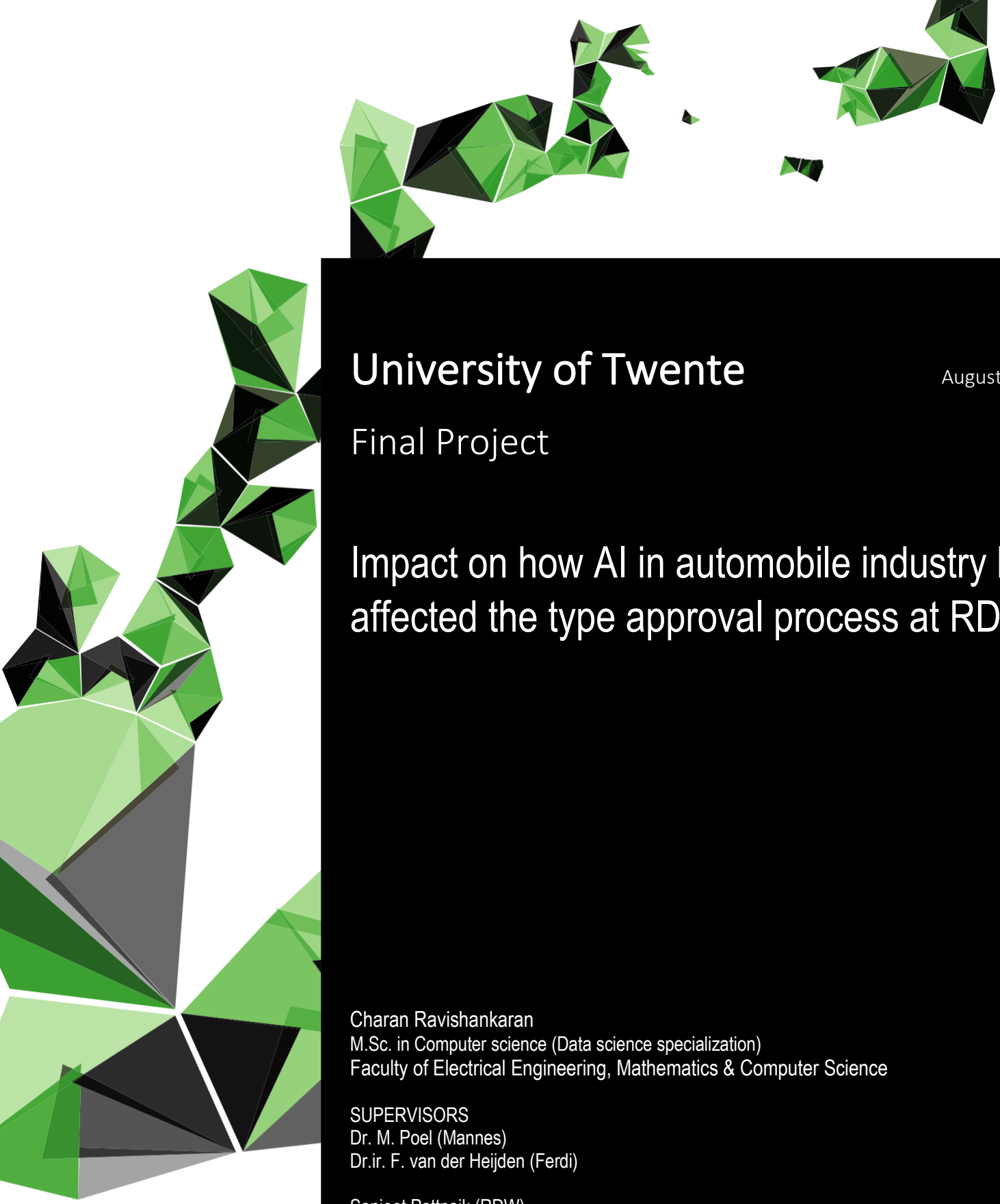


University of Twente

August 2021

Final Project

Impact on how AI in automobile industry has affected the type approval process at RDW.

Charan Ravishankaran
M.Sc. in Computer science (Data science specialization)
Faculty of Electrical Engineering, Mathematics & Computer Science

SUPERVISORS
Dr. M. Poel (Mannes)
Dr.ir. F. van der Heijden (Ferdi)

Sanjeet Pattnaik (RDW)
Shubham Koyal (RDW)

Table of Contents

ACKNOWLEDGEMENT

I would like to take this opportunity to express my heartfelt appreciation to those who assisted me and contributed in various ways and capacities. First and foremost, I'd like to thank my parents for their support throughout the duration of my thesis, despite COVID-19.

I would like to express my heartfelt gratitude to my professor, dr. M. Poel(Mannes), who has guided and supported me in achieving my research objectives. I would like to express my heartfelt gratitude to my RDW supervisors, Sanjeet Pattnaik and Shubham Koyal, for their unwavering support throughout my master's thesis. I was new to the automobile industry, and Sanjeet taught me how it works and how vehicles are approved by per region. I would also like to thank Shubham for his invaluable assistance during my time at RDW. He also taught me how to train and deploy a real-time deep learning model in autonomous vehicles. I would not have been able to complete my master's thesis without this knowledge.

Finally, I would like to thank my friends and family with whose moral support and motivation helped me achieve my research objectives.

Charan Ravishankaran

ABSTRACT

The automobile industry has increased its use of artificial intelligence (AI) over the last decade. One of the primary concerns regarding the use of AI in vehicles is ensuring "safety." Because AI can be subjected to incorrect predictions or make incorrect decisions, this can result in harm to the driver or passenger. Manufacturers test their production units prior to launch in order to avoid such harm or hazardous behaviours. However, in order to establish a manufacturing facility in a region (i.e., country), they must obtain approval from a government body. The government agency certifies that the manufacturing unit is safe. Due to the fact that AI is a type of software, it falls under the software category and must be validated prior to receiving government approval. Artificial intelligence software is based on machine learning, deep learning, and reinforcement learning algorithms. As the use of AI in vehicles increases, validation of the AI software and its capabilities becomes more challenging due to its non-deterministic (black box) behaviour.

The primary objective of this paper is to identify and address the current challenges associated with validating the AI software used in autonomous vehicles. Three factors affecting the validation of AI software in autonomous vehicles during the vehicle approval process were identified through an extensive literature review. The three factors are data-related issues, model-related issues, and security-related issues. This paper will focus on data-related issues, with experiments and recommendations. Security concerns are discussed but not prioritized because they are more concerned with cybersecurity principles than with AI. Model-oriented issues such as the explainability of AI, human-machine interaction, and faults in AI model networks have been discussed.

For data-related issues, the data used to train and test the AI model was evaluated. The impact of data issues was demonstrated through experiments such as labeling quality estimation (for the training set), quality dataset estimation (for the training and testing sets), and spatial uncertainty estimation. To address these issues, a framework and evaluation metrics were proposed. For autonomous vehicles, data will be collected via sensors installed on the vehicle, such as a camera, LiDAR, or RADAR, and used to make decisions. A case study revealed that camera sensors are widely used by the majority of vehicle manufacturers. As a result, all experiments were conducted using the ImageNet dataset [39], because the camera produces video output of the environment, which is then fed into the AI model as images/frames for decision-making. Finally, these experiments were evaluated using real-time deep learning models such as ResNet50 [39] and SSD-MobileNet [35]. From a data perspective, the proposed framework and evaluation metrics provided an adequate assessment of the AI model's robustness. To demonstrate which metrics are best suited for an autonomous vehicle scenario, the proposed evaluation metrics were compared to real-time metrics such as intersection over union (IoU) and mean Average Precision (mAP). Based on the results of the experiment, a recommendation was made to improve the type approval or safety assessment process at RDW.

Chapter 1 – INTRODUCTION

The automobile industry has grown vastly over the years, the evolution of autonomous vehicles paved the way for the future in the automobile industry. Autonomous vehicles run without any human intervention with the help of Artificial Intelligence. Artificial Intelligence is the key factor for autonomous vehicles, with the help of deep learning and machine learning algorithms they provide various features like Advanced Driving Assistance System (ADAS), cruise control, voice control, autonomous driving, lane changing, collision detection, obstacle monitoring and detection and so on. Although these advanced features provide highest automation to the user, achieving safety is one of the biggest concerns. As AI can produce false predictions, it can lead to harmful behaviours. Even though the manufacturers validate and test the vehicle units, these testing results could be in favour of the manufacturers. Hence a third party is required to make sure that safety of these vehicle units is assured. A third party can be a private organisation or a government body that does validation through audits and assessments. They guarantee that the manufactured product or the vehicle unit meets the required specifications based on the International Organization for Standardization (ISO) (safety (ISO 26262) and environmental).

The Netherlands Vehicle Authority also known as RDW is a leading type-approval authority in the Netherlands, and it is designated by the Dutch Ministry of Transport. This paper gives a brief knowledge about the vehicle approval process done in the Netherlands Vehicle Authority (RDW). The main problem faced by these organizations are the validation of AI present in the vehicle. As the usage of AI in the vehicles increases, validating them becomes hard. This is due to the AI's non-deterministic nature. In addition, there are no current ISO standards or procedure to validate the AI present in the vehicles.

This paper aims in creating a standard benchmark on the validation of AI present in the autonomous vehicles. In autonomous vehicles, most of the AI software are trained and developed using deep learning models [26]. Hence this paper focuses on the validation of deep learning-based software present in the autonomous vehicles (i.e., object detection, object classification). The validation of machine learning and deep learning models is achieved by evaluating the robustness of the model. In this paper, the robustness evaluation is done by estimating the quality of dataset [22, 27] (Training and testing), estimating the uncertainty percentage [28] from a deep learning model's output and providing different test environments (weather and road conditions, sensor vulnerabilities like solar glare and aging). These validations are done on a benchmarked ImageNet dataset [39]. Since this paper focuses on deep learning-based software, real-time deep learning models are used for the validations. ResNet50 [40] and SSD-MobileNet [35] are chosen, because they are benchmarked models on object classification and detection. They are also used in autonomous vehicles for self-driving, object detection, pedestrian detection, and lane detection [26]. From the results of the proposed experiments, recommendations are provided to handle the identified data-based issues.

Chapter 2 explains the research questions that helps in achieving the goal of this paper. Also, it provides the organization of this paper.

Chapter 2 - RESEARCH QUESTION (RQ)

The main goal of this paper as mentioned in chapter 1 is achieved through the below research question and its sub questions

RQ 1: How has the introduction of AI in the automotive industry impacted the type-approval process?

- What is type-approval in relationship with RDW?
- Identify the current state of the art approaches used in automotive software.
- Identify current issues in the validation of AI software in automated vehicles.
- Approach to handle the identified issues.
  - Analyze and identify the types of functionalities used in the AI frameworks.
  - Identify different test scenarios for the functionalities.
  - Test the identified scenarios.
  - Provide a comparative study on the test results.

RQ1 mainly focuses on an extensive background knowledge about the current automotive software approaches and current issues faced in validation of the automotive software. Initially a brief knowledge is given on how the vehicles are approved by RDW. From the background reading, knowledge like current frameworks and ISO standards used in creating traditional and AI based automotive software is gained. As a case study, the perception system of the autonomous vehicles are explained in detail. This study on perception system tells how the real-time environment data are collected through sensors and processed for decision making. Also this study provides knowledge about the current sensors used in the automotive market. Another reason for choosing this case study is that this paper deals with image-based data which in real-time is obtained through camera sensors. Another background reading is done to identify the current issues faced in validation of AI based software. Through this, three issues of AI are identified that impacts the safety assessment at RDW. They are data issues, AI model-oriented issues and security issues.

From the above knowledge gained, approaches are provided to handle the issues identified in validation of AI based software. This is done through step-by-step procedure. Initially functionalities of the AI software are identified. In this paper as mentioned in chapter 1, functionalities like object detection and object classification will be used. Test scenarios like estimating the quality of the dataset (training and testing data), estimation of the uncertainty of the model and different testing environments will be used. The identified test scenarios will be implemented through a proof concept (refer Chapter 4). Finally, the results obtained from the proof of concepts will be analyzed and discussed.

REPORT ORGANIZATION

The remaining portion of the report is organized as follows. In Chapter 3, the background reading consists of a detailed explanation of vehicle approval or type approval process at RDW, in section 3.1. Then, the current state of the art approaches used in automotive software is explained, in section 3.2 & 3.3. The case study for perception system is discussed in section 3.4. The issues faced in validation of AI software is explained, in section 3.5. The models and metrics used in this paper is explained in section 3.6 and 3.7. Chapter 4 explains the methodologies used in tackling the issues identified in validation of AI software. Chapter 5 explains the results and analysis observed from the methodologies. Finally, Chapter 6 concludes this paper with future works.

Chapter 3 – BACKGROUND STUDY

3.1 - Type approval process

Type approval is an official confirmation document provided by a government body that ensures that a manufactured product meets the required specifications (safety and environmental). If a manufacturer wants to sell a product in a particular country, then a type-approval is required. RDW is a leading vehicle-approval authority in The Netherlands, and it is designated by the Dutch Ministry of Transport. There are three actors involved in the type approval process. They are the approval authority, the technical services, and the manufacturer. The technical service sends their test report to the RDW assessment unit and if the report issued is according to European and ECE regulation then the appropriate certificate is sent to the applicant.

Type approval process

Initial Assessment

An initial assessment is done if a new manufacturer wants to launch his automobile unit in the Netherlands. This initial assessment is a process where the documents related to the automobile units will be submitted at RDW. RDW will review the documents to verify if all the information is sufficient if it covers all the subjects of the type approval and can assure the future Conformity of Production. An Initial assessment audit in addition to the document review will be done even if the manufacturer has a certified quality system. After the assessment, RDW will issue a compliance statement with a validity of one year. Before the end of this year, a factory inspection will occur to ensure that the implementation of the measures is aligned with the Conformity of Production (a certificate that ensures the manufacturer has produced the approved unit).

Technical service

Once the initial assessment is over, the product must be inspected for giving the type approval. There are two types of technical service one is the internal technical service of RDW and another one is the designated technical service. In general, technical service is a testing laboratory that carries out tests. It is also can be a conformity assessment body to carry out the initial assessment and other tests or inspections on behalf of the approval authority. The technical service uses the UNECE regulations for assessing a vehicle. There are 3 types of the type-approval process. **Component Type Approval** – approval of a component that may be fitted to any vehicle (e.g., seat belts, tires, lamps). **System Type Approval** – approval of a set of components or a performance feature of a vehicle that can only be tested and certified in an installed condition (e.g., restraint system, brake system, lighting system). **Whole vehicle type approval –** the vehicle is tested as a whole. The reports sent by the technical service are reviewed by the certification department at RDW. If the reports result aligns with the UNECE regulations, then the Conformity of Production (CoP) is given. During the factory inspection, if there is any violation of Conformity of Production, then RDW has the power to recall the Conformity of Production certificate and it can take the required actions to mitigate the issue.

Apart from these above-mentioned steps, the unit (vehicles or production unit like brakes, engine, etc) will be tested manually by an inspector on the RDW track and those reports will also be sent to the technical services. Due to the introduction of AI in the vehicles, the manual functionalities have been replaced with automated systems and functionalities, which reduces the visibility for the inspectors and technical services in assessing the unit.

During an audit the inspectors test manually each unit of a vehicle based on the UNECE (United Nations Economic Commission for Europe) regulations. For a braking system the regulations guide the inspectors to check its hardware quality. In addition, it also provides guidelines on how to evaluate its performance on the test tracks.

However, this is not the same with a software. A software present in a vehicle will be validated using the ISO 26262 and ISO 21448 (refer chapter 4). These ISOs have guidelines and framework in developing an automobile software. The audit inspectors during an audit, check whether the software present in the vehicles have adhered to ISO guidelines and framework. But for an AI based software the guidelines of ISO 26262 and ISO 21448 are not applied as these ISOs are not designed for the development, maintaining and validation of AI based software. Although ISO 21448 provides guidelines to validate certain functionalities that require the perception of the environment using sensors. These guidelines focus on validating sensor's properties and the sensor's vulnerability on the road environment.

