Prepared at 18:00, August 23, 2022

Reference Guide to Machine Learning Quality Management

March 29, 2022

Technical Report DigiARC-TR-2022-03
Digital Architecture Research Center

Technical Report CPSEC-TR-2022004
Cyber Physical Security Research Center

Technical Report
Artificial Intelligence Research Center

National Institute of Advanced Industrial Science and Technology (AIST)

Foreword

This reference guide was developed by the Committee for Machine Learning Quality Management in the National Institute of Advanced Industrial Science and Technology (AIST).

This guide was developed in a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

Contents

1 Introduction

AI systems attracted much attention with their remarkably high performance in the last decade and have become expected to be deployed widely in a variety of sectors in society worldwide in coming years. Quality of such AI systems matters a lot because their expected uses often concern situations in which their inappropriate behavior may have serious consequences.

Quality of AI systems, however, cannot be managed well with the conventional software quality management methods. To fulfill the need, the Machine Learning Quality Management Guideline (referred to as MLQM Guideline or Guideline) [1] developed by the Committee for Machine Learning Quality Management presents an approach suitable for quality management of AI systems. The Guideline shows viewpoints to set quality targets and key aspects in the AI lifecycle to operate to attain the targets.

While the Guideline describes the general outline of the AI quality management approach, it does not provide details on how to implement the approach for specific AI systems despite that such details are essential for engineers who want to develop and maintain AI systems of high quality. This Reference Guide aims to remedy that shortcoming.

2 Purpose

The purpose of this Reference Guide is to show examples of quality management using the Guideline, and, through such illustrations, to help AI system engineers understand how to use the Guideline.

Here, the authors play the role of development entrustees also known as service developers. The Guideline is considered as a technical starting point for developing the service in question and will be followed from early development stages to ensure quality standards are maintained throughout the process.

Following outcomes are expected to achieve by applying MLQM Guideline to the development process of an AI based product:

- Clear expression of quality goals in business requirements
- Well-defined standards for assessing qualities during AI product design, development, and operation
- Identification of safety-critical scenarios in advance to reduce risks of accidents from possible erroneous behaviors of AI models
- Better evaluation of end product's quality and detection of deviations from expected quality
- Comprehensible demonstration of the product's quality, safety, and reliability for the final users

3 Approach

To explain the usability of MLQM Guideline, five example AI systems from diverse domains are included in this document. The examples are:

1. Object detection and scene classification for autonomous driving
2. Visual inspection of metal casting
3. Postal code recognition by digit classification
4. Prediction of house prices
5. Automated guided vehicles

At this stage, AI systems developed in these examples are fictional and experimental; they are developed only to demonstrate application strategies of MLQM Guideline to evaluate and ensure quality for typical AI products. Among the three primary axes of external qualities identified by MLQM Guideline, each example may highlight one specific axis more exhaustively than others; but in real world, all these qualities must be properly evaluated according to the requirements.

Each of the first four examples consists of two parts:

I. Business requirements to meet

II. Quality management steps following the Guideline

In this version of the Guide, example 5 contains only part I.

One point to note is that these examples are not created by a single person, rather they are a congregation of different ideas and opinions of many researchers and engineers. While developing these examples, two different perspectives are explored to benefit two important user groups of the Guideline. In the *business requirement document* (BRD) (see Appendix A) presented in part I, the authors played the role of *service providing entities* aka *service providers* who wish to develop an AI-based product for some specific service/application. Hypothetical BRDs are created here to lay out required specifications of the final product for the development team.

The external qualities mentioned in MLQM Guideline can help service providers realize quality goals and quality in use. Therefore, at the end of each BRD, business requirements are expressed in terms of the Guideline's external qualities to make them clear and comprehensive for the developers. This step also reflects how much deviation from the expected quality requirements is allowed in the final product.

Next, the authors take the role of *development entrustee* also known as the *service developer*. Here, MLQM Guideline is considered as a *technical starting point* for developing the AI service in question. Accordingly, in part II, the authors follow MLQM guidelines' internal quality evaluation steps from early development stage to evaluate if quality standards according to the provided BRD are achieved throughout the process.

Finally, the entire quality evaluation and management process should be recorded. This Reference Guide presents *quality assessment sheets*, which should facilitate the recording of the process.

4 The structure of the Reference Guide

The rest of this guide is structured as follows.

Chapter 5 gives a quick summary of the approach advocated by the Guideline.

Chapter 6 presents the AI quality assessment sheets, explaining its role, composition and structure.

Chapters 7 to 11 show examples of AI quality management following the Guideline. AI features handled in those chapters are as follows.

- Autonomous driving in Chapter 7
- Visual inspection of metal casting in Chapter 8
- Postal code recognition in Chapter 9
- House price prediction in Chapter 10
- Automated guided vehicles in Chapter 11

Those among the readers who are already familiar with the content of the Guideline could start reading one of the examples in Chapters 7 to 11 immediately. Those who are curious about the ideas and the approach advocated by the Guideline could find a summary in Chapter 5. Those who are practicing AI quality management may obtain good hints on how to plan, organize and record the practices in Chapter 6.

This version of the Reference Guide has several limitations, which the readers are warned about. First, the descriptions of business requirements are often rather sketchy because they are for fictitious examples. At least their goals and the reason why they matter are explained. Second, efforts of quality management explained in this version do not quite attain the quality targets derived from the business requirements due to lack of sufficient amount of relevant data available to the authors. Third, risk assessment in the examples tends to focus only on safety, paying insufficient attention to other types of risks such as lack of fairness and loss of privacy. Other inappropriate descriptions are occasionally marked as such with footnotes. The authors hope, however, the examples should still serve the purpose of illustrating an outline of the practice.

5 An overview of the Guideline

MLQM Guideline assumes that an AI system contains and uses one or more AI components, *qualities in use* of the AI system depend on *external qualities* of the AI components, and the external qualities are in turn realized by achieving and maintaining *internal qualities* of the AI components during their lifecycle.

Based on the assumption, the Guideline advocates the following process to manage qualities of the AI system:

1. Identify requirements on qualities in use of the AI system.
2. Identify external quality requirements on AI components in the system.
3. Determine the required levels of the external qualities.
4. For each of the internal qualities, look up requirements corresponding to the required levels listed in the Guideline.
5. Try to fulfill the requirements through iterations of measurements and improvements
6. Record the process of attempting to fulfill the requirements together with its results.

The second edition of the Guideline lists three external qualities and nine internal qualities. The external qualities are listed as follows.

- Safety/risk avoidance
- AI performance
- Fairness.

The internal qualities include the following.

A-1 Sufficiency of problem domain analysis

A-2 Coverage of distinguished problem cases

B-1 Coverage of datasets

B-2 Uniformity of datasets

B-3 Adequacy of data

C-1 Correctness of trained models

C-2 Stability of trained models

D-1 Reliability of underlying software systems

E-1 Maintainability of quality in operation

The Guideline also assumes that the development of an AI system requires an iterative process; requirements on qualities in use of the system and required levels of external qualities of the AI components tend to be determined and adjusted only gradually through several trials. Each of such trials should exercises the above process advocated by the Guideline.

6 Quality assessment sheets

In this chapter, we examine how MLQM Guideline can be used specifically as a reference guide in the development of AI-based systems and introduce a development process and a set of quality assessment sheets that were developed to support it. The process is built upon and extends concepts in functional safety development, but is designed to be fundamentally generic, i.e., to help manage the various qualities of AI-based systems.

6.1 Overview

In preparing this chapter, we first examined the concept of development based on functional safety to ensure the "risk avoidance" of AI. In particular, we clarified the assumed use of the system and the functions required therein, designed the hardware and software of the control unit to reduce risks, and clarified the required specifications for the AI module.

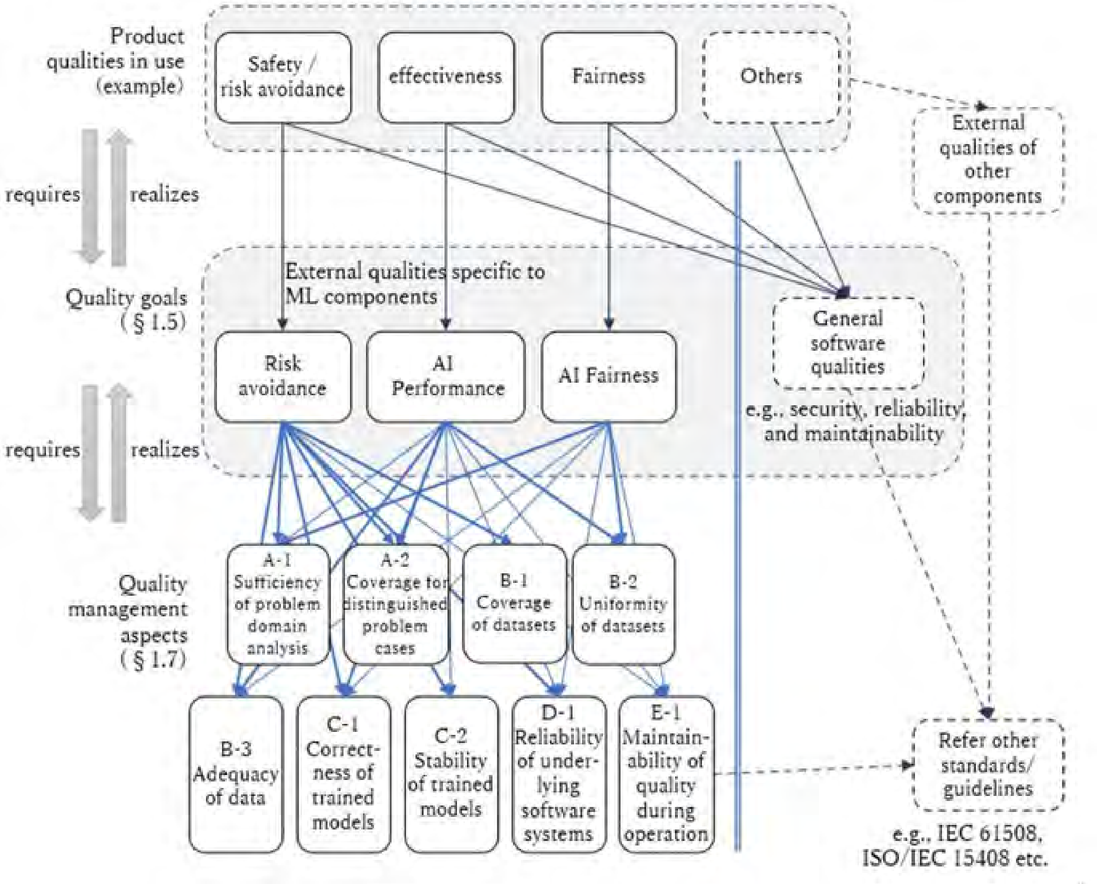
Next, in order to achieve the requirement specifications obtained from the results, we proposed a *development process* to realize an AI-based system based on the internal qualities specified in the Guideline.

Moreover, we have developed *quality assessment sheets* to support and provide evidence for this in the actual promotion of the development process.

6.2 The AI development process

As shown in Figure 1, machine learning quality is understood in the guidelines to be divided into three categories. The first is the *product qualities in use* that the system as a whole is expected to satisfy at that time of use. The second is *external quality*, which is expected to be satisfied by machine-learned components of the system, and the third is *internal quality*, which is inherent to machine-learned components. The external quality is organized to be achieved at the required level through the improvement of the internal quality of the machine learning elements, and the product qualities in use of the final system is to be realized. This includes safety as well as other properties such as effectiveness, fairness, etc.

In the case of AI-based systems covered in this chapter, such as robots, where safety is ensured through control, the development of the system and the design of control devices must comply with functional safety standards such as IEC61508. However, since AI modules are a type of software or programmable element handled by functional safety, we examined the development process based on the concept of functional safety.


Product qualities in use (example)
Safety / risk avoidance
effectiveness
Fairness
Others
External qualities of other components
requires
realizes
Quality goals (§ 1.5)
External qualities specific to ML components
Risk avoidance
AI Performance
AI Fairness
General software qualities
e.g., security, reliability, and maintainability
requires
realizes
Quality management aspects (§ 1.7)
A-1 Sufficiency of problem domain analysis
A-2 Coverage for distinguished problem cases
B-1 Coverage of datasets
B-2 Uniformity of datasets
B-3 Adequacy of data
C-1 Correctness of trained models
C-2 Stability of trained models
D-1 Reliability of underlying software systems
E-1 Maintainability of quality during operation
Refer other standards/ guidelines
e.g., IEC 61508, ISO/IEC 15408 etc.

Source: Machine Learning Quality Management Guideline 2nd Edition

Figure 1. Overall structure of realization of product quality

In addition, in examining the development process, we extended the development process for functional safety as shown in Figure 2 by tying each development process to the internal qualities of machine learning (ML) elements based on the overall structure of quality realization of MLQM Guideline shown in Figure 1. Then, by practicing this development process, we aim to realize the internal qualities and finally quality in use (safety and risk avoidance) of the entire target product.

Although Figure 2 shows workflow for functional safety and labels it as such, mostly similar flows may well be useful to ensure other qualities of AI-based systems, which need to be examined further and validated with additional experiments.

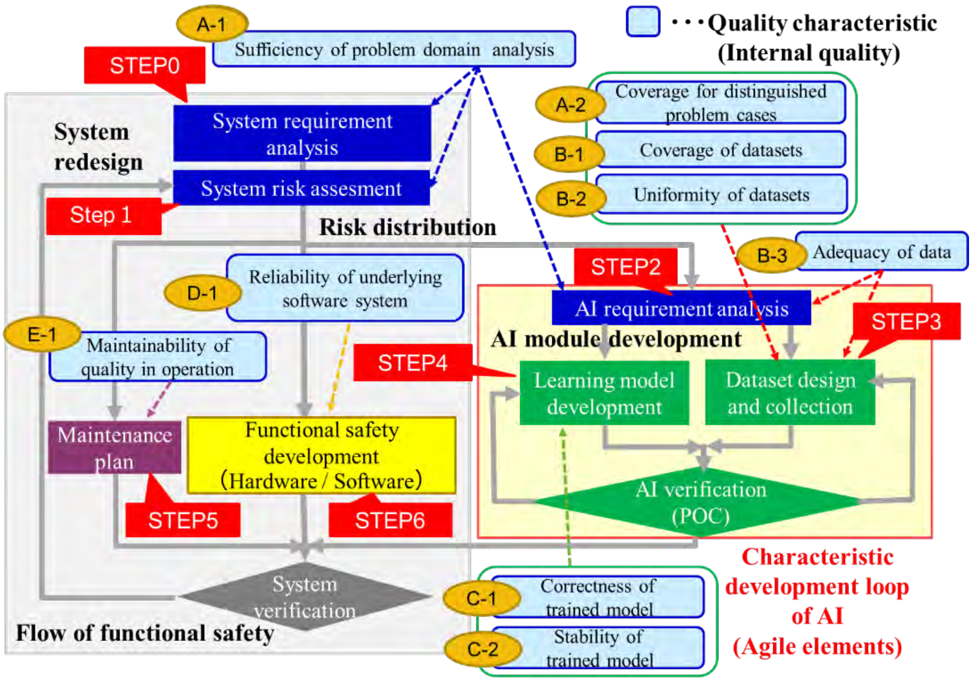

・・・Quality characteristic (Internal quality)
A-1
Sufficiency of problem domain analysis
STEP0
System redesign
System requirement analysis
System risk assesment
Step 1
Risk distribution
A-2
Coverage for distinguished problem cases
B-1
Coverage of datasets
B-2
Uniformity of datasets
B-3
Adequacy of data
STEP2
AI requirement analysis
STEP3
AI module development
D-1
Reliability of underlying software system
E-1
Maintainability of quality in operation
STEP4
Learning model development
Dataset design and collection
Maintenance plan
Functional safety development (Hardware / Software)
AI verification (POC)
STEP5
STEP6
System verification
Flow of functional safety
C-1
Correctness of trained model
C-2
Stability of trained model
Characteristic development loop of AI (Agile elements)

Figure 2. Characteristic development of AI

Figure 2 show the development process proposed in this chapter. The order of development is not limited to the flow shown in Figure 2, and the input/output range of the iterative loops shown in the figure is an example and differs in each stage of development. For example, in the initial stage of Proof of Concept (PoC), when the target system is undecided and only the AI module is considered to verify the feasibility of its function, the development process may be conducted only on the right side of Figure 2.

In each process, the following considerations are made. First, *system requirements analysis* is conducted to define the system requirements. Next, *system risk assessment* (RA) is conducted to identify the system risks and examine the risk reduction measures. Here, risk reduction measures are divided into three major categories. From left to right, there is *maintenance planning* to reduce risk through maintenance and operation, including after the development of the system, *functional safety development* to achieve risk avoidance in the development of hardware and software other than the AI module of the system, and *AI module development* including AI design that is unique to AI. The development of AI modules takes into account the agile elements inherent in AI. In the development of AI modules, *AI requirement analysis* is conducted first for the design of AI modules, followed by *dataset design and collection* and *learning model development* to examine the quality and quantity of datasets, and the learning model and its training method, respectively. The resulting datasets and models will be subjected to *AI verification* including PoC verification, and if the required functionality and performance cannot be achieved, the datasets and learning models will be iteratively reviewed to achieve functionality and performance. In the *development of AI modules*, the scope of process repetition will vary depending on the status of goal achievement in the initial PoC phase, product development phase, and maintenance and operation phase.

After examining the above three risk reduction measures, the achievement of the initially assumed risk reduction for the entire system is confirmed in *system verification*. If the assumed risk reduction cannot be achieved here, the *System RA* is repeated until the risk becomes acceptable, and the risk reduction measures are re-examined.

By following the development process described above, consistent development can be carried out from the upstream, and it becomes clear what needs to be done in each process to achieve the desired performance (quality in use: safety and risk avoidance). Moreover, there will be no omission of considerations, and development will be more consistent.

The left part of the development process basically follows the conventional development process for functional safety, and the right half of the development process for AI-specific AI modules is added to it. As a result, it can be easily integrated with conventional system safety design, and even when AI modules are added to conventional equipment, conventional functional safety design concepts and development processes can be used without modification. Figure 2 also shows the mapping between the internal qualities listed in the Guideline, and each development process, indicating which internal quality should be considered in which process. However, the development flow is shown as a representative example, and a range of iterations other than those described can be assumed.

6.3 The quality assessment sheets

Along with the development of the process, the authors also developed quality assessment sheets to support the implementation of these processes. In these assessment sheets, a sheet has been prepared for each process. Each of these sheets enables evaluation of an internal quality from the Guideline corresponding to the process, and the internal quality can be automatically examined and evaluated as the items on the sheet are described. In this development process, the input and output of each process are defined as shown in Figure 3, and the output of the previous process becomes the input of the following process, making it a coherent process. In the sheets, the outputs are created from those inputs so that the correspondence of the descriptions between processes are traceable.

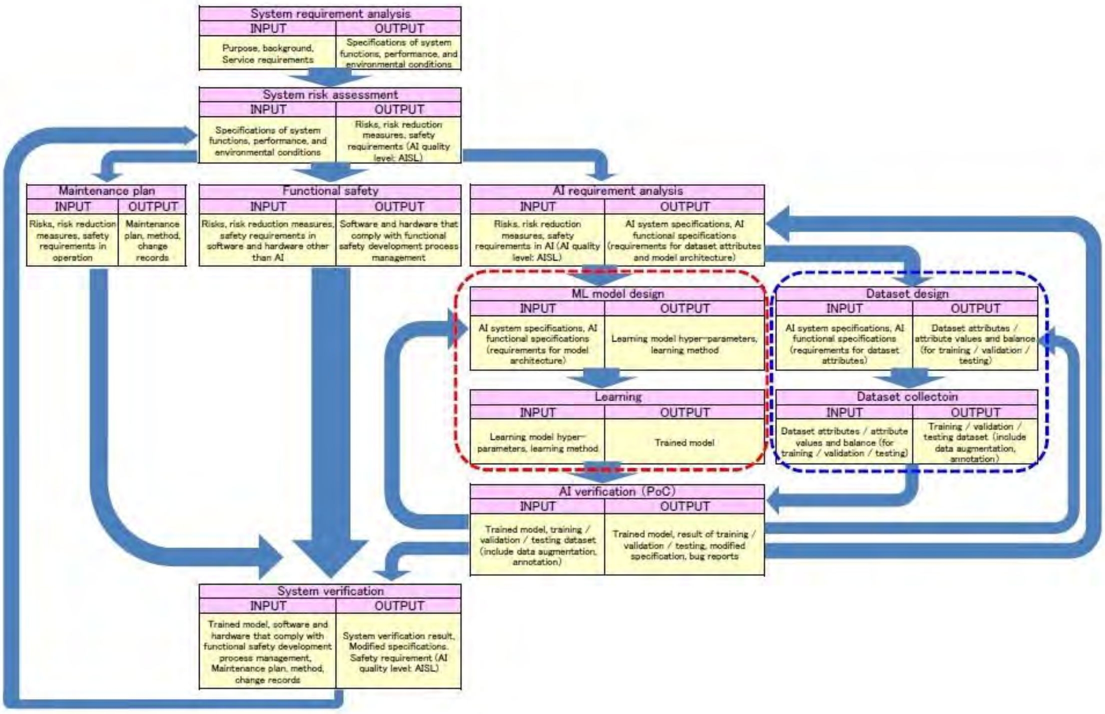

System requirement analysis
INPUT
OUTPUT
Purpose, background, Service requirements
Specifications of system functions, performance, and environmental conditions
System risk assessment
INPUT
OUTPUT
Specifications of system functions, performance, and environmental conditions
Risks, risk reduction measures, safety requirements (AI quality level: AISL)
Maintenance plan
INPUT
OUTPUT
Risks, risk reduction measures, safety requirements in operation
Maintenance plan, method, change records
Functional safety
INPUT
OUTPUT
Risks, risk reduction measures, safety requirements in software and hardware other than AI
Software and hardware that comply with functional safety development process management
AI requirement analysis
INPUT
OUTPUT
Risks, risk reduction measures, safety requirements in AI (AI quality level: AISL)
AI system specifications, AI functional specifications (requirements for dataset attributes and model architecture)
ML model design
INPUT
OUTPUT
AI system specifications, AI functional specifications (requirements for model architecture)
Learning model hyper-parameters, learning method
Learning
INPUT
OUTPUT
Learning model hyper-parameters, learning method
Trained model
Dataset design
INPUT
OUTPUT
AI system specifications, AI functional specifications (requirements for dataset attributes)
Dataset attributes / attribute values and balance (for training / validation / testing)
Dataset collectoin
INPUT
OUTPUT
Dataset attributes / attribute values and balance (for training / validation / testing)
Training / validation / testing dataset (include data augmentation, annotation)
AI verification (PoC)
INPUT
OUTPUT
Trained model, training / validation / testing dataset (include data augmentation, annotation)
Trained model, result of training / validation / testing, modified specification, bug reports
System verification
INPUT
OUTPUT
Trained model, software and hardware that comply with functional safety development process management, Maintenance plan, method, change records
System verification result, Modified specifications. Safety requirement (AI quality level: AISL)

Figure 3. Input/output of each process

The following sections describe the contents of each sheet. Since the contents and details that can be described in the sheet are considered to be different at each stage, such as the initial stage of PoC, the final stage of PoC/start of product development, the completion stage of product development, and the operation stage, the items that need to be described at each stage are indicated in different colors. As for *functional safety development*, there is no difference from the conventional development process for functional safety, so no new assessment sheet has been created. For this process, if necessary, we can divert what we are using for the conventional development process that does not use AI.

6.3.1 System requirement analysis sheet

Target process: System requirement analysis

Input: Intended use, service requirements

Output: System functions (use cases), performance, environmental conditions, usage restrictions, system configuration

In the process of system requirements analysis, service requirements are extracted from use cases corresponding to usage scenarios such as purpose of use, usage method, environment, etc., and system specifications are determined. In the *system requirements* *analysis form*, the contents (use cases) and constraints related to the use in this process are written out and organized, and then incorporated into the system functions, performance, environmental conditions, and system configuration.

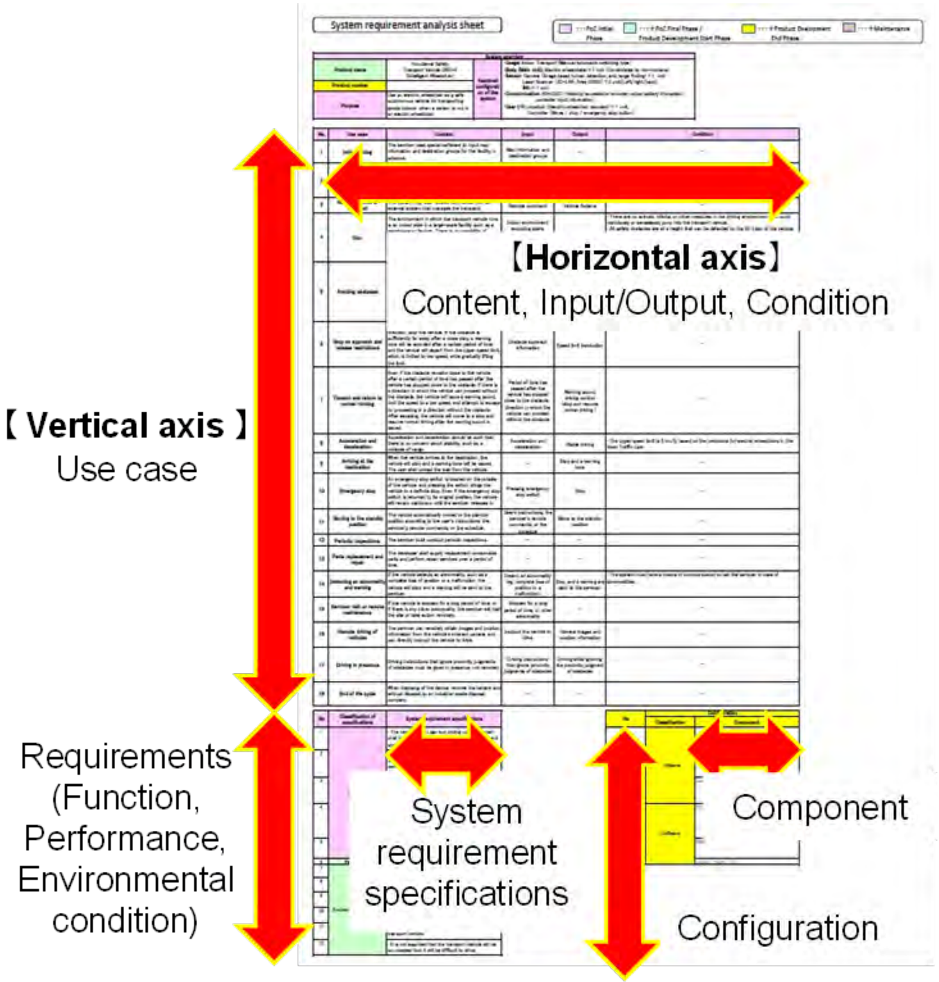


Figure 4. System requirement analysis sheet

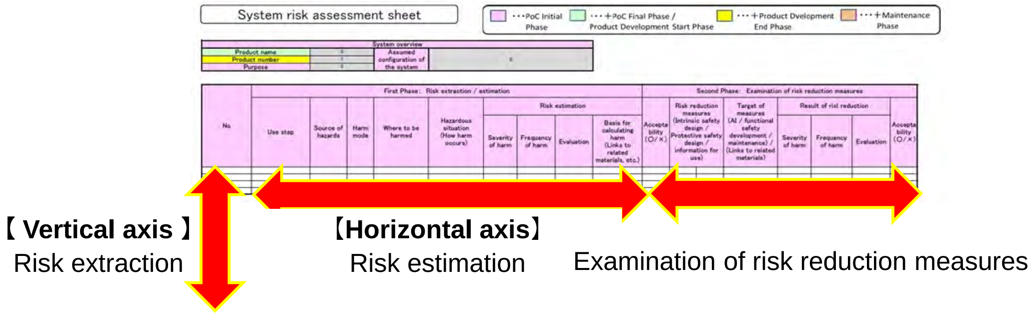
Figure 4 is shown as an example of the format of the form, but as long as the requirements can be organized in the same way as in general requirements analysis, it is not limited to the format shown in this example in particular, and any format that is easy to organize can be used.

