TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214
DOI: 10.26599/TST.2022.90100
Volume xx, Number x, xxxxxxx 20xx

CPT: A Configurable Predictability Testbed for DNN Inference in AVs

Liangkai Liu, Yanzhi Wang, and Weisong Shi*

Abstract: Predictability is an essential challenge for autonomous vehicles' safety. Deep neural networks have been widely deployed in the AV's perception pipeline. However, it is still an open question on how to guarantee the perception predictability for AV because there are millions of DNN model combinations and system configurations when deploying DNNs in AVs. This paper proposes CPT, a configurable testbed for quantifying the predictability in AV's perception pipeline. CPT provides flexible configurations of the perception pipeline on data, DNN models, fusion policy, scheduling policies, and predictability metrics. On top of CPT, the researchers can profile and optimize the predictability issue caused by different application and system configurations. CPT has been open-sourced at: https://github.com/Torreskai0722/CPT.

Key words: predictability; deep neural network; autonomous driving

1 Introduction

Deep neural networks (DNNs) are widely used in autonomous driving due to their high accuracy for perception, decision, and control [51]. Predictability of the perception module is essential for the AV's safety. Predictability generally consists of temporal and functional aspects: [47]. Temporal aspects mean the task should be finished before the deadline. Functional aspects mean the task should make correct decisions.

Variabilities in time are non-negligible during DNN inference [79, 82]. Earlier research pinpointed these variations, especially in mobile devices, and noted that the inference durations typically align with a Gaussian distribution [82]. Another study focusing on the "anytime DNN system" highlighted these timing variations and introduced a Kalman Filter methodology for estimating latency distribution [79]. The D^3 approach offers a solution to these timing variations in autonomous vehicle systems by proposing dynamic instead of static deadlines [26]. This approach allows the perception, planning, and sensing modules to have dynamic deadlines and change adaptively. The Prophet system tackles timing variations in two-stage models by utilizing the insights that intermediate object/lane proposals cause time varaitions [50]. It's important to recognize that understanding DNN inference time variations is a challenging problem, influenced by various elements such as traffic conditions, DNN model categories, scheduling strategies, fusion methods, DVFS settings, and more [52].

In-depth profiling and optimization of the DNN inference variations require a computing testbed that provides flexible configurations of the AV's perception pipeline. The predictability of the perception pipeline is still an open issue, given that the total number of configurations of these factors is enormous [85]. However, current AV testbeds or benchmarks ignore this issue or provide a limited number of perception pipeline configurations, including Autoware.ai, and Autoware.auto, Apollo, NVIDIA DriveWorks, and CAVBench [11, 19, 20, 63, 81]. A testbed that could

• Liangkai Liu and Weisong Shi are with the Department of Computer and Information Science, University of Delaware, Newark 19713, USA. E-mail: liangkai@udel.edu, weisong@udel.edu
• Yanzhi Wang is with the Department of Electrical & Computer Engineering, Northeastern University, Boston, 02115, USA. E-mail: yanz.wang@northeastern.edu
* To whom correspondence should be addressed.
Manuscript received: 2023-Oct-29; accepted: 2024-Feb-07

provide flexible configurations of all these factors to study the predictability issue in AV's DNN inference is still missing.

In this paper, we present CPT, a configurable testbed for quantifying the predictability in AV's perception pipeline. CPT is composed of three modules: the application configurations, the system configurations, and the evaluator. It provides two image datasets covering over 100 scenarios as sensor data. CPT provides Over 19 million DNN model combinations through the integration with the model zoo of OpenMMLab and BDD100K. CPT also supports a flexible configuration of scheduling, policies, and DVFS policies. Finally, several metrics represent the time and performance variations for every task and the fusion task. In summary, this paper makes the following two contributions:

- We observe and redefine the predictability issue for DNN inference, which consists of time and performance predictability.
- We design and implement CPT, a configurable predictability testbed for DNN inference in AVs. We open-source the CPT testbed for the research on predictable DNN inference.

The rest of the paper is organized as follows. Section 3 presents the background and motivation of this work. Section 4 describes the design of the CPT testbed. Section 5 presents the implementation of CPT. Section 6 discusses two use cases for studying DNN inference predictability. The related work is presented in Section 2. Section 8 concludes the paper.

2 Related Work

Autonomous vehicles are proposed to understand the environment and drive without human intervention [53]. DNNs play an essential role in the sensing, perception, decision, and control tasks in autonomous driving. Generally, the prior research on real-time DNNs for autonomous driving can be divided into two categories: model compression and inference time variations.

Model Compression After model training, a fundamental challenge for deployment is how to optimize the model inference in execution time, energy consumption, as well as memory utilization. Han and colleagues [30] have proposed a technique that involves pruning redundant connections and then retraining the DNN models. This approach reduces the computational demands effectively. Another direction for the runtime optimization of DNN inference is to reduce the precision of operations and operands, which is usually achieved by reducing the number of bits/levels representing the data and reducing computation requirements and storage costs [75].