Considering an AI software that detects pedestrian on the road. This AI software decides whether the vehicle has to stop or steer around the pedestrian. The decision is taken based on the data collected through the sensors like camera, LiDAR and RADAR (refer chapter 5). These data will be processed, and the decision will be taken based on trained deep learning model. Using ISO 21448 the sensors will be validated based on their properties like range, clarity etc. Vulnerabilities like weather conditions, aging effect etc. Apart from these validations there are no guidelines or regulations to validate the AI software. The inspectors have very less visibility in the development of AI software. Some manufacturers won't disclose this information as they are confidential. The inspectors have no idea on what type of data the AI software is trained on? What is the test data quality? That is used for testing and how the software takes decisions. In addition, they also question the decision of AI whether it can be trusted.

To conclude, as there are no current regulations and ISOs to validate an AI-based software the audit inspectors find it difficult to approve an AI based vehicle. However, the current audit procedures are done based on the current ISOs. To overcome this problem this paper will provide an approach in validating an AI-based software present in a vehicle.

3.2 – Automotive software

This chapter provides an extensive knowledge about the current state of the art approaches used in the automotive software. By approaches it means the current frameworks and standards used in developing, validating and maintaining automotive software. Also in this chapter the differences between a traditional automotive software and AI based automotive software is discussed.

The manufacturing of any automotive unit (hardware and software) is done based on the ISO standard 26262. During the type approval process, all the assessment and audits verify whether the manufacturer has manufactured the unit according to this ISO standard. Also, software used in automotive vehicles falls under ISO 26262. These automotive software serves as a platform for automated functionalities.

i. What is automotive software?

A vehicle consists of different features (ex. parking assistance) that is supported by different functionalities (ex. Advanced Driving System). For each functionality, there will be an individual or a group of dedicated Electronic Control Units (ECUs) that helps the vehicle in interacting with the real-world entities. These ECUs get data from the elements (i.e. Sensors) that are built in the vehicle and transfer the data via a communication protocol (CAN, LIN, Flex Ray, or Ethernet [9]) to all the underlying vehicle functionalities. In the automotive industry, there are huge number of vehicles been manufactured and each automobile company has its own features and functionalities. To make the ECUs independent of the functionalities an automotive software is required. The automotive software is a collection of data or instructions that runs on top of hardware (ECUs) and helps the software applications to interact with the hardware to provide enhanced safety, performance, and driving experience in a vehicle. In this paper, an automotive software framework AUTOSAR [9, 10] will be discussed. The automotive software applications are built based on the ISO standard 26262, which focuses on-road vehicles and functional safety. This ISO standard also describes the development of a hardware and software component with the help of the "V" model.

ii. different types of automotive software

Traditional automotive software - standards and approaches

a. ISO 26262 standard

With the trend of increasing hardware and software design, content and implementation, there are increasing risks from systematic failures and random hardware failures, these being considered within the scope of functional safety. ISO 26262 series of standards include guidance to mitigate these risks by providing appropriate requirements and processes. ISO 26262 is road vehicles and functional safety standard. If a manufacturer (OEM) intends to create or develop an automotive software application, they have to undergo the safety standards of ISO 26262. ISO 26262 is intended to be applied to safety-related systems that include one or more electrical and/or electronic (E/E) systems and that are installed in series production road vehicles. [11]. ISO 26262 provides requirements for functional safety management, design, implementation, verification, validation, confirmation measures, and requirements for relations between customers and suppliers. ISO 26262 has procedures to develop particular hardware or software component. This is done using the V model approach which is discussed below in section 4c.

b. AUTOSAR framework

AUTomotive **O**pen **S**ystem **Ar**chitecture, an open and standardized software architecture for automotive electronic control units (ECUs). [9], AUTOSAR is one of the leading automotive

software frameworks that is currently been used. According to [9, 15] BMW group, BOSCH, Continental, Daimler, Ford, GM, PSA Group, Toyota, and Volkswagen are the core partners of AUTOSAR. AUTOSAR builds a strong interaction with hardware. AUTOSAR has two types one is the classic AUTOSAR and another one is Adaptive AUTOSAR. Classic AUTOSAR architecture consists of four layers Application layer, Run time environment, the Software layer, and the microcontroller layer. The software layer consists of the service layer, ECU abstraction layer, microcontroller abstraction layer, and complex drivers. The application layer is the highest and it interacts with the software application. The run time environment provides communication services to the application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuator components). The main task of RTE is to make AUTOSAR Software Components independent from the mapping to a specific ECU. The basic software layer runs on top of the microcontroller. It gathers and processes the data from the microcontroller through sub layers present in it.

With the development of Adaptive AUTOSAR, there is a question that will Adaptive AUTOSAR replace classical AUTOSAR? [14]. the answer is no, Adaptive AUTOSAR can co-exist together with classical AUTOSAR. Figure 1 depicts the interactions between the classical and adaptive platform taken from AP AUTOSAR documentation [14]. The classical platform(C) consists of the ECUs and sensors that interact with the Adaptive platform (A) by providing data for the perception system (refer section 3.4). Backend services are provided to the adaptive platform from the backend system. The planning system in the adaptive platform gets the information from the perception system and finally sends the information to the control unit. The classical and adaptive platform can communicate with the help of internet protocols that are already incorporated in the classical platform, which is also supported by the adaptive platform. Ethernet is one of the major changes in the vehicle architecture's communication network. The Adaptive ECUs will communicate over the Ethernet whereas the classical ECUs will communicate via Bus networks like LIN and CAN.

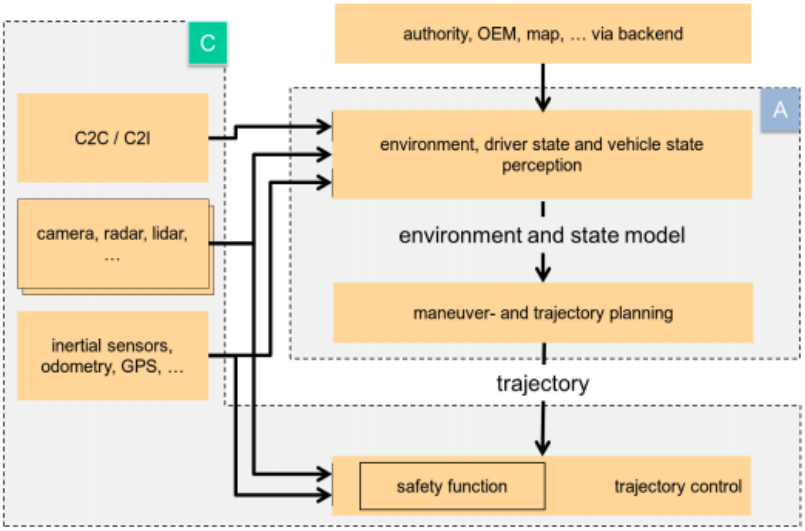


Figure 1: [14]. Interactions between Adaptive (A) and Classical (C) platform

iii. AI automotive software - standards and approaches

AI is a specialized software. The AI software will be trained on the particular use case (classification, regression, recognition, etc.). Based on the data it is trained on, it makes decision when a new data is given. In the automotive industry, AI is growing at an accelerated rate. Most of the automotive industry uses AI in their vehicles to increase the automation functionality. However, the current ISO 26262 which is used for creating and developing an automotive unit (hardware and software) is not designed to support AI software and its functionalities. However, to support such functionality there is another standard called ISO 21448 safety of intended functionality [19].

ISO 21448

ISO 21448 or Safety of the Intended Functionality (SOTIF) [19], is the standard that provides safety of the functionality which requires the perception of an environment (AI functionalities that use the environment data using sensors). Since these functionalities are not adapted to ISO 26262, the safety for these functionalities is provided in this ISO standard. Some hazards can be triggered by a specific condition, scenario, and misuses of an intended functionality for example activation of brake system while the automated driving function is active. ISO 21448 initially identifies whether a particular functionality can be harmful to the environment. Based on the analysis, ISO 21448 categories the functionality into four areas which are known scenario and hazard, unknown scenario and hazard, known scenario and not hazard, unknown scenario and not hazard. One of the main scopes of this ISO is to bring visibility to the unknown scenario and hazard category. Also, to reduce the known scenario and hazard category. Using this ISO 21448, the safety of using the AI functionalities can be provided. However, this ISO deals with safety on a functionality level and it does not provide safety from the software level.

Adaptive AUTOSAR framework is a widely used framework for the development of AI software's and its co-existence with classic AUTOSAR has been mentioned above.

3.3 Differences between traditional and AI automotive software

i. The difference in the development of the software.

As mentioned above, for traditional automotive software the "V model" in ISO 26262 is used. The steps in creating software and software are given below:

- **In item definition**, the description of the item with its functionality, interfaces, environmental conditions, legal requirements, and known hazards is given.
- The second phase is to identify the **Hazard analysis and risk assessment (HARA)**, this process estimates the probability of exposure of the item in the real-world, controllability and the severity of any hazardous events, and finally the ASIL (Automotive Safety Integrity Level) classification of the item.
- The third phase is **functional safety concept** based on the safety goals from the second phase, a functional safety concept is developed considering the preliminary architectural assumptions of the item (this also includes other technologies or external measures).
- The fourth phase is **product development at the system level** in the V model, the specification of the technical safety requirements, the system architecture, and the system design implementation on the left side of the 'V'. The Integration, verification, and safety validation is done on the right side of the 'V'.
- The fifth phase is **product development at the hardware level**, using system design specification, the hardware is developed.
- The sixth phase is **product development at the software level** from the specification of SOFTWARE safety requirements and architecture design, the Software unit design and implementation, Software integration and verification, and the testing of the embedded software are done.
- The final phase the **production, operation, service, and decommissioning**.

For AI software,

The general approach in development of AI software consists of Identify the use case, collect required data, process the data, choose a machine learning or deep learning algorithm, train the algorithm with the processed data, deploy the model (example cloud service) and finally use the model to make decisions. However, the development of an AI software differs based on its use case. For example, different use cases require different types of data (image data, textual data, and voice-based data). According to the author in [20], there are no current ISO standards for the development of AI software. Although certain ISO standards are being under development stage according to the International Organization for Standardization ISO/IEC JTC 1/SC 42. However, in the automotive industry, currently the manufacturers use ISO 26262's software development process for the development of AI software and its functionalities. The first three steps from the "V" model are used (from item definition to functional safety concept). When it comes to development of the AI software the general approach mentioned above is preferred according to the author in [20]. In addition, there is less visibility from the manufacturers on the development of AI software.

ii. Difference in the validation of the software

For traditional software

For validation of traditional software, ISO 26262: part 6 [11] clause is used. In that, the software development will be explained, and validation is a part of the development phase. One of the main goals for validation is to check whether the software has met the requirements given.

The steps in the validation are

- **Unit testing** – The individual units of the software or components are tested in this phase. The main purpose of this unit testing is to verify whether each unit works as expected. In ISO 26262, there are series of methods used for the unit verification which are Walk-through, Pair-programming, Inspection, Control flow analysis, Data flow analysis, Static code analysis, Requirements-based test, and Interface test. The software unit testing can be done in a different environment like Software in the loop, hardware in the loop, model in the loop, etc.
- **Integration testing and verification** – defines the integration steps and integrate the software elements until the embedded software is fully integrated. It also provides evidence that the integrated software units and software components fulfil their requirements according to the software architectural design. Also, it provides sufficient evidence that the integrated software contains neither undesired functionalities nor undesired properties regarding functional safety.
- **Testing the embedded software** – This testing provides evidence that the embedded software fulfils its requirements in the target environment. Hardware in a loop simulation, real-time vehicles are some target environments. Methods like fault injection, requirement based test are some test cases.

For AI software

For validation of AI software functionalities ISO 21448 methods [19] consist series of steps to verify and validate an AI functionality. The system verification and validation activities regarding the risk of potentially hazardous behavior (excluding the faults addressed by ISO 26262) include integration testing activities to address the following scope

- The capability of sensors to provide accurate information on the environment.
- The ability of the sensor processing algorithms to accurately model the environment.
- The ability of the decision algorithms to make appropriate decisions according to the environment model and the system architecture.
- The robustness of the system or function.
- The effectiveness of the fall-back handover scenario.
- The ability of the Human Machine Interaction to prevent reasonably foreseeable misuse.

The validation is divided into two categories, evaluation of known hazard and evaluation of unknown hazard. The main difference in the evaluation of known and unknown hazards is the test cases will be randomized for unknown hazards.

The below are some methods that are followed to validate a sensor.

- Verification of standalone sensor characteristics (e.g. range, precision, resolution, timing constraints, bandwidth, signal-to-noise ratio, signal-to-interference ratio)
- Injection of inputs that trigger the potentially hazardous behaviour. Example input images that depicts a solar glare on the lens of a camera which could result in misclassification or wrong detection.
- In the loop testing (e.g. software in the loop (SIL), hardware in the loop (HIL), model in the loop (MIL)) on selected SOTIF relevant use cases and scenarios considering identified triggering conditions.
- Sensor test under different environmental conditions (e.g. cold, damp, light, visibility conditions, interference conditions)
-

The below are some methods that are followed to validate a decision-making algorithm

- Verification of robustness against input data being subject to interference from other sources, e.g. white noise, audio frequencies, signal-to-noise ratio degradation (e.g. by noise injection testing).
- Requirement-based test (e.g. classification, sensor data fusion, situation analysis, function, the variability of sensor data)
- Vehicle testing on selected SOTIF relevant use cases and scenarios considering identified triggering conditions.

Also under ISO 21448 [19] Annex C, different methods to test a perception system are mentioned which are Sensor Manufacturing Verification, Algorithm Performance Verification, Vehicle Integration Verification, Test Track Verification, Open Road Validation.

However, the above methods validates the output of the AI software's functionalities through different test cases. But there are no standard methods to validate the AI software itself. For example, estimate the quality of the dataset used for training the model, estimating the quality of the dataset used for testing the model, explaining the uncertainty produced by a deep learning model and even identifying how the deep learning model arrives to the decision.

As mentioned before, there are no current standards for validating an AI software but there are ongoing researches and proposed testing frameworks, according to [20,21,22,23,24].

3.4 – Case Study

The purpose of this case study is to provide a background knowledge about the perception system in an autonomous vehicle. Perception system is where the raw data collected from the environment through sensors are processed. The perception system processes the raw data in two ways which is through sensor fusion and localization.

This chapter initially discusses about different types of sensors and its specifications currently in the automotive market. These sensor specifications help the auditing inspectors during the vehicle approval process as the sensors will also be assessed individually. In this chapter camera, LiDAR and ultrasonic sensor specifications are discussed as they are used in most of the automobile industry. Sensor fusion and localization are explained later on in this chapter.

Sensors

The automobile industry is one of the main users of the sensors. With the help of sensor data, an automotive vehicle provides assistance to the user with various automation features (for example lane detection system). Autonomous vehicles (AV) function based on 4 levels which are sensors, perception, planning, and control according to [2, 3]. As shown in Figure 2, the vehicle is sensing the world using many different sensors mounted on the vehicle. These are hardware components that gather data about the environment. The information from the sensors is processed in a perception block whose components combine sensor data into meaningful information. The planning subsystem uses the output from the perception block for decision making. The control module ensures that the vehicle follows the decision provided by the planning subsystem and sends control commands to the vehicle.