6.3.2 System risk assessment sheet

Target Process: System risk assessment

Input: System functions, performance, usage restrictions, system configuration

Output: Risk estimation, risk reduction measures, safety requirement levels



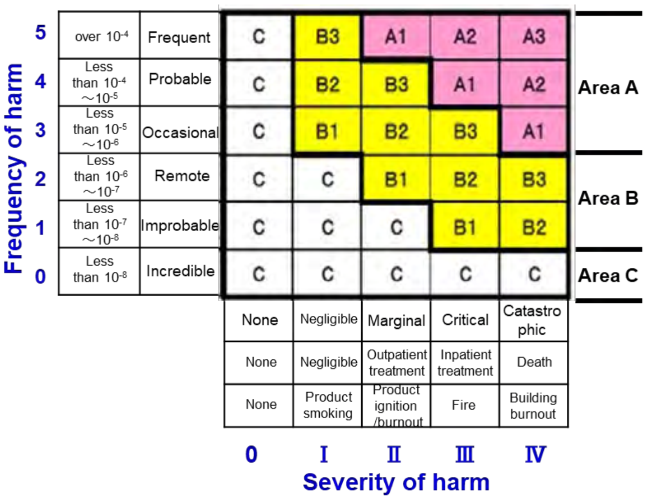
System risk assessment sheet

🟪 ・・・PoC Initial Phase 🟩 ・・・+PoC Final Phase / Product Development Start Phase 🟨 ・・・+Product Dvelopment End Phase 🟧 ・・・+Maintenance Phase

System overview			
Product name	0	Assumed configuration of the system	0
Product number	0		
Purpose	0		

No	First Phase: Risk extraction / estimation									Second Phase: Examination of risk reduction measures						
	Use step	Source of hazards	Harm mode	Where to be harmed	Hazardous situation (How harm occurs)	Risk estimation				Acceptability (O/×)	Risk reduction measures (Intrinsic safety design / Protective safety design / information for use)	Target of measures (AI / functional safety development / maintenance) / (Links to related materials)	Result of risl reduction			Acceptability (O/×)
						Severity of harm	Frequency of harm	Evaluation	Basis for calculating harm (Links to related materials, etc.)				Severity of harm	Frequency of harm	Evaluation	

Figure 5. System risk assessment sheet



Frequency of harm

5	over 10^{-4}	Frequent	C	B3	A1	A2	A3	Area A
4	Less than 10^{-4} ～10^{-5}	Probable	C	B2	B3	A1	A2	
3	Less than 10^{-5} ～10^{-6}	Occasional	C	B1	B2	B3	A1	
2	Less than 10^{-6} ～10^{-7}	Remote	C	C	B1	B2	B3	Area B
1	Less than 10^{-7} ～10^{-8}	Improbable	C	C	C	B1	B2	
0	Less than 10^{-8}	Incredible	C	C	C	C	C	Area C
			None	Negligible	Marginal	Critical	Catastrophic	
			None	Negligible	Outpatient treatment	Inpatient treatment	Death	
			None	Product smoking	Product ignition /burnout	Fire	Building burnout	
			0	I	II	III	IV	

Severity of harm

* This risk map is an example of a risk definition and is not limited to this format.

* "Applying the R-Map Method to Product Safety and Risk Management, Japan", which is referred to as one of the risk management methods from ISO13077:2013(en).

Figure 6. Example of a risk map

In the system RA process, the RA of the conceptually designed system is performed based on the system requirements analysis. This *system risk assessment sheet* identifies the risks of the system in this process, and determines the estimate, reduction method, and reduction target (safety requirement level) for each risk.

The determined risk reduction measures will be passed on to subsequent processes from three perspectives, depending on the content: those related to AI modules, those related to hardware and software design, and those related to maintenance and operation. The format of the sheet shown in Figure 5 is the same as that of a commonly used risk assessment sheet, and it records the extraction of risks, estimation of risks, and consideration of risk reduction measures. Risk estimation methods include, but are not limited to, the use of risk maps as shown in the example in Figure 6 and can be used as long as the policy is clearly stated and can be explained quantitatively to a third party.

The difference is that AI-based risk reduction is added to the risk reduction measures, and AISL, which is the same as SIL in IEC61508 Functional Safety and PL (Performance level) in ISO13849-1, is specified as the required level of safety based on the Guideline.

6.3.3 AI requirement analysis sheet

Target process: AI requirement analysis

Input: Risk estimation, risk reduction measures, safety requirement levels

Output: Requirements for AI system specifications, datasets, and training models

In the process of AI requirement analysis, requirement specifications for AI-based systems, datasets, and learning models are set for the risk reduction measures and safety requirement levels by AI assigned by the system RA.

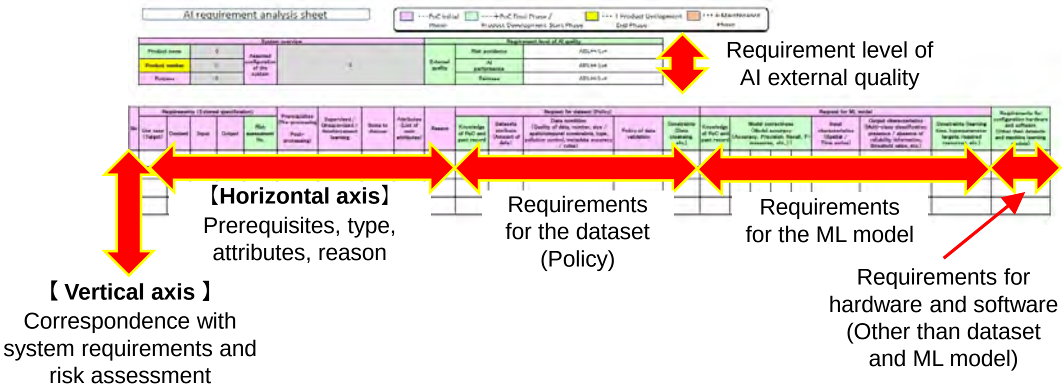

AI requirement analysis sheet
Requirement level of AI external quality
【Horizontal axis】
Prerequisites, type, attributes, reason
Requirements for the dataset (Policy)
Requirements for the ML model
【 Vertical axis 】
Correspondence with system requirements and risk assessment
Requirements for hardware and software (Other than dataset and ML model)

Figure 7. AI requirement assessment sheet

As for the format of the sheet, as shown in Figure 7, in addition to the general requirements analysis, requirements for AI-specific datasets and policies on learning models are required. The *AI requirements analysis sheet* selects the type of AI suitable for the purpose with respect to the AI module in the AI application system. In the case of supervised learning, it identifies the main attributes to be focused on as the characteristics of the learning target, and lists and clarifies the requirements for the dataset, learning model, and other hardware and software, along with their preconditions and reasons, to support the recording. Since it is expected that the range of possible descriptions will change during the development stage, the following guidelines should be used, for example.

In the initial PoC stage, based on the system requirements, the basic roles expected of AI, prerequisites considered necessary (pre-processing, post-processing), learning methods (type of AI) that are candidates for application, policies for collecting and creating datasets used for learning, and input/output characteristics of the learning model are considered and described.

In the final PoC phase/start of product development phase, in addition to the review of the studies in the initial PoC phase, the correspondence with the system RA results, findings from the PoC and past records, constraints, and accuracy requirements of the learning model will be described.

If an organization establishes AI development standards and tailors them for the development project of the target AI-based system in the development planning stage, the scope of description rules for each development stage should be defined in the planning document, and the use of customized forms should be included in the scope of assumption.

6.3.4 Dataset assessment sheet

Target process: Dataset design and collection

Input: AI system specifications, requirement specifications for datasets

Output: Dataset configuration (attributes of the dataset, number of data per attribute)

In the process of dataset design and collection, the attributes of the dataset and the quantity of data for each attribute are recorded against the AI usage system specification and the requirement specification for the dataset set by the AI requirement analysis.

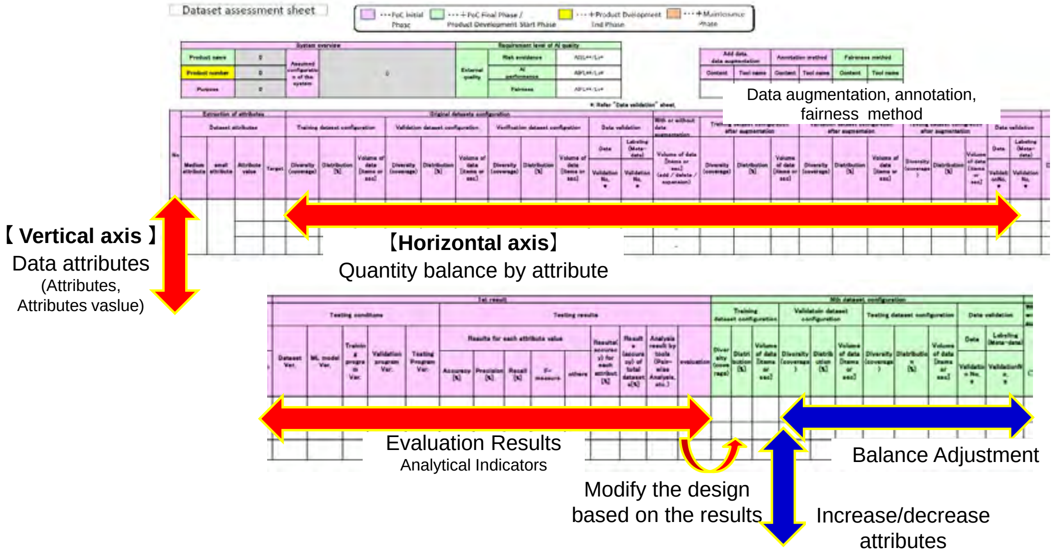
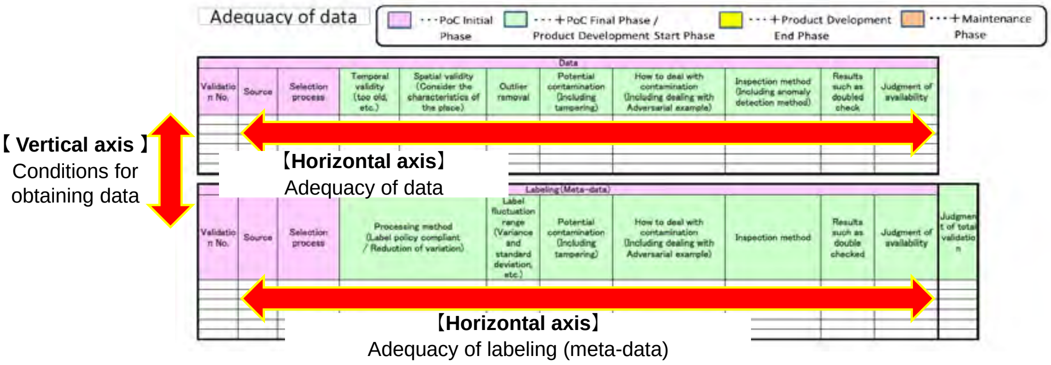

Dataset assessment sheet
PoC initial Phase
PoC Final Phase / Product Development Start Phase
Product Development End Phase
Maintenance Phase
Data augmentation, annotation, fairness method
【Vertical axis】
Data attributes
(Attributes, Attributes vaslue)
【Horizontal axis】
Quantity balance by attribute
Evaluation Results
Analytical Indicators
Modify the design based on the results
Balance Adjustment
Increase/decrease attributes

Figure 8. Dataset assessment sheet

As shown in Figure 8, the *dataset assessment sheet* lists the attributes of the dataset and the initial data quantity for each attribute. Next, the dataset is validated according to the policies specified during the AI requirements analysis, and if necessary, the dataset is expanded, annotations are adjusted, and fairness is addressed. Then, we move on to the next AI verification (PoC) process to verify the AI functions and performance with the dataset. The results are written on the quality assessment sheet, and if the target functions and performance are not achieved, the results are used to increase or decrease the number of attributes, adjust the balance of the number of data for each attribute, check the *adequacy of data*, and conduct the verification again. This is repeated until the target is achieved.

As shown in Figure 8, the format of the sheet is as follows: the attributes are listed on the vertical axis, and for each attribute, the number of data for each attribute and the verification results for that dataset are described on the horizontal axis. With regard to the internal quality of the Guideline, the vertical axis relates to the *coverage of the dataset* and the horizontal axis to the *uniformity of the dataset*, and these internal qualities will be adjusted while checking the contents of the dataset with this sheet.


Adequacy of data
🟪 ･･･PoC Initial Phase
🟩 ･･･＋PoC Final Phase / Product Development Start Phase
🟨 ･･･＋Product Dvelopment End Phase
🟧 ･･･＋Maintenance Phase
【Vertical axis】
Conditions for obtaining data
【Horizontal axis】
Adequacy of data
【Horizontal axis】
Adequacy of labeling (meta-data)

Data										
Validation No.	Source	Selection process	Temporal validity (too old, etc.)	Spatial validity (Consider the characteristics of the place)	Outlier removal	Potential contamination (Including tampering)	How to deal with contamination (Including dealing with Adversarial example)	Inspection method (Including anomaly detection method)	Results such as doubled check	Judgment of availability

Labeling(Meta-data)										
Validation No.	Source	Selection process	Processing method (Label policy compliant / Reduction of variation)	Label fluctuation range (Variance and standard deviation, etc.)	Potential contamination (Including tampering)	How to deal with contamination (Including dealing with Adversarial example)	Inspection method	Results such as double checked	Judgment of availability	Judgment of total validation

Figure 9. Adequacy of data

As shown in Figure 9, for *adequacy of data*, for data, the vertical axis describes the conditions for data acquisition, and the horizontal axis describes the data source (new/processed/appropriated), temporal validity, spatial validity, outlier removal, contamination possibility, contamination handling method, inspection method, confirmation results, and validation details such as judgment on whether the data can be used. For metadata such as labeling information, in addition to the same validation details as for data, it is also checked whether the labeling policy designed in the AI requirements analysis process is followed.

6.3.5 ML model assessment sheet

Target process: ML Model Development

Input: System specification for AI use, requirement specification for training model

Output: Setting up the learning model (including hyper-parameters and learning methods)

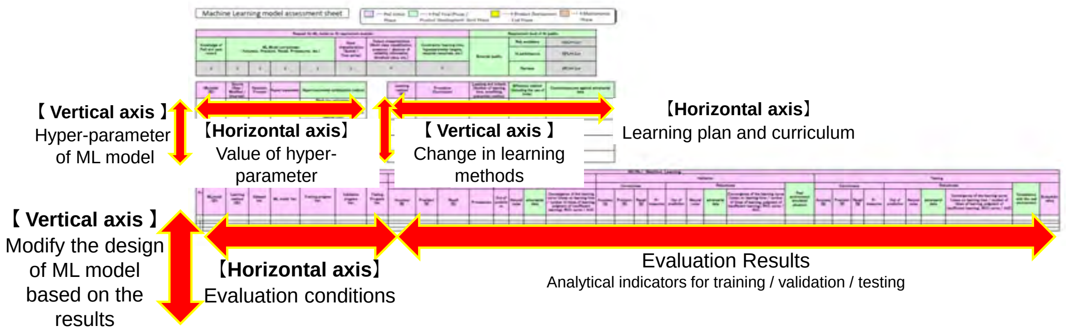

【Vertical axis】
Hyper-parameter of ML model
【Horizontal axis】
Value of hyper-parameter
【Vertical axis】
Change in learning methods
【Horizontal axis】
Learning plan and curriculum
【Vertical axis】
Modify the design of ML model based on the results
【Horizontal axis】
Evaluation conditions
Evaluation Results
Analytical indicators for training / validation / testing
Machine Learning model assessment sheet

Figure 10. ML model assessment sheet

In the process of ML model development, the hyper-parameters of the ML model and the learning method are recorded against the AI-based system specifications and requirements for the ML model set by the AI requirements analysis. Similar to the process of designing and collecting datasets, after recording the hyper-parameters and learning methods in the *ML model assessment sheet*, the AI functions and performance in the ML model will be verified. Then, the results are recorded on the quality assessment sheets. If the target functions and performance are not achieved, the hyper-parameters and learning method are adjusted based on the verification results, and the verification is repeated until the target is achieved.

The format of the sheet is as follows: hyper-parameters are listed on the vertical axis, and the learning method and procedure, as well as the results of verification using the learning model, are described on the horizontal axis. If the learning model itself is to be changed, or if it is to be selected after comparing multiple machine learning models, sheets for the new learning model should be created and the contents of the new learning model should be entered, while leaving the results of the previous learning models study intact.

6.3.6 Maintenance plan and results assessment sheet

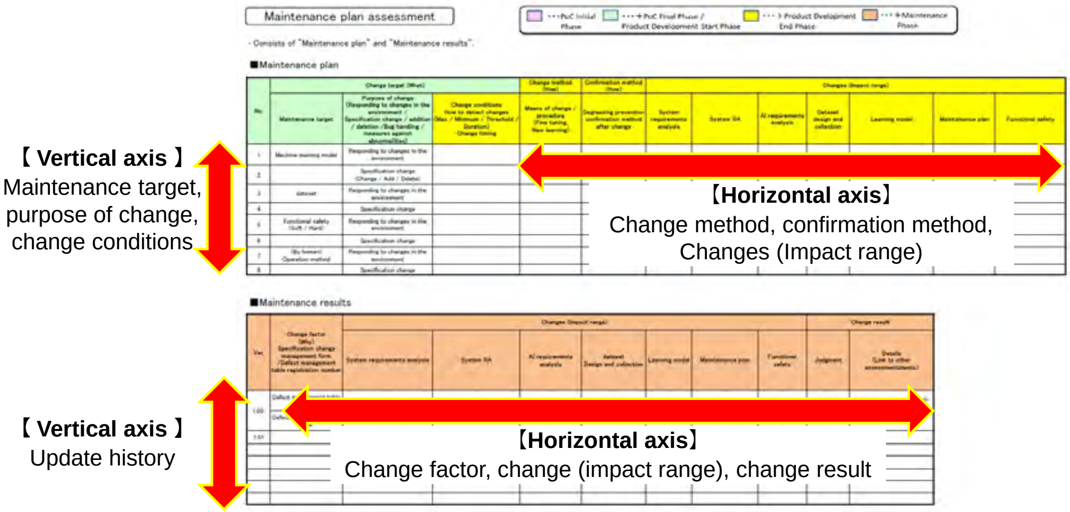
Target process: Maintenance plan

Input: Risks related to maintenance and operation, risk reduction methods, and safety requirement levels

Output: Maintenance (operation) plan and implementation results

In the maintenance planning process, maintenance and operation plans are made and implemented according to the risks, risk reduction methods, and safety requirement levels for maintenance and operation set by the system RA. When planning changes to AI module data sets and learning models, plan how to make changes in accordance with sequential learning and responses to environmental changes. Moreover, in addition to planned implementation, unplanned implementation should also be managed, and the results of changes should be recorded.

The format of the form should be divided into two tables, one for planning and the other for change history. In planning, the plan for operation and maintenance should be recorded. In the plan, record the target of the change, purpose, conditions, means, confirmation method, and scope of influence of the change. For the actual results, the change history, including unplanned changes, should be recorded so that it can be understood, and the scope of influence of the change and the result of the change should also be described. Moreover, for any changes or adjustments to the dataset or learning model in conjunction with this performance, the status of the changes or adjustments is noted on the dataset assessment sheet and learning model assessment sheet so that we can confirm that there is no degradation.



Maintenance plan assessment

・Consists of "Maintenance plan" and "Maintenance results".

■Maintenance plan

No.	Change target (What)			Change method (How)	Confirmation method (How)	Changes (Impact range)						
	Maintenance target	Purpose of change (Responding to changes in the environment / Specification change / addition / deletion / Bug handling / measures against abnormalities)	Change conditions How to detect changes (Max / Minimum / Threshold / Duration) Change timing	Means of change / procedure (Fine tuning, New learning)	Degrading prevention confirmation method after change	System requirements analysis	System RA	AI requirements analysis	Dataset design and collection	Learning model	Maintenance plan	Functional safety
1	Machine learning model	Responding to changes in the environment										
2		Specification change (Change / Add / Delete)										
3	dataset	Responding to changes in the environment										
4		Specification change										
5	Functional safety (Soft / Hard)	Responding to changes in the environment										
6		Specification change										
7	(By human) Operation method	Responding to changes in the environment										
8		Specification change										

■Maintenance results

Ver.	Change factor (Why) Specification change management form /Defect management table registration number	Changes (Impact range)							Change result	
		System requirements analysis	System RA	AI requirements analysis	Dataset Design and collection	Learning model	Maintenance plan	Functional safety	Judgment	Details (Link to other assessment sheets)
1.00	Defect m[illegible]									
	Defec[illegible]									
1.01										

・・・PoC Initial Phase
・・・PoC Final Phase / Product Development Start Phase
・・・Product Development End Phase
・・・Maintenance Phase
【Vertical axis】
Maintenance target, purpose of change, change conditions
【Horizontal axis】
Change method, confirmation method, Changes (Impact range)
【Vertical axis】
Update history
【Horizontal axis】
Change factor, change (impact range), change result

Figure 11. Maintenance plan and results assessment sheet

6.3.7 Functional safety development

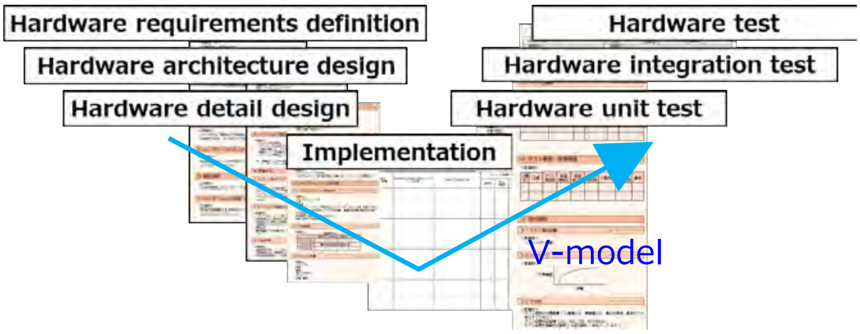
Target process: Functional safety development (Development of other hardware and software)

Input: Risks related to system hardware and software, risk reduction measures, and safety requirement levels

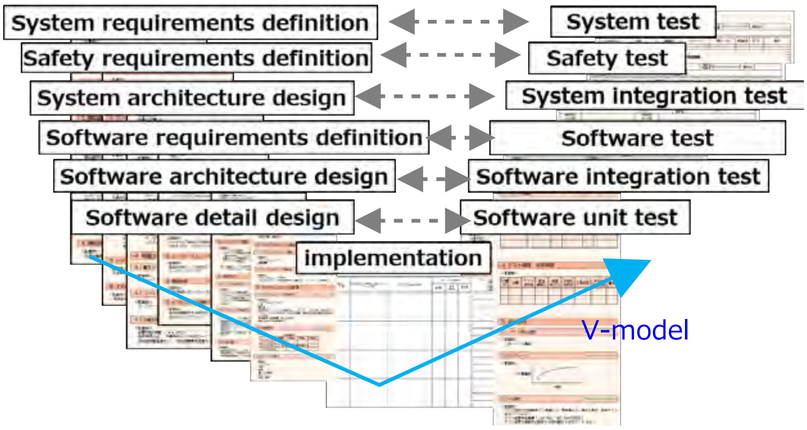
Output: Hardware and software for functional safety

In the functional safety development process, hardware and software are developed for functional safety against the risks, risk reduction methods, and safety requirement levels for the system hardware and software set by the system RA, and there is no difference from general functional safety development except for the processing performance of AI modules. For reference, Figure 12 shows the V-model development that is generally used in functional safety development.

It should be noted that although the V-model shows the positioning and correspondence of each development process, it does not show the order of development (i.e., it includes not only the waterfall type but also the spiral type and other iterative types of development). The sheets used in this process can be the conventional documents used in functional safety development, such as the various design documents and test specifications/results shown in Figure 12, and no new sheets are created.


Hardware requirements definition
Hardware architecture design
Hardware detail design
Implementation
Hardware test
Hardware integration test
Hardware unit test
V-model

(a) Hardware development


System requirements definition
Safety requirements definition
System architecture design
Software requirements definition
Software architecture design
Software detail design
implementation
System test
Safety test
System integration test
Software test
Software integration test
Software unit test
V-model

(b) System and software development

Figure 12. V-model development for functional safety

6.4 Examples of development process and assessment sheet applications

The aforementioned quality assessment sheets were applied to various projects to demonstrate the *development process* and the *quality assessment sheets* that support it. In them, the sheets were used in each development process, as shown in Figure 13.

In this section, we will focus on the process of *dataset design and collection*, which is a part unique to AI, and provide an example of the procedure for describing a *dataset assessment sheet* for the case of application to the development of an intelligent wheelchair.

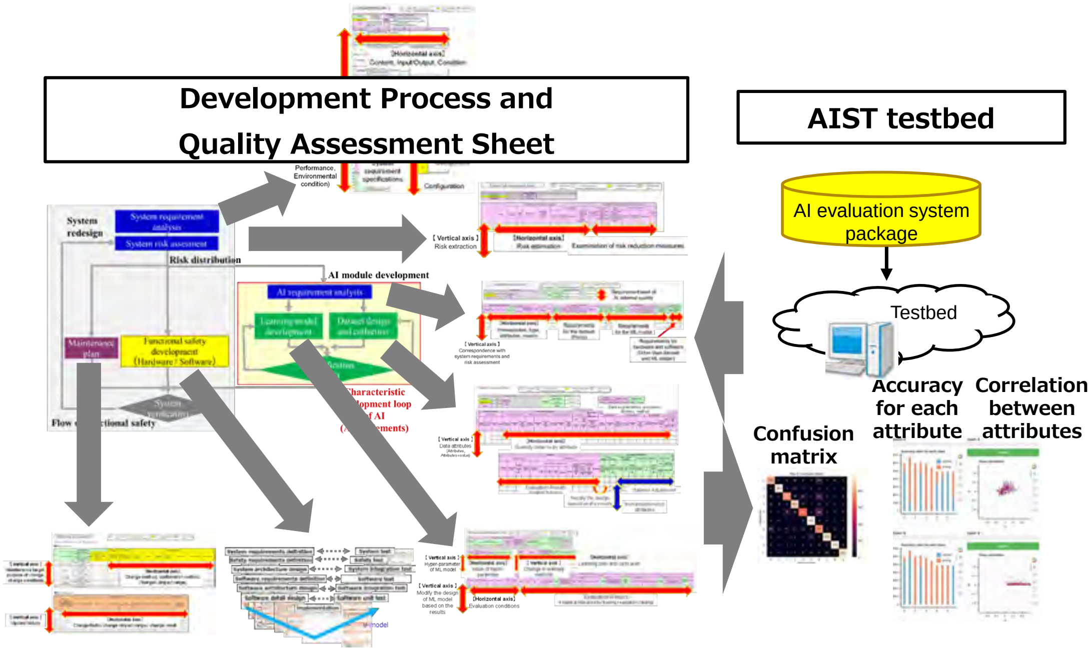

Development Process and Quality Assessment Sheet
AIST testbed
AI evaluation system package
Testbed
Confusion matrix
Accuracy for each attribute
Correlation between attributes
System redesign
System requirement analysis
System risk assessment
Risk distribution
AI module development
AI requirement analysis
Learning model development
Dataset design and collection
Maintenance plan
Functional safety development (Hardware / Software)
Flow of functional safety
System verification
【Horizontal axis】 Content, Input/Output, Condition
Performance, Environmental condition)
Configuration
【Vertical axis】 Risk extraction
【Horizontal axis】 Risk estimation
Examination of risk reduction measures
【Vertical axis】 Data attributes
【Vertical axis】 Hyper-parameter of ML model
【Horizontal axis】 Evaluation conditions

Figure 13. Development process and supporting quality assessment sheets

6.4.1 Details of AI to be applied

In this application example, we show a partial demonstration of the application of safety control to the development of an autonomous wheelchair-type robot (intelligent wheelchair: when not carrying a person, it functions as an autonomous mobile transport vehicle) that has image recognition functions using image sensors and AI, recognizes people around it, and avoids collisions. Figure 14 shows the appearance of the intelligent wheelchair to which it was applied.

An intelligent wheelchair shall use existing AI datasets and training models for human detection, as shown in Table 1.

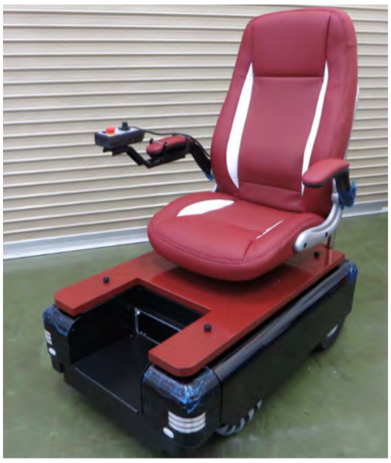


Figure 14. Targeted intelligent wheelchair (2019 prototype)

Table 1. Configuring AI in an Intelligent Wheelchair

Item	**Content**
Dataset	COCO dataset (2014)
ML model	YOLOv3　(Darknet 53) Trained machine learning model（yolov3.weight）

The COCO dataset used as a dataset contains image data and its metadata, which is labeling information (indicating the type of object in the image and the 2D position of the bounding box, which is a frame indicating the target region in the image). The architecture of the learning model, YOLOv3, has the ability to detect objects in 80 different categories (classes), and the "human" detected by the intelligent wheelchair is one of the objects to be detected. In the intelligent wheelchair of this chapter, the pre-trained machine learning models pre-trained by these datasets and learning model architectures were used for actual human detection, and the datasets and learning models were built with these initial values.

6.4.2 System requirement analysis

In this section, we proceed to apply the focus on the data set, and first analyze the requirements of the whole system using the *system requirements analysis form* shown in Figure 4. The functional requirements of the intelligent wheelchair system and the non-functional requirements, namely performance and environmental conditions, are described. In particular, environmental conditions such as the movement of the robot itself, brightness, etc., which affect the acquisition of the camera image that serves as the decision data for recognition, as well as the characteristics of obstacles and people that may appear in the image, should be clarified here. The analysis also mentions the operation of the system, which is also related to the distribution of risks, to visualize the requirements for AI in the end. This analysis identifies the technical points that should be considered in systemizing not only the AI designers and developers, but also the system designers and other technical personnel, in response to the demands and scenes on the user side. By conducting the analysis and system conceptual design with these perspectives in mind, it is possible to smoothly transition to the subsequent processes.

Since what is shown in this section is an example of a quality assessment sheet, the scope is limited to scenarios of scenes narrowed down from commonly used use cases.

6.4.3 System risk assessment

In this process, risks are extracted and methods to deal with the risks are examined for the system envisioned based on the system requirements analysis, using the *system risk assessment form* shown in Figure 5. The details are omitted here because the same considerations are made as in general risk assessment, but the point is that the risks to be addressed by AI and their levels need to be fully considered. Risk reduction measures should not be assigned to AI without sufficient consideration, and risk reduction should be carried out appropriately, such as from inherently safe design to safety design by control, in accordance with the three-step method that is the basis of safety design described in ISO/IEC Guide 51 and ISO 12100. Moreover, we limit the risk reduction to what only AI can do so that the risk that AI takes at the end of the design is reduced to an acceptable level. This is because risk avoidance by AI is not yet at a reliable level with the current technology, and the maximum level is AISL 0.2, which is lower than SIL 1 of actual functional safety.