Inference Time Variations Variabilities in time are non-negligible during DNN inference [79, 82]. Earlier research pinpointed these variations, especially in mobile devices, and noted that the inference durations typically align with a Gaussian distribution [82]. Another study focusing on the "anytime DNN system" highlighted these timing variations and introduced a Kalman Filter methodology for estimating latency distribution [79]. However, it cannot handle huge time variations since understanding roots causing time variations in DNN inference is missing. A scheduling-based approach is leveraged to provide predictable DNN inference [28]. However, the system is based on a cloud server-level GPU platform, which is impractical for an onboard embedded system on an autonomous vehicle. The D^3 approach offers a solution to these timing variations in autonomous vehicle systems by proposing dynamic instead of static deadlines [26]. This approach allows the perception, planning, and sensing modules to have dynamic deadlines and change adaptively. However, since the roots for DNN inference time variations are not studied, the proposed approach still wastes lots of time processing frames that are supposed to miss the deadline. The Prophet system tackles timing variations in two-stage models by utilizing the insights that intermediate object/lane proposals cause time varaitions [50]. It's important to recognize that understanding DNN inference time variations is a challenging problem, influenced by various elements such as traffic conditions, DNN model categories, scheduling strategies, fusion methods, DVFS settings, and more [52].

3 Background and Motivation

Predictability is essential for safety-critical applications like autonomous vehicles [39]. Generally, predictability consists of two perspectives: accurate and real-time [47]. Previous works treated the temporal and functional as separate aspects, which is not enough to achieve predictability [23, 57, 85]. Besides, they

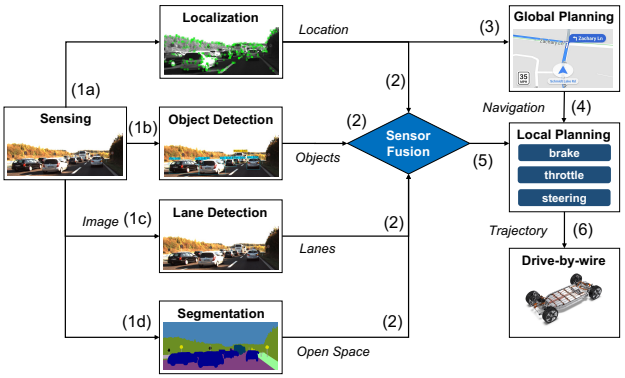

Fig. 1 A general autonomous driving pipeline.

have yet to discuss how to evaluate the predictability of multi-tasks with dependencies. In this section, we present a new definition of predictability.

3.1 DNNs for AV's Perception

AV Pipeline DNNs play a significant role in the whole pipeline of driving autonomously. Figure 1 shows a generalized pipeline for autonomous driving. A sensing node publishes the captured sensor data to all the perception nodes for localization (step 1a) [1, 60, 61], object detection (step 1b) [33, 70, 71], lane detection (step 1c) [41, 62, 64, 90], and segmentation (1d) [16, 72]. Next, the perception results are submitted to a sensor fusion node (step 2), which combines the information on the vehicle's location, object, lanes, and open spaces [29, 39]. The location is also published to the global planning node to calculate a navigation route to the destination [4]. The navigation route (step 4) and sensor fusion results (step 5) are both published to the local planning stage [3], which constructs a local driving space cost map and generates vehicle trajectories and publishes it to the vehicle's drive-by-wire system (step 6) [12, 65]. Finally, the drive-by-wire system sends control messages to ECUs through the Controller Area Network (CAN bus) to drive the vehicle [18].

Object Detection Object detection accuracy is pivotal for environmental perception in autonomous vehicles (AVs) [58]. DNN-based object detection has evolved through the R-CNN series [24, 25, 71], SSD series [21, 54], the YOLO series [68–70], and vision transformer-based [15, 55]. The use of deep learning in object detection was initiated with R-CNN in 2014, leading to faster iterations, Fast R-CNN and Faster R-CNN [24, 25, 71]. YOLO, the premier one-stage object detector, emerged in 2015 [68, 69], progressing to versions such as YOLOv3 and YOLOv4 [13, 70]. SSD balanced speed and accuracy using regression technologies [54]. ViT-based models replace the CNNs with multi-head attention for better feature extraction and classification [15, 55].

Lane Detection and Segmentation Real-time lane detection is a vital function in autonomous driving [62]. Recently, deep learning-based segmentation approaches have dominated this field for their accuracy [27]. VPGNet introduces a multi-task network for lane marking detection [42], while SCNN employs a novel convolution operation to effectively aggregate various dimension information [64]. Light-weight DNNs are developed for real-time applications, like the self-attention distillation mechanism in SAD [35]. Alternative methods, such as sequential prediction and clustering, are also deployed. For instance, an LSTM network addresses the lane's linear structure [44], Fast-Draw predicts the lane direction pixel-wise [66], and in [36], lane detection is redefined as a binary clustering problem, a technique further adopted by [34]. Lastly, a 3D form of lane detection is introduced to tackle non-flat terrains [22]. Segmentation is another pixel-level classification task in AV's pipeline to find open driving space [37]. Mask R-CNN is a generalization of Faster R-CNN for semantic segmentation [33]. PSPNet is another model that leverages the feature pyramid networks with dilated convolutions for segmentation [88]. Deeplab series are recent works leveraging sparse convolutions for segmentation [16].

3.2 Perception Predictability

3.2.1 DNN Inference Time variations

DNN models show non-negligible inference time variations when running on mobile or embedded systems [26, 52, 79, 82].

