, and ϕ^* represents the expected best subset of models in the ensemble for a given input $\mathbf{X}$. We denote the contextual information of a scene (either learned and modeled from the inputs or provided externally) as Ω. We then can define the subsequent equations:

$$\Omega = \pi(\mathbf{X}), \quad \phi^* = \rho(\Omega), \quad \mathbf{Y} = \phi^*(\mathbf{X}) \tag{10}$$

where π represents a context identification model, and ρ represents the mechanism for selecting ϕ^* given the identified context Ω. The goal of π and ρ is to select the optimal subset of branch models ϕ^*

[2] This could be represented in 3D as well, but for the sake of this paper we focus on 2D object detection.

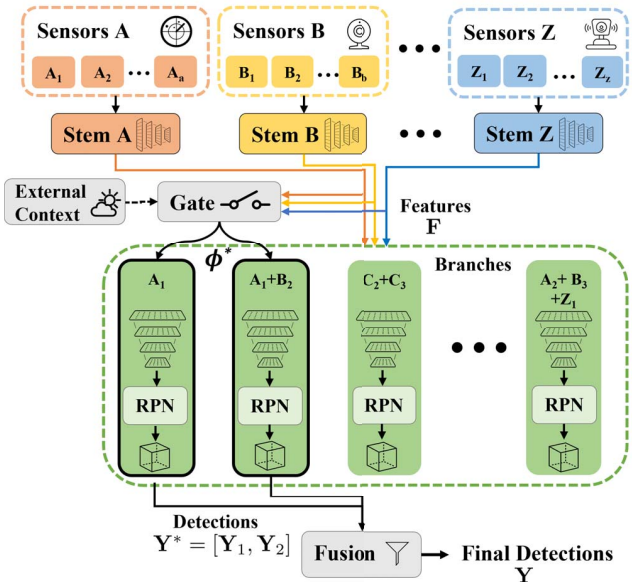

Figure 2: Our Proposed HydraFusion Architecture.

for the inferred context Ω to maximize object detection performance for a given $\mathbf{X}$. We posit that this general problem formulation can be extended to other sensor fusion problems in CPS.

4 METHODOLOGY

The model architecture for our proposed approach, HydraFusion is shown in Figure 2. Algorithm 1 describes how our architecture processes input data from different modalities to produce the desired targets. First, sensor data from each modality is processed by a modality-specific CNN (denoted as "stem") to produce an initial set of features $\mathbf{F}$. Next, these features are used by the gating module (containing π and ρ) to identify the context Ω and select which subset of branches ϕ^* should be executed for this context. Each branch is a deep-learning model capable of converting the features extracted from a subset of sensors $\mathbf{F}^*$ to produce a set of outputs for a specific task (*e.g.*, object detection). After the selected subset of branches is executed, the branches pass their outputs $\mathbf{Y}^*$ to the fusion block, which fuses them to generate the final object detections $\mathbf{Y}$. Next, we discuss the details of each component in our proposed architecture.

4.1 Input Processing and Stems

As shown in Figure 2, HydraFusion accepts any number of sensors and sensing modalities as input. Each stem is implemented as a CNN, which generates an initial set of spatial features for each sensor. We use a shared stem block for processing all the sensors for a given sensor modality. Thus, we will have three stems if our implementation uses camera, radar, and lidar sensors. After the input from each sensor for a given modality is passed through the stem, the gate module uses the resulting features to identify the context and select which branches to execute. Then, the selected branches use the stem output features as inputs to generate their predicted object detections.

Algorithm 1: HydraFusion Algorithm

Input: Sensor measurements $\mathbf{X} = \{A_1, ..., A_a, B_1, ..., Z_z\}$
Output: Object Detections $\mathbf{Y}$

$\mathbf{F} \leftarrow [[], [], [], ...]$ // initialize feature vector
for *s in sensor_types* **do**
  $\mathbf{S} \leftarrow \mathbf{X}[s]$ // get data by modality
  for *m in S* **do**
    $F[s][m] \leftarrow stem_s(m)$ // extract features
$\Omega \leftarrow \pi(\mathbf{F})$ // identify context
$\phi^* \leftarrow \rho(\Omega)$ // select Top-k branches to run
$\mathbf{Y}^* \leftarrow []$
for *branch in* ϕ^* **do**
  $\mathbf{Y}^*[branch] \leftarrow branch(\mathbf{F}^*)$ // pass subset of $\mathbf{F}$
$\mathbf{Y} \leftarrow fusion\_block(\mathbf{Y}^*)$ // fuse branch detections

4.2 Context Identification and Gating Module

Context identification is important for selecting the appropriate subset of branches to maximize performance in a given context. We propose several different gating algorithms for this task. The goal of the gate module is to rank the branches based on their expected performance for the input set of stem features. Next, the top-k branches (where k is configurable) are selected for execution and fusion to maximize object detection performance. The architectures of our three gating models are shown in Figure 3.

