Design Automation for Intelligent Automotive Systems

Shuyue Lan*, Chao Huang*, Zhilu Wang*, Hengyi Liang*, Wenhao Su† and Qi Zhu*
* Northwestern University
† University of Michigan, Ann Arbor

***Abstract*—With rapid advancement of advanced driver assistance systems (ADAS) and autonomous driving functions, modern vehicles have become ever more intelligent than before. Sophisticated machine learning techniques have being developed for vehicle perception, planning and control. However, this also brings significant challenges to the design, implementation and validation of automotive systems, stemming from the fast-growing functional complexity, the adoption of advanced architectural components such as multicore CPUs and GPUs, the dynamic and uncertain physical environment, and the stringent requirements on various system metrics such as safety, security, reliability, performance, fault tolerance, extensibility, and cost. To address these challenges, new design methodologies, algorithms and tools are greatly needed. This paper will discuss the challenges in designing next-generation connected and autonomous vehicles, and the need of design automation techniques to tackle them.**

I. Introduction

The last decades have witnessed the rapid advancement of advanced driver assistance systems (ADAS) [1] and autonomous driving functions in vehicles. Active safety features such as lane keeping, collision avoidance, and road sign recognition have been successfully developed and deployed. A major driving force behind such advancement is the application of machine learning (ML) techniques in vehicle perception of the surrounding environment, and in vehicle planning and control based on the perception results.

However, these ML-based functions also bring significant challenges to the design, implementation and validation of automotive electronic systems. First, many of the current ML techniques, particularly those based on deep neural networks, do not provide sufficient robustness and predictability in their performance (e.g., consistent accuracy in object recognition under changing environment) for ensuring automotive safety. Moreover, the implementation of those ML techniques is often resource-consuming and computationally-expensive. This makes it difficult for the corresponding functions to meet the strict real-time constraints in vehicles (e.g., functions such as pedestrian detection has to be successfully carried out within a hard deadline to avoid fatal accidents), especially when considering the cost sensitivity in automotive industry and hence the always-limited resources in production vehicles.

To address above challenges in applying ML techniques to safety-critical and time-critical automotive systems, a new set of design methodologies, algorithms and tools is greatly needed, to provide an automated, systematic, and formalized process for automotive system design, implementation and validation. These design automation techniques should ensure the system robustness and reliability under dynamic and uncertain physical environment and operation condition, guarantee the system timing behavior with limited computational and communicational resources, and meet the stringent requirements on a variety of other metrics such as security, fault tolerance and extensibility.

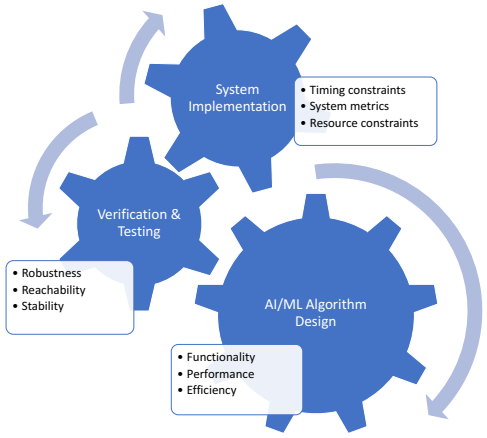

Fig. 1. Addressing ML functions in automotive systems.

In this paper, we will discuss the challenges in applying ML techniques to next-generation connected and autonomous vehicles, and examine the need of design automation techniques to tackle these challenges. As shown in Fig. 1, we believe that addressing ML functions in automotive systems involves multiple aspects across system layers. First, the design of ML algorithms should consider the system requirements on functionality, performance and efficiency. Then, verification and testing methods should be developed to validate the robustness, reachability and stability of the corresponding functions. Finally, automated synthesis methods are needed to ensure that the system implementation of ML functions meets the timing deadlines, hardware resource constraints, and requirements on other system metrics.

The rest of this paper is organized as follows. Section II reviews the application of ML techniques in modern automotive systems and briefly discuss some of the challenges, focusing more on the algorithm design aspect. Section III discusses verification and testing methods for ML-based automotive systems. Section IV addresses the system implementation of ML functions.

978-1-5386-8382-8/18/$31.00 ©2018 IEEE

II. ML Techniques in Automotive Systems

Thanks to the advancement of modern sensors (LiDAR, radar, cameras, etc.) and artificial intelligence technology, automotive systems have become ever more in intelligent than before. In developing vehicle intelligence, the automotive industry has been taking two different paths. In the more conservative path, traditional automotive companies start from aided driving and make their way gradually to high-level intelligence. In particular, ADAS systems have been well developed and adopted by most major automakers, such as BMW, GM, Honda and Toyota. In the more progressive path, IT companies and research institutes directly aim to fully autonomous driving, such as Waymo, Uber, and TuSimple.

Despite the differences in development paths, the major components in intelligent (autonomous or semi-autonomous) automotive systems are similar, as shown in Fig. 2. The in-vehicle systems have three core modules, perception, decision, and control, for collecting information of the surrounding environment via heterogeneous sensors, processing and analyzing the multi-modal information at real-time with advanced computing devices, automatically making planning and control decisions, and continuously actuating the corresponding mechanical components. Each module includes a number of components, some of which are shown in the figure. At inter-vehicle level, vehicle-to-everything (V2X) communication module is utilized to exchange information between vehicles and vehicles (V2V), vehicles and infrastructures (V2I), vehicles and pedestrians, etc. By enabling information sharing among vehicles and surrounding entities, V2X communication can further improve transportation safety and efficiency over single-vehicle intelligence. Furthermore, with V2X, cloud coordination can be adopted to acquire more computing resources for improving vehicle intelligence. In the following, we will discuss the application of ML techniques in each module.

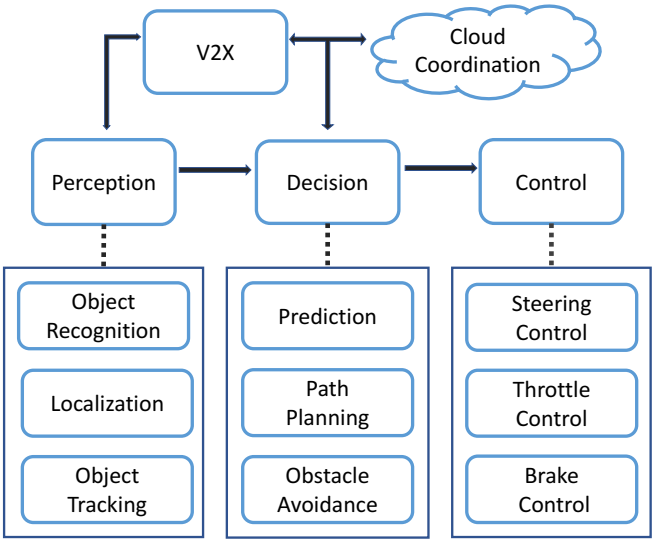

Fig. 2. Major components in intelligent automotive systems.