By deploying the model zoo of the DBB100K dataset [2], we tested eight DNN models for each of the three perception tasks. For object detection, we choose ATSS [87], Cascade R-CNN [14], ConvNeXt with Cascade R-CNN [56], Faster R-CNN [71], FCOS [77], RetinaNet [48], Sparse R-CNN [74], and Swin Transformer with Cascade R-CNN [55]. For lane (drivable space) detection, we choose APCNet [32], CCNet [38], Deeplabv3+ [16], DMNet [31], DNLNet [83], HRNet [80], PSANet [89], and Swin Transformer [55]. For semantic segmentation, we choose Deeplabv3+ [16], EMANet [46], HRNet [80],

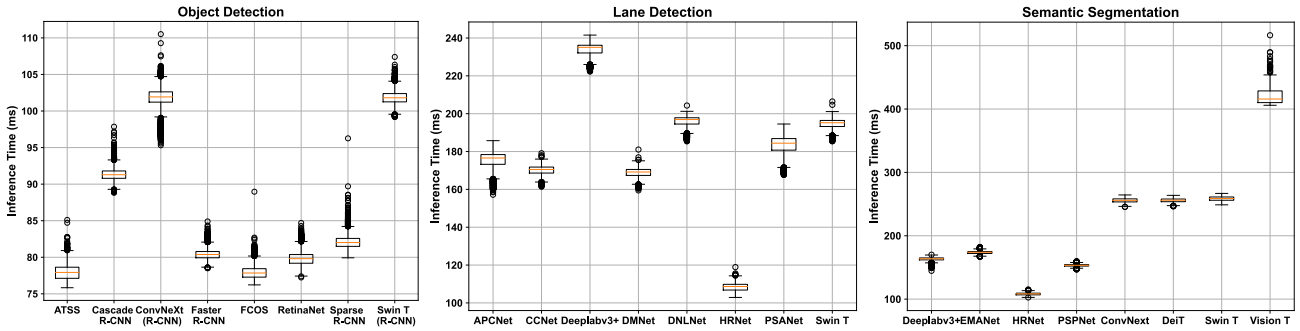

Fig. 2 The box plot of DNN inference time for object detection, lane detection, and semantic segmentation.

PSPNet [88], ConvNeXt [56], DeiT [78], Swin Transformer [55], and Vision Transformer [17]. The input to the DNN model includes over 4800 images from the BDD100K video dataset [84].

The box plot of the inference time for all the DNN models is shown in Figure 2. We can observe non-negligible time variations among all the models. However, the time variations issue shows differently for different tasks. Object detection models show much higher time variations than the other two tasks. However, regarding the average inference time, semantic segmentation is the largest while object detection is the smallest. Among the object detection models, the time variation issue also shows differently. Two stage-based detectors, Cascade R-CNN, ConvNeXt, Faster R-CNN, Sparse R-CNN, and Swin Transformer, show much higher time variations than one stage-based detector [52]. This is because the region proposal network (RPN) generates a dynamic number of object proposals which makes the execution of the second stage dynamic [45,50,71]. Since different traffic scenario has a different number of potential objects, the actual execution time for the same R-CNN model in different scenarios (city, residential, Road, etc.) also has inference time variations [8,45,52].

The inference time variations bring a big challenge for the scheduling of AV's operating system. Enormous resources are wasted if the scheduler is unaware of the inference time variations. If we consider the autonomous driving system a hard real-time system, the scheduler assigns deadlines for each task based on the worst observed execution time [26,52].

Observation 1: *Non-negligible time variations exist in the runtime of DNN models in AV's perception pipeline. The time variation is a cross-layer issue related to the input data, the DNN model's structure, and the runtime system scheduling.*

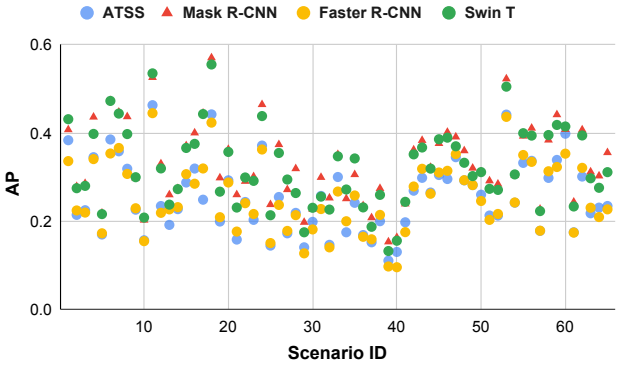

Fig. 3 Performance variations of object detection models under different scenarios.

3.2.2 DNN Performance Variations

DNN-based models exhibit variations in performance across different scenarios and inference times. We examine both offline and online scenarios. In the offline scenario, the impact of inference time is not considered, and the average precision is calculated based on the input image. In contrast, the online scenario accounts for inference time and evaluates the image with the added inference time.

Offline Detection Performance Variations. Here we leverage the Argoverse-HD dataset, which provides high-frame-rate annotations for streaming evaluation [45]. The Argoverse-HD dataset includes 39,384 images covering 65 scenarios. These scenarios cover traffic environments like downtown, rural areas, highways, daytime, and night.

We test four object detection models (ATSS [87], Mask R-CNN [33], Faster R-CNN [71], and Swin Transformer [55]) on each scenario and calculate the average precision. All four models are trained with the MS COCO dataset [49]. Figure 3 shows the scatter plot of AP for all four models under different traffic scenarios. We can observe that all four models show huge performance variations under different scenarios.

Table 1 The AP comparison of different object sizes under scenarios 18 and 40.