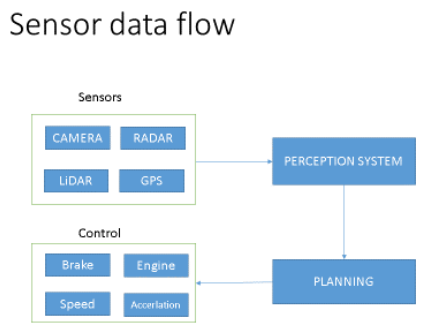


Figure 2: sensor data flow

In this paper, the sensors and perception system will be discussed. Sensor data are used in an AV for detection, classification, measurements, and robust to adverse conditions. According to saFAD [3], there are two types of sensors, they are environmental and apriori sensors. Environmental sensors consist of cameras, RADAR, LiDAR, ultrasonic, and microphones. Apriori sensors consist of High Definition Map and GNSS (Global Navigation Satellite System).

i. Types of sensors and current specifications

The camera enables an autonomous vehicle to visualize its surroundings. Cameras are the first types of sensors used in driverless vehicles. Cameras can also be used for human-machine interaction inside the vehicle. Current high-definition cameras can produce millions of pixels per frame, with 30 to 60

frames per second, to develop intricate imaging which leads to multi-megabytes of data needed to be processed in real-time. There is a huge benefit in using cameras for increasing autonomous vehicle's perception system as it allows the vehicle to identify road signs, traffic lights, lane markings, etc. Cameras are sensitive to low-intensity light and may also be heavily affected by weather conditions. There are stereo cameras, eagle-eyed vision, Time of Flight (ToF), infrared cameras. According to the automotive camera market research [4], cameras can be grouped by application (park assist, advanced driver-assistance systems), by view type (single view, multi-view), by technology (Infrared, thermal and digital cameras), by vehicle type (passenger type, commercial type), by autonomy level (SAE levels 0 to 5) and by region (North America, Asia Pacific, Europe). According to the market research [4], Aptiv PLC, Clarion, Continental AG, Denso Corporation, Magna International Inc., Mobileye, OmniVision Technologies, Robert Bosch GmbH, Samsung Electro-Mechanics, Hitachi Automotive Systems Ltd., Stonkam Co. Ltd, Valeo, Veoneer, ZF Friedrichshafen are some camera manufacturers globally. In this paper, Tesla's state of the art camera specifications is discussed.

- Tesla's Model S [5], uses eight surround cameras to provide 360 degrees of visibility around the car at up to 250 meters of range. There are three cameras mounted (wide, narrow, and forward). Wide cameras are 120-degree fisheye lens that captures traffic lights, obstacles cutting into the path of travel and objects at close range. Particularly useful in urban, low-speed maneuvering. Forward-Looking Side Cameras are 90 degrees they look for cars unexpectedly entering your lane on the highway and provide additional safety when entering intersections with limited visibility. Rearward cameras monitor the blind spots.

LiDAR Light Detection and Ranging use an infra-red laser beam to determine the distance between the sensor and a nearby object. Currently, LiDARs use light in the 900 nm wavelength range, although some LiDARs use longer wavelengths, which perform better in rain and fog. The lasers are pulsed, and the pulses are reflected by objects, these reflected pulses return a point cloud that represents the objects. LiDARs are more affected by weather conditions and by dirt on the sensor. LiDARs can map a static environment as well as to detect and identify moving vehicles, pedestrians, and wildlife. It works according to the time-of-flight (TOF) principle, emitting a pulsed light laser and measuring the time required for the pulse to reflect. They can produce a high-resolution densely spaced network of elevation points called **point clouds**. These point cloud data are essential for accurate positioning. There are two types of sensors solid-state LiDAR and infrared LiDAR. Currently, apart from Tesla, Inc, most of the automobile companies use LiDAR and their global suppliers are Continental AG, Robert Bosch GmbH, First Sensor AG, Denso Corp, Hella KGaA Hueck & Co., Novariant, Inc, Laddartech, Quanergy Systems, Inc., Phantom Intelligence and Velodyne LiDAR, Inc.

In this paper, the current LiDARs manufactured by Velodyne LiDAR, Inc and Ouster will be discussed.

Name	Type	hFoV	vFoV	Range	Others	Advantages
Velodyne - Alpha Prime [6]	Surround sensors	360°	40°	220m	High resolution (0.2° x 0.1°), Class 1 eye-safe 903 nm technology, points per second 2.4M	High-quality perception, advanced sensor-to-sensor interference mitigation, Superior Low Reflectance Object Detection
Velodyne - Ultra-Puck [6]	Surround sensor	360°	40°	200m	Top vertical resolution	Advanced features for minimizing false positives, Firing exclusion, and

					(0.33°), points per second 580K	interference mitigation features
Ouster-OS2 – 128 [7]	Solid-state	360°	22.5° (±11.25°)	240m	Points per second – 2.6M	High resolution, efficient data processing, faster labeling, and streamlined algorithm application

Table 1: Lidar specification from Velodyne and Ouster

Ultrasonic is a device that uses sound waves to measure the distance to an object. A sound wave is emitted towards an object at a specific frequency and the time it takes for the wave to return is utilized to calculate the distance. They are robust in weather conditions and according to the author in [2], It has been used by most car manufacturers as a reliable record of parking sensors for many years.. One of the main disadvantages is the sound waves can be disturbed by the environment, temperature, and humidity. To accommodate this, most sensors use an algorithm to adjust readings depending on the current temperature.

Parameters	**Specifications**
Min Range	15 cm (Ø 7.5 cm)
Max range	5.5 m (Ø 7.5 cm)
hFOV	± 70° @ 35 dB
vFOV	± 35° @ 35 dB
Safety level	ASIL - B

Table 2. Bosch ultrasonic sensor specifications

RADAR Radio Detection and Ranging, RADAR is used for adaptive cruise control, blind-spot warning, collision detection, and avoidance. RADAR uses Doppler Effect to measure speed whereas other sensors measure velocity by calculating the difference between two readings. When in a situation like bad weather, RADAR generates fewer data. Considering the computational requirement, RADAR has lower processing speeds needed for handling data output compared to LIDAR and cameras. RADAR can be used for localization by generating radar maps of the environment, can see underneath other vehicles, and spot buildings and objects that would be obscured otherwise. RADAR is least affected by rain or fog and can have a wide field of view, about 150 degrees, or a long-range, over 200 meters. In the automotive radar market,

ii. Perception System

Once the data from the sensors are collected, the perception system process the data into meaningful information like details of the environment or the vehicle's position (Localization). In the perception system, Sensor fusion and localization are the main methods.

a. SENSOR FUSION

In Sensor fusion, the data from the sensors are fused and it supports the AI functionalities (like object detection and object classification). An Autonomous vehicle (AV) cannot simply rely on a single sensor data [8], if an Autonomous vehicle relies on camera data then it can only visualize the surrounding but it will fail to identify other parameters like the distance between the obstacle and the current speed of the vehicle. But when sensor data are fused say camera and LiDAR data are fused, the AV will now visualize the obstacle, with help of LiDAR data it will identify the distance between the vehicle and the obstacle.

Sensor fusion is done at different levels [8], **an early fusion** which means the sensor fusion is done at the raw data level. **Halfway Fusion,** in this stage the raw data is processed, features are extracted and

these features are fused. **In late fusion,** in this stage the raw data is processed, features are extracted, classifiers are used to make decisions and these decisions are fused.

According to the review done in [8], there are five different levels when comes to data processing for perception and decision applications. For level 1, the raw input data collected from the various sensors is taken and in level 2, the process of filtering, spatial and temporal alignments, and uncertainty modeling is done. Level 3, the output from level 2 is considered and feature extraction, object detection, clustering, data processing occur to generate representations of objects is done. Level 4, the object is identified from the inputs and finally, the decision is made in level 5 (for example whether a vehicle should stop or steer left/ right).

There are several categorization schemes of sensor fusion methods that exist according to a literature study that was done by the authors in [8]. In table 3, fused sensor data for different AV applications has been discussed based on [8].

AV application	**Fused sensors**	**Advantages**
Pedestrian Detection	Camera and LiDAR, Vision and infrared	Ability to measure depth and range, with less computational power; Improvements in extreme weather conditions (fog and rain)
Road detection	Camera and LiDAR, Vision and Polarization camera	Road scene geometry measurements (depth) while maintaining rich color information; Calibration of scattered LiDAR point cloud with the image
Vehicle Detection Lane Detection	Camera and Radar	Measure distance accurately; Performs well in bad weather conditions; Camera is well suited for lane detection applications
SLAM(simultaneous localization and mapping)	Camera and Inertial Measurement Unit	Improved accuracy with less computational load; Robustness against vision noise, and corrective for IMU drifts
Navigation	GPS and INS (inertial navigation system)	Continuous navigation; Correction in INS readings
Vehicle Positioning	Map, Camera, GPS, INS	Accurate lateral positioning through road marking detection and HD map matching.

Table 3: Fused sensor data and AV applications

Sensor fusion is done with different approaches and according to review in [8], it has been categorized into traditional approaches and deep learning approaches (which is discussed later in this paper). From [2, 3, 8], these are some traditional and deep learning sensor fusion approaches.

1. **Traditional approaches in sensor fusion**
    - **Statistical and Probabilistic method**– it uses a statistical and probability-based approach to model the sensory information. Some algorithms are cross-covariance, covariance intersection.
    - **Knowledge-based theory methods** – uses computational intelligence approaches for classification/ decision. Some algorithms are fuzzy logic, genetic algorithms, ant colony

2. **Deep learning approaches in sensor fusion**

The core of deep learning is based on ANN. Deep learning is a subset of Artificial intelligence and a part of Machine learning algorithms. Deep learning mimics the functionality of the human brain that helps in performing complex tasks and take an effective decision.

- **Convolution neural network** – It is a feedforward network with convolution layers and pooling layers which helps in finding the relationship between image pixels. It is widely used in computer vision, speech recognition. There are different types of CNN which are YOLO, R-CNN, Fast R- CNN, Faster R-CNN, SPP-Net, etc.
- **Recurrent neural network** – It uses previous output samples to predict the new output samples. It is used for sequential data. It is widely used in forecasting and natural language processing. Long Short-Term Memory (**LSTM**) and Gated Recurrent Units (GRU)
- **Autoencoders** – used for unsupervised data. Dimensionality Reduction, image retrieval, data de-noising.

b. Localization

Since localization is not in the scope of this paper a small introduction and sensors used for localization are mentioned. The key concept of localization is to identify the location and orientation of the autonomous vehicle (AV). There are three types of localization, global localization, relative location, and simultaneous localization and mapping (SLAM). Sensors used in localization are GNSS (Global Navigation Satellite System), inertial measurement unit (IMU), HD Maps.

Through this case study, the different sensors and their current specifications used in autonomous vehicles are discussed. This helps the audit inspectors when assessing a vehicle unit where individual components (i.e., sensors) will also be assessed. The perception system provides knowledge on how different sensor data's are collected and combined together to form a meaningful information about the environment.

3.5 - Current issues in the validation of AI software in automated vehicles

Artificial intelligence plays a huge role in advancing automotive applications. Due to the increased automation, validating and verification of the software and its functionalities became a concern. In this section, the drawbacks of validating the AI based software used in autonomous vehicles will be discussed. As mentioned in the previous chapters, achieving safety over AI based software is a big concern. According to Salay et al [16], safety is a critical objective. All the automotive software applications are built based on ISO 26262 standards. However, ISO 26262 was not aimed to adapt for artificial intelligence methods. Although AI applications come under the software category, the development of an AI application differs from a traditional software (refer section 3.3). Since the main focus of this chapter is identifying issues in validation of an AI software there are certain characteristics of AI that impacts the safety or safety assessment according to the author in [16].

i. Characteristics of AI that impacts the safety or safety assessment

a. Non-Transparency

Transparency of a software (traditional and AI) application is a requirement for safety assessment. When validating software, it should be a white box (i.e.) all the internal structures and working of each function should be visible. According to Salay et al [16] in machine learning, Bayesian networks show transparency and in contrast, the neural networks are considered non-transparent because of the internal working of the neurons and hidden layers that makes the decision (classification, detection, and tracking) is not transparent.

b. Error rate

The machine learning models do not exhibit correct results all the time, they show some errors. According to Salay et al [16], the estimate of the true error rate is an outcome of the machine learning process. Although this estimation is done statistically, and it can vary on different data.

c. Training based

Machine learning models trained based on supervised, unsupervised, and reinforcement learning approaches. During training, the model can be subjected to overfitting and underfitting. Sometimes the model will be trained over and over for only certain data patterns so if new data comes the model won't perform well.

ii. Problems with AI software

From the safety perspective, problems with the AI software have been categorized into three types, model-based issues, data-based issues, and security-based issues.

a. Model based

1. Behavioural changes

A type of behavioural change hazard is that the driver assuming that the ADAS is smarter than himself/herself which may result in less awareness over the environment. An example, if a vehicle has an automated steering function, there is a high probability that the driver stops monitoring the steering as it has been monitored by the automated function. Although this

can be seen as the driver's error and these types of errors are identified by the ISO 26262 in part 3 [11]. But in according to Salay et al [16], due to the increased automation in vehicles, it creates a behavioral changes to the drivers by reducing their skills and ability to respond to a situation when required. These behavioral changes can impact safety even though when there is no system malfunction.

2. False predictions

As deep learning algorithms are used widely in automotive applications for object detection, object tracking, blind-spot warning, and so on. Due to certain reasons, they can produce false predictions, like the environment data can be corrupted with weather condition (fog, glare, rain) and sensors vulnerability (sensor aging effect). Another common reason where a deep learning model trained in a particular region (US, China) could not perform well or produce false prediction in another region (Europe).

3. Explainability

Due to the AI model's black box nature, visibility during the safety assessment process is reduced. If the audit inspectors are unfamiliar with the AI model used in the vehicle, assessing them will be more difficult. With the aid of Explainable AI, or XAI, a model can explain how and why it makes a decision for a particular use case.

4. Fault in model network

In AI applications, faults in the network topology and learning methods can lead to poor decision making. Faults in the neural network structure like a connection between the neurons and hidden layers, too many dropout layers (step used to overfitting), and so on can lead to faults in the AI application. Other factors in AI that leads to faults are inadequate training set, lack of coverage in rare case scenarios

b. Data based issues

1. Test Oracle

According to Marijan et al [20] machine learning systems are subjected to change their behaviour as they learn over time. Due to this, the generation of test cases for the machine learning model becomes complicated. The term "Test oracle" [20], is a specification that contains test cases for the software, which includes specified inputs and expected results. Due to the black box nature of machine learning models, the parameters (test input, expected output) for tests vary depending on the machine learning model used. This reduces the visibility for the audit inspectors during the safety assessment.

2. Test data

In general, a machine learning and deep learning model’s accuracy can be determined using a test data set (unseen data by the model). In autonomous vehicles, the test data represents the real-world environment (road type, weather, road signs, lane, pedestrian, and other objects). These data are fetched in form of images and point clouds using the sensors (camera, LiDAR) present in the vehicle and fed as inputs for the deep learning models present in the vehicle. However, according to the authors in [22], they question how a machine learning and deep learning model's accuracy for a test data set can be trusted. A test dataset can generally have biased classes, can contain data similar to the training data, and it will not have inappropriate samples (for example, samples like overlapping of classes).

3. Uncertainty

Uncertainty is a state where the AI model is not sure about a particular input and it may produce false predictions [28]. For example, uncertainty in an object detection can occur if the model detects only half of the object and leaves out a portion of the object. In real-time this could lead to serious harms.

c. Security based issues

There are other validation issues that are related with security. In an autonomous vehicle the data collected from the environment can be compromised by a third party (another software or hackers) before they are given to the deep learning model for decision making. Below are few types of security issues that are challenge to the automobile industry and are difficult to validate.

1. Adversarial attacks

Adversarial attacks are perturbations (for example, adding noise) to the input data that cause misclassification and affect the integrity of the AI model. According to the authors in [21, 22, 23, 24], Machine learning and deep learning models are vulnerable to adversarial attacks. These kinds of attacks are common with image recognition and image classification functionalities. According to [23], adversarial attacks are one of the biggest threats in autonomous vehicle systems. If these attacks were to happen in a vehicle that identifies or classifies images, it could lead to severe hazard scenarios to the operator and the environment. These kinds of attacks can be generated using Adversarial networks like GAN, which generates synthetic data based on real-world data. The synthetic data generated look real like the real-world data, but there will be certain small changes that fools the model into making a wrong decision. For example, an autonomous driving system can recognize graffiti as a road and take a wrong decision.