Perception: An intelligent automotive system typically employs a number of heterogeneous sensors for perceiving the environment, such as LiDARs, radars, GPS, inertial measurement unit (IMU), sonars, and cameras. The multi-modal inputs from these sensors are fused and analyzed to improve system reliability and safety [2]. In particular, ML techniques are widely used for processing the data collected by cameras, and these vision-based ML modules significantly contribute to the progress towards automated and cooperative driving [3], [4].

Some of the main perception tasks include object recognition, localization and tracking, which identify and localize the objects in the surrounding environment. In the early days, template-based methods were used to detect vehicles [5] and match objects [6]. Appearance-based methods were proposed later, which feature a three-step pipeline: 1) select proper features to represent data, 2) train a classifier with plentiful training data, and 3) input the extracted features to the classifier for identifying objects in new data.

Classical handcrafted features with classifiers have been widely applied in intelligent automotive systems. A number of approaches use general classifier, such as support vector machine (SVM), with color or histogram of oriented gradients (HOG) features to recognize traffic lights [7]–[12]. In [13], HOG and SVM are used for traffic sign recognition and enhanced by the fusion of camera and LiDAR data. In [14], an SVM-based brake detection method is developed for collision avoidance with the front vehicle. In [15], a vehicle detection algorithm is proposed to detect and track headlights during night time.

Moreover, perception methods have also been used for drivers and pedestrians. To reduce the distraction from interfaces, hand gesture recognition has been explored in vehicles [16]–[20]. Driver activity recognition approaches such as the one in [21] are proposed to monitor driver behavior and determine take-over readiness. Approaches for detecting pedestrians' intention have also been developed [22].

The advancement of deep neural networks, especially convolutional neural network (CNN), has further propelled intelligent automotive systems. For instance, instance-level segmentation for autonomous driving is studied in [23]. CNN is used to detect 3D objects in monocular images [24], with further improvement that fuses LiDAR point cloud with RGB images as inputs [25]. In [26], a unified, small, low-power fully-convolutional neural network called SqueezeDet is developed for real-time object detection in autonomous driving. In [27], recurrent neural network (RNN) is used for inferring the property of a dynamic object on the road.

Decision and Control: Based on the perception and understanding of the surrounding environment, an ML-based decision engine in intelligent automotive systems aims at generating a safe and efficient action plan at real time, conducting tasks such as prediction, path planning and obstacle avoidance.

There are a number of approaches that use ML techniques to decide a driving action directly based on the sensory input. Starting from the late 1980s, neural networks have been used to map the camera image input directly to the steering angles,

with the objective to keep the car on the correct lane [28]–[30]. Moreover, deep CNNs have been successfully applied for obstacle avoidance [31], long-range prediction of traversability [32], depth map estimation [33], estimation of affordance for driving [34], and steering control [35]. More recently in [36], an FCN-LSTM network architecture is adopted for learning a generic driving model and predicting a distribution over future egomotion from monocular camera observations and previous states. As a popular policy learning method, deep reinforcement learning (DRL) has also been applied in deciding autonomous driving policy [37]–[42].

ML techniques have also been used for other control functions in automotive systems such as engine management. In [43], a neural network is used to predict specific fuel consumption and exhaust temperature of a diesel engine for various injection timings.

V2X Communication: ML techniques have been used to address issues in V2X communication and connected vehicle applications in general. In [44], [45], the authors provide a detailed review of machine learning for vehicular networks. In [46], a hybrid centralized strategy using k-means clustering is introduced to control congestion in vehicular ad hoc networks (VANETs). In [47], LTE connectivity prediction and vehicular traffic prediction are addressed with Poisson dependency networks (PDN). A stacked autoencoder model is further used to learn traffic flow features for prediction [48]. Moreover, multiple reinforcement learning models are introduced for user association with load balancing [49], vertical handoff in heterogeneous networks [50], routing for local data storage in vehicular networks [51], and virtual resource allocation on vehicular clouds [52], [53].

Applications and Challenges: Besides the effort from academic community, huge investment and great progress have also been made in the industry for intelligent vehicles, with ML techniques widely applied to various aspects of ADAS and autonomous driving [54]. Companies such as Waymo, General Motors, Mercedes-Benz, Audi, Tesla and Uber have conducted extensive on-road testing of autonomous functions.

However, there are still significant challenges for fully utilizing machine learning and realizing autonomy in vehicles. In particular, the major challenges come from the safety-critical and time-critical nature of automotive systems. First, while ML techniques such as deep learning may provide superb performance in most cases, it is hard to reason about their worst case behavior and robustness in general. This is especially true considering that the surrounding environment is highly uncertain and rare scenarios may not be sufficiently covered in training and testing. Without more assurance of the robustness and stability of ML techniques, the functional safety of vehicles cannot be guaranteed. Second, complex ML algorithms are time-consuming and resource-intensive at runtime. Current autonomous vehicle prototypes have to employ expensive and power-hungry computing platforms that include high-end GPUs and CPUs, together with expensive sensors. The eventual commercialization of autonomous vehicles will require much more economical and efficient platforms, whose resource limitations will present significant challenges for ML algorithms to meet the stringent timing constraints.

In the following two sections, we will discuss how the safety-critical and time-critical aspects may be addressed for applying ML techniques in intelligent automotive systems.

III. Verification and Testing for ML-based Automotive Systems

For safety-critical intelligent automotive systems, it is essential to validate the correctness of both the design of its perception, decision and control algorithms *and* the software/hardware implementation of these algorithms. In this section, we focus on the correctness validation at the algorithmic (functional) level for system control and decision making.

In traditional automotive systems, model-based control (MBC) methods are prevalent. The control modules accept sampled system state as input and make decisions based on explicit models of the system dynamics, which are commonly captured with differential or difference equations via rigorous mathematical deduction. The validation of these MBC modules can then be described as validating reachability properties [55] or liveness properties [56], [57]. In the literature, a number of approaches have been developed for MBC verification [58]–[60] and testing/simulation [61]–[63].