Scenario 18	AP	AP small	AP medium	AP large
ATSS	0.442	**0.028**	0.458	0.664
Mask R-CNN	0.571	**0.198**	0.635	0.733
Faster R-CNN	0.423	**0.038**	0.458	0.646
Swin T	0.555	**0.132**	0.603	0.734
Scenario 40	**AP**	**AP small**	**AP medium**	**AP large**
ATSS	0.130	**0.038**	0.249	0.713
Mask R-CNN	0.163	**0.049**	0.365	0.776
Faster R-CNN	**0.095**	**0.027**	0.270	0.521
Swin T	0.155	**0.051**	0.346	0.764

Table 2 The number of small/medium/large objects under the scenarios 18 and 40.

objects / ratio	small	medium	large
scenario 18	2560 / 0.416	2127 / 0.346	1465 / 0.238
scenario 40	**9966 / 0.579**	**5921** / 0.344	1335 / 0.078

The highest AP is 0.57 by Mask R-CNN under scenario 18, while the lowest AP is 0.095 by Faster R-CNN under scenario 40. Table 1 shows compares AP, AP small, AP medium, and AP large under both scenarios. All the detection models show the highest AP value for large objects and the lowest AP value for small objects. This is understandable since detecting small objects is more challenging than large objects due to their low resolution and noisy representation [43].

One primary reason for the performance variations under different scenarios is the composition of small, medium, and large objects. Table 2 shows the number and ratio of small/medium/large objects under scenarios 18 and 40. We found smaller objects like traffic lights and small vehicles in scenario 40 than in scenario 18.

Online Detection Performance Variations. The inference time significantly impacts the DNN model's performance for safety-critical applications like AV. Because when the detection is finished, the real-world environment has already changed, and there might be insufficient time for the AV to react in some emergencies.

To show the impact of inference time on the streaming perception's accuracy, we leverage the object detection models from the MMDetection project [6] and evaluate the box mAP by comparing the detected bounding boxes with the input image frame and the newest image frame [45]. Figure 4 shows the box mAP result and the average inference time of over 400 object detection models in MMdetection. The online streaming detection results are emulated by multiplying a box-changing ratio for every 33ms. The average box-changing ratio is calculated from the BDD100K video data. We compare the results of offline detection as well as real-time streaming detection. For offline detection, we can find that as the average inference time increases, the box mAP also increases [6, 76]. The DNN model with more parameters is expected to learn more features and patterns from the raw data. However, in real-time streaming detection, the box mAP decreases when the inference time increases. Because the longer the inference takes, the more the difference between the real world and the input frame [26, 45].

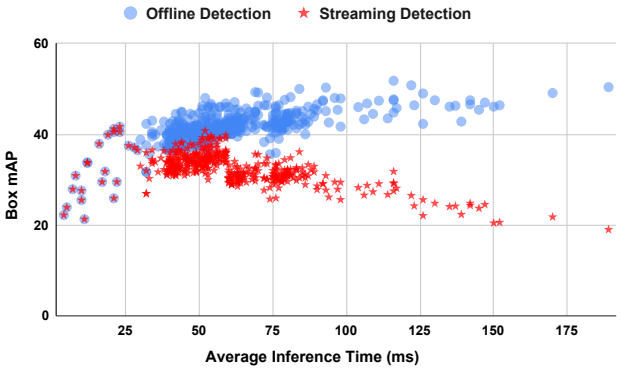

Fig. 4 The box mAP and inference time of offline and streaming online detection for 400+ object detection models in mmdetection [6].

The performance degradation of DNN detection models on streaming detection is mainly caused by the changing traffic environment when the DNN model is executing, which is a fundamental challenge for functional predictability [86].

Observation 2: *Significant performance variations are observed in DNN models used in AV's perception pipeline. The offline DNN performance variations are primarily attributed to the specific running scenario. In contrast, online DNN performance variations combine inference time and offline performance variations.*

4 CPT Design

Predictability is an essential challenge for the safety of AVs. This section begins with the overview and design principles for the CPT testbed, followed by a detailed introduction of each component.

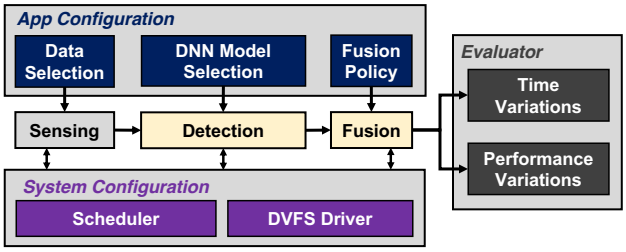

Fig. 5 An overview of the CPT testbed.

4.1 CPT Overview

The DNN inference time variations are related to several factors: the input image, the DNN model's structure, the scheduling policy, and other system configurations [52]. To profile and quantify the impact of these factors on the timing/performance variations, we propose CPT, a configurable testbed for quantifying the predictability in AV's perception pipeline.

The overview of the CPT testbed is shown in Figure 5. The general goal of the testbed is to provide APIs for defining the application, system, and evaluation pipeline. CPT comprises three main modules: the application configuration, the system configuration, and the evaluator. For the application's pipeline, CPT allows configurations on the input data, the DNN model's structure, and the fusion policy. For runtime systems, CPT includes interfaces for setting scheduling and DVFS policies.