As in the case of system requirements analysis, this risk assessment is also conducted by limiting the target to a narrowly selected scene/scenario.

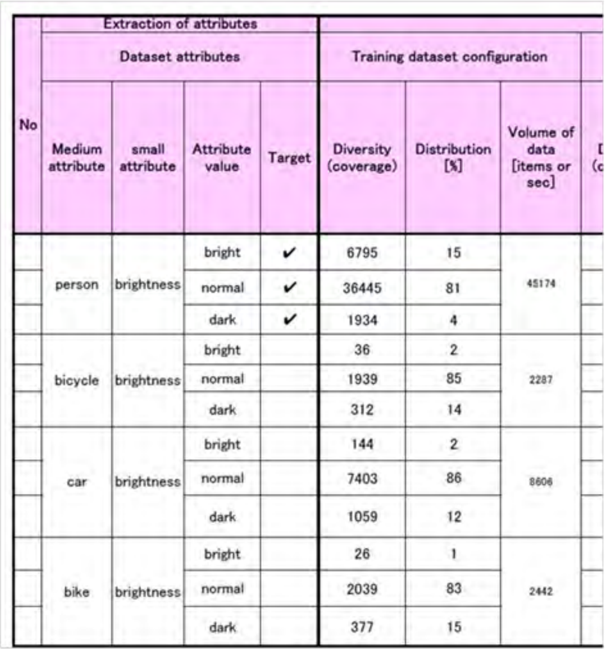
6.4.4 AI requirement assessment

Based on the results of the system risk assessment, we analyzed the requirements for AI with regard to the content of risk reduction measures for AI using the *AI requirements analysis sheet* shown in Figure 7. In this section, we clarify the "attributes" that should be recognized by AI, and classify "person" and other objects to be recognized from the acquired images. First of all, we identify the objects to be recognized and the related attributes to the objects based on the contents of the requirements, which are linked to the scenes in the system requirements analysis. Then, while considering the impact of those attributes on risk, we examine, narrow down, or further classify them to be adopted as AI attributes. The point here is to identify as many attributes as possible, so that if they are not candidates at first, but are not well recognized during the actual development, those items can be reconsidered as attributes.

In AI requirements analysis, data collection and design policies and priorities are first determined for the attributes of the data to be extracted, and then the details of the internal quality of the data are confirmed in the subsequent dataset assessment.

6.4.5 Dataset design and collection

In the process of dataset design and collection, datasets were designed and collected using the *dataset assessment form* shown in Figure 8 based on the attributes extracted in the AI requirement analysis. However, since this chapter uses a dataset that has already been given, the dataset "COCO dataset (2014)" was analyzed using the attributes extracted in the AI requirement analysis.

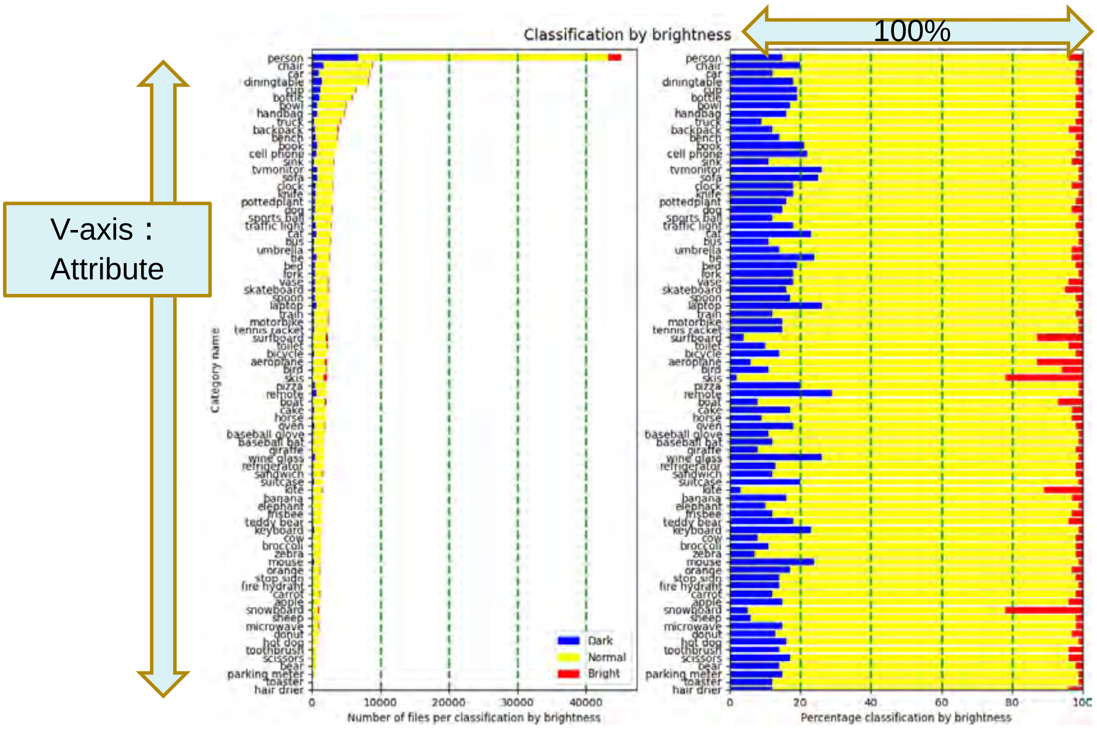


No	Extraction of attributes							
	Dataset attributes				Training dataset configuration			
	Medium attribute	small attribute	Attribute value	Target	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	[(c
	person	brightness	bright	✔	6795	15	45174	
			normal	✔	36445	81		
			dark	✔	1934	4		
	bicycle	brightness	bright		36	2	2287	
			normal		1939	85		
			dark		312	14		
	car	brightness	bright		144	2	8606	
			normal		7403	86		
			dark		1059	12		
	bike	brightness	bright		26	1	2442	
			normal		2039	83		
			dark		377	15		

Figure 15. Examples of data attributes and attribute values in datasets

In this section, we first analyzed the dataset by filling in the attribute values as shown in Figure 15 based on the results of AI requirement analysis for attributes. Figure 16 is a partial example of that analysis, showing the number of data for each attribute (category) in descending order for 82081 training data in the COCO dataset, the number of data for attribute values (eg. brightness) within each attribute (Figure 16 (a)), and the relative distribution of attribute values within attributes in that attribute order (Figure 16 (b)).

However, the small attribute shown in Figure 15 is an example of the viewpoint of attributes when classifying the middle attribute hierarchically, and the small attribute also changes according to the middle attribute. For example, when the middle attribute is "person", "age", "posture", and "body part" are considered as a kind in addition to "brightness" as a small attribute. When the middle attribute is "bicycle", "shape" and "color" are considered as small attributes.



(a) Absolute data amount (b) Relative data amount ratio

Figure 16. Distribution of data amount of attribute value (brightness) for each attribute (object) of training data (COCO datasets for YOLOv3)

An important part of this process is to consider the attributes from various viewpoints through the use of the quality assessment sheet, to identify the characteristics of the dataset needed for the purpose, and to consider the selection of these characteristics. In addition, by recording the results of the examination, two internal qualities (i.e., *Coverage of dataset* in terms of whether there are any data omissions for attributes, and *Uniformity of dataset* in terms of checking the balance of data distribution) can be quantitatively visualized and used as material for evidence.

Furthermore, the results can be used for subsequent quality improvement activities, such as analysis of defects and identification of points for improvement, in comparison with the results of repeated tests.

On the other hand, by confirming the internal quality *Adequacy of data* regarding the data, it is also possible to reduce the risk of labeling errors and other defects that appear when validating machine learning models during data design and collection.

6.5 Summary

In this Chapter, we have introduced the development process we have examined for AI quality management and the use of quality assessment sheets developed to support the promotion of the development process. As for the sheets, the formats illustrated in this chapter should not be considered as fixed but will be updated in the future through the reflection of knowledge and new technologies.

7 Autonomous driving vehicle

AI module for object detection and scene classification for autonomous driving vehicle.

7.1 Overview

An AI product/service development team has received some business requirements for building an AI module for autonomous driving cars from a certain car manufacturing company. The business requirement document (BRD) received by the development team is presented in Section 7.2. Next, in Section 7.3, first a preliminary analysis of the problem based on the received BRD is done. Next, a Proof-Of-Concept phase is performed before planning the prototype development stage. Finally, it is demonstrated how MLQM Guideline's internal quality characteristics can be explored to evaluate the quality of the product throughout the process.

Section 7.2 presents a hypothetical BRD to exemplify how to incorporate MLQM Guideline while preparing BRDs or similar documents and how the incorporation can benefit the overall process.

The following sections discuss the entire development process of the product/service with a special focus on evaluating quality standards in each phase. These sections are the key for understanding how to implement MLQM Guideline for evaluation and management of quality characteristics from planning through development.

- Section 7.3 translates the BRD presented in the preceding section and identifies key technical specifications of the product from the developers' perspective. It includes safety and performance related specifications.
- Section 7.4 describes the Proof-of-Concept phase. It includes initial investigation of the existing datasets and approaches for solving the tasks. Preliminary analysis of data and preliminary training results are discussed preceding insights gained from the PoC phase.
- Section 7.5 explores and discusses development stage along with evaluation of internal quality characteristic axes mentioned in MLQM Guideline.
- Section 7.6 presents a glossary for this example.

7.2 Business requirements

7.2.1 Product name

AI module assigned for performing scene classification and object detection to be used for an autonomous driving vehicle.

7.2.2 Use cases

The AI module is expected to be used for the following cases[1]:

(a) Case 1: Identification and localization of regularly encountered objects in driving scenarios,

(b) Case 2: Recognition of standard environmental conditions (i.e., weather, road type, traffic signal etc.) for a semi-autonomous vehicle

7.2.3 Background

An automotive manufacturing company wants to incorporate some features of autonomous driving in one of its vehicles. The vehicle will have a monocular camera attached to the front which captures continuous video footage of the surrounding environment. Upon receiving extracted frames from the video stream, an AI module is expected to identify and localize objects that are frequently encountered in driving scenarios. The module should also assess environmental conditions related to weather, traffic, and road conditions.

According to SAE (Society of Automotive Engineers) International J3016 “Levels of Driving Automation” standard, the target level of automation SAE level 1. This implies the autonomous vehicle will provide driving assistance for adaptive cruise control feature: steering the vehicle keeping it at a safe distance from the next car but human monitoring is present.

At this stage, the AI module will only be used to improve the driving experience for the driver. If the identified combination of environmental conditions matches with some pre-defined safety-critical scenario (i.e., traffic signal is green but pedestrian on zebra-crossing), an alarm system will provide warnings and the driver will act accordingly to avoid accidents.

The manufacturer agreed to see first what can be achieved in developing the AI using publicly available videos.

[1] Ideally a use case should show not just the system’s purpose but also its input, processing, and output to clarify how it interacts with the target.

7.2.4 Purpose/objectives

Improving driving experience by constantly supervising driving scenarios.

Reducing the possibility of accidents by raising alarms in safety-critical scenarios.

Initializing autonomous driving features in regular vehicles as a first step towards full automation.

7.2.5 Stakeholders of the product

Stakeholders of the product (considered for this Guide) are:

- Car manufacturers
- Customers (drivers/public transport operators/private vehicle owners)

7.2.6 Initial demands of the stakeholders

- Car manufacturers and customers demand that safety of the users and the vehicles should be the highest priority while developing this module.
- Car manufacturers expressed that they want to deploy such a module that will be compatible with the current hardware requirements.
- Along with safety, customers want the AI module interface to be user-friendly and wish to receive maximum functionality at both lower and higher speed limits.
- The highest concern for this product is the safety of the users and the vehicles. The AI module should be able to effectively identify safety-critical incidents in real time and can assist in avoiding life-threatening situations.
- It is recommended that the module should be able to identify objects in regular environments (i.e., sunny, and clear weather) as well as in less encountered environments (i.e., rainy, or snowy weather). If the module fails to provide similar performance in less encountered driving situations, it may be prone to more accidents.

7.2.7 Details of the business requirements

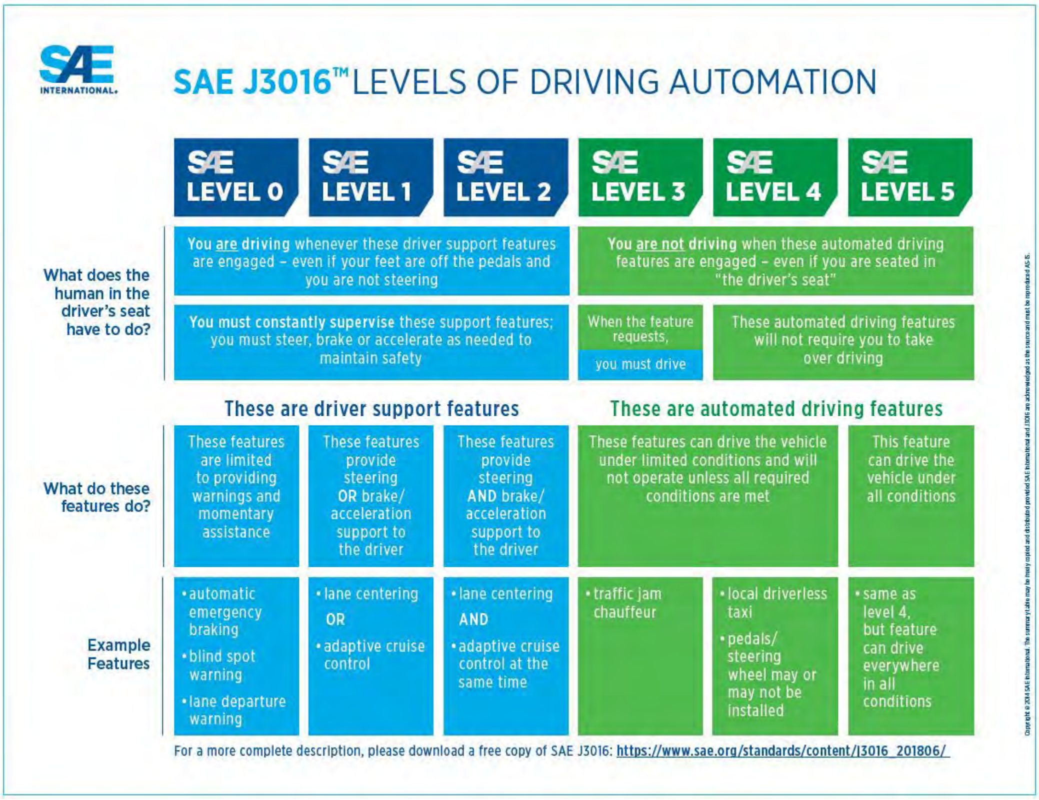
Business requirements for the product to be developed are discussed in detail below:

Functional requirements

Functional requirements of the final AI module are given below:

- It will identify the current driving scenario with features including (but not limited to) weather conditions, road conditions etc.
- It will detect regularly encountered objects in driving situations including (but not limited to) human/pedestrians, cars, traffic lights, traffic signs, buses, bikes etc. in all possible climate conditions, traffic and road conditions, time zones and lighting conditions.

Additional information: The level of automation for the entire vehicle will be SAE level 1 [2] (Figure 17). The AI module's object detection and scene classification features will be used by the vehicle to check if the driving scenario is safe enough for adaptive cruise control or to check occurrence of safety-critical scenarios that needs driver's immediate attention.



SAE INTERNATIONAL

SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/ acceleration support to the driver	These features provide steering AND brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering OR • adaptive cruise control	• lane centering AND • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

For a more complete description, please download a free copy of SAE J3016: https://www.sae.org/standards/content/J3016_201806/

Figure 17. SAE levels of driving automation

Non-functional requirements

Non-functional requirements of the AI model are given below:

- The AI module should maintain the required level of accuracy with a minimum performance speed in terms of FPS.
- The AI module should be most calibrated with the traffic rules and conditions of US urban & suburban areas where the autonomous vehicles will operate.
- AI module will be able to operate well at least at a relatively lower speed range.
- The object detection and scene classification features should be robust also in rare driving scenarios.
- The AI module should be user-friendly with a straightforward interface.

Assumptions

- Examples of a large variety of possible combinations of driving scenarios, weather and road conditions are present in the publicly available driving videos

- Lighting condition is appropriate enough to recognize all objects present in the images by humans.
- Possible accident-prone/safety-critical scenarios where the AI module is supposed to raise warning are present in the publicly available driving videos.

Dependencies

- **During operation:** A monocular camera will capture continuous video of surrounding environment and another system will simultaneously extract frames from the video and send them to the AI module as input images. Malfunction of these systems will result in failure of the AI module. Accurate synchronization of these systems must be checked and maintained regularly. The camera is installed inside the vehicle so that it would not be smudged with dirt or dew easily, and its sight must be kept clear by regular maintenance.

Constraints

- Images collected from the publicly available video driving datasets must be the primary source for building the AI module.

Out-of-scope issues

- **Functionality:** The functions of this AI module are only limited to object detection and scene classification. Other autonomous features like alert raising in safety-critical scenarios or adaptive cruise control in safe scenario will be handled by other individual modules of the vehicle. Those modules will only receive output about the surrounding environment from the AI module to be developed.
- **Driving area/location:** The AI module will be trained using driving footage of specific areas as mentioned previously. Traffic rules and driving conditions might be different around the world. Conditions in wild areas without road such as mountains and sea shore are completely different. Areas where traffic rules and driving conditions are significantly different than the areas mentioned will be considered out of scope for the AI module. Before deployment of vehicles with this AI module to any other area, retraining the module for proper calibration is recommended.
- **Chronological changes:** Traffic rules and driving conditions may change a lot over years or decades. This AI module is not supposed to be used after traffic rules and driving conditions change significantly from the driving footage used to train it.
- **Speed limit:** At higher speeds, performance of the module may degrade. So, vehicles moving at higher speed are not recommended to use the module for

assistance purposes[2].

Risk and safety related concerns

- In case of failure of the AI product, there are risks of human-life threatening incidents and/or economic loss causing incidents. Precautions and measures should be taken throughout the development process to mitigate such risks. For example, low visibility in some conditions (i.e., rainy/foggy weather) may adversely affect the performance of the AI models and cause failure of the module. Identifying and minimizing such risks should be a priority in the development process.
- Safety of human life (both drivers and pedestrians) should be prioritized most in all driving conditions over destruction of property (both the vehicle and surrounding environment) and economic loss. However, unreasonable collisions should be monitored and must be avoided.

7.2.8 Requirements concerning external qualities

The expected level of quality requirements in terms of three major external qualities for the AI software to be developed are given below:

Safety

- According to ASIL (Automotive Safety Integrity Level), the preferred level of safety for the AI module is ASIL D[3]. ASIL D represents potential for severely life-threatening or fatal injury in the event of a malfunction and requires the highest level of assurance that the dependent safety goals are sufficient and have been achieved.

Note: In terms of Severity, Exposure and Controllability, ASIL D is defined as an event having reasonable possibility of causing a life-threatening (survival uncertain) or fatal injury, with the injury being physically possible in most operating conditions, and with little chance the driver can do something to prevent the injury. That is, ASIL D is the combination of Severity Level 3 (S3: Life-threatening (survival uncertain) to fatal injuries), Exposure level 4 (E4: High probability-injury could happen under most operating conditions), and Control level 3 (C3: Difficult to control or uncontrollable)

[2] In a real business situation, such a use should be clearly banned instead of just not being recommended. It does not, however, belong to constraints shown here, which limit actions of the developer, not the system's usage.

[3] The safety level the AI module must attain is much lower than ASIL D because such high level of safety cannot be achieved by the AI module discussed here alone and must be ensured by components other than the module.

Performance

- The final AI system should satisfy agreed upon thresholds of the KPI measures by the stakeholders and the developers. Any deviation should be reported and explained thoroughly.
- Balance in accuracy, precision and robustness to special scenarios is expected. AI models that are highly accurate in general driving cases but have poor robustness to less encountered, accident-prone driving scenarios are discouraged.
- The product must satisfy the expected level of performance for pre-identified safety-critical scenarios.

Fairness

- There are no identifiable requirements for fairness of the product or service[4].

7.2.9 Defining the levels of external quality

The identified levels of external qualities to be ensured in the final product using MLQM Guideline are the following:

Table 2. The levels of external qualities to be realized.

External Quality	Additional specification	Assumed severity	Realized level
Safety[5]	AI Safety Level for human-related risks	Severe injury; possible to avoid through monitoring by humans	AISL 4
	AI Safety Level for economic risks	Considerable; possible to avoid through monitoring by humans	AISL 4
Performance	AI Performance Level in general	KPIs (Key Performance Indicator) will be identified beforehand but thresholds for each KPI may deviate slightly based on other factors and best efforts will be provided	AIPL 2
	AI Performance Level in pre-identified safety-	Required thresholds for the KPIs must be achieved and must not deviate from the reported values	AIPL 2

[4] Indeed, fairness of this product matters. Object detection accuracy must not depend on the target person's age, sex, race, nor whether in a wheelchair or with prostheses. See, however, Section 1.5.3 of the Guideline for whether that should be a fairness requirement.

[5] It is more appropriate to set AISL as less than 1 because the safety level to be achieved by the AI module is low (see footnote 3).

	critical scenarios		
Fairness	AI Fairness Level in general	No identifiable requirements for fairness of the product or service.	AIFL 0

7.2.10 Conclusion

In this section, a hypothetical business requirement document is presented for an AI based product that will perform object detection and scene classification for a car and is expected to achieve some preliminary level of automation.

After setting the requirements, it is also demonstrated how MLQM Guideline can be incorporated in typical business requirement documents. We expect it will assist AI service providing entities to express their expected quality goals more explicitly.

7.3 Preliminary analysis of the problem

This section describes important specifications derived after analyzing the functional and nonfunctional requirements of the AI solution.

7.3.1 Technical specifications

- The final product will consist of two separate machine learning (ML) models.
- Type of learning for the models will be supervised learning.
- The tasks performed by the ML models are scene understanding/scene classification and object detection from images.
- Both the classification model and the object detection model will be sharing the same dataset.
- Since the two modules have specific responsibilities, it must be confirmed that the defined quality standards satisfy requirements related to both tasks.
- The final product is expected to correctly recognize objects regularly encountered in autonomous driving scenarios in all possible climate conditions, traffic and road conditions, time zones and lighting conditions.
- Contender models can include (but not limited to) the following architectures:

  Task 1: Object detection

  - YOLOv3 [3]
  - YOLOv3 + ASFF [4]
  - YOLOv4 [5]
  - YOLOv5 [6]
  - Faster R-CNN [7]

    - M2Det [8]
    - EfficientDet-D2 [9]
    - MobileNetv1 [10]
    - MobileNetv2 [11]

Task 2: Scene classification

    - VGG19 [12]
    - ResNet50 [13]
    - InceptionV3 [14] [15]

Publicly available datasets for autonomous driving tasks (i.e., BDD100, NuScenes, KITTY etc.) will be surveyed and one or combined datasets will be chosen according to suitability.

7.3.2 Safety specifications

According to Automotive Safety Integrity Levels (ASIL) [15], ASIL D is required safety level of the final product. ASIL D is the combination of S3, E4, and C3 where:

- Severity Classifications (S):
    - S0 No Injuries
    - S1 Light to moderate injuries
    - S2 Severe to life-threatening (survival probable) injuries
    - S3 Life-threatening (survival uncertain) to fatal injuries
- Exposure Classifications (E):
    - E0 Incredibly unlikely
    - E1 Very low probability (injury could happen only in rare operating conditions)
    - E2 Low probability
    - E3 Medium probability
    - E4 High probability (injury could happen under most operating conditions)
- Controllability Classifications (C):
    - C0 Controllable in general
    - C1 Simply controllable
    - C2 Normally controllable (most drivers could act to prevent injury)
    - C3 Difficult to control or uncontrollable

In the BRD, along with ASIL quality standards, corresponding quality levels according to MLQM Guideline are also reported. As per requirement, AISL 4 will be considered as the required level of safety. In addition to these, AIPL (AI performance level) and AIFL (AI fairness level) are chosen to be AIPL2 and AIFL0 respectively.

7.3.3 KPI specifications

Key Performance Indicator (KPI) quantifies the attainment level of functional requirements to be attained by output from machine learning components through machine learning based systems. For the two major ML models involved in this examples, initial KPI specifications are given below:

Object detection task

Object detection algorithms are usually evaluated based on metrics such as the mean average precision (mAP) and the F1-score. Regarding the detection ability of an AI model, mAP is considered as the suitable candidate since it is defined as the area under the precision-recall curve computed for a certain intersection over union (IoU) threshold. In other words, mAP involves precision, recall, and IoU, which makes it an attractive choice regarding a measure of the detection strength of a given model.

Scene classification task

Classification models are usually evaluated based on metrics such as confusion matrix, accuracy, precision, recall, specificity, F1 score. ***Accuracy*** should be considered as the KPI when attribute values for a certain feature/attribute are well-balanced. For unbalanced classes, precision, recall or F1 score should be considered to evaluate model's performance. In case of AI application for autonomous vehicles where safety-critical cases should be a prime concern, use of precision and recall can give us more detail information about the model's performance in high-risk cases with respect to false positives and false negatives. So, ***F1 score*** is also considered as KPI where class labels are not well-balanced and while dealing with high-risk cases.

7.4 Proof of Concept (PoC) phase

7.4.1 Initial investigation of existing datasets

For the creation of AI models for the classification task and object detection task explained above, diverse datasets have been considered. There are a lot of datasets related to autonomous driving that are publicly available, but among those, following datasets are used in primary investigation:

Table 3. Autonomous Driving Dataset Comparison

Dataset	# Labels	# Images	has weather	has time of day	3d bounding box	2d bounding box
BDD100k	10	100000	Yes	Yes	No	Yes
CityScapes	30	5000	No	No	No	Yes

Kitti	14	14999	No	No	No	Yes
Semantic Kitti	28	14999	No	No	No	No
Audi A2D2 (segmentic segmentation)	38	41280	No	Yes (timestamp)	Yes	Yes
Audi A2D2 (bounding boxes)	14	12499	No	Yes (timestamp)	Yes	Yes
PandaSet	28	48000	No	Yes (timestamp)	Yes	Yes
NuScenes	23	1400000	No	Yes (timestamp)	Yes	No
Apolloscape	25	146997	No	Yes (timestamp)	Yes	No
Canadian Adverse Driving Conditions Dataset	10	56000	No	Yes (timestamp)	Yes	No
Waymo Open Dataset	4	200000	No	No	Yes	Yes
Lyft Perception Dataset	23	450000	No	Yes (timestamp)	Yes	No
Oxford Robotcar Dataset	-	20000000	Yes	Yes	No	No

As per the given requirement of final product, the scene classification model should be able to detect weather from the input images. Only BDD100k and Oxford Robotcar Dataset accomplish these requisites. However, Oxford Robotcar Dataset does not include 2D bounding box labeling in their images, which is a requirement for the object detection task. Since both scene classification and object detection model will be sharing the same dataset, the best suitable dataset for the product seems to be BDD100k [16] with its GitHub repository [17].

7.4.2 Introduction of the chosen dataset

The chosen dataset is known as ***BDD100K dataset*** which is a large-scale video dataset. The original dataset contains annotations of images for various purposes such as road object detection, lane marking, drivable area, etc. BDD100k has 100,000 of images in total. The official dataset has 3 splits for training, testing and validation using 70%, 20% and 10% of the dataset respectively. The default splitting given by the dataset authors will be used initially. Following information are collected from an initial investigation of the dataset:

(a) General Information:

- Given attributes and attributes values for *classification* task:
  - Weather: rainy, snowy, clear, overcast, partly cloudy, foggy, undefined
  - Scene: tunnel, residential, parking lot, city street, gas stations, highway, undefined
  - Time of day: daytime, night, dawn/dusk, undefined
- This dataset has 10 labels to identify objects: Bus, Light, Sign, Person, Bike,

truck, Motor, Car, Rider, Train[6]

- Additional information on: Occlusion, Truncation, Traffic light color, Lane direction, Lane style and Lane type[7]

(b) Information of training set:

- Number of total images: 70,000
- Number of images with information in .json file: 69,863
- Number of attributes: 3 (weather, scene, time of day)
- Number of categories in each attribute: (excluding undefined)
    - weather: 6
    - scene: 6
    - time of day: 3

(c) Information of validation set:

- Number of total images: 10,000
- Number of images with information in .json file: 10,000
- Number of attributes: 3 (weather, scene, time of day)
- Number of categories in each attribute: (excluding undefined)
    - weather: 6
    - scene: 6
    - time of day: 3

(d) Additional information:

- Occluded: True, False
- Truncated: True, False
- Lane direction: Parallel, Vertical
- Lane style: Solid, Dashed
- Lane types: Crosswalk, Double other, Double white, Double yellow, Road curb, Single other, Single white, Single yellow[8]
- Traffic light color: Red, Green, Yellow, None
- Area type: Alternative, Direct

7.4.3 Distribution of data

Basic distribution analysis for the chosen dataset is given below:

[6] Due to this limited variety of labels, the trained model will not be able to detect objects accidentally lying on the road and will not achieve its safety goal.

[7] Similarly, the lack of uphill and downhill, which significantly constraint the camera view, will result in severe hinderance to achieving it safety goal.

[8] This list shows just lane markings and ignores other road markings such as hatched, which hinders the trained model to achieve its safety goal.