2. Model inversion

According to the author in [25], model inversion is attacks that target the training data of a model. Using this attack, training data can be recreated to access a model. Also, if the access is in a white box form, then the attacker has all the knowledge about the model and its internal structure.

In this paper, the mentioned data-oriented issues will be illustrated using experiments and recommendations will be provided to handle these issues.

Model-oriented issues are not addressed in this paper since it was determined through the literature review [16, 40, 41] that there are fewer concrete studies available to handle model oriented issues. Explainability in a model can be achieved through Explainable AI [40] but however explainable AI is still under development in automobile industry. The term "neuron coverage" [41] refers to the effectiveness with which a deep learning model makes a decision. This is accomplished by calculating the number of neurons active during a decision-making process. However, the authors in [41] express scepticism that neuron coverage is not effective for all deep learning models, as the decision-making process involves additional parameters like the data used for training and testing.

Security related issues are not focussed because they deal with cybersecurity principles more than AI. Additionally, ISO standards like ISO/SAE FDIS 21434 are under development which deals in achieving security in AI powered automotive vehicles.

3.6 Deep learning models

In an autonomous vehicle, a deep learning model plays a huge role in determining the vehicles actions because of the data (images, point cloud from camera or LiDAR sensor) collected from the environment [2,3,26]. The collected data is either used for detection or classification purpose depending on the use case. For example, for a self-driving functionality, data collected (either as images or point clouds) from the environment like vehicles, traffic light, street signs etc should be detected initially and then the detection results will be used for action control (refer section 3.4). In this paper, all experiments will be implemented with the help of deep learning models like ResNet50[39] and SSD-MobileNet [33,34,35]. Based on the literature survey done by the authors in [26,38], it was understood that ResNet50 and SSD-MobileNet is used in autonomous vehicles functionalities like self-driving, object detection, pedestrian detection, lane detection.

ResNet50
ResNet50 is a ResNet [39] (refer A.1 for ResNet architecture) model variant with 48 Convolution layers, 1 MaxPool layer, and 1 Average Pool layer. It has a total of 3.8 x 109 floating point operations. It is a popular ResNet model. ResNet enables the training of hundreds or even thousands of layers while maintaining a high level of performance. One of main advantage of ResNet is it tackles the vanishing gradient problem [43] (When the gradient is backpropagated to prior layers, repeated multiplication may result in an exceedingly tiny gradient. As a result, as the network becomes more established, its performance degrades significantly). ResNet's central concept is to introduce a so-called "identity shortcut link" that bypasses one or more tiers [39]. By leveraging its strong representational capability, the performance of a variety of computer vision applications other than image classification, such as object identification and face recognition, has been improved [39].

SSD-MobileNet
MobileNet [33] is a lightweight deep neural network architecture optimized for mobile and embedded vision applications in SSD-MobileNet [35]. Numerous real-world applications, such as a self-driving car, require identification tasks to be completed quickly on a computationally constrained device. MobileNet was developed in 2017 to meet this purpose.

Single Shot Object Detection [34] or SSD detects several items within an image in a single shot. SSD is a feed-forward convolutional network-based technique that generates a fixed-size collection of bounding boxes and scores for the existence of object class instances within those boxes. SSD is meant to be network-independent, allowing it to run on top of any base network, including VGG, YOLO, and MobileNet.

To handle the practical limitations of running high-resource and power-consuming neural networks on low-end devices in real-time applications, MobileNet was integrated into the SSD framework [35]. So, when MobileNet is used as the base network in the SSD, it becomes SSD-MobileNet [35] (refer A.2 for architecture). This model is already pre-trained with the COCO dataset [31] with a mean average precision (refer section 3.7) of 0.759 or 75.9% [35]. This model is capable of detecting several objects, including buses, cars, motorbikes, pedestrians, and traffic signs [35].

3.7 Metrics

Quality assessment

In this section, the metrics used in the experiments will be explained. For evaluating an object detection model metrics Intersection over Union (IoU), Precision, Recall and mean Average precision (mAP) will be explained in this section.

a. Intersection over Union (IOU)

In object detection to verify that a predicted result of a test sample, is to evaluate how the prediction has covered the actual ground truth object. This can be done using the bounding boxes. A prediction result returns the predicted class name (P_{tc}), confidence level (P_c), and the predicted bounding box vector (P_{bbox}). Compared to the ground truth bounding box (G_{bbox}), the predicted bounding box provides a probability percentage on how much prediction covers ground truth. This is done using IOU.

Consider a target class 't' that should be detected is represented by a ground truth bounding box (G_{bbox}). The detected or predicted area is represented by a predicted bounding box (P_{bbox}). When the predicted and ground-truth bounding boxes area and location are the same, then it is a perfect match. The IOU is equal to the area of overlap (intersection) between the predicted and ground-truth bounding boxes (Pbbox and Gbbox), divided by the size of their union.

$$IOU(P_{bbox}, G_{bbox}) = \frac{area(P_{bbox} \cap G_{bbox})}{area\ (P_{bbox} \cup G_{bbox})}$$

The score of the IOU ranges from 0 to 1. When a score is '1', the bounding boxes ($P_{bbox,}$ G_{bbox}) are perfectly matched, and it is a perfect detection. If the score is '0', the bounding boxes ($P_{bbox,}$ G_{bbox}) do not overlap, and the detection is incorrect. Achieving an IOU score '1' is challenging for most of the benchmarked object detection. Hence, the closer to 1 the IOU score gets, the better the detection. A threshold value is kept to classify the predictions as good or bad. IOU thresholds can be 0.5, 0.6, and 0.75. Preferably 0.5 is chosen as a standard IOU threshold, but it can also be set to different thresholds. If the prediction sample's IOU score is below this threshold value, then predictions are considered as incorrect. Figure 3 represents the IoU scores of the predicted samples. The green bounding box is the ground truth object. The red box is the detected bounding box. Using the IoU formula above, the IoU score is displayed on top of the image.

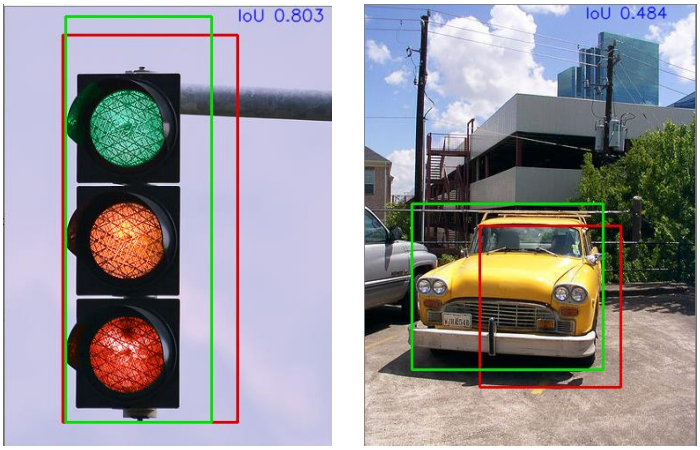


Figure 3 – depicts the IOU scores for predicted test samples.

b. Precision and Recall

Using the IOU threshold, precision and recall values are calculated. From an object detection perspective, precision is a measure that describes how precise the predictions are. The recall is a measure that describes whether all objects are found in an image. Generally, precision and recall are defined using the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) of the prediction samples. With the help of the IOU threshold, a prediction can be categorized, whether it is a TP or FP, or FN. For example, if an IOU score is more significant than 0.5 for a prediction sample, it can be considered a true positive. A prediction sample is false positive when the IOU score is below the threshold (< 0.5). A prediction sample is false negative when there no predictions or detections or if the IOU score is greater than the threshold but has the wrong target class. Using the TPs, FPs, and FNs, the precision and recall are calculated from the below formula.

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad 0 \leq P \leq 1$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad 0 \leq R \leq 1$$

c. Mean Average Precision (mAP)

The weighted average of Average Precision (AP) for all target classes is used to calculate the mean average precision (mAP) for object detection. The formula for calculating mean average precision is as follows. The area under the precision-recall curve [32] can be used to calculate average precision. The precision-recall curve [32] represents a trade-off between precision and recall at various IOU thresholds.

$$mAP = \frac{1}{T_c} \sum_{i=1}^{T_c} AP_i \quad , where\ T_c\ is\ the\ total\ no.\ of\ target\ class, AP\ is\ the\ average\ precisions.$$

Chapter 4 – METHODOLOGY

Introduction

The issues faced in the validation of AI (Deep learning based) software in autonomous vehicles can be categorized into three sections. One is issues caused by data used for training and testing the AI model. Second is issues caused by the black box nature of the AI model. Third is security related issues.

In this chapter, approaches to handle data-oriented issues will be addressed through set of experiments, because these experiments provide a proof of concept on how the data-oriented issues can be handled real-time. The experiments initially illustrate the impact of deep learning-based software related issues in type approval process and then it explains the step-by-step procedure to handle these issues. For data-oriented issues experiments like estimating the quality of dataset used for training & testing the AI model, and estimating the spatial uncertainty of an object detection output are implemented. These experiments are chosen because they depict the AI software's impact on the safety assessment from a data perspective [22, 27, 28].

Overall, this chapter aims to create a standard framework in validating a deep learning software for data-oriented issues present in the autonomous vehicles. The mentioned three experiments have individual data processing methods and proposed metrics based on the papers [22, 27, 28], which will be explained through sub sections below. The three experiments are as follows: creating a quality test dataset, calculating the label quality of an object detector, and estimating the object detector's uncertainty. Each of these studies focuses on a different part of data-driven validation, such as the training and testing data, as well as the model's anticipated output.

Data oriented approaches

4.1 generate a quality test data set.

The main scope of this approach is to create a standard procedure for generating a quality test dataset. This approach is based on the metrics proposed by the authors in [22]. According to the authors in [22] there are three metrics that can estimate the quality of the test data. In this approach using those proposed metrics a quality test data will be generated and tested in real-time pre-trained deep learning model (ResNet50 [39]).

A proof of concept is done with ImageNet [37, 41] - Object Detection dataset to achieve the scope given. ImageNet [37] dataset depicts the reality of camera sensor data as images. ImageNet dataset consists of 1000 classes. The ImageNet dataset is a large database of visual objects created for use in research into computer vision [41]. Over 14 million images have been annotated by the project, and a million of those contain bounding boxes. These annotated bounding boxes play a vital role in the subsequent experiments. Additionally, ImageNet is used in this experiment because ResNet50 is a pretrained model on ImageNet data [39].

For this experiment 7 classes (‘Bicycle’, 'cab', 'scooter', 'sport_car', 'truck', ‘road_signs’, 'signals') are considered for the test data as given in figure 4. These 7 classes are chosen because they depict the real-world environment of an autonomous vehicle. Using the metrics below a quality test dataset is generated from the ImageNet [37]test dataset.

i. Metrics to measure the quality of the test data.

- **Equivalence partitioning (EP)** - it measures the distribution of test data across individual classes. The class level equivalence in the test data set is measured by
  - $EP_i = (ns_i * nc) / ns$,
    where ns_i is the number of test samples belonging to class i, nc is the total number of classes, and ns is the total number of test samples

- **Centroid positioning (CP)** - For each class, measures the percentage of test data that lie close to the centroid of the class cluster (i.e.) test samples close the centroid. The centroid positioning score of a particular class of test data is computed by
  - $\mathbf{CP_i} = \sum_{j=1}^{nsi} cent(ns_i^{(j)})$, where $\text{cent(x)} = \begin{cases} 1, & if\ dist(x, centroid) \leq r \\ 0, & Otherwise \end{cases}$,
    r is the threshold value that classify whether the test point is in the centroid region.

- **Boundary conditioning (BC)** - For each class, it measures the percentage of test data that lie near the boundary. The boundary samples are collected using the convex hull algorithm.

ii. Methodology

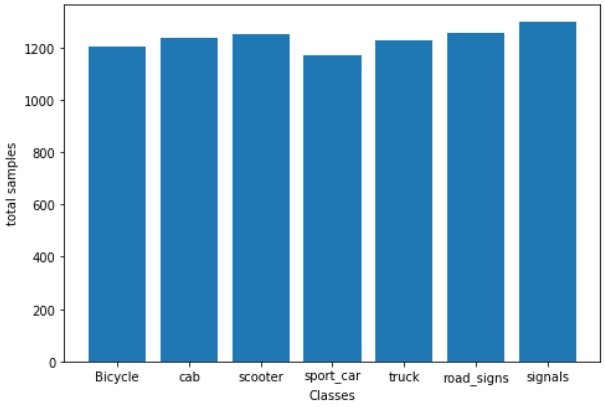


Figure 4: Total test samples for each class – ImageNet

Figure 4 was generated by counting number of test samples for each classes. From figure 4, it can be seen that the test samples are not equally distributed. A score of '1' from the equivalence partitioning metric shows that the classes have equally distributed samples. However, for the ImageNet dataset the score ranged from 0.9 to 1.05 which means most of the test samples are almost equally distributed. Using equivalence partitioning (EP), the equal distribution of classes in the test samples can be identified. A threshold range for the EP score can be set from 0.85 to 1.2. An EP score within this threshold range shows that the classes are equally distributed or almost equally distributed. If the score lies below or above the specified range, then the test samples can be reduced or increased depending on the EP score.

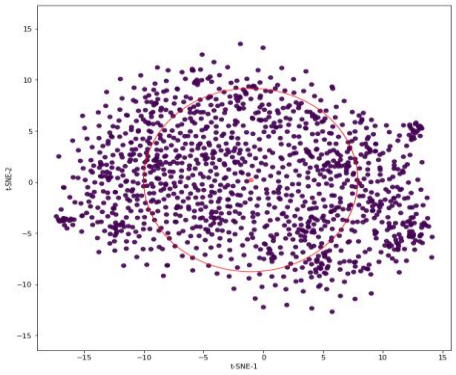


Figure 5 – ‘road signs ’class distribution in 2d feature space for centroid positioning

Figure 5 represents the class 'road signs' from ImageNet data in a 2d feature space. Centroid positioning is applied to the classes in the test samples. For each class, the centroid is calculated by averaging all the feature vectors of points belonging to a single class. The normalized Euclidean distance of all the points belonging to the class is obtained, and a radius threshold of ‘r’ is used to classify whether the test point is in the centroid region. The 'r' value is the average of the distance of all the points in a class. In figure 4, the class 'road signs' consists of 1257 samples. When applied centroid positioning, 620 samples lie near the centroid (i.e.) inside the circle as in figure 5. The ‘r’ in figure 5 is ‘0.5’; means that the radius covers the 50% of the samples in figure 4. Using this the samples that are near the centroid are collected. It is assumed that the samples near the centroid represents a clear visual representation of the target class [22].

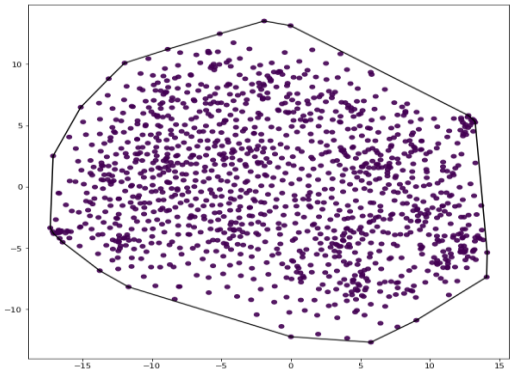


Figure 6 – ‘road signs’ class distribution in 2d feature space for Boundary conditioning

Figure 6 represents the class ‘road signs’ distribution in feature space for Boundary conditioning. The boundary conditioning metric provides the samples near the boundary of the class cluster. These boundary samples provide high robustness evaluation of the model as they are near the boundary [22]. In figure 6 the class ‘road signs’ consists of 30 test samples that lie near the boundary.

iii. Generate perturbed data.