4.2 Application Configurations

4.2.1 Data Selection

Data plays an essential role in DNN's inference time and performance variations. The data should be collected from a real environment under various traffic scenarios. Therefore, CPT provides two datasets: the Argoverse-HD dataset [45] and the BDD100K [84]. The Argoverse-HD dataset contains high-quality and temporally dense annotations for high-resolution videos (1920 x 1200 @ 30 FPS). Overall, there are 70,000 image frames and 1.3 million bounding boxes. The BDD100K dataset is a diverse dataset for multi-task learning. It consists of labels for ten tasks including object detection, semantic segmentation, and lane detection. It also contains high-resolution images (1280 x 720 @ 30 FPS).

4.2.2 DNN Model Selection

Three DNN-based tasks in the perception pipeline are shown in Figure 1: object detection, lane detection, and semantic segmentation. By integrating with the OpenMMLab model zoo and BDD100K model zoo, we use `config` files to load the DNN's structure and trained weights. Table 3 shows the supported DNN models for all three tasks. CPT provides 523 DNN models for object detection, 655 for semantic segmentation, and 56 for lane/drivable detection. These models almost cover all the designs for DNN's structure. Given that the DNN models for these three tasks are chosen individually, the CPT testbed could support over 19 million combinations of models in the perception pipeline.

Table 3 The number of supported DNN models.

Number of Models	Object Detection	Semantic Segmentation	Lane/Drivable Detection
OpenMMLab	464	602	-
BDD100K	59	53	56
Total	523	655	56

4.2.3 Fusion Policy

The fusion node combines the information on the vehicle's location, object, lanes, and open space from the perception tasks. The combination is a message synchronization based on the message timestamp. In CPT, we use a *message_filter* with the Approximate Time Synchronizer to manage the fusion process [5]. There are two parameters in the Approximate Time Synchronizer: the queue size and the slop. The queue size defines how many messages to cache for synchronization, while the slop defines the maximum interval duration. The 100ms slop means the message with a time difference of less than 100ms is considered synchronized.

4.3 System Configurations

4.3.1 Scheduling Configurations

Typically, Linux supports four scheduling policies: SCHED-OTHER, SCHED-FIFO, SCHED-RR, and SCHED-DEADLINE [7]. SCHED-OTHER is Linux's default scheduling policy for supporting user applications and maximizing processors' utilization. SCHED-FIFO schedules in a first-come-first-serve method, while SCHED-RR schedules in a round-robin way. SCHED-DEADLINE is a CPU scheduler based on the Earliest Deadline First (EDF) [73]. For each perception task, CPT provides configurations of their scheduling policies, the priority, the nice, and the interval value.

In addition, the computing device on AV is usually equipped with multiple CPU cores and multiple GPU cards. The CPT testbed also provides the configuration of the CPU affinity and the GPU affinity for each task. Each task can be configured to run on a group of CPU cores and GPU cards.

4.3.2 DVFS Configurations

Dynamic voltage and frequency scaling is a widely used technique that adjusts power and speed settings to optimize resource allotment for tasks and maximize power saving. In some cases, the AV's runtime system could have DVFS enabled for power saving [10]. Therefore, CPT also provides configurations to DVFS drivers to change the DVFS governor, CPU, and GPU frequency.

4.4 Evaluator

4.4.1 Time Variations

The time variation is measured for each task and the end-to-end perception task. The CPT testbed collects the inference time of each DNN model for each image under one scenario. The end-to-end perception task is the time from the image publishing until the fusion finishes. For both cases, we consider the time range (maximum minus minimum) and the coefficient of variations. The coefficient of variation (c_v) is used to evaluate the relative variability, and it is calculated using the standard deviation σ divided by the mean value μ. c_v is a positive value. The higher the value is, the higher the variations the data has.

4.4.2 Performance Variations

The performance variations are evaluated with different metrics on two datasets. For the Argoverse-HD dataset, all three tasks are evaluated with the COCO annotations, so mean average precision (mAP) is used to evaluate the performance of object detection, semantic segmentation, and drivable detection [49]. For the BDD100K dataset, mAP is used to evaluate the object detection performance. In contrast, mIoU (mean intersection of the union) is used to evaluate the performance of semantic segmentation and drivable area detection [2]. All these metrics are calculated in one scenario, and we compare the best, the average, and the worst values among all the scenarios.

We use the fusion ratio to evaluate the performance of the fusion result. The fusion ratio is the number of fusion messages divided by all the messages.

5 Implementation

Autonomous vehicles have various applications for sensing, perception, and decision. As a modular-based system, the system's performance is determined not only by each module but also by the coordination of several modules. We integrate its implementation into an AV perception prototype based on ROS in a multi-card GPU workstation with Linux RT kernel patched.

5.1 Overview of the Perception System

We implement the CPT testbed on top of ROS, as shown in Figure 6. The pipeline starts with the */image* node, capturing and publishing images from the cameras. Three ROS nodes subscribe */image_raw* messages and execute the DNN inference on the images, including object detection, lane detection, and semantic segmentation. Another perception node is responsible for Simultaneous Localization and Mapping (SLAM). After perception, four nodes publish their results, covering the position information, objects, lanes, and semantics. The */fusion* node subscribes to these four topics. It synchronizes them based on the timestamp to get the sensor fusion results, giving the control module the vehicle's obstacles and open driving space. Besides, each perception task and the sensor publisher publish its execution information based on the process context and intermediate inference results. This information includes the PIDs, priorities, nice values, and intermediate results like object proposals and pixel raw points. The *manager* node subscribes to this information and coordinates task execution by updating the policy parameters accessible by all the tasks. It also changes DVFS levels, CPU cores, and GPU card allocations based on the testing configurations.