In emerging intelligent vehicles, data-driven control methods that are based on ML techniques have shown great promises. These methods do not require building explicit physical models, but rather learn the control strategy directly from training data (or sensing data at runtime). Compared with MBC, such data-driven methods could be more effective (and easier to develop) for realizing autonomous functions that involve complex system dynamics and surrounding environment. In particular, methods that are based on deep neural network (DNN) have become popular. They are significantly different from MBC on the following aspects.

- "Black box" property: Even though the parameters in a trained DNN are visible, it is not clear how they relate to properties such as reachability and liveness. When conducting validation, DNNs are often viewed as black boxes.
- End-to-end control: As using multiple sequential neural networks may accumulate inaccuracy, DNNs have been used for end-to-end control, where the system accepts as input the sensor data (particularly camera images), rather than the abstracted system state.

These unique aspects bring more challenges to the validation of DNNs. For better discussion of the state-of-the-arts, we formulate the general validation problem as follows.

Problem 1: Given a neural network that implements a function $y = f(x)$ based on the training data set $\mathcal{X}'$, the associated complete input space $\mathcal{X}$ and a property P, we hope to have the following conclusion:

$$\begin{aligned} & \forall x' \in \mathcal{X}', \quad P(f(x')) = \texttt{true} \\ \Longrightarrow \quad & \forall x \in \mathcal{X}, \quad P(f(x)) = \texttt{true} \end{aligned}$$

Intuitively, we expect all the possible input should share some similarity with the training data on the property P.

In the following, we will discuss current approaches for DNN validation, including both verification and testing.

A. Verification

The verification of DNNs aims at checking the property P of a DNN f under all the possible inputs $\mathcal{X}$. However, we may find it intractable when we hope to use this definition to study DNNs, for the following two reasons:

- Ill-defined $\mathcal{X}$: In most cases, especially in DNNs that accept images or sound as input, it is difficult to define an ideal feasible input space $\mathcal{X}$. For instance, we hope to learn a DNN that can precisely recognize images of "human" to avoid accidents. However, it is almost impossible to define a set that contains all the "human" images.
- Vague P: Different from MBC, the necessary properties of a DNN are not quite clear yet. For instance, reachability is a general property in dynamic systems that essentially describes whether a state can be reached, however, it has no practical meaning for a DNN.

Due to the above reasons, we can find that current works on DNN verification mainly limit to one property, namely *local robustness*. This property receives much attention in the recent years due to a counterintuitive observation that even a very tiny perturbation of a correctly classified input can cause the mislabeling of a DNN [64]. We refer to this kind of mislabeling points as adversarial examples. Local robustness can be formally defined as follows:

Definition 1: A DNN f is said to be r-local robust at an input x_0 with the correct label c_0, if

$$\begin{aligned} &\forall x' \in \mathcal{X}' \triangleq \{x_0\},\ f(x')_{[c_0]} \geq \max_{c \neq c_0} f(x')_{[c_0]} \\ \Longrightarrow\quad &\forall x \in \mathcal{X} \triangleq B_r(x_0),\ f(x)_{[c_0]} \geq \max_{c \neq c_0} f(x)_{[c_0]} \end{aligned} \tag{1}$$

where $B_r(x)$ represents ball centered on x with radius r under certain metric $\|\cdot\|_p$ and $y_{[c]}$ represents the component of y with respect to the label c. We say

$$r_0 = \sup\{r \mid \forall x' \in B_r(x_0), f(x')_{[c_0]} \geq \max_{c \neq c_0} f(x')_{[c_0]}\}$$

is the minimum adversarial distortion at x_0 for f [65].

Local robustness describes that a DNN should be smooth in some degree and keep its judgment within a small neighborhood of an input. Therewith, we discuss two main methodologies that have been proposed to address this problem.

Precise Computing: Some approaches try to compute the exact r_0. Note that for networks with rectified linear unit (ReLU), the activation function ReLU is piece-wise linear and thus can be encoded as linear constraints by introducing slack binary variables based on Big-M convexification methods. Therefore, an intuitive solution is to transform all the ReLU functions by the above technique first and then compute the minimum adversarial examples r_0 by mixed integer linear programming (MILP) [66]–[69].

Alternatively, if we can verify whether there exist adversarial examples inside $B_r(x_0)$ for a given radius r, the bisection method can be used to find the exact r_0 [70]. Based on this idea, researchers have considered use different ways to treat the ReLU function and then verify it using satisfiability modulo theory (SMT) techniques. In [71], the authors linearly over-approximate the ReLU and other piece-wise functions by a triangle to reduce the searching space, and then apply SMT to efficiently solve the problem. In [72], the authors only split the ReLU function into two linear constraints on demand, which we will call it lazy splitting, to reduce the problem scale.

However, as pointed in [72], verifying the local robustness for a general ReLU DNN is NP-complete. The above precise approaches can hardly handle large networks.

Lower Bound Estimation: Another direction is to find the lower bound of the minimum adversarial distortion r_0. Currently works mainly fall into three categories: duality theory based approach, layer-by-layer technique, and function smoothness analysis.

Note that the exact r_0 can be obtained directly by solving a maximization problem that consists of a simple objective function (r) and complex constraints (the network description and the robustness property). Therefore a natural idea is to adopt the duality theory to get the lower bound of r_0. The dual minimization problem has simple constraints and is much easier to develop optimization techniques for higher efficiency. The authors in [73] linearly over-approximate the ReLU network using the similar idea of [71], and encode the overall problem as linear programming. Then, the dual problem of the linear programing is considered. Differently, the authors in [74] consider the Lagrangian dual problem, which does not limit to linear programming and thus can be applied on other DNNs, such as Sigmoid networks.

Layer-by-layer technique is one of the most popular methodologies to solve this problem. The basic idea is to analyze the impact on the propagation of input set $B_r(x_0)$ through layers of the network, and then check if r is a lower bound by observing the output of the final layer. The key is to find an appropriate way to approximate the output set of each layer, otherwise the problem degenerates to precise computation of r_0 and becomes time-consuming. The main difference among the existing works in this category lies on the approximation techniques. In [65], the authors adopt an individual hyper-rectangle to approximate the output set. In [75] and [76], the authors propose to use multiple hyper-rectangles to cover the output set of each layer, but with different construction methods. In [77], polyhedra are used for approximation. In [78], polyhedra are used with a different way of construction based on abstract interpretation technique. In [79], symbolic intervals are used to describe the polyhedra, which makes the over-approximation further tight. It is worthy noting that a tighter over-approximation leads to a better estimation of r_0 in general, but with greater computational load.