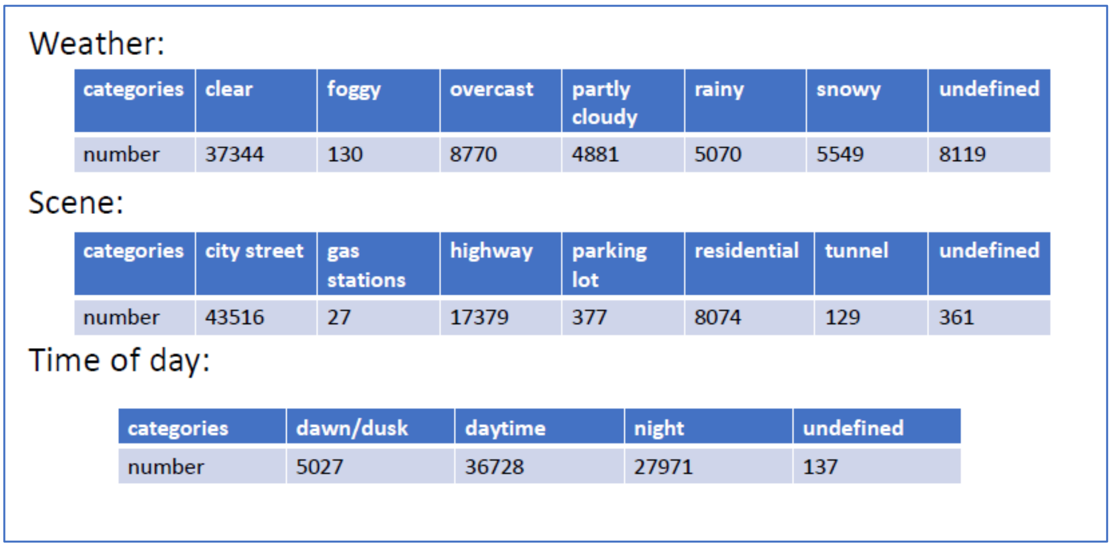
Classification task

For the classification task using BDD100k, the distribution of the attribute values of *Scene*, *weather* and *time of day is given below.*

Training dataset statistics

- Percentage of defined information in each attribute:
  - weather: 61774/69863 (88.42%)
  - scene: 69502/69863 (99.48%)
  - time of day: 69726/69863 (99.8%)
- Number of images having at least one attribute as 'undefined': 8389

Table 4. Training Dataset Statistics for BDD 100k Classification Task



Weather:

categories	clear	foggy	overcast	partly cloudy	rainy	snowy	undefined
number	37344	130	8770	4881	5070	5549	8119

Scene:

categories	city street	gas stations	highway	parking lot	residential	tunnel	undefined
number	43516	27	17379	377	8074	129	361

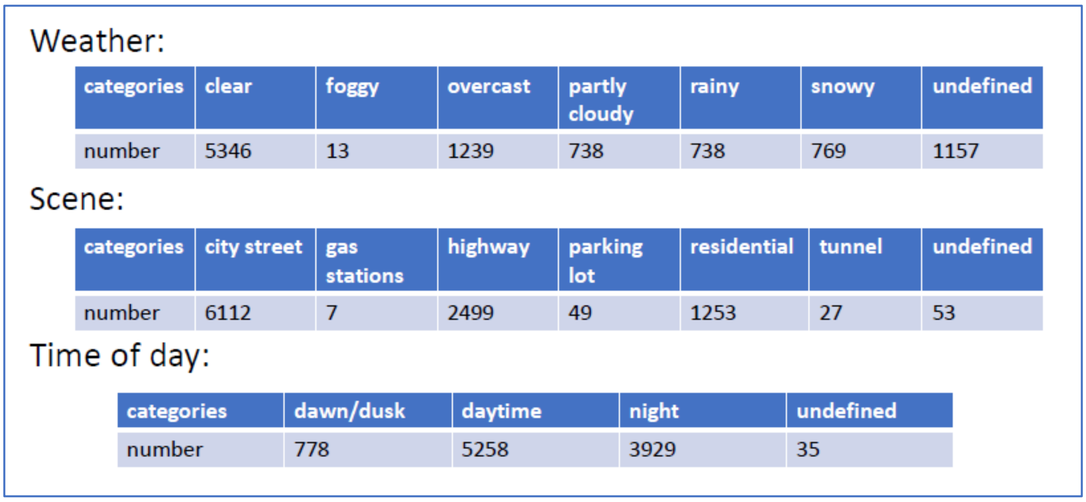
Time of day:

categories	dawn/dusk	daytime	night	undefined
number	5027	36728	27971	137

Validation dataset statistics

- Percentage of defined information in each attribute:
  - weather: 8843/10000 (88.43%)
  - scene: 9947/10000 (99.48%)
  - time of day: 9965/10000 (99.8%)
- Number of images having at least one attribute as 'undefined': 1199

Table 5. Validation Dataset Statistics for BDD 100k Classification Task



Weather:

categories	clear	foggy	overcast	partly cloudy	rainy	snowy	undefined
number	5346	13	1239	738	738	769	1157

Scene:

categories	city street	gas stations	highway	parking lot	residential	tunnel	undefined
number	6112	7	2499	49	1253	27	53

Time of day:

categories	dawn/dusk	daytime	night	undefined
number	778	5258	3929	35

It is evident from Table 4 and Table 5 that there are some attribute values that appear more often than others, such as, *daytime* for *time of day* and *clear* for *weather*. Also, there are attribute values that does not appear often in the dataset, such as *tunnel for* scene. Such attribute values may need a treatment if the number of images is needed to be increased.

Object detection task

This dataset has 100,000 images with the 10 labels: *Train, Motor, Rider, Bike, Bus, Truck, Person, Light, Sign* and *Car*. Table 6 and Figure 18 shows how many times each label appears in the whole dataset.

Table 6. Count of Appearances for Each Object in Training and Validation Images

Label	Training	Validation	Label	Training	Validation
train	136	15	truck	29971	4245
motor	3002	452	person	91349	13262
rider	4517	649	traffic light	186117	26885
bike	7210	1007	traffic sign	239686	34908
bus	11672	1597	car	713211	102506

It is evident that BDD100k has a lot of *car* instances in comparison with other labels. This lack of uniformity[9] in the dataset is encountered as a problem while training the detection models. The main issue was that the detection models were overfitted for the *car* instances and could not recognize properly the other labels. Different solutions were created to solve this issue and will be explained in the upcoming sections.

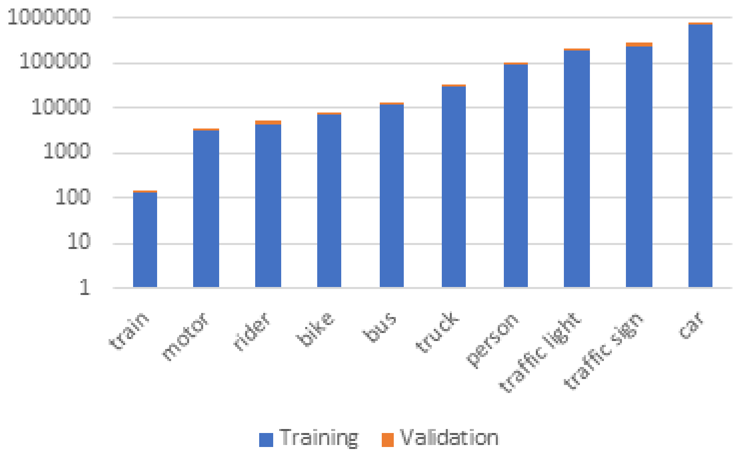


Figure 18. Distribution of Appearances for Each Object in Training and Validation Images

7.4.4 Preliminary training of contender models

The goal of PoC phase is not improving performance, rather it is to check if available data can be directly used in the contender models. The following preprocessing step was taken:

- Images are auto resized to fit by corresponding networks, otherwise no resizing was done.
- Bounding box(bb) annotations for BDD100k were originally in the following format: x and y co-ordinates of the top left and x and y co-ordinates of the bottom right edge of the rectangle. This format was changed to COCO bb annotation format: (x-top left, y-top left, width, height)

After training and validating the model on the entire BDD100k dataset, object detection accuracy has been measured by the overall mAP on the validation dataset. The classification models have been trained using randomly selected 20k images from the BDD100k training dataset for faster training and performance has been evaluated on the validation set based on classification accuracy. In the actual development, entire

[9] In the Guideline, uniformity of datasets means the distribution of a dataset matches the real-world distribution. The author of this chapter, however, uses uniformity to mean small variation in data amount across labels.

dataset can be used. The model proceeds to the development phase for further evaluation and analysis only if the accuracy meets or exceeds the threshold set by the implementing party. The PoC phase will be conducted using the original dataset in hand and pretrained weights for the available state of the art models.

Validation results of classification task

Results shown in Table 7 were obtained from the trained classification models evaluated on the validation dataset with 10k examples:

Table 7. Performance of Classification Models on Validation Set in PoC Phase

	Weather	Scene	Time of day
VGG19	63.34	67.64	91.44
ResNet50	65.35	67.97	91.16
Inceptionv3	71.40	69.86	91.23

Validation results of object detection task

Results shown in Table 8 were obtained from the trained object detection models evaluated on the validation dataset with 10k examples.

Table 8. Performance of Object Detection Models on Validation Set in PoC Phase

Model	mAP@0.5	FPS
YOLOv3	45.7	31.25
YOLOv3 + ASFF	56.55	63.69
YOLOv4	62.3	16.07
YOLOv5	62.9	29.06
Faster R-CNN	59.3	14.58
M2Det	7.2	13.7
EfficientDet-D2	41.2	19
MobileNetv1	79.6	5.3
MobileNetv2	84.5	4.5

Additional information

All the models used in PoC phase are open sourced with established architectures. Since no changes were made to the original architectures, details of each network such as number of layers and neurons, activation functions etc. are not documented in this report for this phase. Following are the pre-trained weights that were used for object detection models:

Table 9. Pretrained Weights Used for The Models in PoC Phase

Model	Weights (URL)
YOLOv3	https://pjreddie.com/media/files/darknet53.conv.74

YOLOv3 + ASFF	Randomized weights
YOLOv4	https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
YOLOv5	https://github.com/ultralytics/yolov5/releases/download/v3.0/yolov5x.pt
Faster R-CNN	https://dl.fbaipublicfiles.com/detectron2/COCO-Detection/faster_rcnn_R_50_FPN_3x/137849458/model_final_280758.pkl
M2Det	https://drive.google.com/file/d/1NM1UDdZnwHwiNDxhcP-nndaWj24m-90L/view
EfficientDet-D2	http://download.tensorflow.org/models/object_detection/tf2/20200711/efficientdet_d2_coco17_tpu-32.tar.gz
MobileNet v1	https://1drv.ms/u/s!AvkGtmrlCEhDhy1YqWPGTMl1ybee
MobileNet v2	https://storage.googleapis.com/mobilenet_v2/checkpoints/mobilenet_v2_1.4_224.tgz

7.4.5 Insights gained from PoC phase

PoC phase can give directions about steps to be taken to begin the development phase of the product. Some insights acquired in the PoC phase are discussed below:

1. **Necessity of creating well-defined problem domain:** Noteworthy differences are observed in available dataset domain attributes in Section 7.4 compared to the expected domain specifications mentioned in Section 7.2.7 in functional requirements. For example, the object detection models are expected to detect objects in all possible lighting conditions, but there is no such attribute in the given dataset that measures brightness/lighting condition of images. Such differences reveal that the existing attributes are not adequate for the domain to be referred as a complete problem domain with all possible combinations that may occur in real life scenarios. The necessity of creating a well-defined problem domain is noted.
2. **Effective balancing between adequate coverage and unbiased distribution:** Records from Section 7.4 reveals the distribution of samples is unbalanced; the dataset has some attribute values with very few examples compared to other attribute values in corresponding attribute (i.e foggy in weather, gas station in scene, train in objects etc). So, while re-defining the problem domain, necessity of merging or deleting attributes and attribute values should be kept in my mind. Also, the data in the newly defined domain should be distributed in a way that the practical/real life distribution of the data is not hampered. But if some

problem cases are found to be very important and critical for the ML models in terms of safety, then adequacy of data in those problem cases must be evaluated and adjusted. So, an effective balance between unbiased dataset distribution and enough examples in safety-critical cases should be exercised.

3. **Necessity of specifying special cases:** The above-mentioned observation also gives rise to the necessity of proper identification of problem cases with distinctive importance. Such as, there may be some combinations of attribute values that can never occur in real life scenarios (i.e., snow in summer). These scenarios should be considered as *impossible case*. Also, there may be some combinations of attribute values that have very few examples in dataset because they occur relatively rarely in real life but carries significant importance in terms of safety, such as pedestrian on road with green traffic signal. These should be identified as *safety-critical* scenarios or *rare cases*. Distribution analysis must be done to expose these cases so that cautious decisions can be taken about their inclusion in training dataset and expected level of performance for the trained models.
4. **Directions for development stage:** Records from Section 7.4 can give insights for next stage of product development such as deciding which models to choose for further improvement, threshold values to be set for KPIs, etc. But it also lacks records of models' performance in specific cases mentioned above. So, while evaluating model's performance in the next stage of development, KPI scores for the safety-critical cases must also be recorded to ensure if the model achieves required level of correctness and stability overall as well as in specific cases.

7.5 AI Development with internal quality evaluation

From the preliminary analysis and insights gained based on the results in the PoC phase, it is evident that the following steps are required to be executed in the upcoming phase of development:

- Creating a well-defined problem domain based on the requirements of the AI product
- Maintaining completeness of the problem domain while using available data in hand
- Designing training and test datasets with enough examples in possible scenarios while maintaining unbiased property of distribution as much as possible
- Identification of specific scenarios like rare cases, impossible cases, and specific distribution analysis of those cases

- Overall performance analysis of trained models as well as evaluation of model performance in safety-critical scenarios

In this section, it is demonstrated how the above requirements can be fulfilled using the *internal quality* aspects set by MLQM Guideline. It is expected that ensuring and evaluating these aspects of internal quality simultaneously throughout the agile development process will consequently create the best version of the AI product in terms of required quality, safety, performance, and fairness.

7.5.1 A-1: Sufficiency of problem domain analysis

The term *sufficiency of problem domain analysis* implies that sufficient requirement analysis is made concerning the situations where machine learning based systems are used in real world and their analysis results cover all possible situations as discussed in Section 6.1.1 of the Guideline.

The following discussion demonstrates procedures to ensure this quality during development. Highlights of this discussion are:

1. Important details of defining a sufficient problem domain
2. Proposed problem domain
3. Sample training data from proposed problem domain
4. Comparison between existing with proposed domain
5. Adaption of existing dataset to with proposed domain
6. Redesigned dataset for the next development stage

Important details of defining a sufficient problem domain

At first the features are analyzed in specific linguistic terms to distinctly define the problem domain of the autonomous driving scenarios that are expected to be encountered in real life. Inspired from the Guideline that envisions a concept of feature tree, the problem domain is defined with some *attributes* and their corresponding *attribute values*. The terms used here as attribute and attribute values carry the same meaning and significance as defined in Section 2.3.6 of the Guideline. Following points were kept in mind during development of the problem domain:

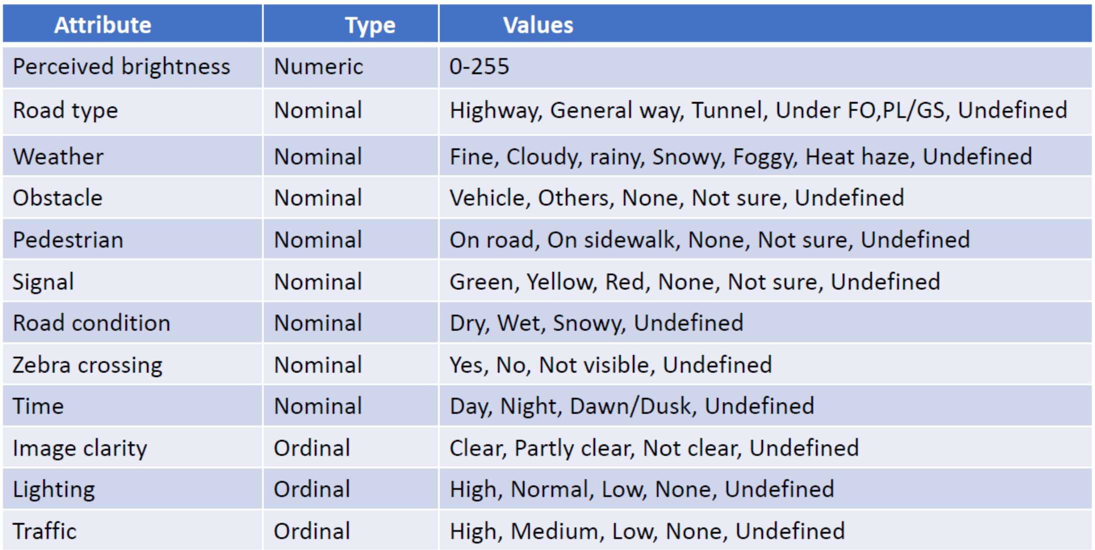
- Thorough investigation of requirements given by the client side as well as available labels of the existing data to include all possible scenarios as combination of attribute and their corresponding values
- Maintaining appropriate balance between too much detail and too little detail while defining the attribute values and their scopes.

Proposed problem domain

The major goal of creating an ideal problem domain is to cover all possible scenarios that an autonomous vehicle may face in real life and scenarios that concern the solution designer significantly.

Table 10 shows the attributes and the corresponding values used to identify such cases considering the initial product specifications, domain knowledge of autonomous driving and solution designer's concerns regarding safety.

Table 10. Problem Domain Specification



Attribute	Type	Values
Perceived brightness	Numeric	0-255
Road type	Nominal	Highway, General way, Tunnel, Under FO,PL/GS, Undefined
Weather	Nominal	Fine, Cloudy, rainy, Snowy, Foggy, Heat haze, Undefined
Obstacle	Nominal	Vehicle, Others, None, Not sure, Undefined
Pedestrian	Nominal	On road, On sidewalk, None, Not sure, Undefined
Signal	Nominal	Green, Yellow, Red, None, Not sure, Undefined
Road condition	Nominal	Dry, Wet, Snowy, Undefined
Zebra crossing	Nominal	Yes, No, Not visible, Undefined
Time	Nominal	Day, Night, Dawn/Dusk, Undefined
Image clarity	Ordinal	Clear, Partly clear, Not clear, Undefined
Lighting	Ordinal	High, Normal, Low, None, Undefined
Traffic	Ordinal	High, Medium, Low, None, Undefined

Here, *under FO* refers to images where the vehicle is under a flyover/bridge. *PL/GS* refers to parking lot/gas station. Due to the numeric property of the attribute *perceived brightness*, additional explanation of definition and calculation of this property is needed.

Calculation of luminance/perceived brightness: Luminance is a photometric measure of the luminous intensity per unit area of light travelling in a given direction. Brightness is the term for the subjective impression of the objective luminance measurement standard. A luminosity function or luminous efficiency function describes the average spectral sensitivity of human visual perception of brightness.

So, calculating the relative luminance of the images using the following formula is considered the most standard way of calculating perceived image brightness from pixel values:

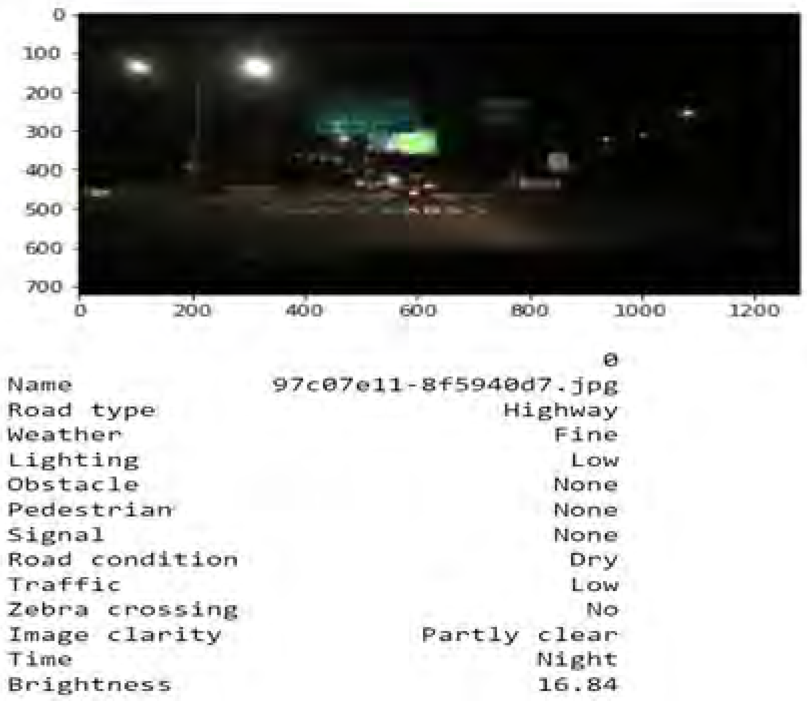
$$L=0.2126R+0.7152G+0.0722B$$

- R, G, and B are computed from sRGB values, standard values to express color in three channels, with a non-linear function. For a whole image, mean values of sRGB of all pixels in it could be used to compute R, G, and B.
- The range of brightness values is from 0 to 255, 0 being complete black and 255 being complete white.

Sample training data from proposed problem domain

The example presented in Figure 19 shows how each datapoint in the dataset can be described using the new attributes and their values.

The name of the image is from the original BDD100k dataset. The axes of the image show the original pixel size (1280x720) and attribute values describe a specific autonomous driving scenario encountered by the vehicle. This is pixel size for all the RGB images in the dataset.



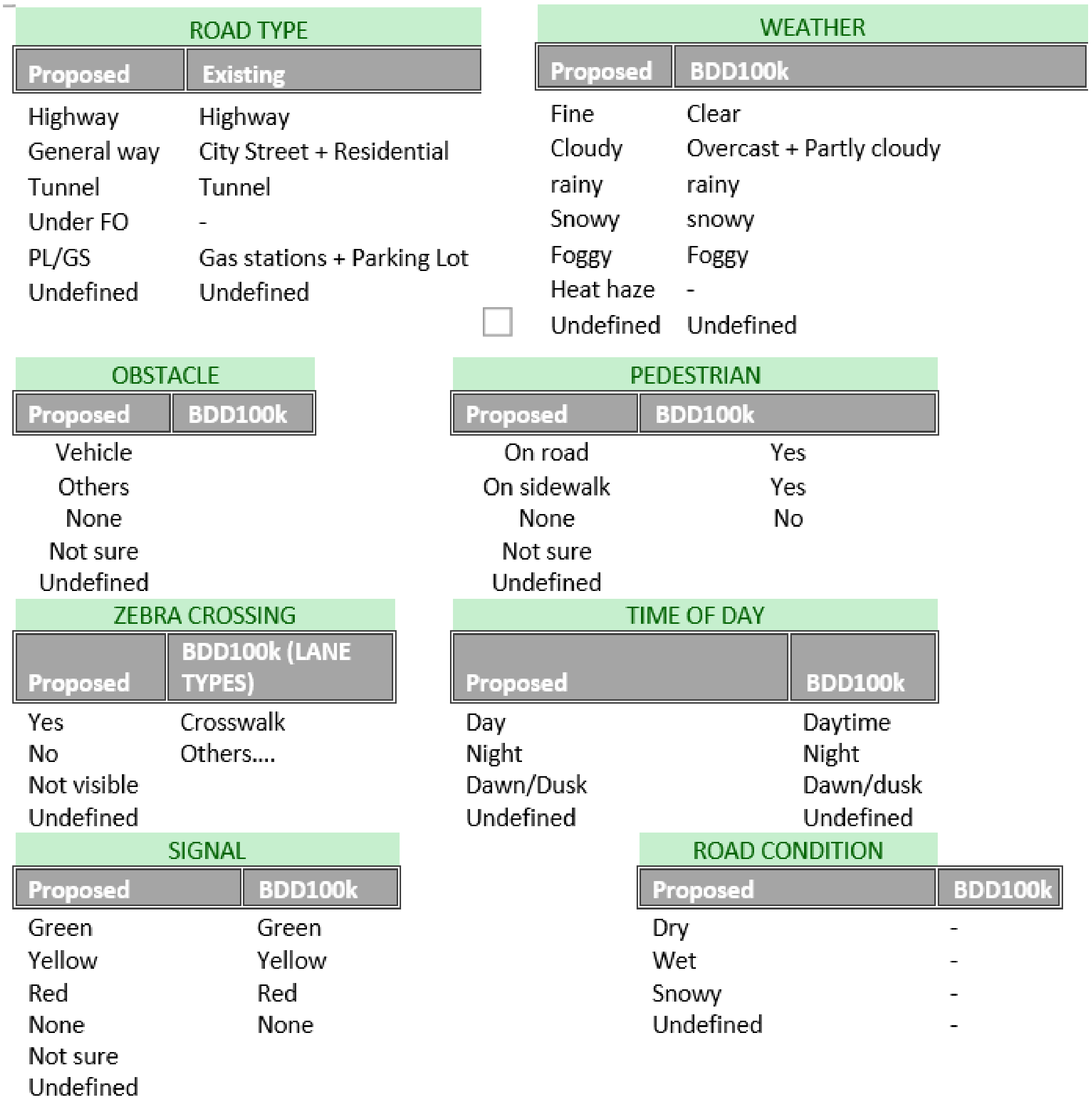
```
                                    0
Name               97c07e11-8f5940d7.jpg
Road type                        Highway
Weather                             Fine
Lighting                             Low
Obstacle                            None
Pedestrian                          None
Signal                              None
Road condition                       Dry
Traffic                              Low
Zebra crossing                        No
Image clarity              Partly clear
Time                               Night
Brightness                         16.84
```

Figure 19. Example Datapoint Described by the New Problem Domain

Comparison between existing dataset with proposed domain

A comparison of the existing problem domain of BDD100k and the proposed problem domain are given in Table 11 to observe their differences and to check if redesigning the dataset is necessary.

Table 11. Comparison of Problem Domain of BDD100k and the Proposed Problem Domain



ROAD TYPE

Proposed	Existing
Highway	Highway
General way	City Street + Residential
Tunnel	Tunnel
Under FO	-
PL/GS	Gas stations + Parking Lot
Undefined	Undefined

WEATHER

Proposed	BDD100k
Fine	Clear
Cloudy	Overcast + Partly cloudy
rainy	rainy
Snowy	snowy
Foggy	Foggy
Heat haze	-
Undefined	Undefined

OBSTACLE

Proposed	BDD100k
Vehicle	
Others	
None	
Not sure	
Undefined	

PEDESTRIAN

Proposed	BDD100k
On road	Yes
On sidewalk	Yes
None	No
Not sure	
Undefined	

ZEBRA CROSSING

Proposed	BDD100k (LANE TYPES)
Yes	Crosswalk
No	Others....
Not visible	
Undefined	

TIME OF DAY

Proposed	BDD100k
Day	Daytime
Night	Night
Dawn/Dusk	Dawn/dusk
Undefined	Undefined

SIGNAL

Proposed	BDD100k
Green	Green
Yellow	Yellow
Red	Red
None	None
Not sure	
Undefined	

ROAD CONDITION

Proposed	BDD100k
Dry	-
Wet	-
Snowy	-
Undefined	-

Adaption of existing dataset with proposed domain

This section summarized approaches taken for adaption of existing dataset with proposed domain:

- **Road Type**, **Weather** and **Time of Day**: both annotations used almost same terminology. Following merging/deletion of some attribute values can be done, so that the existing annotations of BDD100k can be easily adopted to the new domain.
  - o *City street* and *Residential* can be merged as *General way* in Road type
  - o *Overcast* and *Partly cloudy* can be merged as *Cloudy* in Weather. The level of cloudiness in the sky is left to be implicitly learned by the ML program.

  - *Heat haze* will be deleted since the BDD100k does not contain this attribute. The possibility of occurring this situation is very low and the solution designer decides to exclude this from the domain.
  - No change is needed to be made in Time of day.
- **Zebra Crossing**: it has been checked if the BDD100k image annotation has a *Crosswalk* label/description, but it was not present in BDD100k dataset. However, BDD100k has a description per Lane marking containing *Crosswalk*. So, following changes can be done using simple python scripts to extract from Lane marking description the information necessary to adapt the BDD100k dataset for the proposed domain.
  - If there is not a *Crosswalk*, then attribute value is assigned as *No*.
  - If there is a *Crosswalk*, then attribute value is assigned as *Yes*.
- **Signal**: there is a similar situation as with Zebra crossing attribute. The Signal attribute is a description inside Traffic light label annotated in BDD100k. As consequence, if there is a traffic light, BDD100k attach the attribute of the traffic light color. Again, the dataset can be adopted using simple python scripts that extract the attribute if there is a traffic light in the image. If multiple traffic light instances are in the image, multiple Signal attributes are stored, one per instance.
- BDD100k does not have labels for brightness values. Annotating the whole dataset for **Perceived brightness** is easy, because of its numeric nature. So, perceived brightness for all the images in the dataset is calculated using a python script. Later, to reduce ambiguity while testing, the range of brightness values (0-255) is divided into 5 equal sections: *Very Low [0-51]*, *Low [51-102]*, *Moderate [102-153]*, *High [153-204]* and *Very High [204-255]*.
- **Pedestrian**, **Road Condition** and **Obstacle**: in BDD100k, it is only possible to check if there are pedestrians in the image or not, not their position. But, according to the solution designer, the position of the pedestrian is a very important feature and should not be ignored. Because it is very risky if there is a pedestrian *on road* when the signal of the traffic light is *green*. But it is very normal for a pedestrian to be in the *sidewalk* when the signal is *green*. There is no easy way to extract this additional information from BDD100k dataset. Similarly, with the current features of BDD100k, Road Condition and Obstacle labels are also not available. So, one way to resolve this issue is to make manually all the missing annotations.

Re-designed dataset for the next stage of development

Based on the discussions made above, the following domain is obtained from BDD100k for the AI model to be used in training and validation steps.