From the above three metrics a test dataset is generated. The test set now consists of equal test samples for all the classes, samples that have a clear representation of the objects, and test samples that lie near the boundary of the class cluster. However, to evaluate the robustness of the model, the test samples can be perturbed to mimic the environmental reality [22, 23, 24]. For example, the images can be blurred to depict a weather condition that made a camera sensor produce a blurry image. These perturbations are done for the remaining samples from centroid

positioning and boundary conditioning. The perturbations can be done with more real-time scenarios like producing an image that shows solar glare in the camera sensor, different weather climates like snow and rain.

Figure 7 mimics a rainy environment where the camera sensors present in the vehicle has captured a rainy image input. This rainy effect was achieved by drawing small random lines on the image (i.e., to display the rain), by reducing the brightness (i.e., rainy environments are mostly shady) and by adding blur to the image (i.e., rainy views can sometime be blurry). The percentage of rain levels are given from 0 to 1. Figure 7 shows a rain level of 0.6. These levels represent the number lines drawn, the brightness level and the blurriness level on the image. Level of 1 shows heavy rain effect and level of 0 shows a drizzle effect.

Figure 7 – Image depicts rainy environment

Figure 8 mimics a foggy environment where the camera sensors present in the vehicle has captured a foggy image input. This foggy effect was achieved by choosing random coordinates within the image pixels and a circle was drawn for each coordinate with a white background. To mimic the foggy effect the opacity of the white circle was reduced using a transparent overlay with an alpha value of 0.1 (0.1 means less opacity, 1.0 means high opacity) and the beta value is (1-alpha). Figure 8 depicts the creation of fog effect by reducing the opacity. The percentage of fog levels are given from 0 to 1. Figure 9 shows a fog level of 0.8. These levels represent the number circles drawn and the blurriness level on the image. Level of 1 shows heavy fog effect and level of 0 shows a less fog effect.

Figure 8 – Creation of fog effect with a level of 0.4

Figure 9 – Image depicts foggy environment

Figure 10 mimics a dark environment where the camera sensors present in the vehicle has captured a darker image input. This darker effect is achieved by scaling the pixel values to 0 (0 black, 225 white).

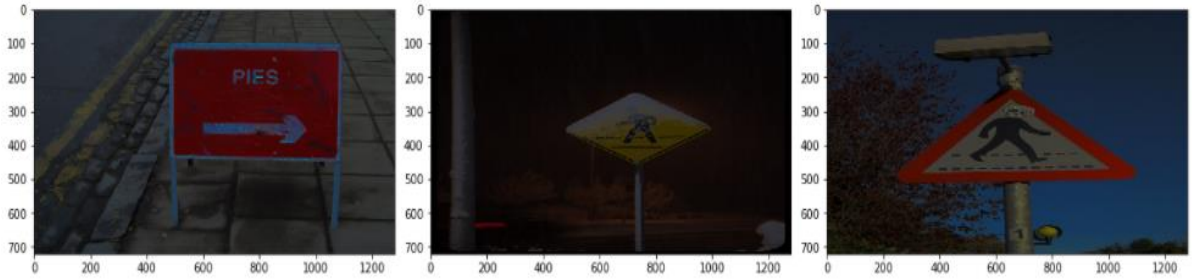


Figure 10 – Image depicts darker environment

Figure 11 mimics a snow environment where the camera sensors present in the vehicle has captured a snowy image input. This was achieved by increasing light or brightness in the image pixel values. Initially a percentage of snow level is chosen from 0 to 1 (0 being less snow, 1 being heavy snow). The snow level is multiplied with the value 255 (represents white colour in the pixel range) and this will be threshold value. Using the threshold value, pixels from the image that are less than this threshold value will be set as 255. This changes the pixels to a white colour; thus it creates a snow effect. Figure 11 shows a snow level of 0.7. However, this snow effect was not helpful for images that have a sunny background because most of the image pixels are brighter.

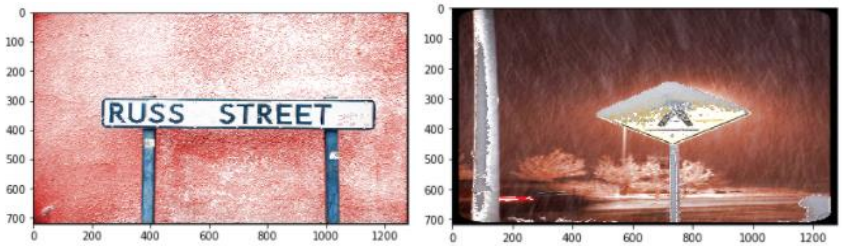


Figure 11 - Image depicts snowy environment

Figure 12 mimics a solar glare effect from the camera sensors. Solar glares are common in a camera lens. Unlike the above effects where the object is visible, a solar glare can hide the whole object making the model blind and resulting in false predictions. Generally, a solar glare will consist of a flare source and it will be accompanied by few smaller glares. To mimic this reality three random coordinates within the image pixels are chosen and three circles were drawn for each coordinate with a white background. Except the flare source rest two flares are made transparent using a transparent overlay as implement in figure 8.

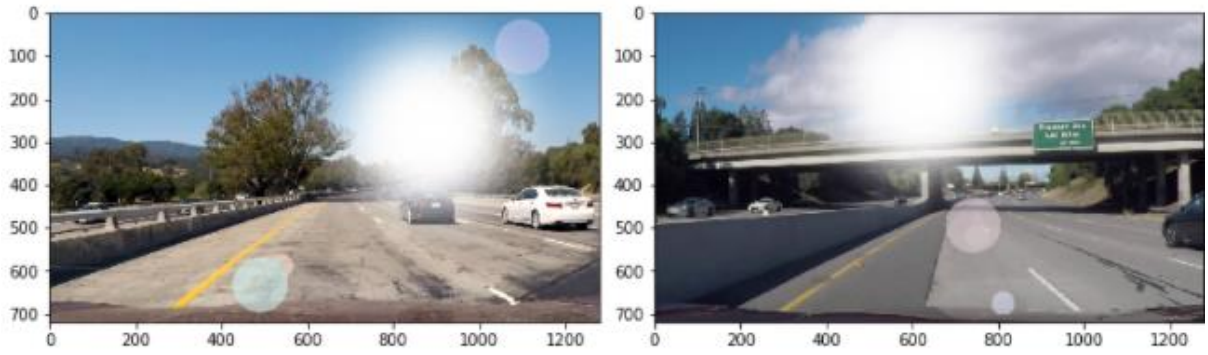


Figure 12 – Image depicts solar glare and speed effects

The above generated data mimics the real-time environment conditions and will provide a good robustness evaluation for the deep learning model present in the autonomous vehicles. However, these effects cannot be applied to all images. For instance, applying a snow effect to a sunny image does not make sense. But this can be achieved by adding two or more effects like adding a dark effect to reduce light or brightness level in the pixels and then a snow effect can be added.

As there were 4102 test samples remaining from the centroid positioning and bounding conditioning. All these 4102 test samples were perturbed with effects mentioned above. 6 sets of perturbed datasets were generated from the remaining 4102 test samples, the first five sets consisted of each effect (rain, snow, fog, dark, solar glare) and the last set consisted of equal distribution of all the effects. These generated perturbed datasets were added to the rest of the test samples (i.e. samples from centroid positioning and boundary conditioning). This procedure is done to understand how the model performs individually on each perturbed effect and how it performs when all the effects were merged together in a dataset. Furthermore, from this approach the model's breaking point (i.e. at what effect level the model produces false predictions) can identified. Finally, the generated test sets were given to the ResNet50 model to classify the inputs.

4.2 Object detector label quality estimation.

The main scope of this experiment is to create a standard procedure that estimates the labelling quality of the training data used in object detection model. Labelling is a key factor in AI models. The process of identifying raw data (images, text files, videos, etc.) and adding one or more meaningful and informative labels to provide context so that a machine learning model can learn from it is referred to as data labeling. In object detection, labelling is done with two steps, first to label an object, second draw bounding boxes or annotations to the object. In figure 13a, the target class is 'traffic light' and the bounding box is drawn in green color.

Autonomous vehicles use object detection to detect an object from the environment through the data (images and point clouds) received from the sensors (camera, LiDAR). The deep learning model present in the vehicle processes the data and detects the object. The detected object contains a bounding box around it. The bounding box depicts that the model knows where the object is located in the data. The bounding boxes are predicted based on the data used for training the deep learning model. The training data consists of the object, the labels, and the bounding box with four values (x, y pixel values with height h and width w) that depict the object's bounding box.

According to the authors in [27], they question the labelling quality of the data used to train the deep learning model. They also explain that the current labelling process, automated process (i.e., using

software to label the data), and manual process (i.e., human annotators) are prone to errors (i.e., wrong labels or wrong bounding boxes). These errors can degrade the deep learning model's performance or provide false detections. In this approach to estimate the labelling quality, a methodology proposed by the authors in [27] was implemented. This methodology validates the training dataset and provides a 'quality estimation procedure'. This approach is achieved using the ImageNet dataset [37]. SSD-MobileNet [35] *(*refer to section 3.6) model is used for training and testing purposes. ImageNet [37] dataset is a benchmarked dataset on object detection, object localization, and classification. In the ImageNet dataset, each target class is categorized and labelled (annotations or bounding boxes) using XML files. This would be helpful when extracting the bounding boxes for introducing errors, as mentioned in the below section.

The proposed procedure for this approach is that the ImageNet data is separated into two sets, 'clean data' and 'modifiable data'. The 'clean data' is annotated correctly (i.e., the bounding boxes for the ground truth object and the target class). The 'modifiable data' is annotated with **bias errors** and **random errors**, as shown in Figure 13. The 'modifiable data' will be termed as 'modified data' after labeling. The SSD-MobileNet is trained on this modified data and the 'clean data' is used for testing the model.

i. Generate annotation errors in the training data

Target classes are chosen for this experiment before generating the label errors in the training data. From the ImageNet dataset, four classes are chosen which are 'sport_car', 'cab', 'traffic_light' and 'street_sign'. There are two reasons why these four classes are chosen, one is since training a real-time deep learning model with more classes is time-consuming, and second is these four classes are used in a day-to-day scenario of an autonomous vehicle. Each of the target classes consisted of 600 to 700 images and most of the images were labeled and annotated with bounding boxes. Some image samples can have two or more target classes. For example, in figure 13, there are multiple traffic lights in that particular image sample. In the ImageNet dataset, each image sample is labeled and annotated in an XML file. Each XML file will have details like the class name, number of target classes, and bounding boxes for each class. Before separating the 'clean data' and the 'modifiable data', a manual check was done to see for samples that are not labeled and annotated. Those samples are labeled and annotated manually using a tool called 'LabelImg' [29]. This tool allows the user to label and annotate the image manually. Once all the images have been labeled and annotated, they were separated as 'clean data' and the 'modifiable data'. For 'clean data', from each target class 100 image samples were taken and there were 400 samples overall. The quality of the 'clean data' was verified manually, by checking the target classes and annotations. The rest of all the remaining samples are kept as 'modifiable data'. There were 2115 training image samples that are kept as 'modifiable data'.

Given an image sample 'i' from 'modifiable data', the target class 't' present in the image consists of label 'l' and bounding boxes 'b'. Each bounding boxes consist of four parameters, x, y coordinate of the target class in the image, a height 'h' & width 'w' that covers the target class 't' as in figure 1. Their corresponding target classes, label names, and bounding boxes are converted to a CSV format (refer A.4) from XML (refer A.3) for each image sample. This is done because processing a CSV file was comparatively easy than an XML file. Nevertheless, this can also be done using XML files. The CSV file consist of data like 'imageFile_name', 'image width', 'image height', 'target class', 'x coordinate', 'y coordinate', 'height', 'width'. After generating the CSV file, these training records 't' are introduced with bias and random errors as given below.

The purpose of creating bias errors is to depict a real-time scenario where a human annotator labels the ground truth object with a false bounding box parameter. The purpose of creating random errors is to depict a scenario where an automated labeling tool generates a randomized bounding box instead of covering the ground truth object. Below are two steps on how to generate these error data.

1. Given a training record 't', for its bounding box **(x, y coordinates, height h, width w)**, a bias value is added to the height and width. The same bias value is given for all the samples in the training record. An observation was made after choosing the bias value. Initially, a bias value of '450' was chosen for height and width. It was seen that most of the training records had no significant differences in their bounding boxes. This was because most of the height and width values were approximately in the range of 430 to 470, irrespective of their target classes for the training records. So a bias of 520 was chosen for the height and width. After manual checking for some random bounding boxes, it was observed that there were some significant changes in their bounding boxes. Also, a smaller bias value can also be provided. The resulting image will look like in figure 13 b. After adding the bias value, the resulting bounding box will be **(x, y coordinates, updated height h', updated width w')**.
2. Given a training record 't', its bounding box **(x, y coordinates, height h, and width w)**. Random values were chosen for each parameter in the bounding box. The maximum value in bounding boxes for all training records was '621'. A range from 0 to 621 was kept, and four random values were generated. These four values are given to bounding box parameters. This can also be done with automated tools like 'LabelImg'[29], where the image can be uploaded and a random bounding box can be drawn anywhere in the image. From the 'LabelImg', the updated bounding box can be imported to a CSV file. The resulting image will look like in figure 13 c. After adding the random values, the resulting bounding box will be **(updated x'& y' coordinates, updated height h', updated width w')**.

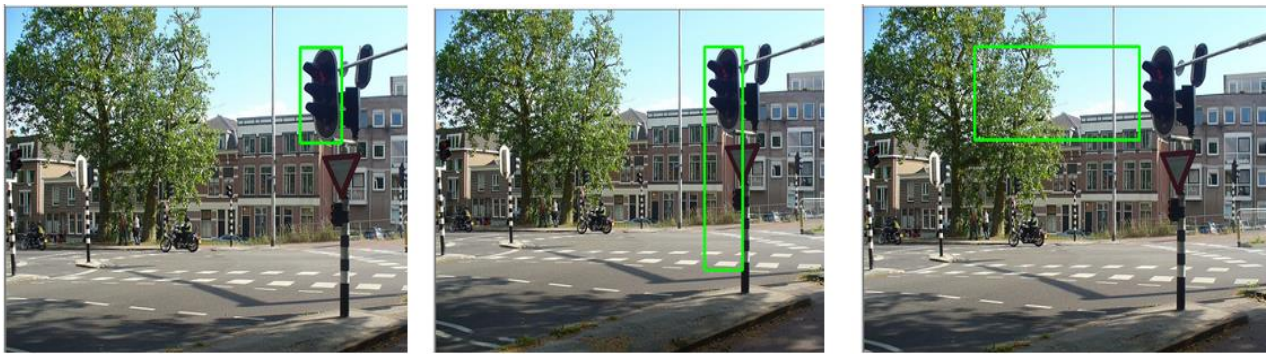


Figure 13, original image – correct Label | b. Image with perturbed bounding boxes | c. Image with randomized bounding boxes

***Note**: The above steps explain the introduction of annotation errors in the target classes. The distribution of the errors in the training records is explained below.*

ii. Procedure

In order to depict the impact of having error labels in the training data, different sets of erroneous training data sets are created [27] (refer table 4). The first training set will have only 25% error data, the second training set will have 50% error data, the third training set

will have 75% error data and the fourth training set will have 100 % of error data. The error distribution process is done for both biased and random errors. Table 4 explains the error distribution percentage in the training records for bias and random errors. There will be eight sets of training data (4 x 2, one for bias errors and another for random errors). The bias and random errors are separated because the purpose was to examine how the model predicts for each type of error.

S.No	Error percentage	Total target classes	Number of records contains annotation errors.
1	25%	2115	528 records
2	50%	2115	1057 records
3	75%	2115	1586 records
4	100%	2115	All the records

Table 4: error distribution percentage in the training records.