Function smoothness analysis is the most recent trend to study the robustness property, which we believe is a promising direction for understanding DNNs. As stated before, the local robustness property indicates the smoothness of DNNs. In [80], the authors discuss the local robustness under the precondition of local Lipschitz continuity and differentiability. This is extended by removing the differentiability requirement based on the extreme theory in [81]. However, how to explore the weakest precondition still remains an open question.

Unfortunately, a good estimation of r_0 (in the case of l_1-norm) of a general ReLU is proven to be unsolvable with polynomial time algorithms if $NP \neq P$ [65]. However, it is still possible to find a fast way for specific ReLU networks.

To conclude, few properties have been discussed for verifying DNNs – currently local robustness is the only property with clear practical meaning. To better understand and validate DNNs, more properties should be defined and analyzed.

B. Testing

In testing, the performance of DNNs are evaluated on specific test suites. In general, the DNN testing problem can be defined as:

Problem 2: Given a neural network that implements a function $y = f(x)$ based on the training data set $\mathcal{X}'$, the associated complete input space $\mathcal{X}$ and a property P. Further given a test suite $\mathcal{X}_t \subset \mathcal{X}$ with test cases $x_t \in \mathcal{X}_t$ and the corresponding expected outputs $y_t \in \mathcal{Y}_t$, the testing result ϵ can be expressed as:

$$\epsilon = \frac{|\{x_t | P(f(x_t)) = y_t\}|}{|\mathcal{X}_t|}$$

Whether ϵ could credibly reveal the performance of the tested DNN greatly depends on the inherent quality of $\mathcal{X}_t$. Koopman and Wagner have pointed out several potential challenges of testing DNNs in [82]:

- Credibility of $\mathcal{X}_t$: Due to the nature of autonomous system, a completely random test suite often leads to misleading performance evaluation. Generally, corner cases are more likely to expose unexpected behaviors. However, manually collecting corner cases could be hard, while automatic synthesis may bring the risk of overfitting.
- Determination of $\mathcal{Y}_t$: Creating correct labels for a test suite often requires huge amount of work. If defects in training data or training programs are found, more validation data has to be collected for revalidation, as the updated system could have a dramatically different set of learned rules.

Traditionally there are two ways to obtain $\mathcal{X}_t$: by manually collecting real-world scenarios and by automatically generating synthesized test suites. For DNN testing, several works have been done to obtain real-world test suites for *vision benchmarks*; while most current works focus on generating high-quality test suites that could reveal unexpected behaviors, based on different *coverage criteria*. The principles of test case generation and evaluation are greatly inspired by techniques used in traditional software validation. We will discuss these two types of efforts more in below.

Vision Benchmark Suites: Recently there has been an increasing availability of large annotated datasets for computer vision tasks relevant to autonomous driving. The *Middlebury* vision benchmark suite provides a dataset whose images are mainly collected from indoor scenes [83]. It provides evaluations for 5 different benchmarks such as stereo and optical flow algorithms. The *KITTI* vision benchmark suite focuses on real-world urban driving scenarios [84]. It provides evaluation for 9 different benchmarks, whose datasets are captured by driving around the mid-size city of Karlsruhe in rural areas and on highways. The *Robust Vision Challenge* is an algorithm testing website that combines 8 different test suites and provides evaluation for 6 benchmarks [85].

Automatic Test Generation based on Coverage Criteria: Manually collected test suites provide real-world emulation, however require high equipment and time cost for gleaning comprehensive data. With the help of automatic test generation based on various coverage criteria, we are more likely to reveal erroneous behaviors in DNNs.

1) Neuron coverage: Generally for a software implementation, if a test suite could have more of its source code executed during testing, it would have a lower chance of containing undetected bugs [86]. For DNN testing, Pei et al. raise the concept of neuron coverage in [87], and develop an algorithm based on this criteria called *DeepXplore*. More specifically, let $ReLU(x)$ denote the set of hidden neurons that are activated according to ReLU function , N_h denote the set of all hidden neurons, and $|N_h|$ denote the number of elements in set N_h. The neuron coverage for test suite $\mathcal{X}_t$ is:

$$Cov_{neuron}(\mathcal{X}_t) = \frac{|\bigcup\{ReLU(x_t)|x_t \in \mathcal{X}_t\}|}{|N_h|}. \quad (2)$$

The key idea of *DeepXplore* is to take unlabeled test input as seeds to generate new tests that maximize neuron coverage, which could be formulated as an optimization problem and solved by a gradient-based algorithm. However, there are two limitations in *DeepXplore*: 1) differential testing requires at least two different DNNs with the same functionality, and 2) whether the generated test cases could fit into real-world situations is not clear.

Several approaches have been proposed to overcome the above limitations. Tian et al. present the framework of *DeepTest* in [88] to eliminate the requirement of multiple DNNs. Inspired by the metamorphic testing in [89], *DeepTest* leverages metamorphic relations to support automatic labeling on a single DNN. Zhang et al. present the framework of *DeepRoad* in [90] to generate accurate driving scenes. *DeepRoad* employs a generative adversarial network (GAN) based technique described in [91] to provide authentic driving scenes with various weather conditions. It leverages UNIT [92], a DNN-based method to perform unsupervised image syntheses.

However, as pointed out in [93], a test suite with high neuron coverage is not sufficient to increase confidence for DNNs in safety-critical domains.

2) MC/DC coverage: Based on the traditional modified condition/decision coverage (MC/DC) criterion [94], Sun et

al. present a MC/DC coverage criteria for DNNs in [93]. The key idea of MC/DC coverage is that if a decision can be made, all the possible conditions contributing to that decision must be tested. In DNNs, the output of a single neuron can be seen as a decision, where contributing conditions are values of neurons in the previous layer. In [93], four MC/DC criteria are given to evaluate whether two test cases could cover the relationship between two single neurons or two layers of neurons. Take one criterion as an example: Let $n_{i,j}$ denote the i^{th} neuron in the j^{th} layer. If two test cases x_{t1} and x_{t2} cause a neuron $n_{1,2}$ to show different activation status but $n_{1,1}$ is the only neuron in layer 1 that shows different activation status under the two tests, then these two neurons are considered covered by the tests x_{t1} and x_{t2}.