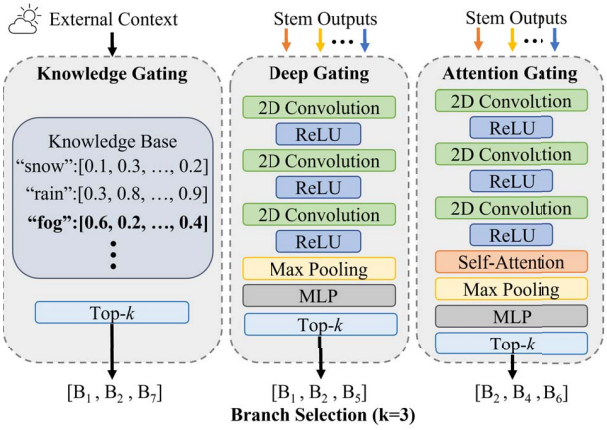

Figure 3: Gating Model Architectures.

Rigid Knowledge-Based Gating. Since there exists some domain knowledge as to how each context will affect each sensing modality, we can implement *Knowledge Gating*, where this domain knowledge is used to statically encode the subset of branches to execute for a given context. This assumes the set of possible contexts is finite, and the current context can be identified via external sources. For example, weather information, time of day, and map data can all be used to define the current context. In our approach, we define the set of fixed contexts based on metadata from the RADIATE dataset [27] describing the type of driving data in each sequence. Thus, our set of fixed contexts is: {*city, motorway, junction, rural, snow, fog,*

and *night*}. We leverage domain knowledge from Table 1 as well as from the RADIATE paper to rank the relative performance of each sensor in each fixed context. Then, at run-time, the external context information (*e.g.*, data from a navigation/weather system) is used to identify the current context. The top-k ranked branches for that context are selected to be executed and fused. The limitation of this gating strategy is that it requires a *fixed* context definition, potentially limiting performance in cases where contexts are less rigidly defined. With our other gating strategies, we define the context as a *continuous* feature space to enable the modeling of more complex contexts.

Learned Dynamic Deep Gating. In *Deep Gating*, we implement a CNN followed by a multi-layer perceptron (MLP) to model the relationship between the features output from the stems and rank the branches based on their expected performance for this feature set. The outputs of the CNN are flattened to one dimension before being passed to the MLP. In this gating method, the context can be viewed as a continuous feature space defined by the stem outputs.

Attention-Based Dynamic Gating. In some contexts, certain regions of the feature map may be more informative than others about the scene's context and, consequently, the branch-wise performance. We implement an attention-based gating strategy, denoted as *Attention Gating*, that infers an attention map over the stem features to evaluate this hypothesis. This attention map is used with CNN and MLP layers to model the relationship between branch performance and stem features. We use the visual attention layer proposed in [31] in our implementation.

Optimal Loss-Based Gating. To serve as a performance target for our gating approaches, we implement a so-called "optimal" gating strategy where, for a given input, the branch ranking output by the gate module is equal to the inverse of the aggregated branch loss for the detections output by each branch. Since the actual branch loss is used to inform the gate *a posteriori*, this strategy is not feasible for real-world deployment. However, it gives the theoretical best-case performance of a gating strategy that can perfectly rank the branches based on their losses for a given input. We denote this gating method *Optimal Gating*.

4.3 Branches

The branches of the proposed framework are designed to be specific to different sensor fusion combinations. These pairings can enforce early fusion in the model by combining the stem features of heterogeneous sensor inputs (*e.g.*, radar and lidar) before performing object detection. Furthermore, some branches use singular sensor inputs (*e.g.*, radar) that the gating module may choose in scenarios where other sensors (*e.g.*, camera and lidar) have poor performance due to situational factors (*e.g.*, weather or obstruction). Each branch is equipped with a Region Proposal Network (RPN) [23] that uses anchor generation techniques to predict detections across a feature map. These predictions are then fed through a region-of-interest layer that generates the following outputs for each detection: *bounding box coordinates* $[\mu_1, \nu_1, \mu_2, \nu_2]$ — expressed in the native coordinate frame, *scores* $[0-1]$ — confidence level of the detected object, and *labels* {1, 2, 3, …, k} — the assigned classification of the object. The outputs from each branch are passed to the fusion block to generate the final set of fused detections.

4.4 Fusion Block

The function of the fusion block in our approach is synonymous with the concept of late fusion. In HydraFusion, we use the following fusion algorithms to fuse the detections output by all of the active branches of the model.

Non-Maximum Suppression (NMS). This algorithm calculates the intersection over union (IoU) of corresponding bounding box estimations, and based on their confidence scores, selects which box estimates to keep. The equation for calculating the IoU (sometimes referred to as the Jaccard index) between two sets, A and B, is given by:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{11}$$

where $\cap$ represents the intersection, and $\cup$ represents the union. In our application, the sets are the rectangular bounding box predictions. By iteratively comparing bounding box predictions and returning a match if the IoU is above a defined threshold, only the box with the highest confidence score is kept among each set.

Soft-NMS. A further refinement of NMS, proposed in [5], which lowers confidence scores using a Gaussian weighting function defined by σ, if the boxes are above a threshold IoU value. Unlike NMS, Soft-NMS does not completely remove box estimates, which can result in more false positives.