5.2 ROS Nodes, Topics, and Parameters

In the ROS framework, nodes (represented as figures) are processes designed for specific computations. Conversely, topics (depicted as arrows) act as named channels allowing ROS nodes to exchange information, as highlighted by the reference [9]. Within the perception system, we have designed seven ROS nodes to gather and process sensor data: */image*, */object*, */slam*, */lane*, */segmentation*, */fusion*, and */manager*. Communication between these nodes relies on a publish-subscribe model in ROS for message exchange. Notably, our SLAM algorithm of choice is ORB-SLAM2, which employs a camera-centric approach for

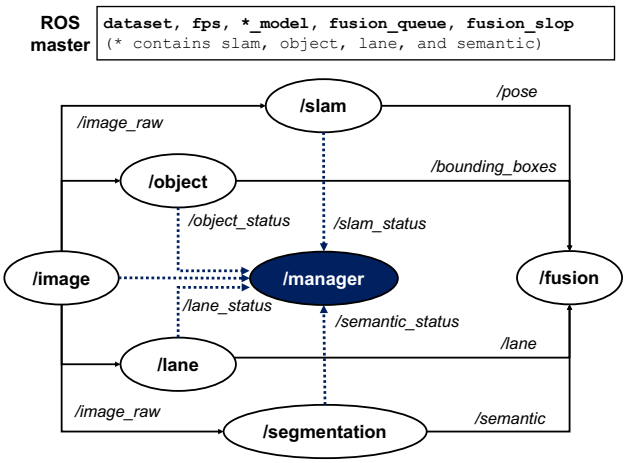

Fig. 6 The ROS implementation of the CPT testbed.

pinpointing pixel key points, vehicle localization, and simultaneous map creation [59]. We've incorporated three DNN models from the model zoo, which contains thousands of combinations.

Defining ROS topics as message conduits between ROS nodes, we've set up ten such topics to facilitate communication encompassing image data, positions, objects, and specialized messages. An overview of several ROS topics can be found in Table 4. Two messages, based on the *Image* type, encompass components like the header, height, width, encoding, and data. Every ROS topic's header is equipped with a sequence ID, timestamp, and frame ID that denote a specific message. To ensure synchronization, which necessitates both the timestamp and sequence ID, we've fashioned the */pose_timestamp* from */pose*, carrying data about position and orientation. When it comes to object detection, bounding boxes visually represent identified objects, based on the minimum and maximum values on the x and y axes, probability, and object category. */bounding_boxes* aggregates these boxes for an individual image and incorporates an internal header. The outcome is visualized with diverse colors within the image, symbolizing varied segments of semantic segmentation. The */semantic* topic, which is rooted in the *Image* type, presents these results alongside its message header. The */lane* topic integrates both a header and a curve, the latter being constructed from pixels described by three float values. Finally, the */*_status* category encompasses four distinct topics: */slam_status*, */object_status*, */lane_status*, and */semantic_status*. This tailor-made topic contains components like a header, process ID, scheduling approach, priority level, image sequence, runtime, proposals, and probability.

CPT leverages ROS parameters to adjust the application configurations. This global variable-based makes it possible to do both offline and online reconfiguring. The online adjustment of the application configurations could be made at the */manager* node when receiving a status message through the callback. As shown in Figure 6, eight ROS parameters are implemented in the perception system. For system configurations, we implement a socket called `proc_manager` which takes requests from the command line to change CPU/GPU affinity, CPU/GPU frequency, priority, and scheduling policy for a given process `PID`.

6 Evaluation

We now show how CPT enables researchers to study: 1) the DNN structure's impact on the inference time variations; and 2) the relationship between performance and scenarios.

Hardware and software setup. The evaluation is conducted on a GPU workstation. It has 28 Intel® Core™ i9-9940X CPUs with the highest frequency at 3.3GHz. The platform has 4 NVIDIA GeForce RTX 2080 Ti/PCIe/SSE2 GPU cards, providing 304 TOPS. Each GPU card has 4352 CUDA cores, supporting 10 Giga Rays/s and 14 Gbps memory speed. The GPU-shared memory has 11GB GDDR6 with 352 memory interface widths. In addition, the platform has 64 GB DDR4 memory. There are four DDR4-based memory devices, each 16GB with a speed of 2666 MT/s.

6.1 DNN structure's impact on inference time variations

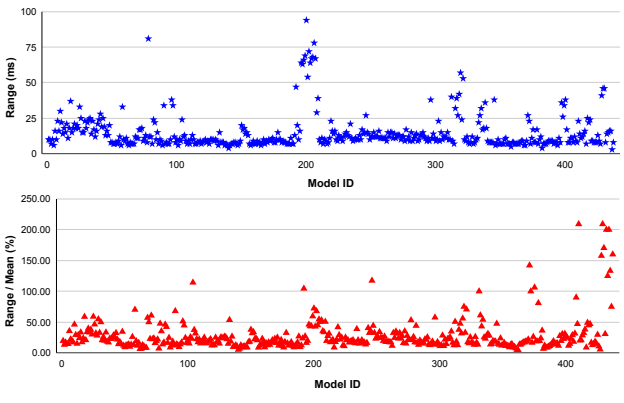

Fig. 7 The range and range/mean for all 464 models in mmdetection.

Table 4 ROS topics implemented in the perception system.