Table 12. Final Problem Domain Considered in This Application

Road Type	Time of Day	Weather	Pedestrian	Traffic Light	Zebra Crossing	Brightness
General way	dawn/dusk	clear	True	Green	True	Very high
highway	daytime	Cloudy	False	Yellow	False	High
parking lot	night	rainy		Red		Moderate
tunnel	undefined	snowy		None		Low
Under FO		undefined				Very low
undefined						

Since BDD100k dataset has labels for all the images for the above-mentioned attributes (except brightness), this domain can be easily used in both training and validation of the classification task.

The original BDD100k dataset has 3 attributes (Road Types, Time of Day and Weather) that can be used to express the driving scenario. But the demonstrated exploratory data analysis and problem domain analysis extracted 3 more attributes (Pedestrian, Traffic Light and Zebra Crossing) as shown in Table 12. So, instead of **3 different classification tasks** as shown in PoC phase, the new domain allows the solution designer to create **6 classification models performing 6 different tasks**. Identification of the driving condition is one of the functional requirements for the final product. The additional attributes express the driving scenario more thoroughly and successfully meets the first functional requirement of the final product.

The **brightness** attribute will be used to evaluate the performance of both classification and object detection models in various brightness levels. These evaluations will demonstrate the model's robustness to brightness fluctuations.

The labels for position of the Pedestrians are missing in the existing dataset but carries great significance. Road Condition and Obstacle labels are also not available. But manual annotation of 100k images (as shown in Section 7.4) will be very time-consuming and require tremendous amount of manpower. So, instead of annotating 100k images, *random 10k images* from the validation set are annotated using a custom annotation tool. These annotation labels can also be used to evaluate how the object detection models perform in specific problem cases. Nevertheless, the Road Condition, Obstacle and

Pedestrians information from the remaining 90k images is expected to be made in the future if necessary.

7.5.2 A-2: Coverage for distinguished problem cases

MLQM Guideline recommends identifying various possible combinations of the attribute values presented in the problem domain analysis phase. Examining the number and details of these combinations of attribute values is the primary theme of evaluating coverage for distinguished problem cases.

MLQM Guideline discusses this quality in Section 1.7.2 and in Section 6.2. Adequate examinations of data design to collect and sort out sufficient training data and test data are required in this phase in response to various situations which systems may need to respond to.

The following discussion demonstrates procedures to ensure this quality during development. Highlights of this discussion are:

1. Steps for evaluating coverage of distinguished problem cases
2. An example evaluation process
3. Identification of special cases

Steps for evaluating coverage of distinguished problem cases

In an ideal scenario, a solution engineer should examine there is enough data for all possible combinations of attribute values. But for a high-dimensional problem domain, it is nearly impossible to give equal importance to each combination. Consequently, to keep a balance between ease of evaluation and preferred level of quality management, a solution designer may perform the following steps:

- Assess the total number of combinations possible for all attributes and their corresponding values. Assess possible combinations of attribute values taking suitable number of attributes in a group:
    - If the number is not high, presence of enough data for each possible combination should be checked.
    - If the number is very high, combinations that carry the most importance and covers most of the dataset should be evaluated first.
- Set cases using combinations of attributes and their values to sort out cases with most importance and cases that can be ignored.

An example evaluation process

The following example is used for demonstration of the evaluation process. For avoiding complexity, a similarly distributed but smaller dataset with ~2k annotated examples (from BDD100k training set) is used. If proper labels for all the images in the dataset can be found, then this evaluation should be done on the whole dataset.

Considered domain details

For this example, let's consider the following attributes and their corresponding values to assess the complexity of the situation.

Total attributes: **7**; Total attribute values: **34**

- **Lighting**: High, Low, Normal **(3)**
- **Road type**: General way, Highway, Parking lot/gas station, Tunnel, Undefined **(5)**
- **Road condition**: Dry, Snowy, wet, Undefined **(4)**
- **Signal**: Green, Red, Yellow, None, Not sure, Undefined **(6)**
- **Pedestrian**: On road, on sidewalk, None, Not sure, Undefined **(5)**
- **Obstacle**: Vehicle, Others, None, Not sure, Undefined **(5)**
- **Weather**: Fine, Cloudy, Rainy, Snowy, Foggy, Undefined **(6)**

Summary of evaluation result

Following are the results obtained by doing a quantitative analysis on a smaller but similarly distributed dataset with 2000 examples.

- Total possible combinations taking one attribute = Total attribute values = **34**
- Total possible combinations taking all 7 attributes and all values = **54,000**

Since the number of total combinations is far too large to investigate, let's consider attribute combinations taking **2** attributes per group from **7** attributes also known as pair-wise analysis.

- Total number of attribute pairs to check combinations: **21**
- Total number of combinations of attribute values: **542**

Let's choose one example pair from the 21 combinations: **Lighting + Signal**. For this pair, there exists 18 possible combinations of attribute values. Table 13 shows the list of those 18 cases and presence of data for each case.

Table 13. Presence of Data for Combinations *Lighting + Signal*

Lighting	Signal	Count	Percentage
High	Green	25	1.23
High	None	70	3.44
High	Not sure	7	0.34
High	Red	4	0.20
Low	Green	345	16.94
Low	None	375	18.42

Low	Not sure	41	02.01
Low	Red	77	03.78
Low	Undefined	1	00.05
Low	Yellow	11	00.54
Normal	Green	326	16.01
Normal	None	626	30.75
Normal	Not sure	41	02.01
Normal	Red	78	03.83
Normal	Undefined	1	00.05
Normal	Yellow	8	00.39
High	Yellow	0	0
High	Undefined	0	0

Data distribution in the above-mentioned combinations are also presented in the following graph.

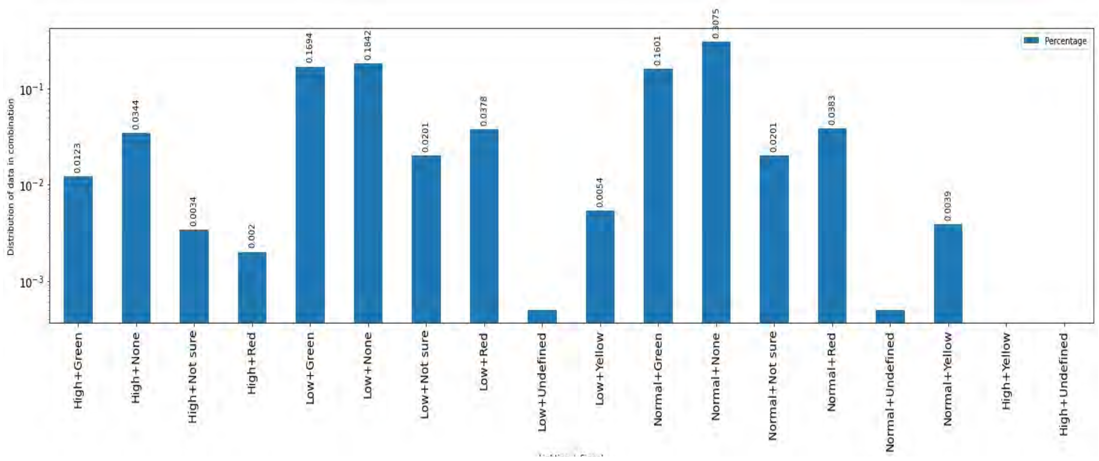


Figure 20. Distribution of Data for Possible Case in Combination *Lighting + Signal*

It is evident that this process of checking coverage can be very time consuming and exhaustive for a solution designer. So, instead, only the special cases for which the system needs to be on high alert will be identified and coverage will be evaluated for those case.

Identifying special cases

Here, the solution designer will check completeness with a rough granularity level by setting out some specific cases with high significance and later investigating the presence of data in only those cases. Carefully choosing these significant cases can allow the solution designer to check the coverage of a large portion of the dataset.

The cases that require special attention are considered significant cases. For example, there can be some combinations of attribute values that may never occur in real life but somehow present in the dataset. The solution designer may choose to exclude the examples of such kind. Also, for some cases, the performance of AI models may degrade in operation. So, presence of enough data for such risky cases must also be thoroughly checked to avoid accidents.

In this section, the goal is to explicitly identify the unsound cases (not possible in real world scenarios) in the problem domain and risky cases (cases where higher level of safety should be maintained due to possibility of performance degradation of models).

Unsound cases

Unsound cases should be identified by the solution designer beforehand so that they can be excluded from training. To describe these cases, some values for a certain attribute (mentioned as *Primary Conditional Attribute*) are set and then they are paired with values of other attributes to check if the created combination can exist in real life. Table 14 shows such combinations observed by the solution designer. Since these combinations must not happen in real life (for this specific application and problem domain), presence of examples from these combinations should be checked and excluded.

Table 14. Unsound Cases that are not Expected to be Present in Data

Case	Primary Conditional Attribute	Primary Conditional Value	Secondary Conditional Attribute	Secondary Conditional Value
0	Weather	Snowy	Road condition	Dry
1	Weather	Rainy	Road condition	Dry
2	Road type	Highway	Signal	Green
3	Road type	Highway	Signal	Red
4	Road type	Highway	Signal	Yellow
5	Road type	Highway	Zebra crossing	Yes
6	Road type	Highway	Pedestrian	On road
7	Road type	Highway	Pedestrian	On sidewalk

Safety critical cases/risky cases

Following are some risky cases identified by the solution designer. These cases are considered risky because AI model performance may degrade in these scenarios and wrong decisions made by AI models in these scenarios can lead to hazardous situations.

#Combinations taking 2 attributes in one group:

1. Road type: Highway + Weather: rainy
2. Road type: Highway + Time: Night
3. Weather: Rainy+ Time: Night
4. Road type: General way + Weather: Rainy
5. Road type: General way + Pedestrian: On road
6. Road type: General way + Time: Night
7. Weather: Rainy + Pedestrian: On road
8. Weather: Rainy + Time: Night
9. Pedestrian: On road + Time: Night

#Combinations taking 3 attributes in one group:

1. Road type: General way + Weather: Rainy + Pedestrian: On road
2. Road type: General way + Weather: Rainy + Time: Night
3. Road type: General way + Pedestrian: On road + Time: Night
4. Weather: Rainy + Pedestrian: On road + Time: Night

After calculating the distribution, the solution designer should set a standard of coverage or a **threshold value.** This threshold, if set after proper observation, can give the solution designer an estimate about if the number of data present in combinations with greater importance (i.e., risky cases) are enough or not. In the next stage of development, based on the comparison between the threshold values and the actual values, decisions need to be made about the following issues:

- If there is any combination that needs more data by augmentation or data duplication or other methods
- If there is any combination that should be ignored completely
- Actions to be taken for combinations with no data

7.5.3 B-1: Coverage of datasets

In MLQM Guideline, a property is defined as *coverage of datasets* where enough data (especially, test data) is given to each *combination of situations that require response* designed in the previous paragraph without any missing situation.

The purpose of configurating this axis of characteristic is to guarantee that the shortage of learning due to the shortage of data or any oversight of learning in specific conditions due to biased data does not occur in any situation or case identified in requirement analysis or data design.

The following discussion demonstrates procedures to ensure this quality during development. Highlights of this discussion are as follows.

1. Steps for evaluating coverage of dataset

2. An example evaluation process
3. Insights from the recorded results

Steps required for evaluation

Following approaches can be taken to ensure if enough amount of data is available for each important scenario, especially the cases identified in Section 7.5.2.

- Quantify number of data points present in each group
- In each group,
  - Assess number of combinations with datapoints more than threshold
  - Assess number of combinations with datapoints significantly less than threshold
  - Assess number of combinations with no data

From above quantities, re-evaluation of threshold values previously defined for data coverage can be done if needed and features can be omitted if necessary.

Example evaluation process

As discussed earlier, in the next step of original development process of an AI product, instead of doing pair wise analysis, it can be considered to calculate distribution for possible combinations taking all 7 attributes and all values(**~54k**). The following analysis should be done on validation dataset as well. But for now, to keep the evaluation process brief for this hypothetical product, let's continue with the evaluation of coverage of dataset only for the specific scenarios for the same dataset used in Section 7.5.2. Following are the results obtained after investigating the coverage for various scenarios mentioned in the previous section.

Presence of data in unsound cases

Table 15 shows the presence of data in unsound cases.

Table 15. Data Coverage in Unsound Cases

Case	Primary Condition Attribute	Primary Conditional Value	Secondary Conditional Attribute	Secondary Conditional Value	Percentage in the Dataset
0	Weather	Snowy	Road condition	Dry	0
1	Weather	Rainy	Road condition	Dry	0.098
2	Road type	Highway	Signal	Green	1.031
3	Road type	Highway	Signal	Red	0.295
4	Road type	Highway	Signal	Yellow	0.049
5	Road type	Highway	Zebra crossing	Yes	0.344

6	Road type	Highway	Pedestrian	On road	0.098
7	Road type	Highway	Pedestrian	On sidewalk	0

As mentioned earlier, the training data should not contain such unsound cases. The inclusion of this kind of data may happen due to different reasons, such as errors in annotation, mislabeling, etc. The solution designer can exclude these examples for the dataset. If the number of data is very low in such cases, additional investigation can be done (considering reasonably minimum effort) to check the source of error and if possible, the labels can be corrected.

Presence of data in risky cases

Results of data coverage are summarized below for high-risk cases mentioned in Section 7.5.2.

Group 1: Combinations taking 2 attributes in one group:

1. Road type: Highway + Weather: rainy
2. Road type: Highway + Time: Night
3. Weather: Rainy+ Time: Night

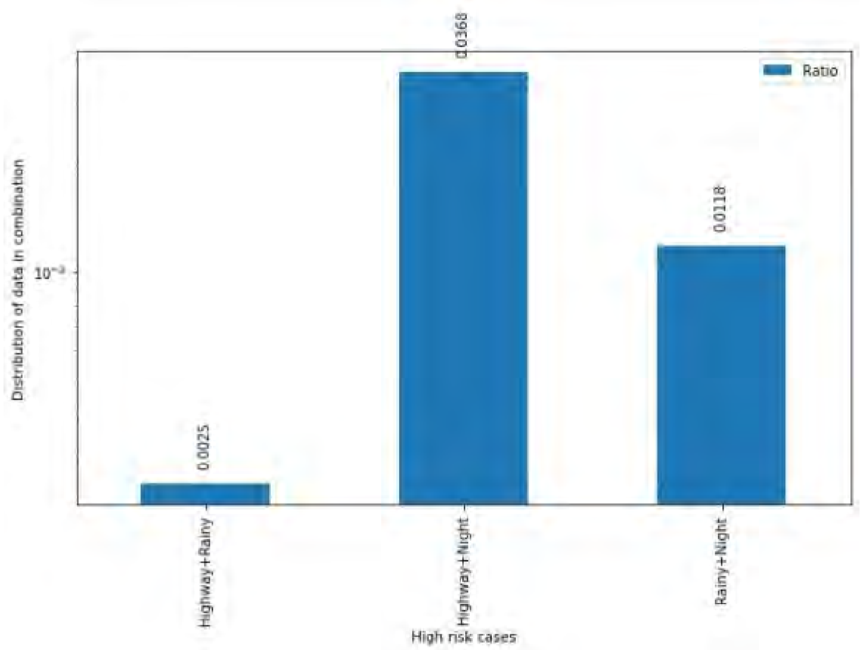


Figure 21. Data Coverage across Some High-risk Cases

Group 2: More combinations taking 2 attributes in one group

1. Road type: General way + Weather: Rainy
2. Road type: General way + Pedestrian: On road

3. Road type: General way + Time: Night
4. Weather: Rainy + Pedestrian: On road
5. Weather: Rainy + Time: Night
6. Pedestrian: On road + Time: Night

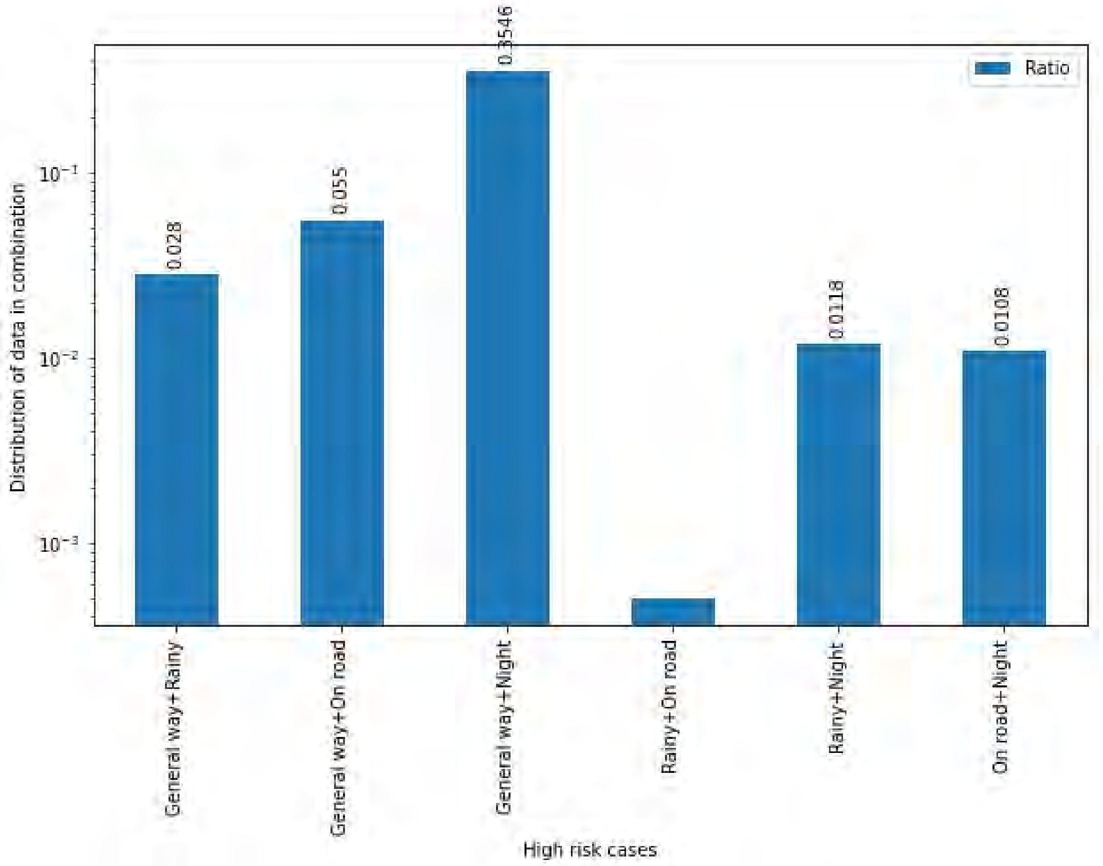


Figure 22. Data Coverage across Some High-risk Cases

Group 3: Combinations taking 3 attributes in one group

1. Road type: General way + Weather: Rainy + Pedestrian: On road
2. Road type: General way + Weather: Rainy + Time: Night
3. Road type: General way + Pedestrian: On road + Time: Night
4. Weather: Rainy + Pedestrian: On road + Time: Night

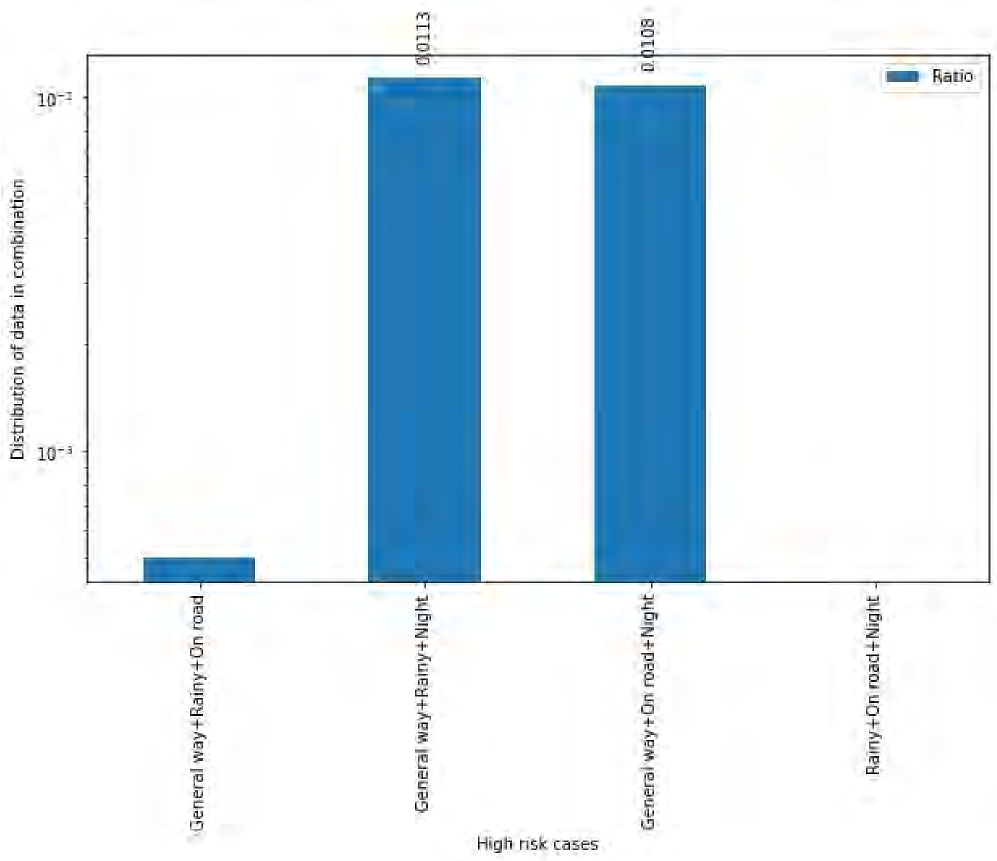


Figure 23. Data Coverage across Some High-risk Cases

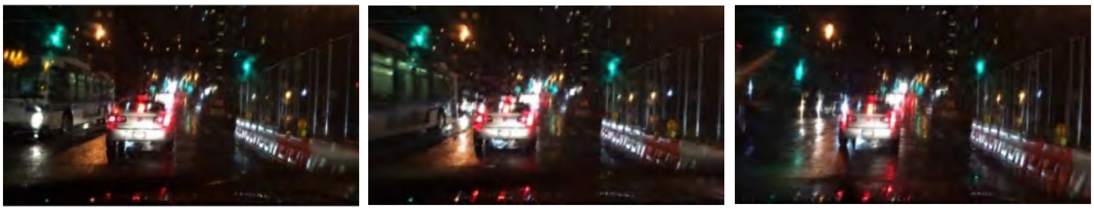
Insights from recorded results

It is observed from the results of coverage analysis that some combinations have significantly less examples than others. While this kind of distribution is expected in real situations, the model performance in these specific scenarios may suffer due to shortage of data. Setting specific thresholds for these cases can help in sorting of cases that will need additional data to increase the coverage of these rare cases. For this research, the threshold has been set to 100 images.

It is necessary to have rare inputs in the dataset in an appropriate amount so that model performance does not degrade in those specific cases. The results presented in the above section identified rare images that do not occur in the dataset that often but those images are extremely important for autonomous driving scenarios. Since acceptable number of such rare images are needed to be taken into training, effective ways to provide these inputs in good amount should be analyzed by solution designer. One way can be using data augmentation processes to generate such corner cases. In fact, with adapting proper augmentation technique, adversarial examples can also be generated to check the robustness and stability of trained models.

For BDD100k dataset, another way is to extract more examples of similar scenarios from BDD video dataset by sampling more frames from the desired clips. All the images from BDD100k dataset are obtained from BDD videos. Additionally, each image in BDD100k has an ID pointing which video was used in the BDD video dataset. In consequence, it is possible to find more images related to an image of BDD100k if the video of that image is found. All the video frames of BDD video dataset are labeled and can be extracted with a python script.

For example, if the solution designer wants to include some unclear images due to water on the windshield to make the trained models robust to noises, then some images from the same video but with different timestamp can be used. But data coverage investigation shows that such cases rarely happen. So, what you could do is as follows. First, identify an unclear image due to water on the windshield in the video dataset. Next, extract two frames, 1 second earlier and later frames than the identified image. These three images will have the same disturbances with different road position. The video dataset has all object labeled using the same annotation as BDD100k. In consequence, the extracted images can be included in the dataset for training or validation. In this way, the number of images of the combinations that have few available images can be increased.



(a) 1 second before (b) Original (c) 1 second after

Figure 24. Consecutive Frames of 02701fba-809c39f3.mov from BDD video

7.5.4 B-2: Uniformity of datasets

MLQM Guideline tells us that *uniformity of dataset* is a concept contrary to *coverage* mentioned in the previous section. Evaluating *uniformity* of a dataset involves investigation of the original distribution of dataset and inspecting if there is any bias present in the data. When each case in a dataset is populated in accordance with the frequency of its occurrence in the whole input data, the dataset is considered *uniform*. Here the primary concern is to evaluate the balance between *coverage* and *uniformity*. MLQM Guideline states that it is necessary to consider which one receives priority, coverage or overall uniformity, and how to strike a balance between them.

The following discussion demonstrates procedures to ensure this quality during development. Highlights of this discussion are:

1. Steps for evaluating uniformity of dataset
2. An example evaluation process

Steps required for evaluation

The priority mentioned above will have a significant effect on the models' performance. Depending on the given requirements, the solution designer may choose to:

- Either give priority to overall performance of the models. Here, if there is such a case that occurs very rarely in the dataset, then due to shortage of enough data, the ML model may fail to learn that case properly.
- Or give priority to performance of the models for specific cases with greater significance, even if there are not enough data present for that case. To achieve this goal, the solution designer would be biased to the specific case and would try to include more data from this case ignoring the natural frequency of occurrence for that case. As a result, specific performance may improve, but overall performance may deteriorate.

Considering the above scenarios,

- The solution designer should evaluate the natural frequency of occurrence, the distribution of the dataset, and check if the steps taken to improve coverage of the specific cases introduce bias in the dataset.
- If significant bias is found, an expected distribution of the dataset can be calculated considering a smaller portion of the data or considering data from other similar sources. Then, a comparison should be made on the expected distribution and the biased distribution.
- The solution designer can set a level of allowance for the fluctuation of distribution for the problem cases depending on the requirement and priority level discussed before.

Example evaluation process

Let's consider a simple portion of the data to demonstrate evaluation procedure of uniformity analysis. In real case of product development, the distribution should be measured across the whole training and validation dataset and then the frequency of occurrences for cases that can be compared with another dataset coming from a separate source. Say, for this research, the following attributes describe the problem domain with around ~2k data in the dataset:

- **Obstacle**: None, Vehicles, Other, Not sure, Undefined

- **Pedestrians**: None, Not sure, On sidewalk, On road, Undefined
- **Road type**: General way, Highway, Under bridge, Gas station, Undefined, Tunnel
- **Lighting:** High, Normal, Low

Evaluating general distribution of the dataset

First let's evaluate the general distribution of data across attribute values for different attributes. The following graphs show the distribution of data for *Lighting* attribute across attribute values.

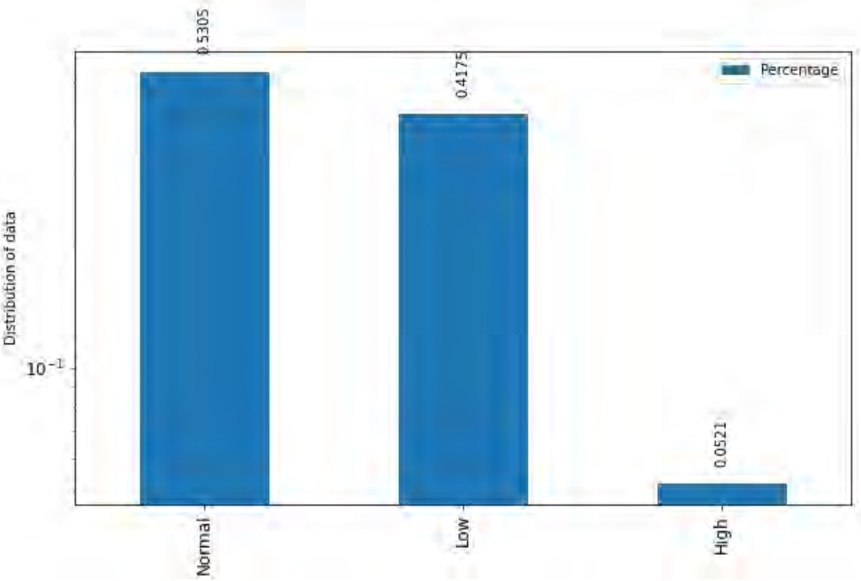


Figure 25. Distribution of Data across Attribute Values of Attribute *Lighting*

The following graphs show the distribution of data for *Road type* attribute across attribute values.

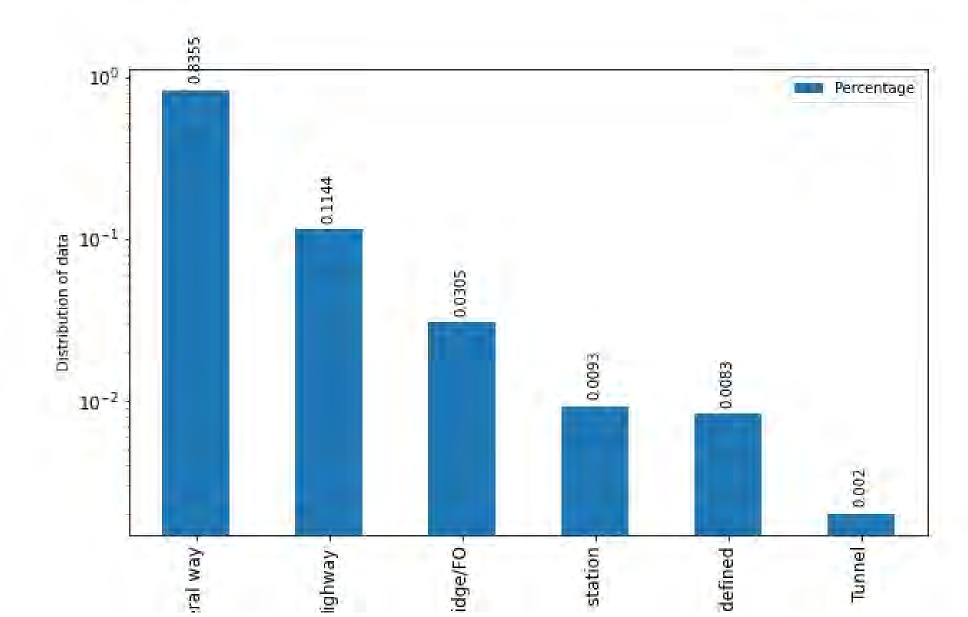


Figure 26. Distribution of Data across Attribute Values of Attribute *Road type*

Let's assume the solution designer has a different but similarly distributed dataset from another source, a dataset with around 6k data in it. We can consider the distribution of this secondary dataset as a reference or an expected distribution to acquire a comparison.