Weighted Box Fusion (WBF). This approach, proposed in [29], clusters the bounding box predictions into distinct lists by iterating over the boxes and calculating IoUs with respect to thresholds. From each cluster, the fused bounding box predictions, $[\mathbf{f}_\mu, \mathbf{f}_\nu]$, are computed as weighted sums of each detection and its confidence score:

$$\mathbf{f}_{\mu_j} = \frac{\sum_{i=1}^{n} C_i \cdot \mu_{i,j}}{\sum_{i=1}^{n} C_i}, \quad \mathbf{f}_{\nu_j} = \frac{\sum_{i=1}^{n} C_i \cdot \nu_{i,j}}{\sum_{i=1}^{n} C_i} \tag{12}$$

where $j \in \{1,2\}$, $\mu_{i,j}$ and $\nu_{i,j}$ are the corresponding locations of the bounding box points, and C_i is the confidence score for the ith box. WBF also has a skip-box threshold that defines which boxes to exclude if they are below a certain confidence score. Furthermore, each branch can be assigned varying weights that can be tuned within the overall model or application being used. In our experiments, covered in the next section, the tunable threshold parameters across fusion methods were found to have insignificant effects on the results within reason.

5 EXPERIMENTS

In this section, we discuss our experiments. In Section 5.1 we elaborate on the dataset used to conduct the experiments. Sections 5.2 and 5.3 detail our model implementation process and training procedures. In Section 5.4 we present our experimental results. Finally, in Section 5.5 we discuss the practicality of our approach and future work.

5.1 Dataset

The RADIATE dataset [27] contains annotated data from a Navtech CTS350-X radar, a Velodyne HDL-32e LiDAR, and a ZED stereo camera. With this dataset, we trained and evaluated our models on object detection using supervised learning. The RADIATE dataset contains data for various driving contexts, including urban driving, snow, rain, fog, night, and motorway driving. In some cases, several sensors are visually obstructed by fog, rain, or snow. The dataset contains the following annotated object classes: {*car, van, truck, bus, motorbike, bicycle, pedestrian, group of pedestrians*}. This dataset provides a challenging benchmark on which the robustness of object detection models can be evaluated for a range of driving contexts. They additionally present object detection results using radar in varying weather conditions. Since [27] uses a different problem formulation, model size, and metrics, its results are not directly comparable to ours; however, our results for radar-only are representative of the model evaluated in their work. Please refer to the dataset for further details [27]. We used a 70:30 train-test split for training and evaluating our models.

5.2 Model Implementation

5.2.1 Model Specification. To evaluate HydraFusion in comparison to the baseline fusion approaches, we implemented each stem and branch as a Faster R-CNN [23] model with a ResNet-18 [13] backbone. We split the ResNet-18 models at the first block and use it as the stem for each modality. Then, the remaining ResNet-18 layers and the RPN of Faster R-RCNN are used in each branch.

With four sensors (two cameras, lidar, and radar), the total number of possible unique branches is $2^4 - 1 = 15$. However, the training and space complexity of a 15-branch model may be much larger without providing noticeable improvements in precision. Thus, we use domain knowledge to identify the best branches for the application by picking branches that can cover the limitations of other branches in difficult contexts. Thus, our HydraFusion implementation contains four single-sensor branches and three early fusion branches, for a total of seven branches. The single-sensor branches are: *Left Camera, Right Camera, Lidar*, and *Radar*; the early fusion branches are *L/R Cameras, Lidar+Radar*, and *L/R Cameras + Lidar*. For single-sensor branches, the stem features for the sensor are used as the input for the branch. For branches with early fusion, we concatenate the stem features for each sensor to be fused across the channel dimension. Then, we use a 2D convolution layer to fuse this concatenated output before passing the result to the remaining ResNet-18 layers in the branch.

Regarding the fusion block, the three fusion algorithms we implemented used the following thresholds during the experiments: IoU threshold = 0.4, skip-box threshold = 0.01, $\sigma = 0.5$. Due to computation constraints, we only evaluated ResNet-18 in this work; however, this architecture can be directly used with larger image-processing models (*e.g.*, ResNet-34/50/152, DenseNet-169, VGG-16) by simply changing the image processing backbone and picking a different split-point to divide it between the stems and the branches.

5.2.2 Gating Module Specification and Training. We implemented deep convolutional networks for the Deep and Attention Gating methods. As shown in Figure 3, the Deep Gating model is implemented as a 3-layer CNN with an MLP layer to map the CNN output to seven output channels, corresponding to the branch ranking for the seven branches. The Attention Gating method differs in that a self-attention layer is added after the CNN but before the max pooling and MLP layers. We trained the Deep and Attention Gating methods to estimate the aggregated loss of each branch for a given input using regression with mean absolute error as the loss function. The top-k lowest loss branches predicted by the gate were selected for fusion. To prevent the gate model training process from affecting the training process of the HydraFusion model, we trained and evaluated the gating modules separately using the stem outputs and branch losses of a fully trained HydraFusion model as the inputs and targets for the gate. After training, the gate model can be re-introduced into the HydraFusion model for deployment.