Using this approach, a graph is plotted to view the deep learning model's performance for each set of training data, as given in figure 22 and figure 23. Once the training sets are generated, they are trained one by one using a real-time deep learning model, "SSD-MobileNet" [35]. SSD-MobileNet is a pre-trained model that is widely used for object detection, and it is used in this experiment to detect the target classes mentioned above. The trained model is tested with the 'clean data'. The predictions from the trained model are evaluated using the metrics, Intersection over Union and mean average precision (See below for more details). The mean average precision from the prediction results is plotted in a graph like in figure 22 and figure 23. A low mean average precision depicts that the model performance is poor, which means the data used for training the model has a low quality (error in the label or in the bounding boxes). This will be shown with a performance graph (figure 22 and 23) that shows how the model performed for each distribution of erroneous training dataset (*refer to the results and discussion section*).

For this experiment, benchmarked object detection metrics are used. Pascal VOC [30] and COCO object detection metrics [31]. Pascal VOC object detection metric is evaluated on the Pascal VOC dataset [30]. COCO object detection metrics are evaluated on the COCO image dataset [31]. These standard metrics works based on the metrics in section 3.7. One difference between Pascal VOC and COCO detection metrics is their IOU thresholds. For Pascal VOC, default IOU threshold is 0.5 and for COCO it is 0.75. They are widely used because they are used as an evaluation metric on many real-time deep learning models using various Image datasets including ImageNet dataset [37].

4.3. Spatial uncertainty estimation of an object detector.

Uncertainty is one of the issues faced in autonomous vehicles [28]. As mentioned in the previous chapters (refer to section 3.5), it is a situation where the machine learning or deep learning model is unsure about its decision. This could lead to serious hazards. In figure 14, it is seen that an object detection model has detected a car. The confidence is high, and the predicted bounding box is correct. However, the detection is incomplete; there are certain portions of the object ignored by the model. Model is unsure about the decision for that small portion which leads to uncertainty. The main scope of this experiment is to estimate the spatial uncertainty of a real-time object detector. According to the authors in [28], most autonomous vehicles use probability-based object detection. They also state

that the current benchmarked evaluation measures like Intersection over Union (IoU), average precision measures have weaknesses in estimating the accuracy of an object detector. Some weakness like the current evaluation methods measures the label score only without seeing the spatial quality. This evaluation could lead to sub-optimal results [28].

Figure 14: depicts the uncertainty for a detected object.

This measure will evaluate the spatial quality of the object detections. In this approach, the proposed probability-based detection quality (PDQ) measure will be implemented and tested on a real-time object detection model, "SSD-MobileNet"[35] using the ImageNet [37] dataset. A comparison between the existing evaluation measures and spatial quality estimation will be done with the prediction results given by the object detection model.

i. What is spatial quality estimation?

Spatial quality is a measure that shows how well an object detector covers the spatial range of a ground truth object. The spatial quality [28] is calculated with the help of two-loss terms, one is the **foreground loss**, and another is the **background loss**. The foreground loss is the average negative log probability the detector assigns to the pixels for a ground-truth segment [28]. The background loss is calculated by penalizing any probability mass that the detector incorrectly assigned to pixels outside the ground-truth bounding box [28]. Finally, the spatial quality is the exponentiated negative sum of the two-loss terms.

ii. Foreground loss, background loss & spatial quality

Foreground loss (FG) is a loss term that calculates the number of ground truth pixels ignored by the deep learning model's prediction. Bounding boxes enclose the ground truth objects. The prediction consists of the predicted class name, accuracy or confidence, and the predicted bounding box. Figure 15 represents an assumption of a predicted bounding box in red color and a ground truth object in a green-colored box. The foreground loss provides a probability of how much ground truth pixels inside the bounding box was ignored by the model's prediction.

The background loss (BG) is a loss term that calculates how many pixels the model's prediction covered other than the ground truth pixels. From figure 15, it is seen that the predicted bounding box in red color has covered some portion outside of the ground truth box. The calculation of background loss is similar to foreground loss with one difference. In background

loss, the pixels that are not part of the ground truth bounding box but are part of the predicted bounding box are calculated, and a probability score is given.

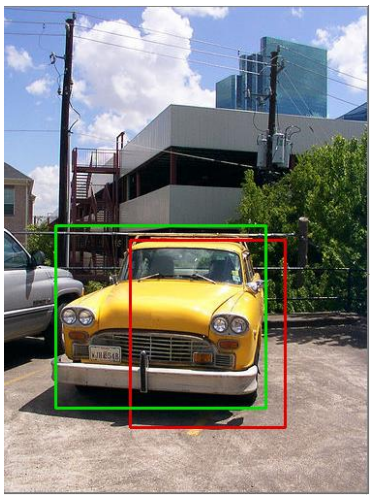


Figure 15: predicted and ground truth bounding boxes.

This is calculated by assigning the probability value 0 or 1 to the pixels. If a predicted pixel (i.e., pixel inside the predicted bounding box) is inside the ground truth bounding box, a probability score "1" is assigned to each pixel. If the predicted pixel is outside the ground truth bounding box, then a probability score "0" is assigned to each pixel. A foreground loss value is then calculated by dividing the number of pixels (the pixel values that are 0) that are ignored by the model's prediction with the total number of pixels that are present inside the ground truth. This is achieved using the formula below.

Formula

- **Spatial quality** [28]: $S(G_i, P_i) = -exp\ (-(S_{FG}(G_i, P_i) + S_{BG}(G_i, P_i)\ ))$.
- **Foreground loss** [28]: $S_{FG}(G_i, P_i) = -\frac{1}{|S_i|}\sum_{x\ \in\ S_i} log(P(x\ \in\ S_i))$, where foreground loss (S_{FG}), S_i is segmentation masks, G_i ground truth object, P_i predicted object (i.e., detected object from the deep learning model)
- **Background loss** [28]: $S_{BG}(G_i, P_i) = -\frac{1}{|S_i|}\sum_{x\ \in\ S_i} log((1 - P(x\ \in\ S_i)))$, where background loss (S_{BG}).

The implementation of foreground loss and background is given below.

iii. Procedure

- To calculate the foreground loss (S_{FG}) and background loss (S_{BG}), a ground truth (G_i) bounding box of the image sample, predicted bounding box (P_i) of that image sample is required.
- A segmentation mask of the ground truth bound box and the predicted bounding box is created. Figure 16a represents an image sample that consists of a ground truth image enclosed in a bounding box. Figure 16b represents a segmentation mask of that ground truth object enclosed in the bounding box. Figure 16c represents a segmentation mask of the predicted bounding box.

- The segmentation mask is created by assigning a white pixel value (255) for the pixels inside the bounding box. For pixels outside the bounding box, the black pixel value (0) is assigned. The resultant mask image will look like in figure 16b, figure 16c. When calculating the two-loss functions, these segmentation masks will compare the predicted bounding box and return the number of ground truth pixels ignored by the predictor or the model.
- The foreground loss formula mentioned above is applied for each pair of the G_i and P_i. Using their respective segmentation masks, number of ground truth pixels ignored by the predictor or by the model is identified. A probability is provided by dividing the number of ignored ground truth pixels by the total number of pixels present in the ground truth bounding box. Figure 17 represents the foreground loss probability for the image sample in figure 15. It is seen that there is a 42% (0.42) foreground loss. Figure 15, figure 16b, and figure 16c show visually that the model prediction approximately covered only 60% of the ground truth.
- The background loss formula mentioned above is applied for each pair of the G_i and P_i. The number of pixels that are not a part of G_i but part of P_i is calculated, and it is divided by the total number of pixels present in the ground truth bounding box. Figure 17 presents the background loss probability for the image sample in figure 15. It is seen that from figure 15, there is less background loss because the model predicted bounding box covered only a few of the pixels that are not a part of the ground truth. Hence the background quality was 0.89 or 89 %, and the background loss is (1-background quality), which shows there is an 11% of background loss for figure 15.
- Finally, the spatial quality is calculated with the help of the two-loss term results. If spatial quality is 1, then it is understood that the predictor has covered the spatial range of the ground truth object (i.e., the ground truth bounding box). If the spatial quality is 0, then it is understood that the predictor has not covered the spatial range of the ground truth object. The closer the spatial quality score gets to 1, the better the prediction is spatially.
- When applied spatial quality formula mentioned above for figure 15, the spatial quality score was 0.48 or 48%. Figure 17 represents the spatial quality of figure 15. This was because the foreground loss was low. Since the predictor partially covered the ground truth, it resulted in a low foreground loss.
- In the below sections real-time deep learning models will be used for predicting image samples, and those samples will be checked for their spatial quality score.

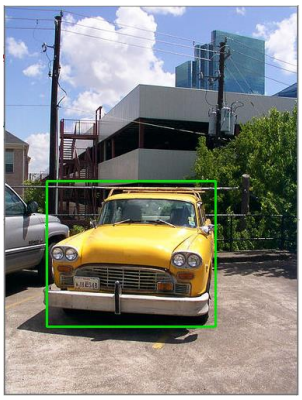


Figure 16a – Image sample with

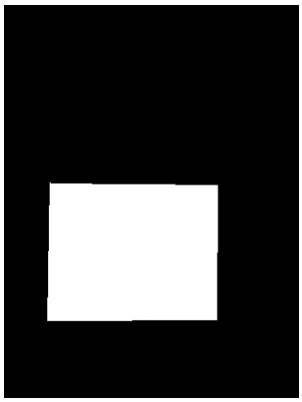


Figure 16b – Segmentation mask of the

ground truth

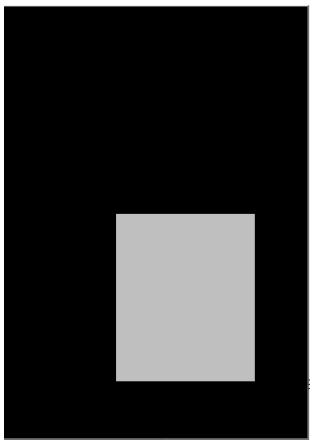


Figure 16c – segmentation mask of the predicted bounding box.

ground truth

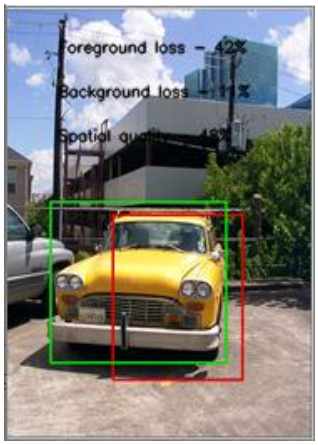


Figure 17 – Spatial quality, foreground Background loss for image sample in figure 2.

iv. Dataset

From the above section, foreground loss, background, and spatial quality calculation were explained in Figure 15. However, Figure 15 was an assumption of a model's prediction. In this section, the spatial quality will be calculated using a real-time model's (SSD-MobileNet [35]) prediction. The ImageNet [37] dataset is used for calculating the spatial quality because it is a benchmarked dataset in object detection and the image samples are annotated with a proper bounding box covering the ground truth object. The target classes used in the previous experiment were used for calculating the spatial quality. This is because the chosen target classes and their image samples were verified previously for bounding boxes, and it has been processed from XML to CSV. The target classes are 'sport_car', 'cab', 'traffic_light' and 'street_sign'. For each target class, 100 test image samples were chosen to calculate the spatial quality. Each image sample will be tested in a real-time deep learning model, and the prediction result will be stored in a CSV file. The generated CSV will consist of 'image name', 'width', 'height', 'predicted target class', 'accuracy', 'predicted bounding box'. For each predicted sample, the ground truth target class and ground truth bounding box were appended to this CSV. This was done because calculating the spatial quality of a predicted bounding box and ground truth bounding of an image sample is required.

For the above experiment methodologies, their results and analysis will be discussed in next chapter.

Chapter 5 – RESULTS and DISCUSSION

5.1. Generate a quality test data set - Results and discussions

ResNet50 is a pre-trained model on ImageNet dataset the prediction results consisted of more classes than the chosen 7 classes. There were certain misclassifications for the perturbed data that was created from the above effects. In figure 18, the data was perturbed with a fog effect level of 0.6 and the ResNet50 model was able to predict it as 'Street sign' with an accuracy of 98%. However, for figure 19 when the fog level effect was increased to a level of 0.9 the model misclassified the object as a 'web site' with a very low accuracy of 15%. From figure 18 and 19 it is seen that the model's prediction becomes incorrect from between the effect level 0.6 to 0.9. Furthermore, to see at which fog level the model breaks and gives wrong prediction, the fog levels were increased to different levels (0.65, 0.7, 0.75, 0.8 and 0.85) and were tested with the ResNet50. It was seen that for the image on the figure 18 & 19 the model's prediction accuracy started to decrease on the fog level of 0.8 (the accuracy was 56%). However, for every perturbed image input the model's breaking point differs based on image's background, colour, brightness, type of effects used, and effects level. Table 5 provides the details about the ResNet50's performance on the generated test datasets. It also provides the information about the types of effects used and the average breaking point range for ResNet50 on each effect. Refer A.5 for ResNet50 model accuracy for all perturbation effects from table 5.

Figure 18 – 60% (0.6) Fog effect prediction

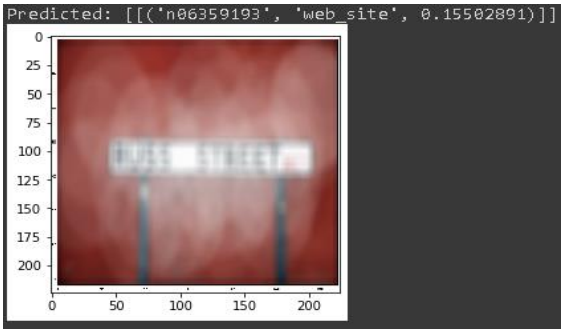


Figure 19 – 90% (0.9) Fog effect prediction

s.no	Type of effect	Overall Model accuracy	Breaking point (i.e. effects level where the model breaks and produces false prediction for the perturbed data)
1	Rain	73%	(0.75 to 0.8)
2	Fog	71%	(0.8 to 0.85)
3	Snow	81%	(0.9>=)
4	Dark	79%	(0.8 to 0.9)

Table 5- ResNet50 performance details on perturbed data

For the solar glare effect unlike other effects, it does not have an effect level (from 0 to 1). So hence the model's breaking point cannot be identified. However, through manual verification of the predicted test results for solar glare effect it was found that, for images that has solar glares covering the ground truth object the prediction was incorrect or the accuracy was low (below 60 %) like in figure 20. For solar glares that does not cover the ground truth object the predictions were correct (like in figure 21). The overall accuracy for the test set that contains only solar effect perturbation was

90% as most of the solar glares did not cover the ground truth object. Finally, for the test set that consisted of all the perturbed effects with a level 0.7 (70% perturbation level), the accuracy was 78%.

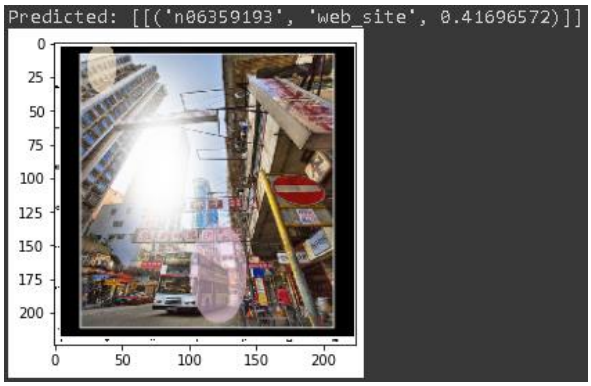


Figure 20 – Incorrect prediction for solar glare. Figure 21 – Correct prediction for solar glare

i. Evaluate a model for type approval using quality test data.