Table 16. Comparison of Frequency of Occurrences for Attribute Values of *Lighting*

Attribute Value (Lighting)	Original dataset distribution	Secondary dataset distribution	Difference in percentage
Normal	53.05	62.15	-9.1
Low	41.75	35.07	6.68
High	05.21	02.75	2.46

Table 17. Comparison of Frequency of Occurrences for Attribute Values of *Road type*

Attribute value	Original dataset distribution	Secondary dataset distribution	Difference in percentage
General way	83.55	81.84	01.71
Highway	11.44	13.45	02.01
Under bridge/Flyover	03.05	02.9	00.15
Parking lot/Gas station	00.93	00.84	00.09
Undefined	00.83	00.72	00.11
Tunnel	00.2	00.26	-00.06

Depending on the differences observed, the solution designer can next set threshold values for allowed fluctuations. Later, if any biasness is found for any attribute values due to data augmentation or other data distribution manipulation, the solution designer can check if the biasness is within the allowed fluctuation.

Evaluating distribution of data across combined cases

Similar comparison should be made taking various combinations of attributes, especially including the cases with special significance like risky cases. The possibility of risky cases being rare in the dataset leads to the solution designer's decision of creating a biased dataset. So, to strike a balance between uniformity and coverage, detail investigation like the above one should be done for the specific combinations with higher importance. A similar analysis on combination attribute for *lighting + road* type is given below:

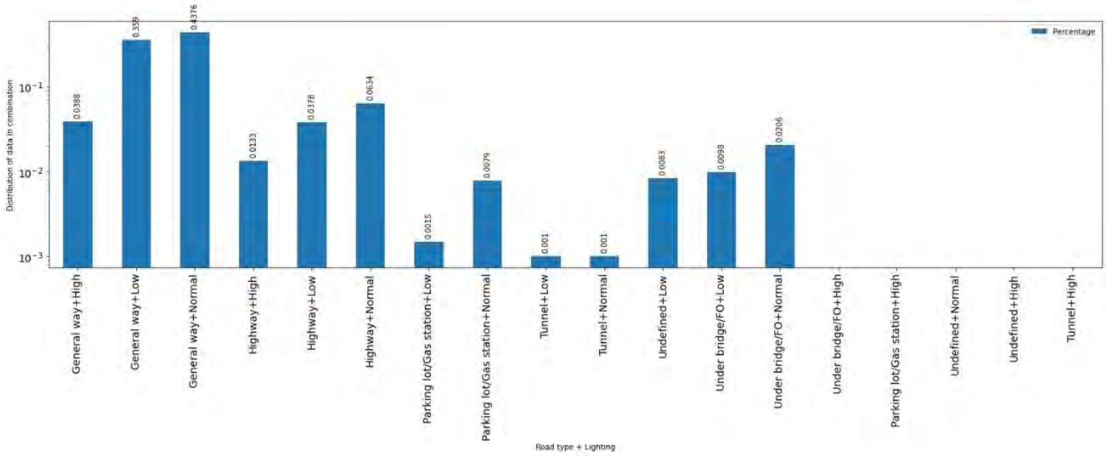


Figure 27. Distribution of Data across Combinations of Attribute Values of *Road type + Lighting*

A similar comparison can be done on differences in expected vs actual distribution.

Table 18. Comparison of Frequency of Occurrences for Combined Attribute Values

Road type	Lighting	Original dataset distribution	Secondary dataset distribution	Difference in percentage
General way	High	3.88	2.05	1.83
General way	Low	35.9	29.08	6.82
General way	Normal	43.76	50.72	6.96
Highway	High	1.33	0.7	0.63
Highway	Low	3.78	4.45	0.67
Highway	Normal	6.34	8.28	1.94

Parking lot/Gas station	Low	0.15	0.12	0.03
Parking lot/Gas station	Normal	0.79	0.72	0.07
Tunnel	Low	0.1	0.09	0.01
Tunnel	Normal	0.1	0.17	-0.07
Undefined	Low	0.83	0.58	0.25
Under bridge/FO	Low	0.98	0.75	0.23
Under bridge/FO	Normal	2.06	2.15	0.09
Under bridge/FO	High	0	0	0
Parking lot/Gas station	High	0	0	0
Undefined	Normal	0	0.12	0.12
Undefined	High	0	0	0
Tunnel	High	0	0	0

It should be noted that with availability of actual data from the actual operational environment and with adequate time, these evaluation procedures should be executed on the real world datasets that will be used for training and validation of the final AI product. This can be considered as future work in next versions of the reference guide. This version of the Reference Guide does not include detailed experiments or exact evaluation procedures in most cases. Rather it discusses how the evaluation process and model development process can be carried out. Also, in some cases only analysis for the object detection task or the classification task is done. Similar results for the left-out task are expected to be produced as well. More structured ways of recording PoC results, evaluation results and model performance are expected to be used in each step of the agile development process in future.

7.5.5 B-3: Adequacy of data

This version of the reference example does not demonstrate any evaluation procedure for the internal quality *B-3: Adequacy of data*. It is expected to be included in the next version.

7.5.6 C-1: Correctness of the trained models

The term *correctness of a trained model* represents that a machine learning component functions as intended upon the input from the learning dataset (consisting of training data, validation data, and test data). In MLQM Guideline, this notion also includes the convergence of the training and the quality of training data (e.g., the dataset has only a small number of outliers and incorrectly labeled data).

Highlights of the following discussion are:

1. Decisions from PoC phase
2. Evaluation procedure for correctness of object detection model

Decisions from PoC phase

Using the results shown in the PoC phase of Section 7.2.2, it has been decided to only continue experimenting with the models that achieve 50% or more mAP and can work with real time images (meaning FPS is bigger than 15). The reason is that if a model has less than 50% mAP, the solution designer needed to expend a lot of time to improve them in comparison to improve a model with more than 50% mAP. Regarding the FPS, though it is possible to improve the FPS of the models, improving the FPS may decrease accuracy. In consequence, for this example evaluation, it is decided to focus only on improving accuracy and leave the improvement and study of FPS for future work. Yolov3, M2Det, EfficientDet-D2, MobileNetv1 and MobileNetv2 are removed from the experiments because they cannot achieve the minimum mAP and/or their FPS is too low. The remaining models (Yolov3+ASFF, Yolov4, Yolov5, Faster R-CNN) were the objective in this reference example. Their accuracy is to be improved to achieve enough mAP to accomplish ASIL D.

Evaluation procedure for correctness of object detection models

Removing noise information in the dataset

The image in Figure 28 shows how some bounding boxes overlap which makes the training process difficult. This issue can bring noise to the training process of each detection model reducing the accuracy obtained:

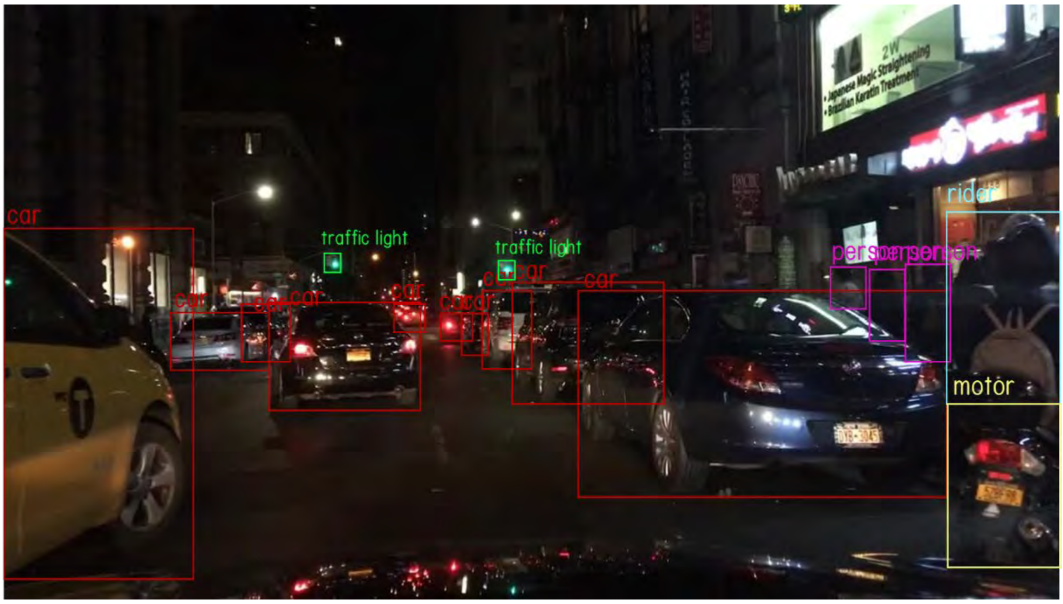


Figure 28. Image Example of Overlapping Bounding Boxes

An appropriate solution to increase the accuracy can be to reduce the number of boxes and make the detection models learn about objects that are close to the car. But calculating the distances of objects from the vehicle adds another level of complexity and may need depth estimation. For reducing the number of bounding boxes, let's follow approach that can be named as *Reducing boxes*.

The idea is if a bounding box is smaller or equal to the created artificial window, then that bounding box is removed from the training dataset. A new dataset is created excluding those bounding boxes and it is used to re-train the detection model. The 6 different measures of the artificial windows are:

0: All objects are used

10: objects inside 10x10 pixels are not used for prediction or validation

20: objects inside 20x20 pixels are not used for prediction or validation

30: objects inside 30x30 pixels are not used for prediction or validation

40: objects inside 40x40 pixels are not used for prediction or validation

50: objects inside 50x50 pixels are not used for prediction or validation

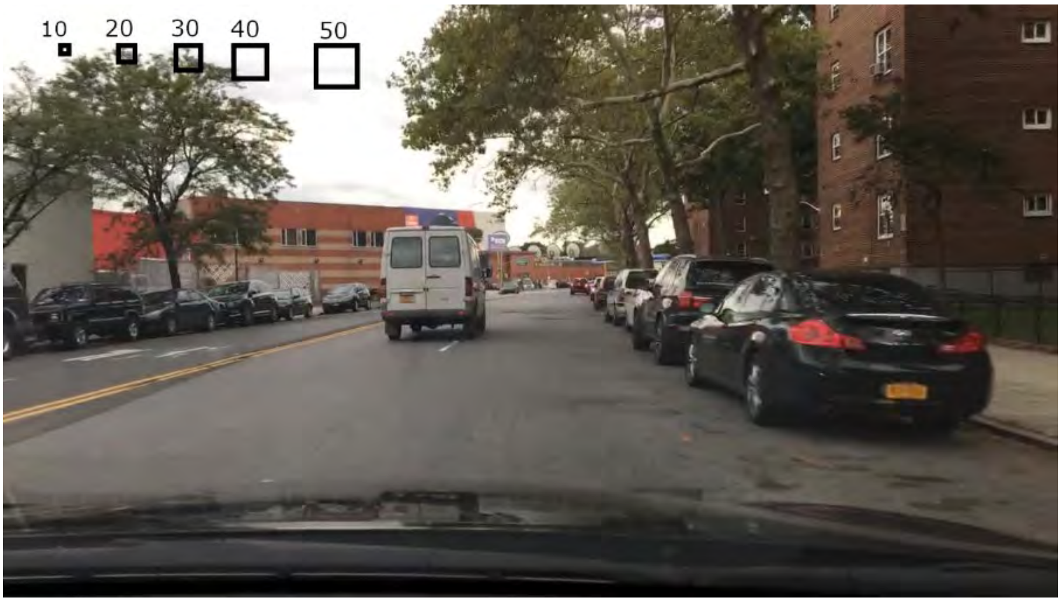


Figure 29. Different Artificial Window Sizes Used to Remove Bad Bounding Boxes

Table 19. mAP Comparison After Reducing Boxes Mechanism

Model	Whole dataset	Removed 10x10	Removed 20x20	Removed 30x30	Removed 40x40	Removed 50x50
YOLOv3 + ASFF	56.55	58.43	63.13	65.11	53.00	40.74
YOLOv4	62.3	64.17	69.70	72.93	58.78	45.15
YOLOv5	62.9	64.57	70.62	73.68	59.87	45.43
Faster R-CNN	59.3	60.66	66.14	70.41	54.58	41.08

Table 19 shows the performance of object detection models after removing the bounding boxes and re-training each detection model selected. There is a mAP increment of about 1.7%, 7.1% and 10.2% after removing bounding boxes smaller than 10x10, 20x20 and 30x30 respectively. However, if the bounding boxes smaller than 40x40 and 50x50 are removed, the mAP percentage decrease about 3.7% and 17.1%, respectively. With this result, it is demonstrated that there are bounding boxes that bring noise to the detection models training process. This mechanism can be used with other dataset to reduce overlapping annotations and check if the mAP increases.

Table 20. Label mAP Result After Reducing Boxes Mechanism is Performed

Yolov4 mAP(%)	Whole dataset	Removed 10x10	Removed 20x20	Removed 30x30	Removed 40x40	Removed 50x50
Traffic light	51.92	52.65	59.86	66.84	47.25	31.73
Traffic sign	66.47	67.65	74.15	76.64	60.78	46.46
Car	79.25	78.26	84.16	86.9	71.62	59.84
Person	51.15	55.84	60.18	62.52	49.71	38.26
Bus	63.13	64.58	68.92	71.87	60.86	43.27
Truck	47.81	50.63	57.27	60.24	45.32	32.61
Rider	61.78	62.42	66.58	69.36	58.75	46.97
Bike	72.43	75.68	79.91	81.34	69.64	57.31
Motor	67.5	69.84	76.62	80.85	64.93	50.18
Overall mAP	**62.3**	**64.17**	**69.70**	**72.93**	**58.78**	**45.15**

Table 20 shows the mAP results per label of using *Reducing Boxes* algorithm on Yolov4. This result shows how removing some of the bounding boxes helps the detection model to improve the accuracy of all labels.

Specific training for the detection models

Another way that can improve the accuracy is to train the detection models using images of specific attribute values. In this way, the detection model can be trained to work for specific situations. Regarding BDD100k, the images have 7 different attributes, such as *road, weather*, *time of day*, *has pedestrians*, *traffic light color*, *zebra crossing* and *brightness*. However, for this experiment, only the attributes *road type*, *weather*, and *time of day* are used because they have more priority regarding ASIL D requisite.

Let's assume that training a detection model require at least 20,000 images. Figure 30 shows the number of images per attribute value, and also shows that this experiment can be made using only the attribute values of *city street (Road Type)*, *daytime (Time of day)*, *night (Time of day)* and *clear (Weather)*.

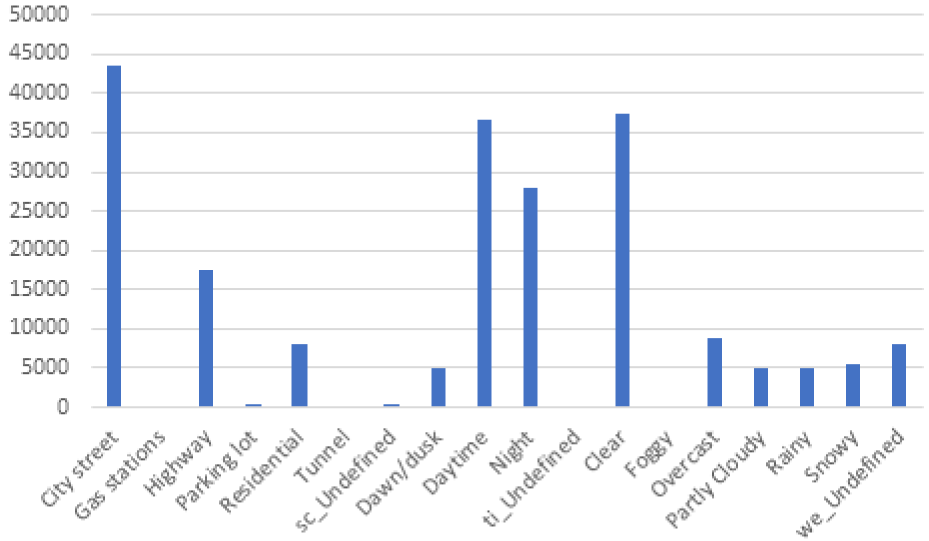


Figure 30. Number of Images per Attribute Value

Figure 15 shows the mAP comparison of training each detection model only with images with attribute values *city street*, *daytime* or *night*, against the same detection model trained using the whole dataset. To calculate the mAP, that uses the images from validation dataset, only images of the specific attribute value are used. As a result, there is an increment in all cases. This increment is higher for *daytime* having an average increment of 10% while *night* has the lowest increment about 6%.

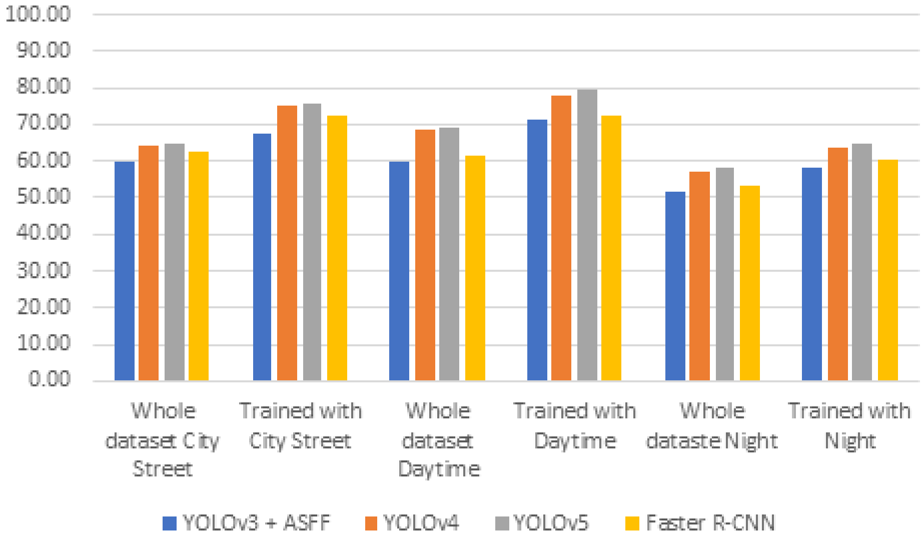


Figure 31. mAP Comparison of Specific Training Using Attribute Values to the Whole Dataset of Attribute Values Used for Specific Training

Summarizing, with this experiment it is demonstrated that there are situations where it is better to use specific dataset for training instead of using a dataset with diverse situations. Using this idea, a system could use multiple detection models and use them depending on the situation, for example, a model that works in *city street* attribute value when the car is inside the city, or two models, one that works in daytime and the other at night. The only requisite is that the system will need to have a mechanism to detect if the car is in a city street or it is daytime. In this experiment, *Reducing Boxes* algorithm has not been combined with specific situation training; but it is expected to be done in the future version of the reference example. Additionally, in the future, combinations of attribute values will be tested, such as, having a model that works in *daytime* and when the car is in a *city street*.

7.5.7 C-2: Stability of the trained model

According MLQM Guideline, the term *stability of the trained model* means that a machine-learning component shows appropriate reactions to input data that are neither included in learning datasets nor sufficiently similar to data in learning datasets.

Ensuring this quality of a model includes evaluating its generalization ability, its reaction to corner cases/rare cases, and its performance on adversarial examples. Assessing a model's robustness to these issues or the stability of the models is not a single step process in the agile AI development cycle. Rather, various techniques and measures are encouraged to be integrated in different stages of the AI development life cycle.

Highlights of the following discussion are:

1. Steps required for evaluation
2. Evaluating generalization capability
3. Evaluating robustness to adversarial images

Steps required for evaluation

Steps that can be executed by the solution designer to evaluate this internal property is discussed below:

- **Evaluating generalization capability:** In real world cases, there will be a lot of variations in input data. It is not possible to include all of them in training. Primarily, it is essential to include input data from all over the domain and hope that the model will behave properly within a proximity of training dataset. Next, overfitting tendency of models should be kept in check while training. Last but not least, the model's generalization ability should be tested on inputs never encountered by the model while training and validation period. Effectiveness of the used evaluation techniques needs to be analyzed as well.
- **Evaluating robustness to noise/adversarial examples:** There is always a possibility of added noise in input data. So, the noise handling capabilities or robustness to specific noises of concerns may need to be measured as well. Evaluation procedures of measuring robustness and the level of robustness needs to be described to reflect the response of the model to adversarial examples.

Evaluating generalization capability

For this section, a different dataset called nuImage [18] has been used to evaluate the performance of the detection models after they have been trained using BDD100k. nuImage has the following properties.

Table 21. *nuImage* Dataset Description

# Labels	# Images	has weather	has time of day	3d bounding box	2d bounding box
23	93476	No	Yes (timestamp)	Yes	No

This dataset contains 93476 images, which are divided into train, test, and validation data.

Table 22. nuImage Dataset Image Distribution

	Number of images	Percentage on dataset (%)
Train	67279	71.97462
Val	16445	17.59275
Test	9752	10.43262

However, there are images that do not have any label at all.

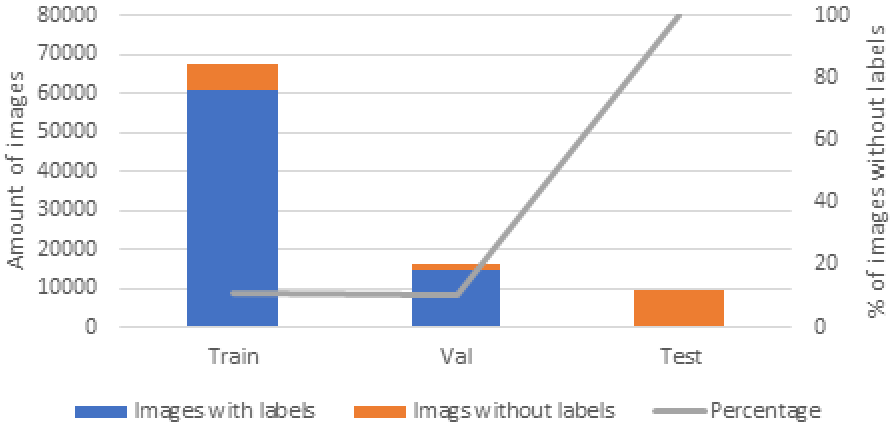


Figure 32. nuImage Distribution of Images that Have and not Have Annotation

So, only 60,668 images can be used for training the detection models and 14,884 images for validating, respectively.

Regarding the number of annotations, BDD100k is a dataset with 10 labels while nuImage has 23 labels. To evaluate the generalization capabilities of the detection models, the ground truth of the labels that are common to both BDD100k and nuImage are used. Therefore, the solution designer has reduced the 23 labels to the ones that can match the detection models trained by BDD100k and they have been highlighted in green color in **Table 23**. As consequence, only the following BDD100k labels can be detected: *bike*, *bus*, *car*, *motor*, *person* and *truck*.

Table 23. nuImage Label Distribution and (in Green) Labels Used for Converting to BDD100k Labeling

Label	Training	Validation	Total	Percentage
animal	173	82	255	0.04
human.pedestrian.adult	121200	28721	149921	21.61
human.pedestrian.child	1683	251	1934	0.28
human.pedestrian.constru	10465	3117	13582	1.96

human.pedestrian.persona	1828	453	2281	0.33
human.pedestrian.police_	368	96	464	0.07
human.pedestrian.strolle	293	70	363	0.05
human.pedestrian.wheelch	33	2	35	0.01
movable_object.barrier	70112	18433	88545	12.76
movable_object.debris	2461	710	3171	0.46
movable_object.pushable_	3030	645	3675	0.53
movable_object.trafficco	69016	18587	87603	12.63
static_object.bicycle_ra	2461	603	3064	0.44
vehicle.bicycle	13708	3352	17060	2.46
vehicle.bus.bendy	203	62	265	0.04
vehicle.bus.rigid	6538	1823	8361	1.21
vehicle.car	202809	47279	250088	36.05
vehicle.construction	4768	1303	6071	0.88
vehicle.emergency.ambula	34	8	42	0.01
vehicle.emergency.police	104	35	139	0.02
vehicle.motorcycle	13682	3097	16779	2.42
vehicle.trailer	3285	486	3771	0.54
vehicle.truck	29456	6858	36314	5.23

To compare the results, 2 different models of Yolov4 has been trained. One model has been trained using BDD100k training dataset, while the other model has been trained using nuImage training dataset after the labels has been converted on BDD100k annotations. The model trained using BDD100k is used to evaluate how effectively the detection model performs on images from outside the dataset, and the other model is used as a ground truth to know the difference if the model was trained using the same dataset. Figure 33 shows this comparison. The labels *car*, *bus* and *truck* achieve similar accuracy; the differences are less than 5%. *Motor* is the label with the highest difference between the models (about 16%). The reason is because BDD100k has *rider* label as well, and the model sometimes annotates the rider and not the motor.

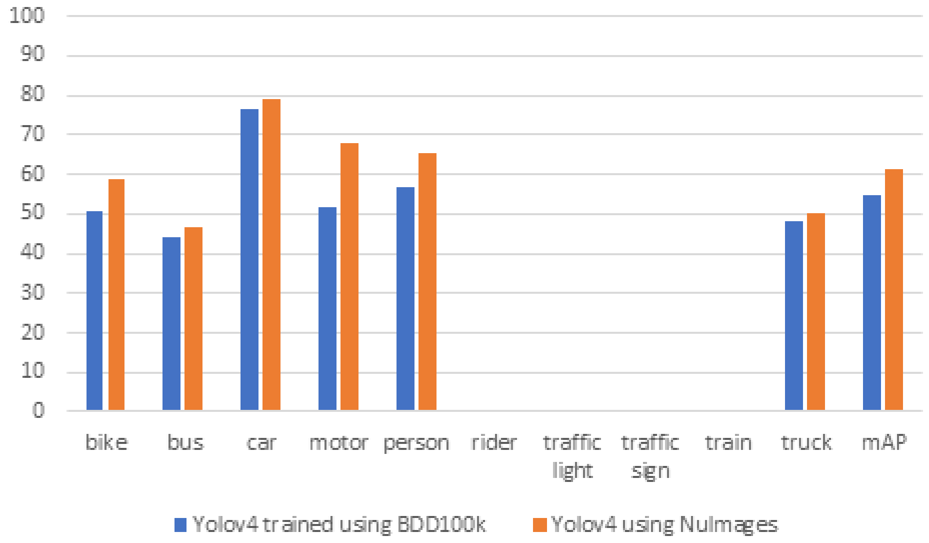


Figure 33. Accuracy Comparison of Yolov4 on nuImage Validation Dataset after being Trained by BDD100k and nuImage Training Datasets

Comparing the mAP of both models, the BDD100k achieves 54.64% and the other model achieves 61.32%. This difference is less than 7%. As a result, Yolov4 achieved a good generalization capability after being trained using BDD100k to work with images outside the dataset.

Evaluating robustness to adversarial images

To evaluate the robustness of the object detection models, it has been decided to use surprise adequacy [19] (see also Appendix B) that uses adversarial examples. Adversarial example data needs to be generated introducing specific noise patterns or using available technologies for adversarial attack. Some popular techniques of adversarial attacks mentioned in recent literatures are:

- Fast Gradient Sign Method (FGSM)
- Basic Iterative Method (BIM-a, BIM-b)
- Jacobian-based Saliency Map Attack (JSMA)
- Optimization-based attack (Opt)

In this example, FGSM is used, leaving the other three adversarial attacks for future work.

Complications of adversarial attack on object detection models

One matter of concern is that even the most recent literatures on adapting adversarial attacks for NN architectures (FGSM, BIM-a, BIM-b, JSMA and Opt (C&W)) only deals with traditional convolutional neural networks. These adapting adversarial attacks use the last layer of the NN to generate adversarial examples. Standard state-of-the-art object detection models are unique and complex in terms of their architecture. Especially the divergent structure of last few layers makes it very difficult to use these available methods to generate adversarial examples. Also, output predictions in Yolo are different than other NN. In other NNs, normally, the output layer gives the scores of each label, while Yolo, with all its versions, has 3 parallel output layers that consists of three detection tensors, each with its own prior boxes and each twice the resolution of the previous. After non-max suppression method is used, only the boxes with higher label score are kept. For the creation of adversarial examples, non-max suppression cannot be used because it is a selection method and not a layer. In consequence, the generation of adversarial examples using Yolo neural networks is an issue that this example tried to resolve.