As mentioned in Section 4.2, the Knowledge Gating approach uses external context and domain knowledge to inform the branch ranking. During inference, we query the knowledge base using the external context for each input and return the branch rankings defined for that context. For the Optimal Gating method, we take the loss between the ground-truth boxes and the branch outputs for each branch and use this information to rank the branches — the branches with the lowest aggregated loss are ranked the highest.

5.2.3 Perspective Mapping. Since the RADIATE dataset contains data from both forward-facing (stereo cameras) and birds-eye view (radar and lidar) perspectives, we used a transformation matrix to transform the predicted bounding boxes from the birds-eye view (BEV) sensors to the forward-facing perspective (FWD). This enabled us to fuse the detections from both perspectives in the fusion block. To allow a fair assessment in our analysis across the different sensor modalities, we chose the cameras' field of view to be the fused reference frame as it was the limiting factor since it covers the least area. This prevents the objects detected by the lidar and radar branches from dominating the model when objects are detected outside the cameras' field of view. The transformations from the various other sensors to the reference frame are detailed further in Appendix A. We postulate that our approach could be directly applied in scenarios with full 360-degree camera coverage without loss of generality.

5.3 Training and Scoring Metrics

We built, trained, and evaluated each model in PyTorch using a batch size of 1 with a learning rate of 5e-3 for training the stem/branch models and 5e-5 for training the gate models. We trained all of the branches simultaneously on the dataset and averaged the loss across the branches before backpropagating in each training update step. We computed the classification and box regression loss for each branch using the multi-task loss function defined and used in Faster R-CNN [23]. Please reference their work for more details on the loss calculation.

To score the models on object detection, we used the mean average precision (mAP) score, which is widely utilized as the primary metric for benchmarking object detection models [8, 23]. We compute the mAP for boxes with an intersection-over-union (IoU) $\geq$0.5, which aligns with the PASCAL Visual Object Classes (VOC) Challenge [8]. Precision (P) and recall (R) for each class in the dataset are defined as:

$$P = TP/(TP + FP), \; R = TP/(TP + FN) \tag{13}$$

where TP, FP, and FN represent the number of true positive, false positive, and false negative classifications, respectively, by the model at a set confidence threshold. Average precision (AP) is a measure of the area under the precision-recall curve and is calculated as follows:

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{14}$$

where R_n and P_n correspond to the recall and precision at threshold n on the precision-recall curve. We calculate the mAP as the mean of the AP across all object classes where every object instance is weighted equally.

5.4 Results

Here we present our experimental results for object detection, evaluate our proposed gating methods, and benchmark our model's performance on industry-standard AV hardware.

5.4.1 Object Detection Results. In Table 2, we show the mAP achieved by different model configurations on the dataset. Results are shown for (i) individual sensors, (ii) early fusion between sensors, (iii) late fusion between sensor-specific branches, and (iv) our proposed HydraFusion approach. For the results in sets (i) and (ii), the mAP is calculated from a single ResNet-18 FasterRCNN model taking the stated sensor data as input. The late fusion results are computed by processing each sensor modality separately through a ResNet-18 FasterRCNN model and fusing the outputs of each model using one of the three fusion algorithms (WBF, NMS, or Soft-NMS), with the best performing result shown in the table. All-Branches (Early + Late) is the result from running all of the branches in HydraFusion and fusing the results using the fusion algorithms. Set (iv) shows the results for our selective sensor fusion approach using the Attention Gating method to select the Top-3 branches for each input.

Fusion Method	Model	mAP %
(i) No Fusion	Single Camera	65.33
	Radar	69.42
	Lidar	61.86
(ii) Early Fusion	L/R Cameras	65.33
	Radar + Lidar	71.63
	Camera + Lidar	65.99
(iii) Late Fusion	L/R Cameras	65.71
	Radar + Lidar	65.33
	L/R Cameras + Lidar	66.20
	Radar + Lidar + L/R Cameras	71.16
	All-Branches (Early + Late)	65.47
(iv) HydraFusion (Ours)	**Top-3 Branches w/ WBF**	**74.54**
	Top-3 Branches w/ NMS	**78.51**
	Top-3 Branches w/ Soft-NMS	**81.31**

Table 2: Object detection mAP scores on the RADIATE dataset for: (i) single sensors, (ii) early fusion, (iii) late fusion, and (iv) HydraFusion (ours) with Attention Gating.

Interestingly, All-Branches performs *worse* than all the results in (iv), supporting our hypothesis that using less sensor data can improve robustness. The tradeoffs between early and late fusion approaches are also shown. Early fusion can perform better with fewer sensors if the sensors provide good quality data (Radar + Lidar). In comparison, late fusion is more robust to bad data but requires more sensors to achieve good performance (Radar + Lidar + L/R Cameras). The table also shows the benefits of fusion compared to single-sensor approaches, as all fusion variants outperform (i) in at least one configuration. Figure 4 shows the average mAP for each fusion method. As shown, HydraFusion significantly outperforms both early and late fusion approaches on average (by 13.66% and 14.54%, respectively), achieving a peak mAP of **81.31%**. Overall, the results support our hypothesis that a context-aware selective sensor fusion architecture is significantly more robust and accurate than existing fusion methods.