ROS Topics	Library	Type	Fields
/image_raw	sensor_msgs	Image	header, height, width, encoding, data, etc.
/pose	geometry_msgs	PoseStamped	header, position (x, y, z float64) orientation (x, y, z, w, float64)
/bounding_boxes	darknet_ros_msgs	BoundingBoxes	header, image_header, bounding_boxes
/bounding_box	darknet_ros_msgs	BoundingBox	probability, xmin, ymin, xmax, ymax, id, class
/semantics	sensor_msgs	Image	header, height, width, encoding, data, etc.
/lane	geometry_msgs	Points	header, curve
*/*_status*	process_status_msgs	ProcessStatus	header, pid, priority, image_seq, runtime, proposals, etc.

We measured the inference time of 464 DNN models in the mmdetection using the same input images to study the impact of DNN structure on inference time variations. Figure 7 shows the results of these models' inference time range and range/mean percentage on the GPU workstation. We found that many models have a variation range exceeding 50ms and a variation range/mean percentage larger than 50%. We also observed that models with larger than 50ms time variations share the same HTC structure, which features a skipping design that causes inference time variations. It also enables effective modeling of temporal dependencies in sequential data through its hierarchical architecture with temporal convolutions and skip connections [67].

6.2 The relationship between the performance and scenarios

In order to understand the relationship between AP performance and scenarios, we deployed four object detection models on all 65 scenarios and collected their AP results for small, medium, and large objects. Besides, we collected the ratio of small, medium, and large objects for each scenario. From the comparison of scenario 18 and scenario 40 results in Table 1 and Table 2, the breakdown of different object sizes contributes significantly to the final average AP value. Next, we calculated the correlation coefficients between the ratio of small, medium, and large objects and the results of AP for all scenarios. Table 5 shows the correlation analysis results. We can observe that both small and large object ratios have a high impact on the AP, while the small object ratio has a negative impact, and the large object ratio has a positive impact.

Table 5 The correlation coefficient between the AP and the small/medium/large object percentage.

r	*Mask R-CNN*	*ATSS*	*Faster R-CNN*	*Swin T*
small	-0.413	-0.398	-0.403	-0.388
medium	0.121	0.101	0.099	0.100
large	0.475	0.471	0.480	0.457

7 Future Work

One significant direction for future work involves the integration of CPT with Autoware [40], a leading open-source software for autonomous vehicles. This integration aims to test and evaluate the end-to-end autonomous vehicle (AV) pipeline, providing a more comprehensive assessment of predictability in real-world scenarios [19, 20].

By incorporating CPT with Autoware, researchers can delve deeper into the complexities of the entire AV pipeline, extending beyond the perception module to include decision-making, path planning, and control systems. This comprehensive approach will allow for a holistic evaluation of the predictability of AV systems in diverse and dynamic driving environments. The integration will enable researchers to test various DNN model combinations and system configurations within the full context of an operational AV, thereby addressing some of the most pressing challenges in AV safety and reliability.

Furthermore, this future work will explore the scalability of CPT in more complex AV scenarios, including urban driving and adverse weather conditions, where predictability becomes even more critical. The findings from these studies are expected to significantly

contribute to the advancement of AV technology, ensuring safer and more reliable autonomous driving solutions.

8 Conclusions

This paper introduces CPT, a testbed designed to evaluate the predictability of AV's perception pipeline. CPT consists of three modules, namely application configurations, system configurations, and an evaluator. It offers two image datasets that include over 100 scenarios as sensor data, and it provides more than 19 million DNN model combinations through its integration with OpenMMLab and BDD100K's model zoo. CPT allows for flexible scheduling, fusion, and DVFS policies. Additionally, CPT defines various metrics representing time and performance variations for each task and fusion task.

References

References

[1] AMCL. http://wiki.ros.org/amcl.

[2] BDD100K Model Zoo. https://github.com/SysCV/bdd100k-models.

[3] dwa_local_planner. http://wiki.ros.org/dwa_local_planner.

[4] global_planner. http://wiki.ros.org/global_planner.

[5] message_filters. http://wiki.ros.org/message_filters.

[6] MMDetection. https://github.com/open-mmlab/mmdetection.

[7] sched(7) — Linux manual page. https://man7.org/linux/man-pages/man7/sched.7.html.

[8] The KITTI Vision Benchmark Suite. http://www.cvlibs.net/datasets/kitti/.

[9] Robot Operating System(ROS), Powering the World's Robots, 2019.

[10] Aporva Amarnath, Subhankar Pal, Hiwot Kassa, Augusto Vega, Alper Buyuktosunoglu, Hubertus Franke, John-David Wellman, Ronald Dreslinski, and Pradip Bose. Hetsched: Quality-of-mission aware scheduling for autonomous vehicle socs. *arXiv preprint arXiv:2203.13396*, 2022.

[11] Baidu. Apollo Open Platform. http://apollo.auto/index.html, 2021.

[12] M Bertoluzzo, Paolo Bolognesi, Ottorino Bruno, G Buja, Alberto Landi, and A Zuccollo. Drive-by-wire systems for ground vehicles. In *2004 IEEE International Symposium on Industrial Electronics*, volume 1, pages 711–716. IEEE, 2004.

[13] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[14] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[15] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[18] Mohammad Farsi, Karl Ratcliff, and Manuel Barbosa. An overview of controller area network. *Computing & Control Engineering Journal*, 10(3):113–120, 1999.