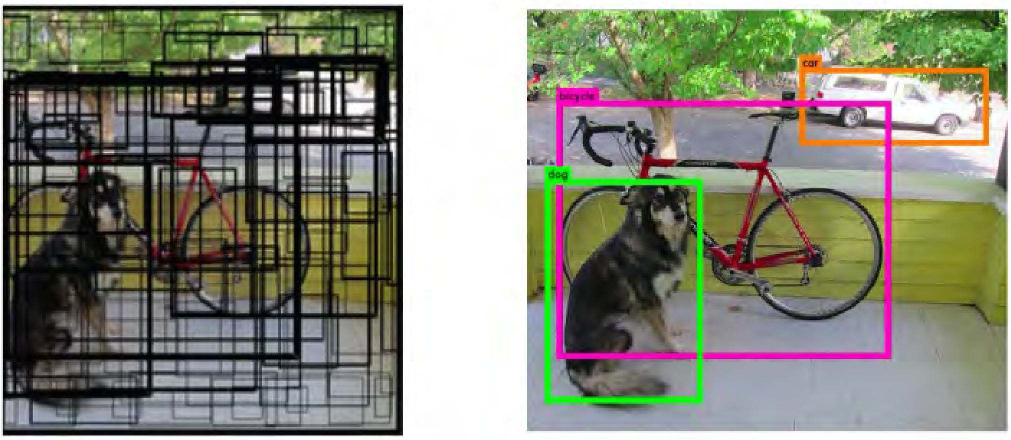


Figure 34. Example of Yolo Last Layer Performance

Attempted approaches

The solution designer attempted to adapt FGSM attack for YOLOv4 architecture but was unsuccessful because the FGSM attack cannot work with NN that has multiple parallel output layers. Different approaches were taken to fix this issue:

1. **Changing the framework** from Tensorflow-Keras to Pytorch
2. Using different architectures such as YOLOv3, YOLOv5
3. **Modifying the prediction output**, making other labels 0 and keeping the confidence of the label found. The problem is that it cannot generate the adversarial example because it cannot determine which label has the closest confidence score because all the other labels have confidence 0 and they are not

closed to determine the adversarial example.

4. **Generating gradients manually** instead of using tf.gradients in TensorFlow
5. Using different adversarial attacks like BIM-a and JSMA etc.

Possible solutions

All previous mentioned solutions did not work. However, the solution designer proposed other solutions that help to create adversarial examples and/or detect corner cases.

- **Using a different NN model**, such as Inception or MobileNet. These models do not have the same last parallel layer problem as Yolo. As a trial, MobileNetv2 was used to generate FGSM adversarial examples, and it works perfectly. Adapting surprise adequacy to these models do not represent any problem. So, using a different model to determine the corner cases and after that using those corner cases in Yolo can be a suitable solution.

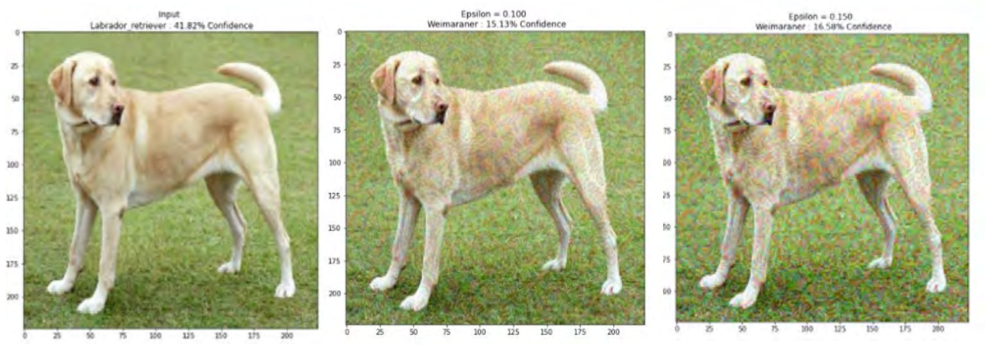

Input
Labrador_retriever : 41.82% Confidence
Epsilon = 0.100
Weimaraner : 15.13% Confidence
Epsilon = 0.150
Weimaraner : 16.58% Confidence
0
25
50
75
100
125
150
175
200

Figure 35. Example from Successful Implementation of FGSM Attack on MobileNet

- **Trying different attacks** can be a solution to the challenge. However, all the methods that generate adversarial examples to perform surprise adequacy use output label confidence and, therefore, they only work if the NN has one last output layer and not multiple last output layers. So, defining new attacks that are different from the ones mentioned might be necessary. For example, using ***1-pixel change*** method (introduced later in this section) to detect the ***corner cases*** that has a high percentage to cause a bad labelling.

Adapting adversarial attacks using a different NN

To evaluate the robustness of the detection models trained, the solution designer decided to use MobileNetv2 to create the adversarial examples and examine the robustness using surprise adequacy. In this case, MobileNetv2 has been trained using BDD100k and Tensorflow-Keras framework. After training, the trained model obtains an accuracy of 84.49% but with an FPS of 4.5. Due to the lower number of FPS, MobileNetv2 was never considered to be used as an autonomous driving car detection model. However, we can use MobileNetv2 to generate adversarial attacks such as FGSM. After adversarial examples are generated, surprise adequacy is used to examine the corner cases. These corner cases have been assumed to be the same for all detection models and they will be treated in the same way for all detection models. The following images show examples of adversarial examples generated using MobileNetv2 and FGSM.

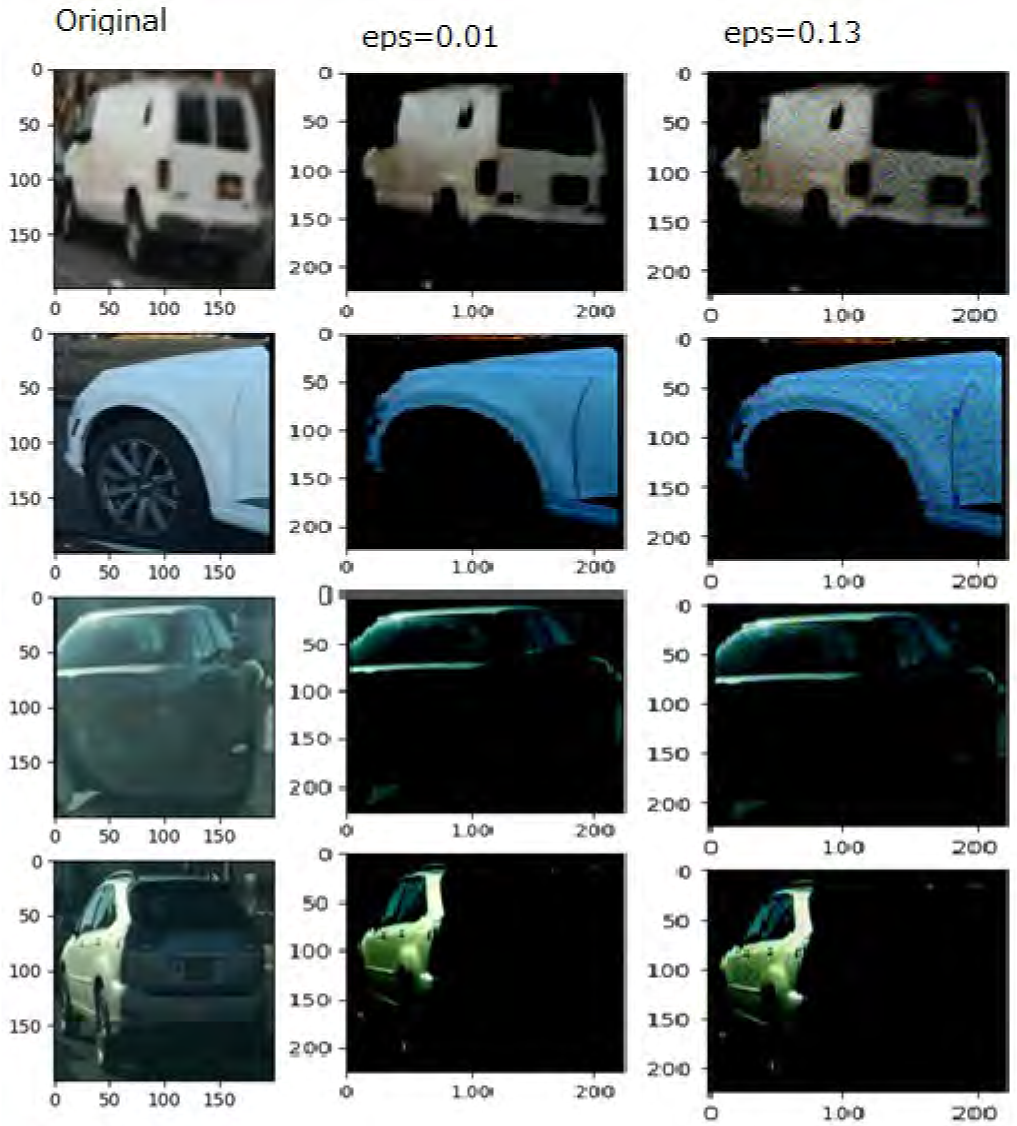


Figure 36. Data Generated Using FGSM Attack on BDD 100k Dataset for eps = 0.01 & 0.13

Adapting surprise adequacy to work with MobileNetv2

In this section, *surprise adequacy* [20] is used to detect the corner cases of the training data, remove them from the training dataset and re-train all detection models: Yolov3 + ASFF, Yolov4, Yolov5, Fast R-CNN and MobileNetv2.

Regarding BDD100k, there are 1.286.871 bounding boxes in the training dataset (dataset contains 69.863 images). **Table 24** shows the distribution of bounding boxes in the validation data that has been used for the Surprise attack.

Table 24. Bounding Box Label Distribution of BDD 100k

Label	# of BB
bike	7210
bus	11672
car	713211
motor	3002
person	91349
rider	4517
traffic light	186117
traffic sign	239686
train	136
truck	29971

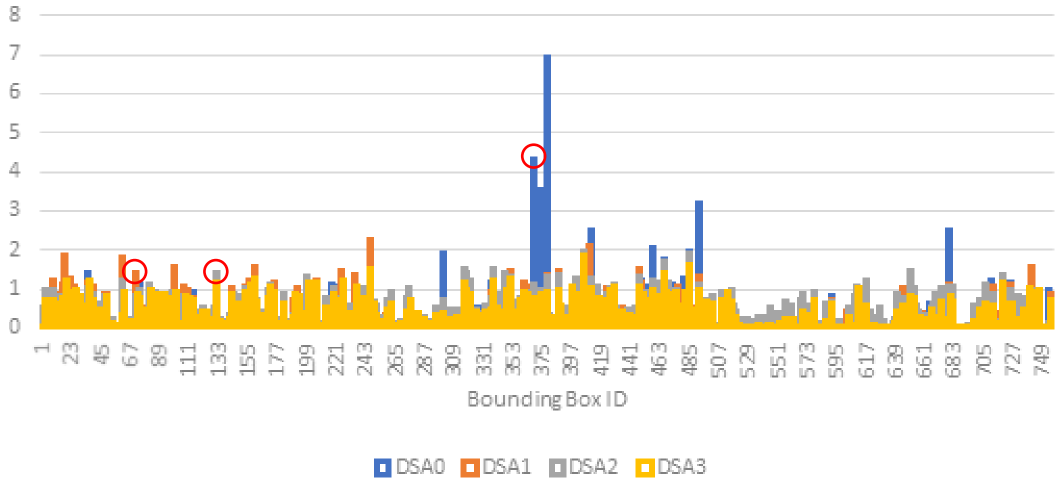
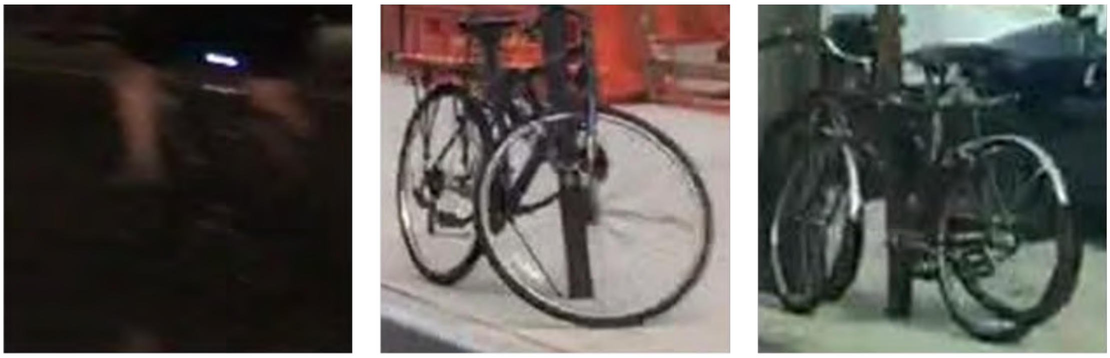


Figure 37. Surprise Adequacy Calculation of the First 750 Bounding Boxes of *Bike* in the Training Data

There are too many *car* bounding boxes. Nevertheless, there are enough bounding boxes to detect the corner cases of all labels, including train. This is because with 100 images, surprise adequacy has enough data to determine the corner cases among the labels.

Figure 37 shows surprise adequacy calculation of DSA_0, DSA_1, DSA_2 and DSA_3 of the first 750 bike bounding boxes in the training dataset. If the distance is higher than 1.5 in any of the DSAs calculated, then the bounding box is defined as a corner case. For example, three corner cases are marked with red circles in Figure 37 and presented in Figure 38.



ID	DSA0	DSA1	DSA2	DSA3
73	1.54	0.82	0.97	0.79

ID	DSA0	DSA1	DSA2	DSA3
131	0.52	0.39	1.6	1.15

ID	DSA0	DSA1	DSA2	DSA3
368	4.32	0.98	1.11	0.78

Figure 38. Bike Corner Case Examples Detected Using Surprise Adequacy

As a result, checking all labels of BDD100k:

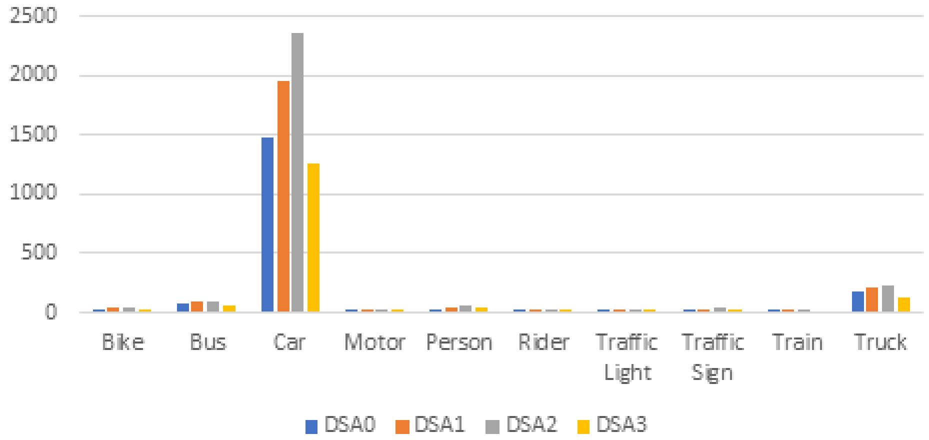


Figure 39. Amount of Corner Cases Detected by Labels

The main issue with the graph above is that the BDD100K has too many *car* labels compared to other labels. As consequence, it is difficult to determine what is the impact of DSAs regarding each label.

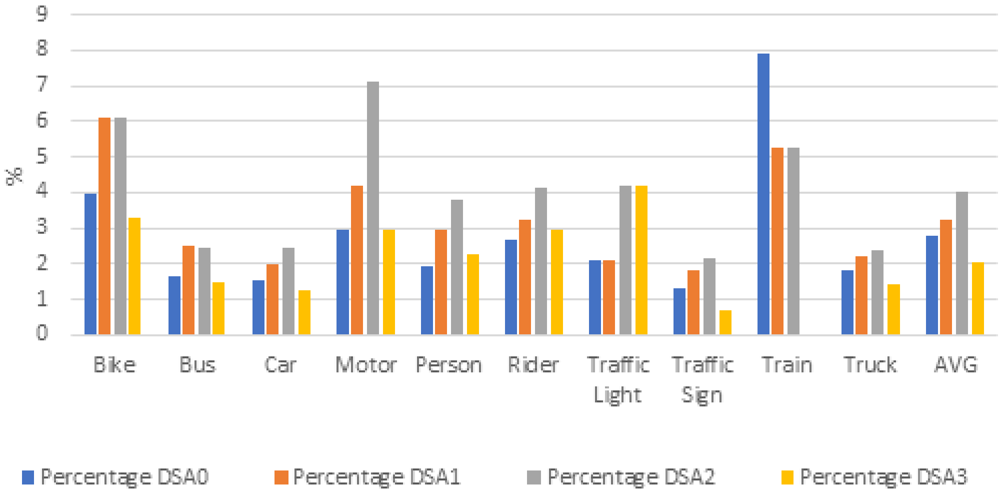


Figure 40. Percentage of Corner Cases in BDD 100k Dataset

However, using the percentage, to determine how many corner cases are detected per label, it is possible to see that depending on the label the DSA works better than other DSAs. For example, DSA_2 can detect more corner cases for label *Motor*. Overall, DSA_3 is not detecting corner cases for BDD100K very well, for example, it cannot detect any corner case for *Train* label.

To make use of corner case detection, the solution designer trained 6 different MobileNetv2:

- one model per each DSA detection, trained with the original dataset except all corner cases detected by the detection
- one model trained with the original dataset except all corner cases detected by any of the four DSA detections
- one model trained with the whole original dataset

The model used here is a MobileNetv2 that has been already trained to identify images from ImageNet dataset. In this way, the training is faster and only the last layers need to be re-trained.

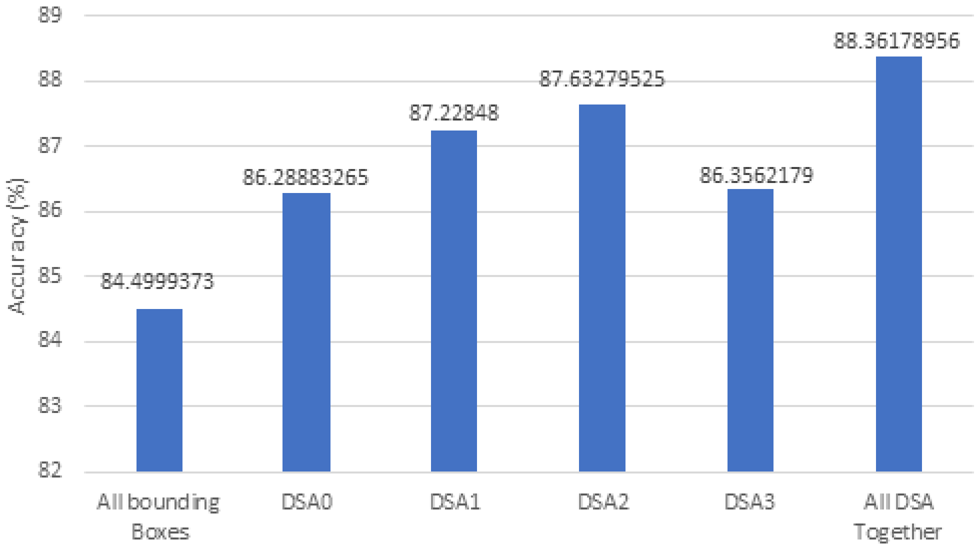


Figure 41. Model Accuracy after Removing Corner Cases Using MobileNetv2

Without removing any image when training (*All bounding Boxes*), MobileNetv2 obtains 84.5% of accuracy. All DSAs improve the accuracy of MobileNetv2 in different amounts. DSA_2 is the one that obtains the highest (87.635) with 3.13% better result than using all images. This can be because DSA_2 is the one that detects the most corner cases. At this point, what happen if any image detected as corner case is removed independently of the DSA and the model is re-trained? The result is *All DSA Together*. In this case, there is an improvement of 3.86% over the model having all images when training. These results show that surprise adequacy can detect the corner cases and if they are removed from the training dataset, the accuracy is increased.

The second part of this section is to remove the corner cases detected from the training dataset and re-train Yolov4. This is made because MobileNetv2 has less than 15 FPS and it cannot be used for autonomous driving detection for safety reasons (ASIL D). For this reason, Yolov4 is used to check if the corner cases detected by surprise adequacy and MobileNetv2 can be used for other detection methods.

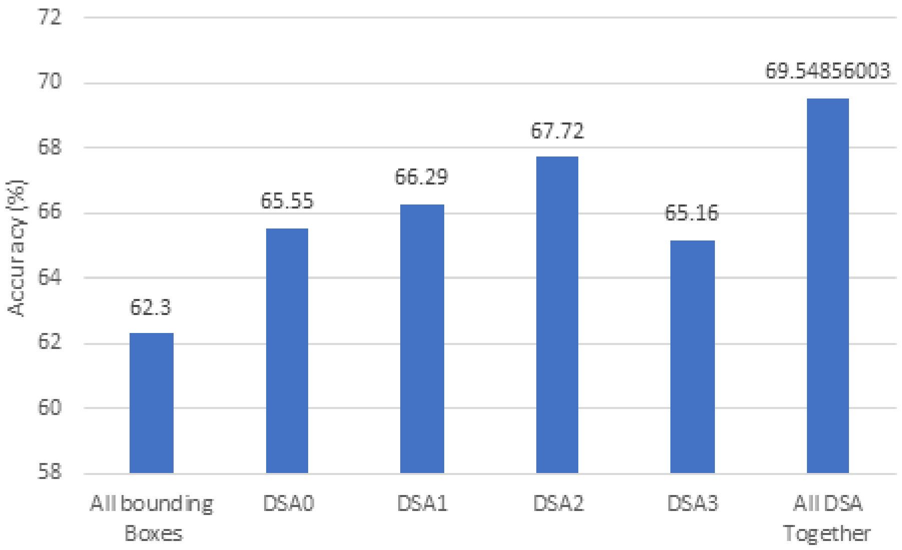
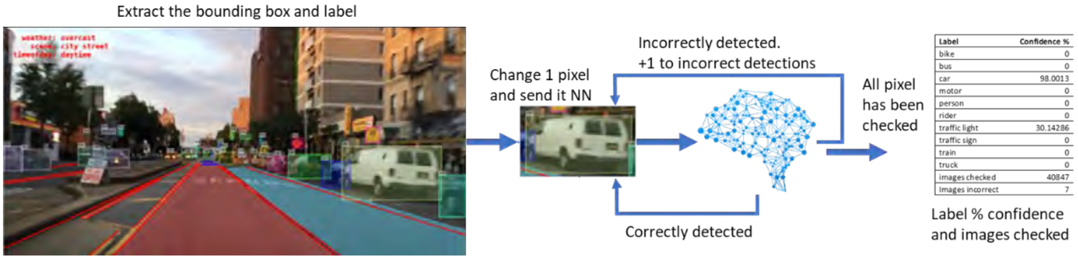


Figure 42. Model Accuracy after Removing Corner Cases Using Yolov4

Figure 42 shows the result of removing the corner cases detected with MobileNetv2 from the training dataset and then re-training Yolov4. There are increments of accuracy for all models trained with corner cases removed. The increments are similar to the one obtained on MobileNetv2 results. This demonstrates that surprise adequacy can detect corner cases using one neural network and the corner cases can be used to increase the robustness of another neural network.

It is necessary to note that surprise adequacy is used in this case to improve the accuracy, however, the correct functionality of surprise adequacy is to increase the robustness of the models. The improvement of accuracy in this case is because BDD100k has too many *car* labels and the highest number of corner cases detected are cars. In a normal case, the robustness might be reduced because the corner cases have been removed from the dataset and the training process is losing information. However, in BDD100k, removing the corner cases make the detection models more confident when they are detecting cars. This definition is made because the boundary among *car* labels and others is removed.

1-pixel change



Label	Confidence %
bike	0
bus	0
car	98.0013
motor	0
person	0
rider	0
traffic light	30.14286
traffic sign	0
train	0
truck	0
images checked	40847
Images incorrect	7

Figure 43. 1-Pixel Attack Method Using BDD 100k

Figure 43 shows how 1-pixel attack is working with BDD100k training dataset (see also Appendix C). First, for each bounding box in the dataset, an image delimited by the box is extracted. Then, 1 pixel of the images is changed. Finally, the modified image is sent to the detection model, and it determines if it still can recognize the object correctly. For changing the color of the pixel, the solution designer decided to swap the RGB color of the pixel to the opposite RGB color of that pixel.

Table 25. Confidence in 1-Pixel Attack

Label	Confidence % by the NN
bike	0
bus	42.86
car	62.59
motor	21.68
person	0
rider	0
traffic light	0
traffic sign	29.48
train	0
truck	95.15
images checked	86390
Images incorrect	68327
Failed Percentage	79.09133

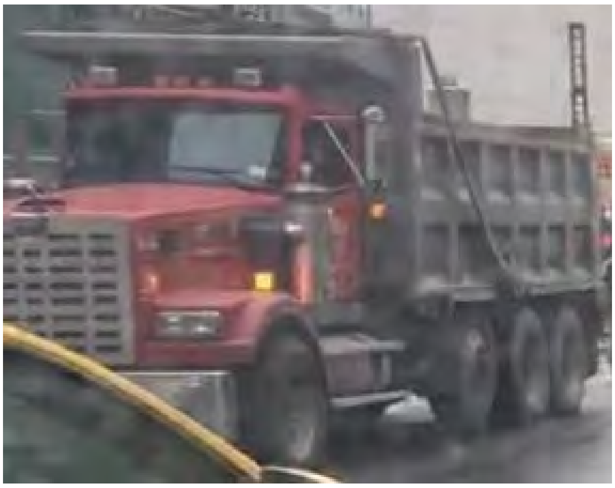


Figure 44. 1-Pixel Attack Truck Example Image

This process is made per each pixel of the bounding box. Keep in mind that only 1 pixel is changed in the image every time. After all pixels in the image has been processed, the method summarizes the confidence label of checking all modified images and how many images have been checked. For example, for an image of 300x200 pixels, this process forces the detection model to check 60,000 images, gathering the results of the 60,000 images.

Using the bounding box of Figure 43 and Yolov4 as the detection model, 1-pixel attack produced the confidence percentage of the image as shown in the figure. It shows that the detection model recognizes the image as *car* or *traffic light*. However, when the image is recognized as *car*, the detection model is 98% sure it is a car while if it is recognized as *traffic light*, only 30.14% of confidence. For this process, the model checked 40,847 images and in 7 of them it made incorrect detection, that is, only 0.017% was affected by noise. The result shows that this bounding box is unsusceptible to noise and is good to be used for training the model.

For another example, when 1-pixel attack uses the image shown in Figure 44 and Yolov4 as detection model, the detection model can recognize it as *truck*, *car*, *bus*, *motor* and *traffic sign*. As shown in **Table 25**, the confidence of recognizing the image as *car* is bigger than 60%, a result of an incorrect training process for the detection model because this image is a corner case between *truck* and *car*. In addition, when this image has noise, the system fails about 79% of the times to recognize it properly. This image is a clear example of how 1-pixel attack can recognize corner cases and bounding boxes unsuitable to be used for the training process.

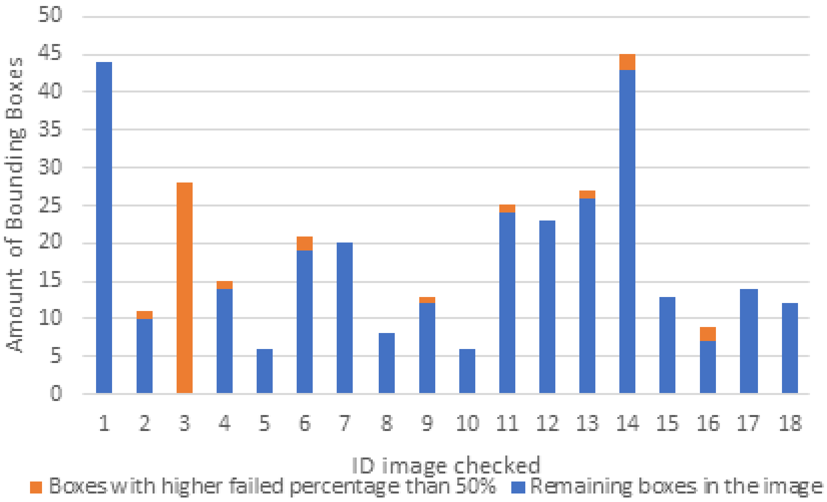


Figure 45. ABCI Experiment Using 1-Pixel Attack to Detect Incorrect Bounding Boxes

1-pixel change can detect images that are creating noise in the NN and detect corner cases. However, it is a long process, especially when the user wants to examine all the bounding boxes in the dataset. For example, the next experiment used the ABCI server (powerful server machine) for 8 hours. In that time 1-pixel attack managed to check 18 images that contain 340 bounding boxes. This is around 43 boxes per hour. There are 1,272,818 bounding boxes in the whole training dataset. That means about 29,600 hours (1233.33 days or 3.38 years) to check the whole training dataset of BDD100k.

Furthermore, after this detection of bad bounding boxes, it is necessary to deal with them, for example, by retraining the models.

Figure 45 shows the result of the ABCI experiment of using 1-pixel attack in BDD100k for 8 hours. The result shows that 11% of the bounding boxes of these 18 images are unsuitable to be used for the training process. Additionally, note that image 3 adds only noise to the detection model. Unfortunately, due to time constraints, the solution designer could not check all bounding boxes in the BDD100k dataset and, in consequence, could not re-train the detection models after removing the corner cases and check the accuracy obtained. Nevertheless, there are two ways to resolve this issue. One is to improve the algorithm, for example, adopting parallelization to make it faster or focusing only on one label instead of all labels at the same time. The second way to resolve this issue is to combine 1-pixel change with surprise adequacy. The idea is to obtain the corner cases using surprise adequacy and then use 1-pixel change to classify the corner cases detected. In this way, 1-pixel change only checks a few bounding boxes and prioritizes the corner cases that have the worst failure rates. Additionally, using 1-pixel change with surprise adequacy helps to determine which labels are close to the corner case thanks to the confidence obtained on the examination.