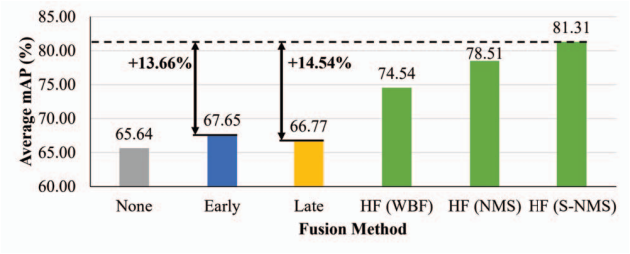

Figure 4: Average mAP for each fusion method compared against HydraFusion (HF).

Gate Model (Fusion Alg.)	mAP %			
	$k = 1$	$k = 3$	$k = 5$	$k = All$
Knowledge Gating (NMS)	77.59	76.37	76.53	75.81
Deep Gating (NMS)	67.43	78.14	73.31	75.81
Attn. Gating (Soft-NMS)	67.27	**81.31**	69.88	65.71
Optimal Gating (Soft-NMS)	73.03	**81.57**	72.93	65.71

Table 3: Gating evaluation for different k. The highest mAP indicates which gate configuration is best for real-world deployment of HydraFusion.

5.4.2 Gating Method Evaluation. Next, we evaluate our proposed gating strategies. To evaluate the impact of different subset sizes k on each of our proposed gating methods, we computed the mAP after fusion for $k \in \{1, 3, 5, All(7)\}$ with WBF, NMS, and Soft-NMS fusion. The results for the best performing fusion algorithm for each gating approach are shown in Table 3. Optimal Gating represents the theoretical best performance if the k lowest-loss branches are selected for each input.

As shown, Attention Gating using Soft-NMS achieves the best mAP for 3-branch fusion, with a score of **81.31%** (only 0.26% less than Optimal Gating). This likely results from its capability to identify the regions in the input that are most relevant to the output. Deep Gating was the second-best approach with a mAP of 78.14% for 3-branch fusion as it was still able to identify the context well using the stem features.

Interestingly, Knowledge Gating performed best for $k = 1$, likely because the domain knowledge was sufficient to determine the best modality for each context. However, Knowledge Gating did not achieve as high of a mAP score as Deep and Attention Gating for any k, meaning that its performance across contexts is generally worse. Besides, in real-world deployments, $k = 1$ would be insufficiently

robust to sensor obstruction or failures, so $k = 1$ performance is less relevant to real-world use cases than $k \in \{3, 5, All\}$ performance. For our application, $k = 5$ and $k = All$ did not perform as well as $k = 3$. Overall, the results in Table 3 show that Attention Gating with 3-branches results in the highest object detection mAP score (4.94% higher than Knowledge Gating) and is thus the best configuration to use on an actual vehicle.

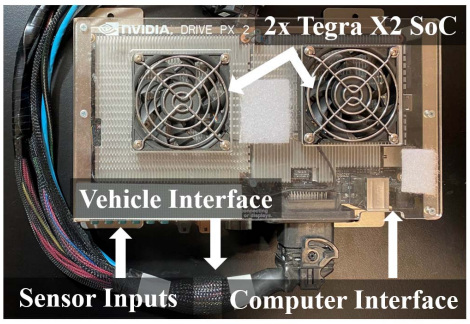

Figure 5: Nvidia Drive PX2 Testbed.

Fusion Method	Configuration	Energy (J)	Latency (ms)	Memory (MB)
None	Radar or Lidar	0.954	21.85	769
	Single Cam.	0.945	21.57	767
Early Fusion	L/R Cam.	1.192	27.36	768
	L/R Cam. + Lidar	1.379	31.36	694
	L/R Cam. + Lidar + Radar	1.615	36.86	750
Late Fusion	L/R Cam.	1.959	43.99	923
	L/R Cam. + Lidar	2.878	64.09	1087
	L/R Cam. + Lidar + Radar	3.769	84.32	1239
HydraFusion	3-Branch (Deep Gating)	3.317	73.84	1271
	3-Branch (Attn. Gating)	**3.284**	**73.02**	**1080**
	5-Branch (Deep Gating)	5.008	110.58	1390
	5-Branch (Attn. Gating)	4.897	107.28	1390

Table 4: Hardware evaluation on the Nvidia Drive PX2. Reported numbers are for processing one input through the model.

5.4.3 Hardware Energy and Latency Evaluation. To demonstrate that our approach is practical for real-world deployment, we analyze the energy consumption, latency, and memory usage of our model on an industry-standard AV hardware platform, the Nvidia Drive PX2, shown in Figure 5. To perform hardware analysis, we compiled each model specification using TensorRT and used built-in tools to measure its end-to-end latency and memory usage. Then, we multiply this value by the power consumption of the system measured via an external power meter to obtain the energy consumption.

In Table 4, we show the results for running different model variations including single sensor models, early fusion models, late fusion models, and our HydraFusion methodology. The HydraFusion 3-branch results shown are for the worst-case energy and latency scenario where all three branches selected by the gate are early-fusion branches. Similarly, the HydraFusion 5-Branch result is with three early-fusion branches and two single-sensor branches selected. The HydraFusion results are shown with Deep Gating and Attention Gating modules.