- A quality test data can be generated by following the steps outlined above (refer to section 4.1, i, ii, iii). During the type approval process, this test data can be used to evaluate a model.
- Initially, the target classes that needs to be evaluated should be determined. For example, the target classes used in this experiment can be used to generate a quality test data.
- Using equivalence partitioning (EP) function, the equal distribution of test samples across the target classes are verified. As mentioned in section 4.1 ii, the EP score should be in range of 0.85 to 1.2.
- The centroid positioning (CP) function provides the test samples that are near the centroid. A default of 0.5 is given as a radius, but this can be modified.
- The boundary conditioning (BC) function provides the test samples near the boundary.
- After applying these three functions, perturbation effects are introduced into the test samples that do not fall within CP or BC.
    - Perturbation function is applied to the test samples. The type of perturbation, such as rain, fog, or solar glare, can be specified in the function, along with its intensity on a scale of 0 to 1
- After the generation of perturbation effects, all the test image samples are combined from CP, BC. The generated test data is given to the model present in the autonomous vehicle. The performance of the model is calculated using metrics as mentioned in section 3.7

5.2. Estimate the labelling quality of an object detector - Results and discussion

In this experiment, the SSD-MobileNet model will be trained with the ImageNet data with erroneous annotations that are generated from the above steps. The model is downloaded from the TensorFlow repository hub [36]. The downloaded model is configured to accept the ImageNet classes on its pipeline configuration file. Initially, the target classes will be given as '0' in the pipeline configuration file. This was changed to '4' as there were four target classes. Finally, the input path for training samples is provided in the pipeline configuration file, and the model is trained. The model is trained until a stable model loss of 0.05 is achieved. The trained model is then saved for testing.

Following the generation of training datasets for bias and random errors (refer to table 4), these error sets are trained on the SSD-MobileNet. The trained models are tested using the 'clean data' and their prediction results are evaluated using the metrics above. Using the Pascal VOC object detection metric, the default IOU threshold of 0.5 was used and calculated, as mentioned in [30]. For the COCO object detection metric, the default IOU threshold of 0.75 was used and calculated, as mentioned in [31]. Figure 22 and figure 23 represent a graph plotted on mean average precision for each distribution of erroneous training dataset. Table 2 shows the mAP achieved by the model for bias and random erroneous training dataset.

S.No	Error distribution	mAP for bias		mAP for random	
		COCO	Pascal VOC	COCO	Pascal VOC
1	25%	0.68	0.71	0.65	0.7
2	50%	0.54	0.6	0.43	0.51
3	75%	0.32	0.39	0.19	0.23
4	100%	0.15	0.28	0.02	0.08

Table 6 – mAP for bias and random errors

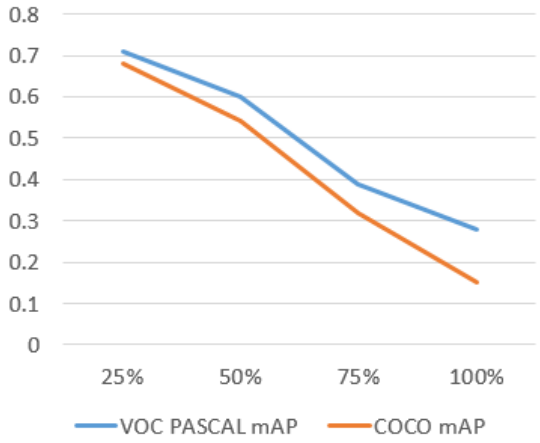


Figure 22 – mAP vs. error distribution for Bias errors

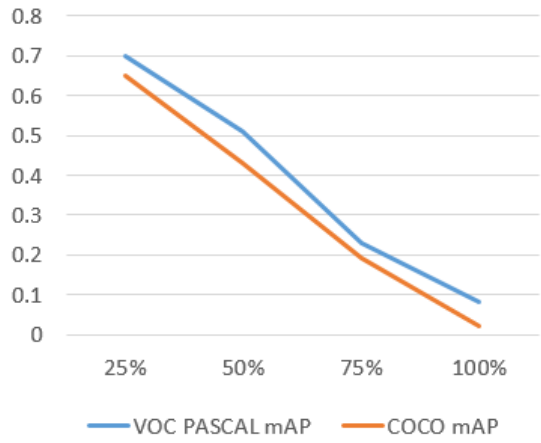


Figure 23 – mAP vs. error distribution for random

For bias and random errors, as shown in Table 6, Figure 22, and Figure 24, the model's performance decreases linearly as the error percentage increases. In figure 23 & 24, the X-axis is the error distribution, and the Y-axis is the mean average precision (mAP). However, when the bias value or random values are changed, this linear decrease is subject to change. Table 6, Figure 22, and Figure 24 illustrates the impact of having erroneous data in the training set (mislabelling, incorrect annotation, or bounding box).

- The model performed satisfactorily for 25% of error data (bias and random errors) for Pascal VOC and COCO metrics. This is because 75% of training data has samples that are correctly labeled and annotated.
- This not the case when the error percentage is 50%. Particularly for random errors, there is a significant drop in their mAPs because 50% of the training data has error annotations and all of them are random, unlike bias errors.
- For 75% of error distribution, there is more significant drop in the mAP for random errors and bias errors.

- For 100% error distribution, the mAP for random errors drops significantly because all the training samples have random annotations (i.e., bounding boxes). However, for bias errors, the mAP drops to 0.28 for Pascal VOC and 0.15 for COCO; this is because of the bias value ('520'). Some of the predicted training samples were inside the IOU threshold within the range of (0.5 to 0.57) and very few samples were in the range of (0.75 to 0.79). In table 6, this can be seen for Pascal VOC the mAP is 0.28 because the IOU threshold was '0.5'. For COCO, the mAP is 0.15 because the IOU threshold was '0.75'. If a higher bias value or a low bias value were chosen, then there would be significant changes in the mAPs of bias error distribution samples.
- Figure 24 and figure 25, shows the model's predicted output for bias and random errors. The model correctly predicted the target class with a low accuracy of 34%. In Figure 24, it can be seen that the predicted bounding box's height h is greater than the target class. In Figure 25, the predicted bounding box is randomly specified in the image. In a real-time autonomous vehicle scenario such erroneous detection could cause dangerous behaviour to the environment or to the driver.

Figure 24: predicted output for bias error

Figure 25: predicted output for random error

i. Label quality estimation for the type approval

a. From the above steps, it was understood that having errors in the annotations of an object detection training dataset will hinder the model's performance and gives false detections. On the other hand, the 'clean data' helped to evaluate the prediction results by calculating the mean average precisions of the test samples present in it. From table 6, for error distribution of 25% the model performed satisfactory despite of having erroneous annotations. Their mAP for bias and random errors is from 0.65 to 0.7 (Pascal VOC and COCO). The optimal mAP threshold can be kept as 0.7.

b. During the type approval process, if a deep learning model of an autonomous vehicle is to be validated, the below steps can be followed to ensure the labeling quality of an object detector.

   i. The 'clean data' generated from this experiment can be used. The target classes in the 'clean data' are used in a day-to-day scenario of an autonomous vehicle. If required, more target classes can be added. However, it should be verified that the samples are labeled and annotated perfectly.

   ii. The deep learning model is tested with 'clean data'. The prediction results are evaluated by calculating the IOU scores. Pascal VOC or COCO

detection metrics can be used. The IOU threshold can be set within an optimal of (0.50 to 0.75).

1. If the resulting mAP score of the test samples is below 0.65 then keeping table 6 as reference it can be assumed that model had erroneous data (incorrect annotations or label) on its training dataset. In table 6, for 50% error distribution and above the mAPs were below 0.65.
2. If the resulting mAP score for the test samples is between 0.65 and 0.7, it is reasonable to assume that the model performed satisfactorily with few erroneous data in its training dataset. Manual verification of the IOU scores is possible to determine which samples scored below the IOU threshold. Additionally, a visual analysis can be performed to determine which target class performed poorly, which can be communicated to the manufacturers later.
3. If the resulting mAP score of the test samples is above 0.7, it can be assumed from table 6 that the model performance is good.

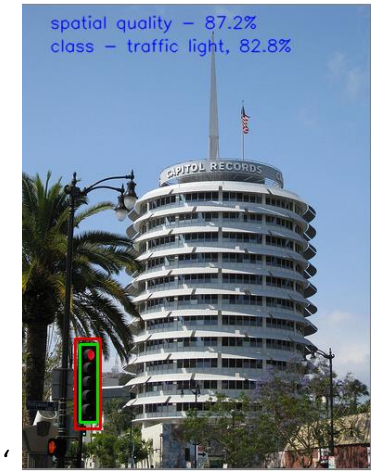
5.3. Estimate the spatial uncertainty of an object detector - Results and discussion

An SSD-MobileNet [35] deep learning model is used for predicting the image samples. As mentioned in previous experiment, SSD-MobileNet is a pre-trained model, and it is capable of detecting objects, including the target classes mentioned in this experiment. The SSD-MobileNet model is downloaded from the TensorFlow repository [36]. Each image samples are predicted for the ground truth objects. The predicted results are stored in a CSV along with the ground truth object details (i.e., bounding box, original target class).

The generated CSV is given to the spatial quality estimation functionality, which takes the predicted bounding box, ground truth bounding, image sample as input and returns the spatial quality as the output. Figure 26 represents the spatial quality for the predicted samples for each target class. From figure 26, it is seen that for SSD-MobileNet predictions, the spatial quality is good. The model's prediction has covered the ground truth objects for almost all the image samples. This is because SSD-MobileNet is a pre-trained model on the target classes, hence good predictions.

a: for target class – street sign

b: for target class - cab



c: for target class – sports car

d: for target class – traffic light

Figure 26 – spatial qualities for the predicted image samples

- From figure 26a, it is seen that the model has predicted the target class with 98% accuracy or confidence. It is clearly seen that the prediction has covered the ground truth object completely with some minor loss in its foreground around the edges, hence a 97% spatial quality.
- From figure 26b, it is seen that model has predicted the target class with 94% accuracy. The spatial quality is 94%; this is because there is a minor loss in its foreground. However, on visual inspection, the model has covered the ground truth object entirely.
- From figure 26c, the prediction accuracy is 95.6%. However, its spatial quality was reduced to 75.8% because there is a significant loss in its background (i.e., the predictor has covered pixels that are not part of the original ground truth) and a minor loss in its foreground.
- From figure 26d, the prediction accuracy is 82.8%. The spatial quality is 87.2%; this is because the model's prediction has completely covered the ground truth, but there is a significant loss in the background.

From all the above images in figure 26, it is understood that the model's prediction covered the ground truth object almost completely. However, due to some loss in its background, the spatial quality is reduced. Including these above four image samples, the remaining image samples from the ImageNet dataset were given to the SSD-MobileNet model for prediction, and their results were checked for spatial quality. The spatial quality was calculated for each image sample, and an average was taken for all the 400 image samples. It was observed that for 400 image samples that consisted of 4 target classes, the spatial quality was 79.3 %. Through visual observation, it was seen that few image samples had a significant foreground loss and some image samples had background losses. Nevertheless, overall, the model achieved a good spatial quality for the image samples.

In the below section, a comparison is done with other benchmarked object detection metrics like IOU and mean Average Precision (refer to section 3.7). Since the model used in this experiment performed well for spatial quality, the model used in previous experiment (refer to experiment 2) (because the SSD-MobileNet models were trained with annotation errors) is taken to show the impact of having less spatial quality. The spatial quality score from the erroneous model predictions will be compared with the benchmarked object detection metrics. It will be concluded which metrics are best for an object detection scenario.

i. Comparison of IoU, mAP with Spatial quality.

From experiment 2, an erroneous model trained with 75% bias-based annotation errors was chosen. This is because the model itself was trained with error data (i.e., annotation); hence the prediction would consist of erroneous predictions. These predictions will help to understand the usage of spatial quality and it will also help to compare the benchmarked object detection metrics result.

The image samples are given to the erroneous model for prediction. The prediction results consisted of detections like in figure 27. Since the model was trained with 75% of annotation errors, the predictions consisted of false detection (either not covering the ground truth object or covering more pixels apart from the ground truth object). Through visual inspection, most of the predictions covered the ground truth object, but there was a significant loss in their background due to annotation errors. Due to this, the average spatial quality for the model's prediction was 32.4%.

The IoU and mAP were calculated using Pascal VOC [30] and COCO objection [31] metrics with an IoU threshold of 0.5 and 0.75. The resultant mAP for Pascal VOC was 0.385, and for COCO, it was 0.32. The score obtained from mAP and average spatial quality are similar, but compared with each image sample, the spatial quality score and IOU score differ.

For some image samples, the IoU and the spatial quality are similar. However, like in figure 28, the IoU score and spatial quality are different for some image samples. For the image sample with a cab, the IoU score was 0.78, and the spatial quality obtained was 56% or 0.56. This is because, for spatial quality, the prediction has left out some portions of the cab, leading to a significant loss in its foreground. However, the model's prediction had no major background loss; only a few pixels other than the ground truth image were covered. In the case of IoU it checks the overlap between two bounding boxes (i.e., predicted and ground truth) and returns the probability. An issue was observed in this since the IoU score is 0.78; it satisfies Pascal VOC and COCO metrics' threshold, making this detection a true. Although this is a good detection with good target class accuracy and overall coverage in IoU, this detection is not good spatially. In a real-time autonomous vehicle scenario, this could lead to dangerous behavior to the environment or the passenger.

Similarly, for the image sample in figure 28 that has street sign detection, the IoU score was 0.59, and the spatial quality was 48.3% or 0.483. This is because the model's prediction has a significant loss in its foreground and its background, resulting in low spatial quality. However, the IoU score for this image sample is 0.59, which satisfies the threshold set by Pascal VOC.

Figure 27 – predictions from the erroneous model.

Figure 28: Image sample with predictions & ground truth boxes, IoU score, spatial quality and target class details.

It was observed that for some image samples, it had high IoU score and low or average spatial quality (approx. 50%). From a software application perspective, these kinds of detections are acceptable because the action taken based on the detection will not affect the user or the environment. For example, a mobile application that detects flowers or fruits will not have a huge impact on the user or the environment even if there is a partial detection over the ground truth object. This because the purpose of the application is to detect a flower or fruit and provide details to the user. If there is a partial detection, but the target class prediction has good accuracy, the application will provide the details about the object. From an autonomous vehicle perspective, the vehicle's functionality like self-driving, obstacle detection, parking assist depends on the deep learning model that is used for detection. If these detections are not spatially covered with the ground truth like in figure 28, it could lead to dangerous behavior.

To conclude, IoU and mAP are good object detection metrics depending on the application's use case. However, IoU & mAP are not enough to validate the detection results from a deep learning model in autonomous vehicles. Additionally, the calculation of foreground and background loss is critical because it determines pixel-by-pixel whether a prediction has covered the ground truth. In contrast to IoU, which only provides the percentage of overlap between two boxes. Hence, the spatial coverage of the ground truth object is equally important, and it should be used alongside IoU and mAP.

ii. Spatial quality estimation for the type approval

From the above comparison of IoU, mAP with spatial quality, it is understood that estimating the spatial coverage quality of a model's prediction is vital for an autonomous vehicle. This section recommends a step-by-step procedure to estimate the spatial quality for an object detection model present in an autonomous vehicle during the type approval.

- During the type approval, an autonomous vehicle detects objects from the environment, and these detected objects are collected for estimating the spatial quality.
- If an autonomous vehicle is tested using a simulator, then the prediction results can be collected from the simulator itself. If an autonomous vehicle is tested on a real-time track, the prediction results can be fetched from the vehicle’s cloud storage.
- The collected samples are given to the spatial quality checking functionality. As mentioned in the above sections, this function calculates the foreground loss, background loss and returns the spatial quality for each predicted image sample.