Based on the MC/DC criteria, a test suite generating algorithm *DeepCover* is also proposed in [93]: For each neuron pairs in a DNN, recursively select a test case x_t in the given test suite and solve a linear programming problem to derive a new test case x'_t that maximizes MC/DC coverage and minimizes deviation value. If x_t and x'_t are "close" under certain metric and give different output values, then erroneous behavior is detected. This algorithm outputs a generated test suite $\mathcal{X}_t$ with maximum MC/DC coverage and a subset $\mathcal{X}'_t$ that contains test cases revealing erroneous behaviors.

3) CT coverage: Ma et al. present a set of coverage criteria for DNNs in [95] based on the concept of combinatorial testing (CT), together with a test generation algorithm *DeepCT*.

Assume there are $|N|$ neurons in a DNN. Each neuron can be activated or inactivated, resulting in altogether $2^{|N|}$ different neuron statuses. Instead of exhaustively testing all possible cases, CT aims at taking each pair of or each triple of neurons into consideration and test their combinations.

Three CT criteria with similar logic are given in [95] to evaluate coverage behavior of test cases on DNNs. Take one criterion as an example: Given a constant $t < |N|$ where N is the set of all neurons, we call the subset $\theta \in N$ a t-way combination if $|\theta| = t$. Let Θ_t be the set of all t-way combinations of neurons in N, and let $FCNA(T, \theta)$ to represent that all neuron-activation configurations of θ are covered by test suite $\mathcal{X}_t$. The t-way combination sparse coverage of $\mathcal{X}_t$ is given by

$$Cov_{comb-sparse}(\mathcal{X}_t, t) = \frac{|\{\theta \in \Theta_t | FCNA(\mathcal{X}_t, \theta)\}|}{|\Theta_t|}. \quad (3)$$

Based on a given distance metric, *DeepCT* recurrently generates test cases that are considered "close", until all CT coverage targets are covered or processed, or a time limit is hit. This algorithm outputs a generated test suite $\mathcal{X}_t$ with maximum CT coverage and a subset $\mathcal{X}'_t$ revealing erroneous behaviors.

Test Suite Evaluation: Besides above works on developing coverage criteria, Ma et al. explore a mutation testing technique for DNNs in [96], which focuses on developing a systematic way to evaluate and understand the quality of test suite. Specifically, given a program f, a set of faulty programs F' (mutants) are created based on predefined rules (mutation operators), each of which slightly modifies f. A given test suite $\mathcal{X}_t$ is executed on both f and F', and a mutant $f' \in F'$ is called killed if its test result is different from that of f. Define mutation score as the ratio of killed mutants to all generated mutants. A test suite $\mathcal{X}_t$ with a higher mutation score is more likely to capture real defects in the original program and provide better quality. Since both training suite and neuron structure can influence the performance of a DNN, two types of mutation testing, source-level (injecting error in the training data) and model-level (injecting error in the training program), are needed. Two DNN-specific mutation testing metrics are proposed to provide quantitative measurement of test quality.

While the above testing approaches have shown promising results under certain conditions, they are mainly empirically based. The discovery of a theoretically credible test suite generation method will be valuable and will need further development of the interpretation of DNN structures.

IV. System Implementation of ML Functions in Automotive Systems

Automotive systems are time-critical systems, in which many functions have strict timing constraints. For example, an obstacle avoidance function will need to sense the environment, process the input, and make a control decision within a hard deadline. The consequence of violating such end-to-end (sensor-to-actuator) latency deadline could be disastrous. Thus, validating the software/hardware implementation of ML functions, in particular the timing property of such implementation, is as important as validating those functions themselves.

However, the functional, software and hardware complexity has been rapidly increasing in intelligent automotive systems, especially with the development of ML techniques. This presents significant challenges for system implementations and for the validation of their timing behavior, as discussed below.

A. Increasing Functional and Software Complexity

The development of ML techniques has significantly increased functional complexity in automotive systems. The ML-based functions collect a large amount of data from various sensors and generate many intermediate results for perception, decision making and control. Typical neural networks may contain millions of parameters [97], [98] and are extremely computational intensive. Even the previously-mentioned SqueezeDet, a light-weight DNN, requires about 10^{10} FLOPs (float-point operations) for each image. When the sampling frequency is 20 FPS, the computation capacity needed for this single detection task will be at least 200 GFLOPS (giga-FLOPs per second).

At the software layer, the implementation of automotive functions is subject to standards such as ISO 15504 [99] and ISO 26262 [100]. ISO 15504 provides a reference model at a high abstraction level for software development management, while ISO 26262 standardizes function safety for the automotive development process. However, these standards usually lack specification and interpretation for neural networks [101]. Furthermore, while software structures such as dynamic objects, implicit conversion and recursions are not recommended

in these standards, the conventional development of neural network functions usually ignores these aspects.

Besides implementation standards, another issue is the certification of ML functions. As automotive systems are safety-critical and time-critical, a systematic way for source code inspection and worst-case scenario estimation is needed for deterministic system analysis.

B. Automotive Hardware Complexity

Fig. 3 shows an illustration of possible in-vehicle hardware architecture for future automotive systems, where computing resources such as ECUs and GPUs are connected via buses such as Ethernet, CAN, and FlexRay.

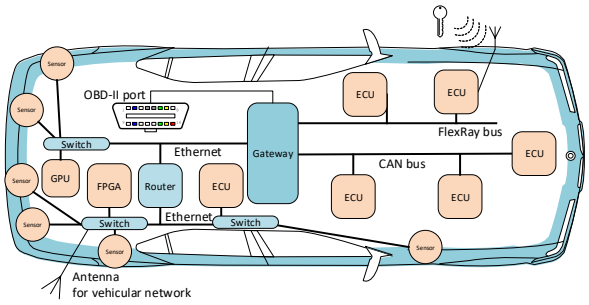

Fig. 3. Illustration of in-vehicle hardware architecture.

In the design of automotive system, there is a transition from the federated architecture, where each function is implemented on an individual ECU as a black box, to the integrated architecture, where multiple functions can be executed on the same ECU/GPU. Furthermore, with the development of computationally-intensive ML functions, the computing platform has to become more advanced and support more concurrent execution. For instance, the latest Nvidia platform for autonomous application, Jetson AGX Xavier [102], has an 8-core ARM 64-bit CPU, a 512-core GPU and two deep learning accelerators. This level of parallelism presents new challenges to timing behavior prediction. For instance, the blocking effect due to GPU synchronization is poorly documented [103], [104] and some implicit synchronization mechanisms may not be clear to software developers. This makes it challenging for the worst-case execution time analysis of ML functions on GPUs. The adoption of multi-core ECUs [105], [106] also leads to more resource sharing and contention among different functions, and increases the difficulty for timing prediction.