As expected, the single-sensor and early fusion methods are the least demanding on hardware since they only use a single ResNet-18 Faster R-CNN model; however, they also achieve lower mAP scores overall, as shown in Table 2. The results show that the HydraFusion 3-Branch configurations have energy consumption, latency, and memory usage that is comparable to 3-sensor and 4-sensor late fusion models. This result means 3-branch HydraFusion can reasonably be used in cyber-physical systems where late fusion approaches are currently deployed. Since 3-branch HydraFusion achieves significantly higher mAP than both early and late fusion methods, it presents clear benefits over state-of-the-art methods. The 5-branch HydraFusion was slower and less energy efficient than 3-branch and also achieved a lower mAP score (as shown in Table 3), so 3-branch would be preferred for real-world implementation. For both 3-branch and 5-branch HydraFusion, Attention Gating was slightly more efficient than Deep Gating, likely because TensorRT better optimized its architecture.

5.5 Discussion

5.5.1 Practicality. As mentioned in Section 5.4.3, the energy, latency, and memory usage of HydraFusion on the industry-standard Nvidia Drive PX2 is comparable to that of late fusion — meaning that HydraFusion can be used in any CPS where late fusion is currently in use. Thus, to implement HydraFusion in a real AV, the trained model and hardware can be installed in the vehicle and integrated with the perception module of the existing modular AV software stack. Although we evaluated one hardware platform with four sensors in our experiments, our approach is hardware- and sensor-agnostic. It can be used with any hardware platform and sensor configuration by using the corresponding model compilation tools and aligning the sensor data to the model's input. Additionally, our approach can be applied to a wide range of CPS problems besides object detection. Any CPS application using sensor fusion can potentially benefit from our context-aware selective sensor fusion approach. The model size and memory requirements will increase proportionally with more sensors due to the increased number of branches. However, they will likely still be comparable to late fusion with the same number of sensors. The branches must also be defined using domain knowledge for the new task; for example, sensors that cover the same FOV or complement each other can be combined to form early fusion branches.

5.5.2 Limitations and Future Work. We statically defined the set of branches used in HydraFusion for AV object detection using domain knowledge. Thus, our approach does not enable selecting between every possible set of sensor combinations for each branch. Doing so would not be computationally feasible as the space complexity would be $O(2^n)$, and the training time would increase similarly. Thus, our approach currently requires domain knowledge to identify the subset of branches that provide the most coverage across scenarios without exceeding model complexity or size requirements. Future research could explore automated techniques for defining the optimal set of branches to use in the model.

Additionally, we only explored the use of top-k branch selection for multiple fixed k in this work. It would be valuable to explore if the branch selection parameters are learnable based on the data in future work. In this paper, we focused on the problem of object detection for AVs; however, our approach can be directly applied to a wide range of multi-modal CPS and internet-of-things (IoT) problems. Different backbone models or fusion methods can be used to enable HydraFusion to model new tasks, such as tracking, localization, and control. We also believe that improved gating strategies with temporal modeling components could provide avenues for improving context identification, task performance, and resource utilization. It would also be prudent to evaluate the difference in safety between our approach and existing methods, especially in challenging driving conditions.

6 CONCLUSION

In this paper, we present HydraFusion — a sensor fusion framework that can selectively fuse sensor inputs in a context-aware manner. We validate our approach through theoretical, qualitative, and quantitative analysis on the task of object detection performed by AV perception systems on a challenging and diverse real-world dataset. On average, our selective sensor-fusion approach achieved a mAP score **13.66%** and **14.54%** higher than early fusion and late fusion approaches, respectively, supporting our hypothesis that a context-aware selective sensor fusion approach improves robustness. Additionally, we proposed and evaluated several gating models to perform context identification and branch selection, finding that an attention-based deep learning gate model was **4.94%** more effective than static selection methods. Lastly, we evaluated our proposed approach on industry-standard AV hardware, showing that our approach had comparable energy consumption, latency, and memory usage to existing fusion architectures. Ultimately, HydraFusion offers a novel sensor fusion approach for multi-modal CPS that can not only improve performance but also help support safer autonomous driving.

REFERENCES

[1] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. 2016. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*. Springer, 516–532.

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3366–3375.

[3] Eduardo Arnold, Omar Y Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. 2019. A survey on 3D object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3782–3795.

[4] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. 2004. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons.

[5] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. 2017. Soft-NMS–improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*. 5561–5569.

[6] Changhao Chen, Stefano Rosa, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. 2019. SelectFusion: a generic framework to selectively learn multisensory fusion. *arXiv preprint arXiv:1912.13077* (2019).

[7] Changhao Chen, Stefano Rosa, Yishu Miao, Chris Xiaoxuan Lu, Wei Wu, Andrew Markham, and Niki Trigoni. 2019. Selective sensor fusion for neural visual-inertial odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10542–10551.

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (VOC) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[9] Di Feng, Ali Harakeh, Steven Waslander, and Klaus Dietmayer. 2020. A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving. *arXiv preprint arXiv:2011.10671* (2020).

[10] Daniel D Fong, Kourosh Vali, and Soheil Ghiasi. 2020. Contextually-aware fetal sensing in transabdominal fetal pulse oximetry. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 119–128.