[19] The Autoware Foundation. Autoware AI. [Online]. https://github.com/Autoware-AI/autoware.ai.

[20] The Autoware Foundation. Autoware Auto. [Online]. https://gitlab.com/autowarefoundation/autoware.auto/AutowareAuto.

[21] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. DSSD: deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[22] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. 3D-LaneNet: end-to-end 3D multiple lane detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2921–2930, 2019.

[23] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[24] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[25] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[26] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. D3: a dynamic deadline-driven approach for building autonomous vehicles. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 453–471, 2022.

[27] Raghuraman Gopalan, Tsai Hong, Michael Shneier, and Rama Chellappa. A learning approach towards detection and tracking of lane markings. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1088–1098, 2012.

[28] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving DNNs like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462, 2020.

[29] Farrukh Hafeez, Usman Ullah Sheikh, Nasser Alkhaldi, Hassan Zuhair Al Garni, Zeeshan Ahmad Arfeen, and Saifulnizam A Khalid. Insights and strategies for an autonomous vehicle with a sensor fusion innovation: a fictional outlook. *IEEE Access*, 8:135162–135175, 2020.

[30] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[31] Junjun He, Zhongying Deng, and Yu Qiao. Dynamic multi-scale filters for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3562–3572, 2019.

[32] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive pyramid context network for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7519–7528, 2019.

[33] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[34] Yuenan Hou. Agnostic lane detection. *arXiv preprint arXiv:1905.03704*, 2019.

[35] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection CNNs by self attention distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1013–1021, 2019.

[36] Yen-Chang Hsu, Zheng Xu, Zsolt Kira, and Jiawei Huang. Learning to cluster for proposal-free instance segmentation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[37] Thomas E. Huang. BDD100K model zoo. https://github.com/SysCV/bdd100k-models.

[38] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF*

international conference on computer vision, pages 603–612, 2019.

[39] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[40] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[41] Yeongmin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection. *arXiv preprint arXiv:2002.06604*, 2020.

[42] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1947–1955, 2017.

[43] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017.

[44] Jun Li, Xue Mei, Danil Prokhorov, and Dacheng Tao. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE transactions on neural networks and learning systems*, 28(3):690–703, 2016.

[45] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pages 473–488. Springer, 2020.

[46] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9167–9176, 2019.

[47] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. 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*, pages 751–766, 2018.

[48] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[49] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[50] Liangkai Liu, Zheng Dong, Yanzhi Wang, and Weisong Shi. Prophet: Realizing a predictable real-time perception pipeline for autonomous vehicles. In *The IEEE Real-Time Systems Symposium (RTSS)*, 2022.

[51] Liangkai Liu, Sidi Lu, Ren Zhong, Baofu Wu, Yongtao Yao, Qingyang Zhang, and Weisong Shi. Computing systems for autonomous driving: State-of-the-art and challenges. *arXiv preprint arXiv:2009.14349*, 2020.

[52] Liangkai Liu, Yanzhi Wang, and Weisong Shi. Understanding Time Variations of DNN Inference in Autonomous Driving. *arXiv preprint arXiv:2209.05487*, 2022.

[53] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[54] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu,

and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[55] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[56] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[57] Huizi Mao, Xiaodong Yang, and William J Dally. A delay metric for video object detection: What average precision fails to tell. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 573–582, 2019.

[58] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[59] Raul Mur-Artal et al. ORB-SLAM2: an open-source SLAM system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[60] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[61] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[62] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 286–291. IEEE, 2018.

[63] NVIDIA. NVIDIA driveworks. [Online]. `https://developer.nvidia.com/drive/driveworks`.

[64] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[65] Ya-Jun Pan, Carlos Canudas-de Wit, and Olivier Sename. A new predictive approach for bilateral teleoperation with applications to drive-by-wire systems. *IEEE Transactions on Robotics*, 22(6):1146–1162, 2006.

[66] Jonah Philion. FastDraw: addressing the long tail of lane detection by adapting a sequential prediction network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11582–11591, 2019.

[67] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10213–10224, 2021.

[68] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[69] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[70] Joseph Redmon and Ali Farhadi. YOLOv3: an incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

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

[72] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[73] Marco Spuri and Giorgio C Buttazzo. Efficient aperiodic service under earliest deadline scheduling. In *RTSS*, pages 2–11, 1994.

[74] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021.

[75] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[76] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[77] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[78] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[79] Chengcheng Wan, Muhammad Santriaji, Eri Rogers, Henry Hoffmann, Michael Maire, and Shan Lu. ALERT: Accurate learning for energy and timeliness. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 353–369, 2020.

[80] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[81] Yifan Wang, Shaoshan Liu, Xiaopei Wu, and Weisong Shi. CAVBench: A benchmark suite for connected and autonomous vehicles. In *Proceedings of the Third IEEE/ACM Symposium on Edge Computing*, pages 1–13. IEEE, 2018.

[82] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. Machine learning at Facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344. IEEE, 2019.

[83] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020.

[84] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

[85] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, PP:1–1, 03 2020.

[86] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[87] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on*

computer vision and pattern recognition, pages 9759–9768, 2020.

[88] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[89] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European conference on computer vision (ECCV)*, pages 267–283, 2018.

[90] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. Resa: Recurrent feature-shift aggregator for lane detection. *arXiv preprint arXiv:2008.13719*, 5(7), 2020.