The in-vehicle communication for future automotive systems is also becoming more powerful and more complex. A video stream of 640×480 RGB images in 20 FPS requires a bandwidth of at least 60 MBps, which far exceeds the capacity of currently prevalent CAN bus protocol (typically allows up to 1Mbps). New high-speed automotive bus protocols, such as time-sensitive networking [107], [108] and time-triggered Ethernet [109] have been proposed, but they also present new challenges to system design and timing analysis. Furthermore, techniques could be developed to reduce the communication requirements, e.g., by transferring compressed data rather than raw sensing input or by skipping the processing of unimportant sensing input at the first place (similarly as the fast-forward strategy proposed in [110]).

C. Timing Validation

It is essential to ensure that the software/hardware implementation of automotive functions, in particular ML functions, meets the strict system timing constraints. This requires comprehensive timing analysis and validation across system layers. For instance, the worst-case execution time of a software task needs to be carefully analyzed. The worst-case response time of a task, which takes into account of the interferences from other tasks, needs to be derived and ensured to be within the task's deadline. Similar analysis also needs to be conducted for communication messages. Then, the end-to-end latency along a functional path (with a chain of tasks and messages) needs to be analyzed and checked against its deadline.

In the literature, a vast number of works have been proposed for timing analysis and validation of traditional automotive systems. The development of ML functions brings new challenges, some of which result from the growing complexity of functionality, software and hardware, as discussed above. In addition, the usage of ML functions also has significant impact on a variety of system metrics, such as security, fault tolerance, extensibility, and energy consumption. These metrics closely relate to system timing behavior, and their tradeoffs with schedulability and performance have to be addressed in a holistic fashion. For instance, we have demonstrated the significant tradeoffs among latency, fault tolerance, and extensibility in designing automotive software in [111]–[114], the tradeoffs among security, control performance, and schedulability in [115]–[118], and the tradeoffs among modularity, reusability, robustness, performance, and schedulability in [119]–[121]. The consideration of ML functions will require new models of these metrics that take into account of ML functionality (e.g., execution behavior of neural networks) and the underlying hardware platform (e.g., GPUs and new bus protocols), new analysis techniques, and new tradeoff/optimization algorithms. New timing models may also be needed to more accurately capture the timing behavior of ML functions.

V. Conclusion

In this paper, we reviewed the application of machine learning techniques in intelligent automotive systems, and discussed the resulting challenges to system design and validation. We highlighted those challenges that come from the safety-critical and time-critical nature of automotive systems, and elaborated the importance of developing new verification, testing, and timing analysis techniques to address these challenges. As intelligent automotive systems becoming reality, the development of such new design automation methods and tools has become a critical and pressing need for automotive industry.

ACKNOWLEDGMENT

This work is supported partially by the National Science Foundation grants CNS-1839511, CCF-1834701, CCF-1834324, and IIS-1724341.

REFERENCES

[1] A. Lindgren and F. Chen, "State of the art analysis: An overview of advanced driver assistance systems (adas) and possible human factors issues," *Human factors and economics aspects on safety*, pp. 38–50, 2006.

[2] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Caad: Computer architecture for autonomous driving," *arXiv preprint arXiv:1702.01894*, 2017.

[3] B. Ranft and C. Stiller, "The role of machine vision for intelligent vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 8–19, 2016.

[4] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[5] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous systems*, vol. 32, no. 1, pp. 1–16, 2000.

[6] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet, "A cooperative approach to vision-based vehicle detection," in *IEEE 4th International Conference on Intelligent Transportation Systems (ITSC)*, 2001, pp. 207–212.

[7] Z. Ozcelik, C. Tastimur, M. Karakose, and E. Akin, "A vision based traffic light detection and recognition approach for intelligent vehicles," in *IEEE International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 424–429.

[8] J. L. Binangkit and D. H. Widyantoro, "Increasing accuracy of traffic light color detection and recognition using machine learning," in *IEEE 10th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 2016, pp. 1–5.

[9] M. Salarian, A. Manavella, and R. Ansari, "A vision based system for traffic lights recognition," in *IEEE SAI Intelligent Systems Conference (IntelliSys)*, 2015, pp. 747–753.

[10] M. Michael and M. Schlipsing, "Extending traffic light recognition: Efficient classification of phase and pictogram," in *International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[11] Y. Ji, M. Yang, Z. Lu, and C. Wang, "Integrating visual selective attention model with hog features for traffic light detection and recognition," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 280–285.

[12] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 606–611.

[13] L. Zhou and Z. Deng, "Lidar and vision-based real-time traffic sign detection and recognition algorithm for intelligent vehicle," in *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 578–583.

[14] X. Wang, J. Tang, J. Niu, and X. Zhao, "Vision-based two-step brake detection method for vehicle collision avoidance," *Neurocomputing*, vol. 173, pp. 450–461, 2016.

[15] X.-Z. Chen, K.-K. Liao, Y.-L. Chen, C.-W. Yu, and C. Wang, "A vision-based nighttime surrounding vehicle detection system," in *IEEE 7th International Symposium on Next Generation Electronics (ISNE)*, 2018, pp. 1–3.

[16] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE transactions on intelligent transportation systems*, vol. 15, no. 6, pp. 2368–2377, 2014.

[17] A. Riener, A. Ferscha, F. Bachmair, P. Hagmüller, A. Lemme, D. Muttenthaler, D. Pühringer, H. Rogner, A. Tappe, and F. Weger, "Standardization of the in-car gesture interaction space," in *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. ACM, 2013, pp. 14–21.

[18] F. Althoff, R. Lindl, L. Walchshausl, and S. Hoch, "Robust multimodal hand-and head gesture recognition for controlling automotive infotainment systems," *VDI BERICHTE*, vol. 1919, p. 187, 2005.

[19] C. Endres, T. Schwartz, and C. A. Müller, "Geremin: 2d microgestures for drivers based on electric field sensing," in *Proceedings of the 16th international conference on Intelligent user interfaces*. ACM, 2011, pp. 327–330.

[20] M. Zobl, R. Nieschulz, M. Geiger, M. Lang, and G. Rigoll, "Gesture components for natural interaction with in-car devices," in *International Gesture Workshop*. Springer, 2003, pp. 448–459.

[21] C. Braunagel, E. Kasneci, W. Stolzmann, and W. Rosenstiel, "Driver-activity recognition in the context of conditionally autonomous driving," in *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, 2015, pp. 1652–1657.