[11] Vineet Gokhale, Gerardo Moyers Barrera, and R Venkatesha Prasad. 2021. FEEL: fast, energy-efficient localization for autonomous indoor vehicles. *arXiv preprint arXiv:2102.00702* (2021).

[12] Yiping Gong, Zhifeng Xiao, Xiaowei Tan, Haigang Sui, Chuan Xu, Haiwang Duan, and Deren Li. 2019. Context-aware convolutional neural network for object detection in VHR remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing* 58, 1 (2019), 34–44.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[14] Radoslav Ivanov, James Weimer, and Insup Lee. 2018. Context-aware detection in medical cyber-physical systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 232–241.

[15] Sanghoon Lee, Dongkyu Lee, Pyung Choi, and Daejin Park. 2020. Accuracy–power controllable lidar sensor system with 3D object recognition for autonomous vehicle. *Sensors* 20, 19 (2020), 5706.

[16] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. 2019. Scale-aware trident networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6054–6063.

[17] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. 2018. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. 751–766.

[18] Enrique David Martí, David Martín, Jesús García, Arturo De la Escalera, José Manuel Molina, and José María Armingol. 2012. Context-aided sensor fusion for enhanced urban navigation. *Sensors* 12, 12 (2012), 16802–16837.

[19] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. 2018. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8080–8089.

[20] National Transportation Safety Board. 2019. *Collision between vehicle controlled by developmental automated driving system and pedestrian*. Technical Report NTSB/HAR-19/03. National Transportation Safety Board.

[21] National Transportation Safety Board. 2020. *Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator*. Technical Report NTSB/HAR-20/01. National Transportation Safety Board.

[22] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. 2019. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE, 1–7.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015), 91–99.

[24] Francisca Rosique, Pedro J Navarro, Carlos Fernández, and Antonio Padilla. 2019. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors* 19, 3 (2019), 648.

[25] Sara Saeedi, Adel Moussa, and Naser El-Sheimy. 2014. Context-aware personal navigation using embedded sensor fusion in smartphones. *Sensors* 14, 4 (2014), 5742–5767.

[26] Babak Shahian Jahromi, Theja Tulabandhula, and Sabri Cetin. 2019. Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles. *Sensors* 19, 20 (2019), 4357.

[27] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, and Andrew Wallace. 2020. RADIATE: A radar dataset for automotive perception. *arXiv preprint arXiv:2010.09076* 3, 4 (2020), 7.

[28] Lauro Snidaro, Jesús García, and James Llinas. 2015. Context-based information fusion: a survey and discussion. *Information Fusion* 25 (2015), 16–31.

[29] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. 2021. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing* 107 (2021), 104117.

[30] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. 2018. Pointfusion: Deep sensor fusion for 3D bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 244–253.

[31] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention generative adversarial networks. In *International Conference on Machine Learning*. PMLR, 7354–7363.

A SENSOR COORDINATE FRAME TRANSFORMATIONS

The transformations to convert the detections from one sensing modality to the reference frame occur in three main steps, which are detailed for the radar sensor as follows. Firstly, a point in the radar pixel coordinates, $[u, v]^r$, is transformed into the radar Cartesian frame:

$$[x, y]^r = \gamma \cdot ([u, -v]^r - [w/2, -h/2]), \tag{15}$$

where γ is the radar resolution expressed as meters/pixel, w is the width of the radar image in pixels, and h is the height of the radar image in pixels. An additional step to add a height, z^r, is computed by mapping the object's classification to a defined set of average class heights. Secondly, this Cartesian representation in the radar coordinate frame must be expressed in the chosen reference frame — the camera Cartesian frame. This is accomplished by subsequent the translation and rotation of coordinate frames via the following:

$$[x, y, z]^c = R_r^c \cdot ([x, y, z]^r + T^r), \tag{16}$$

where the superscript c indicates the world camera frame, R_r^c is the 3×3 rotation matrix from the radar to the camera frame, and T^r is the 1×3 translation vector between the radar and camera. Thirdly, the intrinsic parameters of the camera are used to convert the points from the Cartesian camera frame to the pixels in the image frame:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}^c = P \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}^c, \tag{17}$$

$$P = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

where s is an arbitrary scaling factor and P is the camera intrinsic projection matrix, which is constructed during calibration of the camera using camera focal length parameters (f_x, f_y) along with the principal point, or optical center of the camera (c_x, c_y). Using two stereo cameras provides the ability to derive depth information from the image to complete the necessary transformation presented above. The same procedures are repeated for the other sensors, with the respective adjustments to the translation and rotation vectors as needed. We note that this transformation process introduces additional uncertainties into the fusion as factors such as road elevation can alter the result in certain scenarios.

B ADDITIONAL RESULTS

B.1 Extended Object Detection Results

Table 5 expands on the late fusion results from Table 2 and shows a comparison between different late fusion algorithms for the chosen sensor combinations. Here, we examine results using WBF, NMS, and Soft-NMS across the different models. Only the fourth model, Radar + Lidar + L/R Cameras, has a noticeable improvement using WBF, while the other variations remained within similar score ranges. We attribute the closeness of these results to the three chosen fusion algorithms having similar statistical techniques embedded within them.