- A spatial quality score of 0.75 can be kept as a threshold value. A threshold value of 0.75 means that the prediction has spatially covered 75% of the ground truth object. Besides spatial quality, calculating the IoU and mAP should also be done to ensure a quality model prediction.
- For any image samples, if the spatial quality and IoU differ significantly and the IoU score is higher than the spatial quality score, they should be verified manually. If there is any significant loss in its foreground or background, those predictions can be invalid.
    - For predictions that are made as invalid, the target class is identified, and the model is made to predict for the same target class to verify whether for all image samples there is a less spatial score (below threshold).
    - If there a spatial quality loss for all image samples, then that target class can be reported back to the manufacturer.
    - If only certain image samples with low spatial quality occur, then those image samples' background details are identified, and those samples can be removed.

CHAPTER 6 – CONCLUSION and FUTURE WORKS

6.1 Conclusion

The goal of this paper is to identify the impact of AI in the RDW safety assessment or type approval process and provide solutions to the identified impacts. An extensive literature review revealed that AI has an impact on safety assessment based on three factors. There are three types of issues: model-oriented issues, data-oriented issues, and security issues. Model oriented issues like non explainability, model's behavioural change and faults in network structure of the model have been explained. Data oriented issues like the quality of the test and training data and the uncertainty of the model output has been explained and addressed through experiments. Finally, security issues like model inversion, adversarial attacks have been mentioned.

As mentioned above, this paper concentrated on data-oriented issues such as generating a high-quality test dataset, estimating an object detector's labeling quality, and estimating an object detector's spatial uncertainty. The ImageNet dataset was used in these experiments and real-time deep learning models like ResNet50 and SSD-MobileNet were used for training and testing. In general, the impact of data-oriented issues was illustrated by the proposed experiments. Nevertheless, each experiment has their own conclusions.

- **Generate a quality test dataset.**

  A high robustness evaluation of ResNet50 was achieved from this experiment by generating high quality test data based on the proposed metrics and by the proposed perturbation effects. These above metrics and procedure could be the first step in creating standard test data for validating a deep learning model.

- **Estimation of an object detector's labeling quality.**

  Using an SSD-MobileNet [35] object detection model, the impact of having erroneous data in image labels (i.e., annotations or bounding boxes) was demonstrated. It was discovered using benchmarked object detection metrics such as Pascal VOC [30] and COCO [31] detection metrics that when the error on the training sets increased, the model's performance decreased, resulting in a low mAP score. Using the obtained mAP scores, an optimal mAP threshold of 0.7 was determined. With this mAP threshold, a step-by-step procedure for label quality estimation of an object detector model in an autonomous vehicle that comes for audit during type approval was recommended. This experiment provides more visibility towards the estimation of labeling quality in an object detector.

- **Spatial uncertainty estimation**

  A new object detection metric was implemented in this experiment, which evaluates the spatial coverage of prediction results from an objection detection model over the ground truth object. The spatial quality is calculated using the SSD-MobileNet model's prediction results and two loss terms. After comparing IoU, mAP, and spatial quality, it was concluded that spatial quality estimation is required in conjunction with IoU and mAP for autonomous vehicle scenarios (such as obstacle detection and self-driving) in which the vehicle's action is dependent on the object detection model. This provided a step-by-step guide for estimating spatial quality during the type

approval process. Finally, this experiment introduced a new metric, which could be the next step in evaluating an autonomous vehicle's object detection model.

Overall, a framework with proof concepts and recommendations is proposed as a result of these experiments. Using this paper, the audit inspectors at RDW will have more data-oriented knowledge in evaluating deep learning models in autonomous vehicles.

6.2 Future works

With experiments and recommendations, this paper focuses on data-related issues. However, the AI model's black box nature is a significant impediment during the safety assessment process. Vehicle manufacturers create their own AI models and keep them confidential. The audit inspectors have no visibility into the AI model during the safety assessment or type approval process. They lack information about the model's network architecture, the decision-making process, or why the AI model made a particular choice. These questions may be addressed in the future with the aid of explainable artificial intelligence. Explainable artificial intelligence (XAI) is one of the fastest growing areas of artificial intelligence [40]. There is a great deal of research being conducted in the medical field with XAI, where the model explains why it made a decision. For instance, why did the model predict that a person would develop heart disease? However, XAI's use in the automobile industry is still in development. When combined with the above-mentioned experiments, XAI in the automobile industry will significantly improve the safety assessment process. The more white-box the model, the more visibility it is during the safety assessment.

Apart from model issues, the proposed data-driven experiments can be performed on manufacturers' real-time deep learning models. For instance, evaluating Tesla's HydraNet [40] using the proposed experiments. Additionally, each experiment was carried out using a single deep learning model. In the future, a comparative study can be conducted using various models in conjunction with the proposed experiments, and their results can be analyzed further.

There were no stable ISO standards for artificial intelligence during the time period covered by this paper, which made the literature review extensive and time consuming. However, ISO has recently proposed new approaches to artificial intelligence. In the future, the proposed experiments could be adapted to ISO standards for artificial intelligence.

References

1. *SURFACE VEHICLE RECOMMENDED PRACTICE J3016 JUN2018 Issued 2014-01 Revised 2018-06 Superseding J3016 SEP2016 (R) Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.*
2. *Sensors and Sensor Fusion in Autonomous Vehicles Jelena Kocić, Nenad Jovičić, and Vujo Drndarević 26th Telecommunications forum TELFOR 2018 Serbia, Belgrade, November 20-21, 2018.*
3. *SAFETY FIRST FOR AUTOMATED DRIVING, 2019. (saFAD)*
4. *Automotive Camera Market by Application (ADAS, Park Assist), View Type (Single View, Multi-Camera), Technology (Thermal, Infrared & Digital), Level of Autonomy (L1, L2&3, L4, L5), Vehicle & Class, Electric Vehicle and Region.*
5. *https://www.tesla.com/autopilot*
6. *https://velodynelidar.com/products/*
7. *https://ouster.com/products/os2-lidar-sensor/*
8. *Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review Jamil Fayyad, Mohammad A. Jaradat, Dominique Gruyer and Homayoun Najjaran*
9. *https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf*
10. *.V, Siddhartha & Kalappa, Naveen & Yaji, Sitaram. (2019). Comparison of CAN, LIN, FLEX RAY, and MOST In-vehicle bus protocols.*
11. *ISO 26262 road vehicles and functional safety*
12. *https://www.autosar.org/fileadmin/ABOUT/AUTOSAR_EXP_Introduction.pdf*
13. *A Survey on the Benefits and Drawbacks of AUTOSAR - Silverio Martínez-Fernández, Claudia P. Ayala, Xavier Franch Universitat Politècnica de Catalunya (UPC) – BarcelonaTech Barcelona, Spain {smartinez,cayala,franch}@essi.upc.edu Elisa Y. Nakagawa University of São Paulo (USP) São Carlos, Brazil elisa@icmc.usp.br*
14. *https://www.autosar.org/fileadmin/user_upload/standards/adaptive/19-11/AUTOSAR_EXP_PlatformDesign.pdf*
15. *https://www.autosar.org/fileadmin/ABOUT/AUTOSAR_EXP_Introduction.pdf*
16. *Salay, R., Queiroz, R., and Czarnecki, K., "An Analysis of ISO 26262: Machine Learning and Safety in Automotive Software," SAE Technical Paper 2018-01-1075, 2018, doi:10.4271/2018-01-1075*
17. *Characterizing the Safety of Self-Driving Vehicles: A Fault Containment Protocol for Functionality Involving Vehicle Detection Juan Pimentel Electrical & Computer Engineering Kettering University Flint, Michigan, U.S.A. jpimente@kettering.edu Jennifer Bastiaan Mechanical Engineering Kettering University Flint, Michigan, U.S.A. jbastiaan@kettering.edu 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES) September 12-14, 2018, Madrid, Spain.*
18. *Report on the safety and liability implications of Artificial Intelligence, the Internet of Things and robotics - REPORT FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE COUNCIL, AND THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE*
19. *ISO 21448, the safety of the intended functionality*
20. *Zielke, Thomas. (2020). Is Artificial Intelligence Ready for Standardization?*
21. *Software Testing for Machine Learning Dusica Marijan, Arnaud Gotlieb Simula Research Laboratory, Norway {dusica, arnaud}@simula.no.*

22. *Coverage Testing of Deep Learning Models using Dataset Characterization Senthil Mani, Anush Sankaran, Srikanth Tamilselvam, Akshay Sethi*

23. *DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars*

24. *DeepXplore: Automated Whitebox Testing of Deep Learning Systems*

25. *Improved Techniques for Model Inversion Attacks Si Chen, Ruoxi Jia, Guo-Jun Qi*

26. *A Survey of Deep Learning Techniques for Autonomous Driving, Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, Gigel Macesanu*

27. *Haase-Schuetz, Christian & Hertlein, Heinz & Wiesbeck, W.. (2019). Estimating Labeling Quality with Deep Object Detectors. 10.1109/IVS.2019.8814144.*

28. *Hall, David & Dayoub, Feras & Skinner, John & Zhang, Haoyang & Miller, Dimity & Corke, Peter & Carneiro, Gustavo & Angelova, Anelia & Sunderhauf, Niko. (2020). Probabilistic Object Detection: Definition and Evaluation. 1020-1029. 10.1109/WACV45572.2020.9093599.*

29. *Lin, T. LabelImg. 2015. Available online: https://github.com/tzutalin/labelImg*

30. *The PASCAL Visual Object Classes (VOC) Challenge, by Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn and Andrew Zisserman*

31. *http://cocodataset.org/#detection-eval*

32. *https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html*

33. *Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.*

34. *Liu, Wei & Anguelov, Dragomir & Erhan, Dumitru & Szegedy, Christian & Reed, Scott & Fu, Cheng-Yang & Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. 9905. 21-37. 10.1007/978-3-319-46448-0_2.*

35. *Y. -C. Chiu, C. -Y. Tsai, M. -D. Ruan, G. -Y. Shen and T. -T. Lee, "Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems," 2020 International Conference on System Science and Engineering (ICSSE), 2020, pp. 1-5, doi: 10.1109/ICSSE50014.2020.9219319.*

36. *https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md*

37. *ImageNet Large Scale Visual Recognition Challenge - https://www.image-net.org/challenges/LSVRC/2017/index.php*

38. *Barba Guamán, Luis Rodrigo & Naranjo, José & Ortiz, Anthony. (2020). Deep Learning Framework for Vehicle and Pedestrian Detection in Rural Roads on an Embedded GPU. Electronics. 9. 589. 10.3390/electronics9040589.*

39. *He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.*

40. *F. K. Došilović, M. Brčić and N. Hlupić, "Explainable artificial intelligence: A survey," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 0210-0215, doi: 10.23919/MIPRO.2018.8400040.*

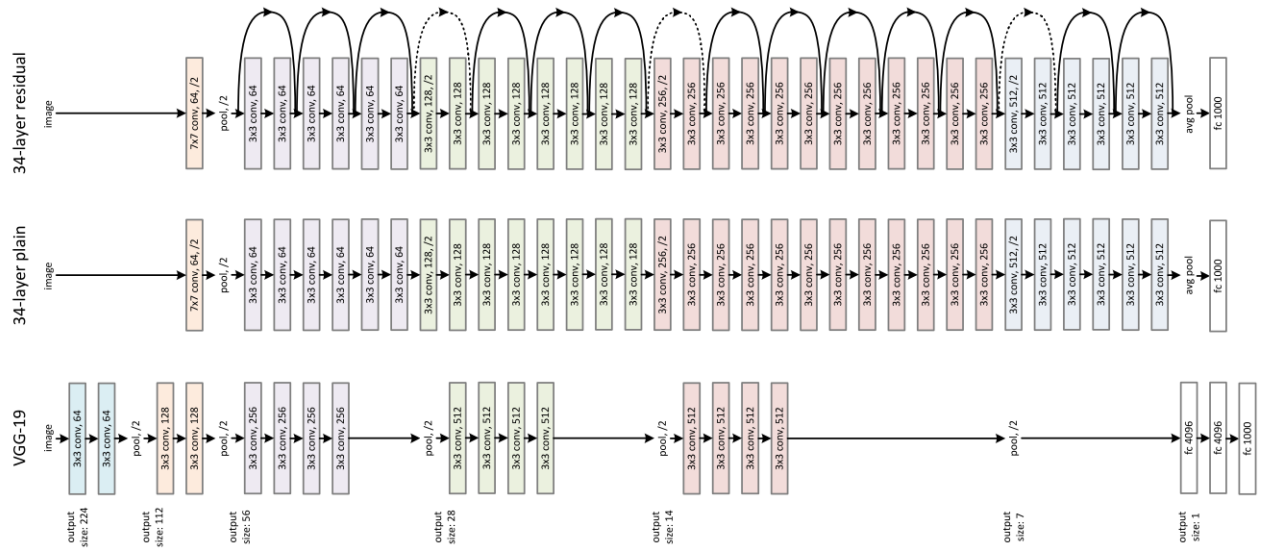
41. *Deng, Jia & Dong, Wei & Socher, Richard & Li, Li-Jia & Li, Kai & Li, Fei-Fei. (2009). ImageNet: a Large-Scale Hierarchical Image Database. IEEE Conference on Computer Vision and Pattern Recognition. 248-255. 10.1109/CVPR.2009.5206848.*

42. *Harel-Canada, F. Y. (2019). Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks?* UCLA. *ProQuest ID: HarelCanada_ucla_0031N_18382. Merritt ID: ark:/13030/m58m2h01. Retrieved from https://escholarship.org/uc/item/3c8107x5*

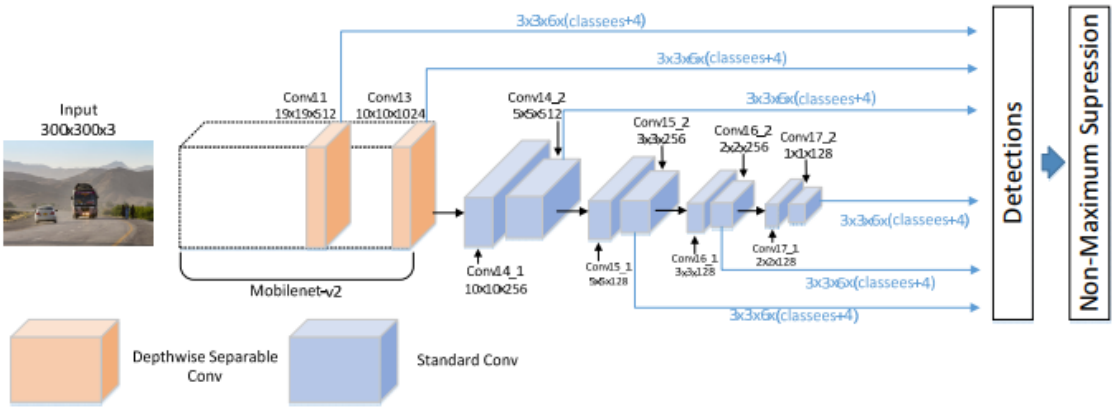
43. *Hochreiter, Sepp. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 6. 107-116. 10.1142/S0218488598000094.*

Appendices A.

A.1 ResNet [39]



A.2 SSD-MobileNet architecture [35]



A.3 Xml file

```
<annotation>
    <folder>n02930766</folder>
    <filename>n02930766_1406</filename>
    <source>
        <database>ILSVRC_2012</database>
    </source>
    <size>
        <width>375</width>
        <height>500</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>n02930766</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>53</xmin>
            <ymin>225</ymin>
            <xmax>270</xmax>
            <ymax>413</ymax>
        </bndbox>
    </object>
</annotation>
```

A.4 CSV file

filename	width	height	class	xmin	ymin	xmax	ymax
n04285008_4829	500	276	n04285008	103	76	407	470
n04285008_4736	500	334	n04285008	3	85	470	470
n04285008_5154	400	300	n04285008	20	43	385	470
n04285008_4704	1920	1200	n04285008	132	435	470	470
n04285008_4331	500	307	n04285008	29	88	466	470
n04285008_4390	602	402	n04285008	30	71	470	470
n04285008_4416	280	173	n04285008	8	30	273	470

A.5 ResNet50 model accuracy vs the perturbation effects.