[22] S. Köhler, M. Goldhammer, K. Zindler, K. Doll, and K. Dietmeyer, "Stereo-vision-based pedestrian's intention detection in a moving vehicle," in *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, 2015, pp. 2317–2322.

[23] Z. Zhang, S. Fidler, and R. Urtasun, "Instance-level segmentation for autonomous driving with deep densely connected mrfs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 669–677.

[24] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2147–2156.

[25] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 2, 2017, p. 3.

[26] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving." in *CVPR Workshops*, 2017, pp. 446–454.

[27] M. Fathollahi and R. Kasturi, "Autonomous driving challenge: To infer the property of a dynamic object based on its motion pattern," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 40–46.

[28] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[29] ——, *Neural network perception for mobile robot guidance*. Springer Science & Business Media, 2012, vol. 239.

[30] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, "Evolving large-scale neural networks for vision-based torcs," 2013.

[31] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[32] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[33] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 1385–1392.

[34] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2722–2730.

[35] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[36] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[38] G.-H. Liu, A. Siravuru, S. Prabhakar, M. Veloso, and G. Kantor, "Learning end-to-end multimodal sensor policies for autonomous navigation," *arXiv preprint arXiv:1705.10422*, 2017.

[39] C. Ye, H. Ma, X. Zhang, K. Zhang, and S. You, "Survival-oriented reinforcement learning model: An effcient and robust deep reinforcement

learning algorithm for autonomous driving problem," in *International Conference on Image and Graphics*. Springer, 2017, pp. 417–429.
[40] R. Liaw, S. Krishnan, A. Garg, D. Crankshaw, J. E. Gonzalez, and K. Goldberg, "Composing meta-policies for autonomous driving using hierarchical deep reinforcement learning," *arXiv preprint arXiv:1711.01503*, 2017.
[41] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *arXiv preprint arXiv:1704.03952*, 2017.
[42] A. Mehta, A. Subramanian, and A. Subramanian, "Learning end-to-end autonomous driving using guided auxiliary supervision," *arXiv preprint arXiv:1808.10393*, 2018.
[43] A. Parlak, Y. Islamoglu, H. Yasar, and A. Egrisogut, "Application of artificial neural network to predict specific fuel consumption and exhaust temperature for a diesel engine," *Applied Thermal Engineering*, vol. 26, no. 8-9, pp. 824–828, 2006.
[44] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, 2018.
[45] L. Liang, H. Ye, and G. Y. Li, "Towards intelligent vehicular networks: A machine learning framework," *arXiv preprint arXiv:1804.00338*, 2018.
[46] N. Taherkhani and S. Pierre, "Centralized and localized data congestion control strategy for vehicular ad hoc networks using a machine learning clustering algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3275–3285, 2016.
[47] C. Ide, F. Hadiji, L. Habel, A. Molina, T. Zaksek, M. Schreckenberg, K. Kersting, and C. Wietfeld, "Lte connectivity and vehicular traffic prediction based on machine learning approaches," in *IEEE 82nd Vehicular Technology Conference (VTC Fall)*, 2015, pp. 1–5.
[48] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang *et al.*, "Traffic flow prediction with big data: A deep learning approach." *IEEE Transactions Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
[49] Z. Li, C. Wang, and C.-J. Jiang, "User association for load balancing in vehicular networks: An online reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 2217–2228, 2017.
[50] Y. Xu, L. Li, B.-H. Soong, and C. Li, "Fuzzy q-learning based vertical handoff control for vehicular heterogeneous wireless network," in *IEEE International Conference on Communications (ICC)*, 2014, pp. 5653–5658.
[51] C. Wu, T. Yoshinaga, Y. Ji, T. Murase, and Y. Zhang, "A reinforcement learning-based data storage scheme for vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 6336–6348, 2017.
[52] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.
[53] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Reinforcement learning for resource provisioning in the vehicular cloud," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 128–135, 2016.
[54] W. Brenner and A. Herrmann, "An overview of technology, benefits and impact of automated and autonomous driving on the automotive industry," in *Digital Marketplaces Unleashed*. Springer, 2018, pp. 427–442.
[55] C. Le Guernic and A. Girard, "Reachability analysis of linear systems using support functions," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250–262, 2010.
[56] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Falsification of ltl safety properties in hybrid systems," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 4, pp. 305–320, 2013.
[57] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta, "Verifying ltl properties of hybrid systems with k-liveness," ser. Computer Aided Verification. Springer International Publishing, 2014, Conference Proceedings, pp. 424–440.
[58] C. Huang, X. Chen, W. Lin, Z. Yang, and X. Li, "Probabilistic safety verification of stochastic hybrid systems using barrier certificates," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–19, 2017.
[59] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," ser. Computer Aided Verification. Springer Berlin Heidelberg, 2011, Conference Proceedings, pp. 585–591.
[60] Z. Yang, C. Huang, X. Chen, W. Lin, and Z. Liu, "A linear programming relaxation based approach for generating barrier certificates of hybrid systems," ser. FM 2016: Formal Methods. Springer International Publishing, 2016, Conference Proceedings, pp. 721–738.
[61] C. G. Cassandras, M. I. Clune, and P. J. Mosterman, "Hybrid system simulation with simevents," *IFAC Proceedings Volumes*, vol. 39, no. 5, pp. 267–269, 2006.
[62] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," ser. Computer Aided Verification. Springer Berlin Heidelberg, 2010, Conference Proceedings, pp. 167–170.
[63] F. Bergero and E. Kofman, "Powerdevs: a tool for hybrid system modeling and real-time simulation," *SIMULATION*, vol. 87, no. 1-2, pp. 113–132, 2011.
[64] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644*, 2016.
[65] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel, "Towards fast computation of certified robustness for relu networks," *arXiv preprint arXiv:1804.09699*, 2018.
[66] C.-H. Cheng, G. Nührenberg, and H. Ruess, "Maximum resilience of artificial neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, Conference Proceedings, pp. 251–268.
[67] M. Fischetti and J. Jo, "Deep neural networks as 0-1 mixed integer linear programs: A feasibility study," *arXiv preprint arXiv:1712.06174*, 2017.
[68] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv preprint arXiv:1706.07351*, 2017.
[69] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.
[70] R. Bunel, I. Turkaslan, P. H. Torr, P. Kohli, and M. P. Kumar, "Piecewise linear neural network verification: a comparative study," *arXiv preprint arXiv:1711.00455*, 2017.
[71] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," ser. Automated Technology for Verification and Analysis. Springer International Publishing, 2017, Conference Proceedings, pp. 269–286.
[72] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*, 2017, Conference Proceedings, pp. 97–117.
[73] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.
[74] K. Dvijotham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks," *arXiv preprint arXiv:1803.06567*, 2018.
[75] S. W. X. Huang, M. Kwiatkowska and M. Wu., "Safety verification of deep neural networks," in *International Conference on Computer Aided Verification*, 2017, Conference Proceedings, pp. 3–29.
[76] W. Xiang and T. T. Johnson, "Reachability analysis and safety verification for neural network control systems," *arXiv preprint arXiv:1805.09944*, 2018.
[77] W. Xiang, H.-D. Tran, and T. T. Johnson, "Reachable set computation and safety verification for neural networks with relu activations," *arXiv preprint arXiv:1712.08163*, 2017.
[78] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai 2: Safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy (SP)*, 2018, Conference Proceedings.
[79] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," *arXiv preprint arXiv:1804.10829*, 2018.
[80] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems*, 2017, Conference Proceedings, pp. 2266–2276.
[81] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.