Model	mAP %		
	WBF	NMS	Soft-NMS
L/R Cameras	65.71	65.71	65.71
Radar + Lidar	65.33	65.33	65.33
L/R Cameras + Lidar	66.06	**66.20**	66.18
Radar + Lidar + L/R Cameras	**71.16**	67.11	65.42
All-Branches	64.85	63.64	**65.47**

Table 5: Object detection mAP scores on the RADIATE dataset for different late fusion algorithms: (i) WBF, (ii) NMS, (iii) Soft-NMS.

Gate Model	Fusion Alg.	mAP %			
		$k = 1$	$k = 3$	$k = 5$	$k = All$
Knowledge Gating	WBF	76.30	75.56	74.66	73.96
Knowledge Gating	NMS	**77.59**	76.37	76.53	75.81
Knowledge Gating	Soft-NMS	76.95	68.75	68.75	65.71
Deep Gating	WBF	67.62	75.19	72.95	73.96
Deep Gating	NMS	67.43	**78.14**	73.31	75.81
Deep Gating	Soft-NMS	67.27	77.36	74.70	65.71
Attn. Gating	WBF	67.86	74.54	72.93	73.96
Attn. Gating	NMS	67.43	78.51	73.47	75.81
Attn. Gating	Soft-NMS	67.27	**81.31**	69.88	65.71
Optimal Gating	WBF	75.57	74.62	72.45	73.96
Optimal Gating	NMS	74.69	77.10	73.20	75.81
Optimal Gating	Soft-NMS	73.03	**81.57**	72.93	65.71

Table 6: Evaluation of different gating methods for selecting the k best branches for across different fusion methods. For each input, the top k branches selected by the gate are fused to produce a set of detections scored using mAP.

B.2 Extended Gating Results

Table 6 shows our extended gating results. It includes mAP scores evaluated with the four gating strategies, each used with WBF, NMS, and Soft-NMS fusion. The results indicate that NMS and Soft-NMS with $k = 3$ result in the highest mAP score for most of the gating methods evaluated. Interestingly, NMS seems to be more robust to different k values than Soft-NMS, which varies by up to 15% depending on k. WBF works well with Knowledge Gating for all k and works decently well for the other gates with $k \in \{3, 5, All\}$. As mentioned in Section 5.4.2, regardless of the performance of other configurations, only the highest scoring configuration would be deployed in an actual vehicle. Thus, these extended results confirm that Attention Gating with Soft-NMS and $k = 3$ is the best configuration to deploy in the real world.

B.3 Branch Selection

In Table 7, we show the frequency at which each branch was selected by each gate model for different values of k. The branches listed are the seven branches explicitly defined in our experiments (four single-sensor and three early fusion). The branch selection rate is the percent of inputs in the test dataset for which a specific branch was selected as part of the top-k. The selection results for a single input can vary depending on the context; however, these aggregated results illuminate which sensors contributed more to the final detection results than others. The deep learning-based

k	Gate Model	Branch Selection Rate (%)						
		Radar	L Cam.	R Cam.	Lidar	L/R Cam.	L/R Cam.+ Lidar	Radar+Lidar
1	KnowledgeGating	8.61	0.00	0.00	0.00	**76.82**	0.00	14.57
	DeepGating	7.95	0.00	0.00	18.54	1.32	19.87	**52.32**
	AttentionGating	9.93	0.66	0.00	6.62	8.61	12.58	**61.59**
	Optimal Gating	19.87	4.64	1.99	**25.83**	13.25	9.27	25.17
3	KnowledgeGating	23.18	**76.82**	**76.82**	23.18	**76.82**	0.00	23.18
	DeepGating	78.81	3.31	12.58	66.23	25.17	31.79	**82.12**
	AttentionGating	78.81	1.32	7.95	72.19	28.48	28.48	**82.78**
	Optimal Gating	74.17	16.56	15.23	62.91	32.45	23.84	**74.83**
5	KnowledgeGating	23.18	76.82	76.82	23.18	**100.00**	**100.00**	**100.00**
	DeepGating	**97.35**	32.45	21.19	87.42	82.78	94.04	84.77
	AttentionGating	89.40	29.14	19.87	**96.03**	76.82	95.36	93.38
	Optimal Gating	**87.42**	54.30	47.68	80.13	67.55	78.81	84.11

Table 7: Evaluation on how often each gate model selected each branch as part of the top-k for various k. The branch selection rate is expressed as a percentage over the number of inputs for the test dataset.

gating models heavily favored selecting radar and lidar branches. We propose that this dependence on sensors that are traditionally more robust to severe weather was reinforced throughout the model's learning process as feedback taught the model that the cameras were susceptible to high amounts of error in specific contexts. Knowledge gating tends to favor the camera selection more often but does not perform as well as the deep learning models (see Table 6). This illuminates a limitation in using domain knowledge to define the gate as some sensors, like cameras, dominate the selection process. Optimal Gating shows the most consistent responses across the branches as expected due to its *a posteriori* knowledge of each branch's loss.