[82] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[83] D. Scharstein and R. Szeliski, "Middlebury stereo vision page," 2002.

[84] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3354–3361.

[85] (2018) Robust vision challenge. [Online]. Available: http://www.robustvision.net/index.php

[86] S. Devadas, A. Ghosh, and K. Keutzer, "An observability-based code coverage metric for functional simulation," in *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, 1997, pp. 418–425.

[87] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 1–18.

[88] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, pp. 303–314.

[89] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, Tech. Rep., 1998.

[90] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic autonomous driving system testing," *arXiv preprint arXiv:1802.02295*, 2018.

[91] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[92] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[93] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.

[94] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, and L. K. Rierson, "A practical tutorial on modified condition/decision coverage," 2001.

[95] L. Ma, F. Zhang, M. Xue, B. Li, Y. Liu, J. Zhao, and Y. Wang, "Combinatorial testing for deep learning systems," *arXiv preprint arXiv:1806.07723*, 2018.

[96] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao *et al.*, "Deepmutation: Mutation testing of deep learning systems," *arXiv preprint arXiv:1805.05206*, 2018.

[97] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural networks*, vol. 32, pp. 333–338, 2012.

[98] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3367–3375.

[99] K. El-Emam and I. Garro, "Iso/iec 15504," *International Organization for Standardization*, 1999.

[100] I. ISO, "26262: Road vehicles-functional safety," *International Standard ISO/FDIS*, vol. 26262, 2011.

[101] R. Salay and K. Czarnecki, "Using machine learning safely in automotive software: An assessment and adaption of software process requirements in iso 26262," *arXiv preprint arXiv:1808.01614*, 2018.

[102] NVIDIA Corporation, "Jetson agx xavier faq," https://developer.nvidia.com/embedded/faq#xavier-faq, 2018.

[103] N. Otterness, M. Yang, T. Amert, J. Bakita, J. H. Anderson, and F. D. Smith, "Implicit gpu synchronization: A barrier to real-time cuda workloads."

[104] Tesla, "NVIDIA. CUDA 9.0 RC C Programming Guide," http://docs.nvidia.com/cuda-rc/cuda-c-programming-guide/index.html, 2017.

[105] N. Navet, A. Monot, B. Bavoux, and F. Simonot-Lion, "Multi-source and multicore automotive ecus-os protection mechanisms and scheduling," in *IEEE International Symposium on Industrial Electronics (ISIE)*, 2010, pp. 3734–3741.

[106] S. Schliecker, J. Rox, M. Negrean, K. Richter, M. Jersak, and R. Ernst, "System level performance analysis for real-time automotive multicore and network architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 7, pp. 979–992, 2009.

[107] R. Queck, "Analysis of ethernet avb for automotive networks using network calculus," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, July 2012, pp. 61–67.

[108] J. Diemer, D. Thiele, and R. Ernst, "Formal worst-case timing analysis of ethernet topologies with strict-priority and avb switching," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, June 2012, pp. 1–10.

[109] P. Hank, T. Suermann, and S. Müller, "Automotive ethernet, a holistic approach for a next generation in-vehicle networking standard," in *Advanced Microsystems for Automotive Applications 2012*, G. Meyer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–89.

[110] S. Lan, R. Panda, Q. Zhu, and A. K. Roy-Chowdhury, "Ffnet: Video fast-forwarding via reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6771–6780.

[111] Q. Zhu, Y. Yang, M. D. Natale, E. Scholte, and A. Sangiovanni-Vincentelli, "Optimizing the Software Architecture for Extensibility in Hard Real-Time Distributed Systems," *the IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 621–636, 2010.

[112] Q. Zhu, Y. Yang, E. Scholte, M. D. Natale, and A. Sangiovanni-Vincentelli, "Optimizing Extensibility in Hard Real-Time Distributed Systems," in *RTAS '09: Proceedings of the 2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009, pp. 275–284.

[113] B. Zheng, Y. Gao, Q. Zhu, and S. Gupta, "Analysis and Optimization of Soft Error Tolerance Strategies for Real-Time Systems," in *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, October 2015, pp. 55–64.

[114] H. Liang, Z. Wang, B. Zheng, and Q. Zhu, "Addressing extensibility and fault tolerance in can-based automotive systems," in *2017 IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, Nov 2017.

[115] C. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli, "Security-Aware Mapping for CAN-Based Real-Time Distributed Automotive Systems," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, 2013, pp. 115–121.

[116] C. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-Aware Mapping for TDMA-Based Real-Time Distributed Systems," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*, Nov 2014, pp. 24–31.

[117] C. Lin, B. Zheng, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-Aware Design Methodology and Optimization for Automotive Systems," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 1, pp. 18:1–18:26, December 2015. [Online]. Available: http://doi.acm.org/10.1145/2803174

[118] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti, "Cross-Layer Codesign for Secure Cyber-Physical Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 699–711, May 2016.

[119] Q. Zhu, P. Deng, M. Di Natale, and H. Zeng, "Robust and Extensible Task Implementations of Synchronous Finite State Machines," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1319–1324.

[120] P. Deng, F. Cremona, Q. Zhu, M. D. Natale, and H. Zeng, "A Model-Based Synthesis Flow for Automotive CPS," in *Cyber-Physical Systems (ICCPS), 2015 ACM/IEEE International Conference on*, April 2015, pp. 198–207.

[121] P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, and M. D. Natale, "An Efficient Control-Driven Period Optimization Algorithm for Distributed Real-Time Systems," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3552–3566, December 2016.