7.5.8 D-1: Reliability of underlying software systems

In MLQM Guideline, the term *reliability of underlying software systems* represents that the underlying conventional software (e.g., training programs and prediction/inference programs) functions correctly. This notion includes the software quality requirements such as the correctness of algorithms, the time/memory resource constraints, and the software security. Therefore, to ensure the soundness of components the solution designer needs to perform the following things, which are the highlights of this discussion:

- Check correctness of the algorithms.
- Choose well-evaluated and qualified resources while using open-source implementations.
- Solve the bugs due to frequent version-ups of the libraries.
- Check similarity between testing environments and actual operational environments.

Correctness of algorithms

Concerning correctness of algorithms, first it is necessary to understand what *correctness* means. In this case, the correctness of an algorithm is achieved when the algorithm gives expected output correctly with respect to its specification and terminates without any error. This is called *total correctness*.

For example, in this research, some additional algorithms were designed for various purposes. Namely, robustness evaluation (1-pixel change), data annotations (annotation tool) and data label conversion (the conversion of nuScene labels to BDD100k labels). The only way to prove the correctness of these algorithms is to test them using all possible inputs and analyzing their outputs. The algorithms indeed produced expected outputs with no error or warning involved.

Soundness of open-source elements

Python language has been used for developing this AI. It uses various open-source packages, which should be version compatible with each other. So, the list of used packages and their versions should be provided by the developer.

Table 26. List of Open-source Packages Used in This Research

Programing language	Version
Python	3.8.3
Package	Version
NumPy	1.18.5
TensorFlow	2.3.1
PyTorch	1.6.0
SciPy	1.5.2
Pandas	1.1.0
Matplotlib	3.3.0

To avoid any kind of problem that can occur because some libraries and dependencies may differ among the detection models, docker images have been created that contains all the libraries and dependencies of each detection model used.

Dependability of hardware in training and operational environment

The hardware used is significantly important for verification of soundness of components. Due to the complexity of object recognition and image classification models, it is recommended to use GPU for training and inference. With the current state of the research, it is possible to use the docker images without GPU when only inference is performed. However, if the user wants to re-train a detection model, the solution designer recommends using a machine or server with GPUs.

The dockers have been tested in 5 different machines and servers with different hardware specifications. It has been demonstrated that the dockers can work without any issue adapting to the new hardware.

Soundness in usage of memory

Proper documentation of maximum usage of memory during training and inference time should be recorded and evaluated so that the program does not face memory inadequacy in operational environment. Based on these records, efficient memory allocation of the device in operational environment can be accomplished.

- **Model architectures and weights:** Hierarchical Data Format (HDF) file (.h5) is one of the file formats AI developers use to store trained AI networks and their weights. When that is used, for example, the saved trained network for YOLOv4 has 490,961 parameters and takes about 52.43 MB space on the hard drive.
- **Source codes:** Different algorithms written by programing languages are part of the workflow of the machine. These codes usually do not take much space on a storage device compared to data and trained weights.
- **Input data:** Memory required for the training and validation datasets can be very large depending on the number of samples and the resolution of the images. For this research, all the images originally extracted from BDD100k video dataset have a resolution of 1280 x 720 pixels. The memory usage for the training dataset (70k images), validation dataset (10k images), and testing dataset(20kimages) are 3.77GB, 553 MB, and 1.07 GB respectively. Since input data resolution and size can change in operational environment due to different camera settings or other reasons, memory usage of input data should be recorded during training and inference time.
- **Processing Unit specifications:** Minimum RAM and GPU needed during both training and inference time should be examined and recorded.

Efficiency in training time and inference time

Time is a critical issue for an ML model when being applied to a real-life situation. In case of an autonomous vehicle, critical decisions may have to made within a split second after correctly identifying a life-threatening case. So, the lower the inference time, the better the dependability of the AI product. Inference time should be recorded for all the candidate models and evaluation should be done to check if the inference procedure is taking sufficiently little time.

Time of training should also be recorded to check if any unnecessary time loss is occurring in any part of the training program. Faster algorithms are encouraged to be used in development process.

7.5.9 E-1: Maintainability of quality in operation

According to Section 6.9 of MLQM Guideline, *maintainability of quality in operation* means that internal qualities satisfied at the beginning of operation are maintained throughout operation.

It is the process to update the system developed or incorporate new data into the training or validation dataset to improve the models. Therefore, it is required to continuously monitor behaviors of machine learning based systems and machine learning components during operation.

There are two operational patterns to keep the system updated. One is to return to the development phase and modify the whole process after adding the new changes and then re-deploying the system. The other is to update the necessary software components to keep update the system. That requires updating the necessary components in real time during operation. The second approach is quicker but the risk of getting incorrect dependencies is higher.

The following tasks are highlights of the next discussion:

- Accuracy monitoring
- Model output monitoring and input data monitoring
- KPI monitoring

Accuracy monitoring

Accuracy monitoring is a method to monitor the accuracy of the detection models and update the information according to the accuracy improvement. In this research, the accuracy of the detection models has been measured using mean Average Precision (mAP). In previous sections, we have discussed how the mAP have been monitored and used to improve the detection models. These improvements have been made improving the images in the dataset together with re-training the models.

Model output and input data monitoring

Model output and *input data monitoring* refer to the monitoring of results of inferences made by a trained machine learning model and the monitoring of its input data, respectively. In both cases, the solution designer has used two different mechanisms: 1-pixel change and surprise adequacy. These mechanisms helped to update the training dataset to detect, modify or remove the corner cases that added noise to the training process.

KPI monitoring

KPI monitoring focuses on monitoring the detection models from the viewpoint of KPI. Key Performance Indicator (KPI) is an indicator which quantifies the attainment level of functional requirements to be attained by output from machine learning components through machine learning based systems. In other words, a KPI is a measurable value that demonstrates how effectively a project is achieving key objectives. Defining KPI can be tricky. The operative word in KPI is “key” because every KPI should related to a specific project outcome with a performance measure. KPIs need to be defined according to critical or core business objectives.

To define a KPI it is necessary to reply to the following questions:

- What is your desired outcome?
- Why does this outcome matter?
- How are you going to measure progress?
- How can you influence the outcome?
- How will you know you've achieved your outcome?
- How often will you review progress towards the outcome?

This research focuses on autonomous driving, and, in consequence, the solution designer has decided to use a standard definition of security levels when defining autonomous driving machine learnings, namely ASIL (Automotive Safety Integrity Level) D. ASIL D represents likely potential for severely life-threatening or fatal injury in the event of a malfunction and requires the highest level of assurance that the dependent safety goals are sufficient and have been achieved.

For this research, the following attributes from BDD100k have been used:

Weather: rainy, snowy, clear, overcast, partly cloudy, foggy, undefined

Scene: tunnel, residential, parking lot, city street, gas stations, highway, undefined

Time of Day: daytime, night, dawn/dusk, undefined

Traffic light color: red, yellow, green, none

Are there pedestrians: True, False

Is there Zebra Crossing: True, False

Brightness: super high, high, mid, low, super low

A total of 7 attributes and 31 attribute values.

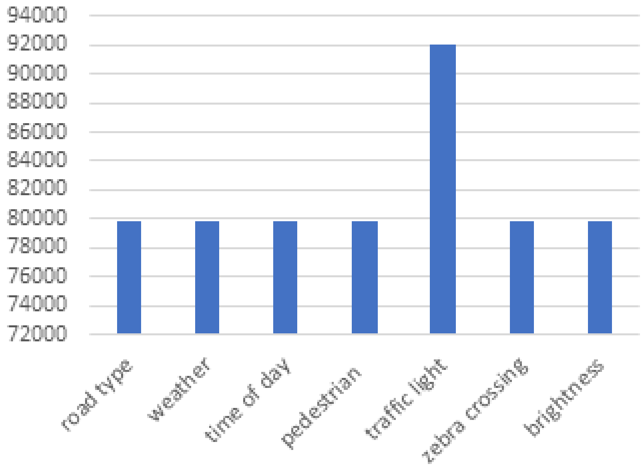


Figure 46. Number of Images per Attributes on BDD 100k

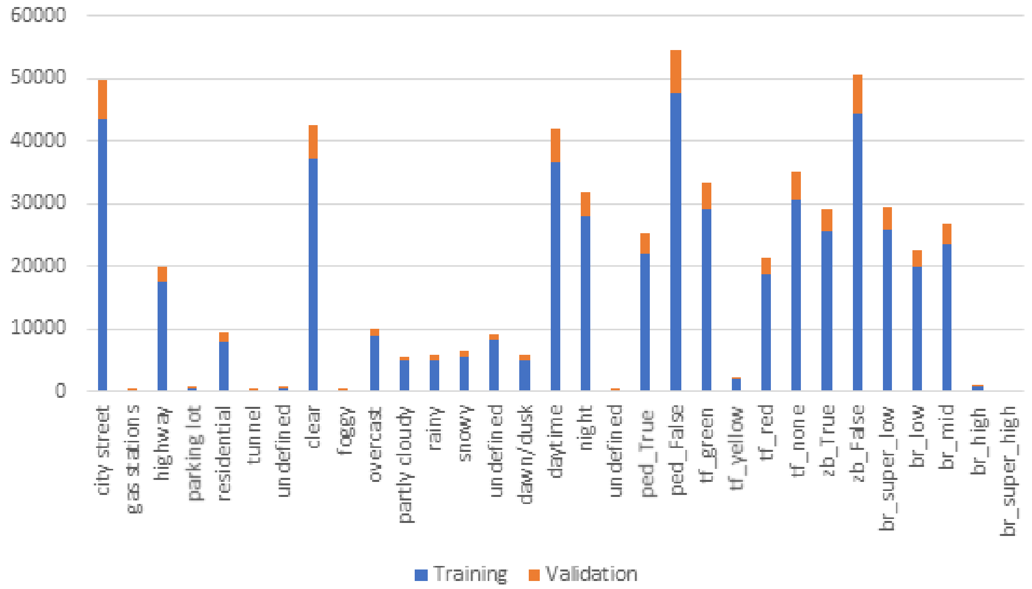


Figure 47. Distribution of Images per Attribute Values and per Training or Validation Data of BDD 100k

As you can see in Figure 46 and Figure 47, there are attributes that appear more or less often than others, such as city street (49628 images) and gas station (34). However, for the definition of KPI, we cannot use individual attribute values. We want to find situations to define where the KPI will focus. As consequence we combine the attribute

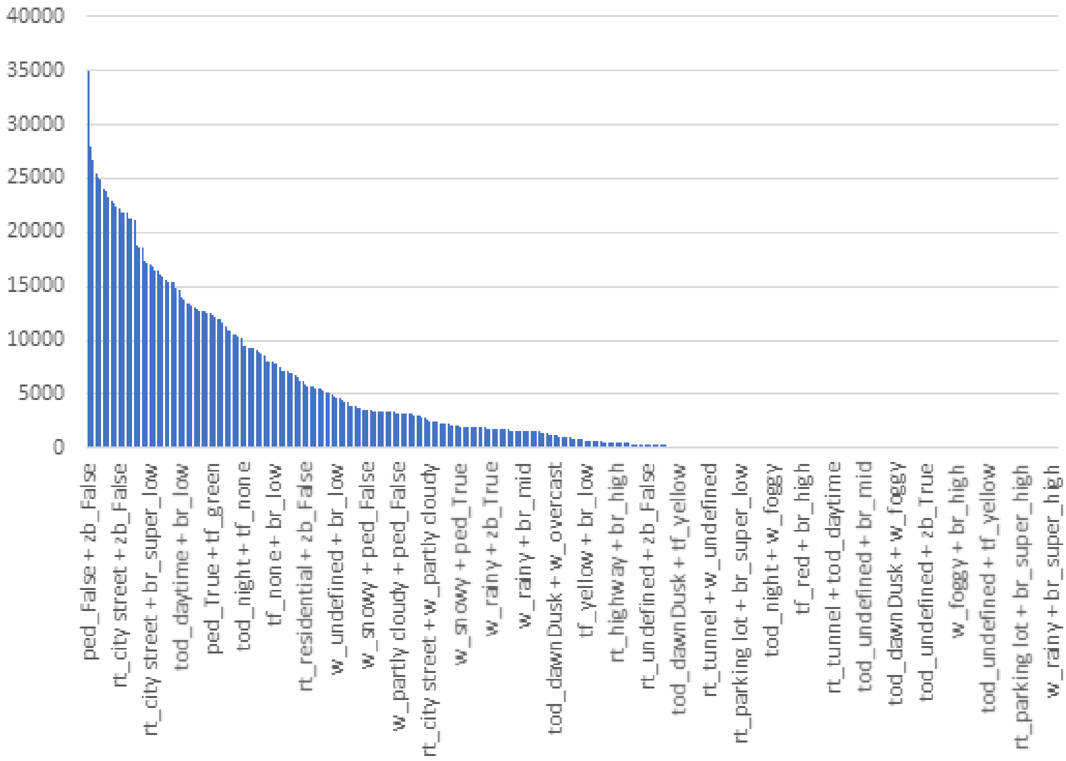


Figure 48. Amount of Images per Pair-wise Combination of BDD 100k

values in pairs. Doing this, we have 375 combinations.

Because it is difficult to correctly see Figure 48, let's focus on the first 30 combinations.

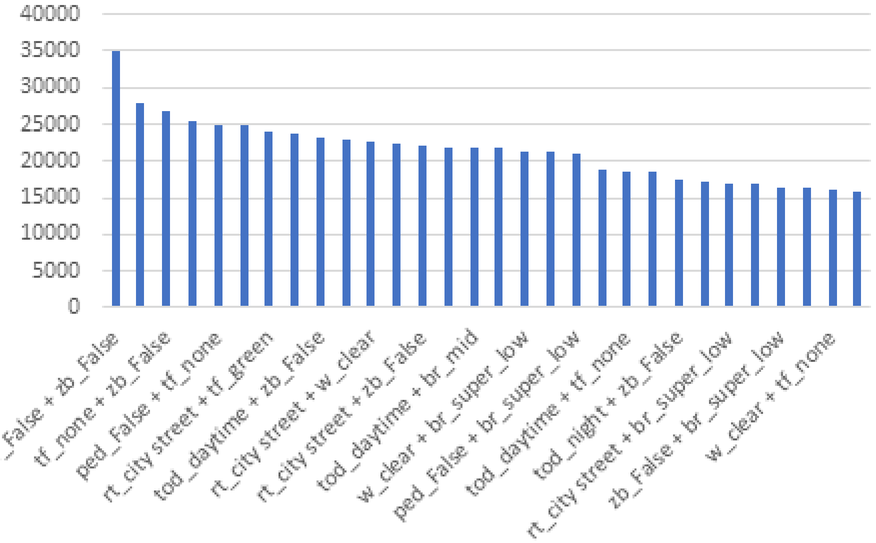


Figure 49. Top 30 Amount of Images per Pair-wise Combination of Attribute Values of BDD 100k

Because there are not enough images in all combinations, it is not possible to use all the combinations for the KPI definition. For this reason, using the amount of images per combination and ASIL D, the solution designer has focused only on *Road Type, Weather* and *Time of Day*. These 3 have more priority than the others in terms of the visualization of the images. As consequence, they are more related to ASIL D than the others. It is true that maybe *Brightness* could be included. In case that is included, it will be included in the future.

Additionally, we are not counting the attribute values of *undefined* because it is not a real situation. Then, there are 63 different combinations.

Table 27. Amount of Pair-wise Images
when *Road Type*, *Weather* and *Time of Day* are Combined

ID	Combination	Amount	ID	Combination	Amount
1	tod_night + w_clear	22884	33	rt_residential + w_partly cloudy	580
2	rt_city street + w_clear	22750	34	tod_dawnDusk + w_partly cloudy	570
3	rt_city street + tod_daytime	21811	35	rt_residential + w_rainy	487
4	rt_city street + tod_night	18748	36	tod_dawnDusk + w_snowy	436
5	tod_daytime + w_clear	12454	37	tod_dawnDusk + w_rainy	328
6	rt_highway + w_clear	10422	38	rt_parking lot + tod_daytime	228
7	rt_highway + tod_daytime	8905	39	rt_parking lot + w_clear	169
8	tod_daytime + w_overcast	7551	40	rt_parking lot + tod_night	94
9	rt_highway + tod_night	7025	41	tod_night + w_overcast	72
10	rt_residential + tod_daytime	5658	42	tod_night + w_foggy	67
11	rt_city street + w_overcast	5121	43	rt_parking lot + w_overcast	62
12	tod_daytime + w_partly cloudy	4262	44	rt_city street + w_foggy	61
13	rt_city street + w_snowy	3996	45	tod_night + w_partly cloudy	49
14	rt_residential + w_clear	3800	46	tod_daytime + w_foggy	48
15	rt_city street + w_rainy	3395	47	rt_highway + w_foggy	41
16	rt_city street + tod_dawnDusk	2950	48	rt_parking lot + w_partly cloudy	33
17	tod_daytime + w_snowy	2862	49	rt_parking lot + w_snowy	32
18	rt_city street + w_partly cloudy	2561	50	rt_tunnel + tod_daytime	32
19	tod_daytime + w_rainy	2522	51	rt_parking lot + tod_dawnDusk	30
20	rt_highway + w_overcast	2336	52	rt_residential + w_foggy	27
21	tod_night + w_snowy	2249	53	rt_parking lot + w_rainy	21
22	tod_night + w_rainy	2208	54	rt_tunnel + tod_night	20
23	tod_dawnDusk + w_clear	2004	55	tod_dawnDusk + w_foggy	15
24	rt_residential + tod_night	1813	56	rt_tunnel + w_clear	8
25	rt_highway + w_partly cloudy	1705	57	rt_tunnel + w_rainy	7
26	rt_highway + tod_dawnDusk	1439	58	rt_parking lot + w_foggy	1
27	rt_residential + w_overcast	1239	59	rt_tunnel + tod_dawnDusk	1
28	tod_dawnDusk + w_overcast	1147	60	rt_tunnel + w_foggy	0
29	rt_highway + w_rainy	1105	61	rt_tunnel + w_overcast	0

30	rt_residential + w_snowy	795	62	rt_tunnel + w_partly cloudy	0
31	rt_highway + w_snowy	707	63	rt_tunnel + w_snowy	0
32	rt_residential + tod_dawnDusk	599			

There are situations that are impossible to know or are unrealistic, such as, inside a tunnel know the weather is partly cloudy (number 62). In contrast, there are situations that are realistic and there are not enough images to be sure the accuracy of that combination. We are using a threshold of 100 images. For example, Road Type (Residential) + Weather (Foggy) [ID 52] has 27 images and the following mAP per label:

Table 28. Label Accuracy Example of the Pair-wise Combination of Residential and Foggy Attribute Values

Combination	bike	bus	car	motor	person	rider	traffic light	traffic sign	train	truck	Average	IoU
residential + foggy	0	0	57.61	0	0	0	0	100	0	0	56.98	15.76

According with this label accuracy, the combination *residential + foggy* needs more images to detect objects that are in a residential area, such as, *persons, bikes, etc.* However, this label accuracy is not correct because there are not enough images to obtain the correct accuracy.

Focusing on the remaining attributes, the following Road Type with any kind of Weather and Time of Day are candidates to be part of a KPI:

- City Street
- Highway
- Residential
- Parking lot
- Gas Stations

Regarding *Tunnel*, this attribute value is independent and needs to be checked alone in a KPI. For the remaining combination of Time of Day with Weather, all are valid. As consequence, **55 different KPIs can be created**.

Example: KPI - Residential + foggy

Figure 50 is an example of residential and foggy attribute value in BDD100k dataset. The solution designer has examined all images with residential and foggy attribute value, including the one in the example. After having the data, the solution designer is ready to define the KPI related to this combination.

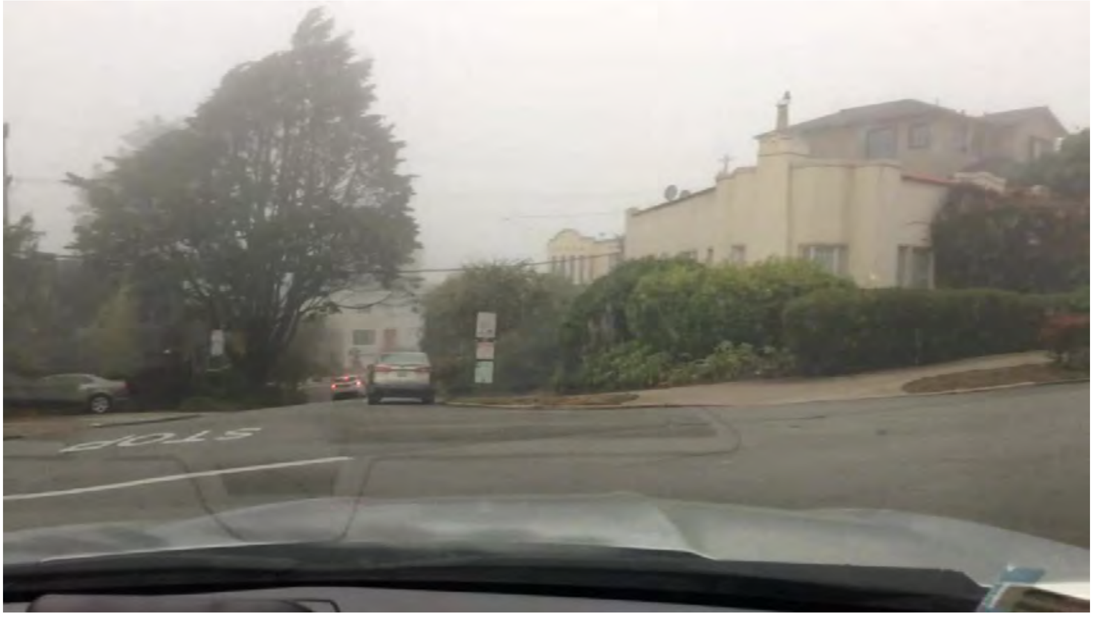


Figure 50. An Example of *Residential + Foggy* Attributes

- What is your desired outcome?
    - Detect with a minimum value of 60% all labels except *train* and *truck*. These 2 are not important because both are not allowed in residential areas.
- Why does this outcome matter?
    - Because it is important to achieve ASIL D and avoid any health problem from detection failure.
- How are you going to measure progress?
    - Using Accuracy of detecting objects and surprise adequacy.
- How can you influence the outcome?
    - Adding images of specific labels and attributes that are low. For this purpose, we will use augmentation data. In this case, with the augmentation, we only found 21 more images that shares *residential* and *foggy* attributes. It is possible to force the augmentation mechanism and obtain more of these images.
    - Additionally, we can include more images if we use image processor applications to modify the current images in the dataset and add the modified ones to the dataset.

    - In case of bad images such as images impossible to see anything, we can remove those images because they have too bad accuracy (<30%) and they only add noise to the training process.
    - Train individual detection model to work only with *residential* and/or *foggy* situations.
- How will you know you've achieved your outcome?
    - When the augmentation of images is achieved
    - When the accuracy is better than 60% in all labels or at least in almost all of them.
    - Check the trained detection model using images from other datasets, such as nuScene.
- How often will you review progress towards the outcome?
    - Every time a new KPI is checked.
    - If a new group of images with *residential + foggy* attribute from outside the dataset is used for inference and one of the labels (except *truck* and *train*) obtains lower accuracy than 60%

With these questions answered, the KPI of residential and foggy combination is defined and it can be used for the solution designer or/and anyone that want to achieve robustness in BDD100k. In the future, all KPIs definition will be tested in other dataset to check how well it performs. In case the KPIs will need to be updated, they will be.

7.6 Glossary

Below is a list of technical terminology from the protocol that is described in Chapter 7. For definitions not mentioned here, Section 2.3 of MLQM Guideline can be checked.

Object Detection

The task of identifying the presence and location of an object in an image.

Annotation

A set of ground-truth data identifying the area, location, and type (class) of an object in a given image; usually given as a rectangle enclosing the object (bounding box) and a label.

Dataset

A set of images and annotations used to train and validate a machine learning algorithm.

Training

The process of teaching a model to detect objects by iteratively providing it with images and annotations from a subset of the dataset and correcting its parameters to improve results.

Validation

The process of evaluating the ability of a model to detect objects on a portion of the dataset that has not been seen during training.

IoU (Intersection over Union)

Also known as Jaccard index, IoU is the ratio between the size of the intersection of two sets and the size of their union. In the context of object detection, it measures the accuracy, in terms of location and size, of a predicted region with respect to a human-annotated region of interest.

Confusion Matrix

A matrix reporting the number of True Positives (TP, existing predictions with corresponding ground-truth objects), False Positives (FP, existing predictions without corresponding ground-truth objects), True Negatives (TN, non-existing predictions without corresponding ground-truth objects) and False Negatives (FN, non-existing predictions for existing ground-truth objects).

Precision and Recall

Two measures of the object detection accuracy of a machine learning algorithm; defined from the confusion matrix.

Here, Precision = $TP/(TP + FP)$ and Recall = $TP/(TP + FN)$.

mAP (mean Average Precision)

A measure of object detection performance; given an IoU threshold, it is calculated as the mean of the average precision values for all classes of objects in the evaluation dataset, computed for predictions with IoU above the threshold. Average precision for all detections is the area under the precision vs. recall curve.

FPS (Frames Per Second)

The number of images that a machine learning algorithm can evaluate in one second.

KPI (Key Performance Indicator)

A specific target mAP that should be achieved by a machine learning algorithm.

Contender Model

A machine learning algorithm that is being considered as an object detection method for an automated driving system.

Candidate Model

A machine learning algorithm selected for development from a set of contender models.

8 Visual inspection of metal casting

8.1 Business requirements

8.1.1 Background

In manufacturing process, some newly produced products may have defects due to various reasons. For example, in the surface of metal casting products, there exist small holes, crack, break and other defects. These defects negatively affect not only the appearance but also other qualities of the casting products. Therefore, defect detection based on visual inspection is a significantly meaningful topic in manufacturing. To detect these defective products, all industries have their quality inspection department. But the main problem is this inspection process is usually carried out manually. It is a very time-consuming process and due to variation or instability of human accuracy, this is not 100% accurate. If the ratio of defective products does not match the customer's requirement, it may bring a big damage to the company. Therefore, there is a growing need for using machine-learning based methodology for the automated defect inspection.

8.1.2 Purpose/objectives

- realizing the fast and automatic defect detection on every newly produced metal casting product
- classifying the types of detected defects for equipment maintenance.
- realizing a higher accuracy on casting defect detection than conventional manual detection, and reducing the human and economic cost on quality detection
- realizing to detect product defects under human-unfriendly environments (e.g. night, strong reflection), and improve the safety on manufacturing process.

8.1.3 Stakeholders of the AI system

- Manufactories
- Casting product users

8.1.4 Initial demands of the stakeholders

- Manufactories as well as casting product users expect this system can capture most of defective products so as to reduce cost of unqualified products and increase revenues.
- Moreover, manufactories expect to have casting products' test reports generated by the AI system to understand the quality of this series of casting products.

8.1.5 Details of the business requirements

Assumptions

- Images captured from metal casting product's surfaces are enough to train a high accuracy AI model

Dependencies

- From the perspective of operation, the camera for image capturing should be set up on the top of casting product assembly line to take the top-view images
- The captured images should be taken from the right angle
- Images input to the AI system should be free of noise due to dusts.

Constraints

- Financial constraint: the overheads of AI system's development and maintenance should be lower than that of detection by human inspection.

Functional requirements

- To recognize defective casting products

Non-functional requirements

- The accuracy should reach an adopted standard.
- In the recognition process, different error criteria should be satisfied, e.g., false alarm rate and hit rate, should also reach the threshold given by the standard.
- The system should be robust to images with different resolutions under the domain
- The system should be robust to images captured from different brands of cameras.

Out-of-scope issues

- Locating the positions of defects on casting products' surface is currently out-of-scope of the system.

Risk and safety related concerns

- Unreliable AI systems may lead to shipment of unqualified products and unfulfilled expectations of customers.

8.1.6 Requirements concerning external qualities

The expected level of quality requirements in terms of three major external qualities for the AI software to be developed are given below:

Safety

- There is no concern about physical injury related to this system. Consequently, there is no requirement on that point, either.

- Shipping too many unqualified products brings negative impact on business. To prevent such shipment, the ratio of undetected defects must be sufficiently small.

Performance:

- The system should satisfy commonly used standards in metal casting industry. To this end, defect detection should perform according to the standards.
- If the ratio of false rejection of qualified products is large, yield ratio drops or human confirmation may have to be introduced, which defeats the purpose of automation. Therefore, the false rejection ratio must be sufficiently small.

Fairness

- There is no concern for the fairness of the system, and there is no requirement on that point, either.

8.1.7 Identifying the levels of external quality to be achieved

Table 29: The levels of external qualities to be realized.

External Quality	Additional specification	Assumed severity	Realized level
Safety	AI Safety Level for human-related risks	No physical damage is expected	AISL 0
	AI Safety Level for economic risks	Minor loss of profit; possible to avoid through monitoring by humans	AISL 0.2
Performance	AI Performance Level in general	KPIs will be identified beforehand but thresholds for each KPI may deviate based on other factors and best efforts will be provided	AIPL 1
Fairness	there are no identifiable requirements for fairness of the product or service		AIFL 0

8.2 Quality management steps

In visual inspection study, there are a lot of applications, like fabric inspection, manufacture product inspection, car defect inspection, and so on. In this Reference Guide, we could focus on one specific application, namely focusing on the metal casting products and detecting the product's surface defects. The dataset is based on the casting defect data [21]. Casting defect is an undesired irregularity in a metal casting process. There are many types of defects in casting like blow holes, pinholes, burr, shrinkage defects, pouring metal defects, metallurgical defects, etc. The selected dataset contains total 7,348 image data. All these photos are top view of submersible pump impeller, and with the size of (300*300) pixels grey-scaled images. This dataset is already split for training and testing into two folders. Both train and test folder contain def-front and ok-front subfolders, as below

train: 3758 images def-front and 2875 images ok-front

test: 453 images def-front and 262 images ok-front

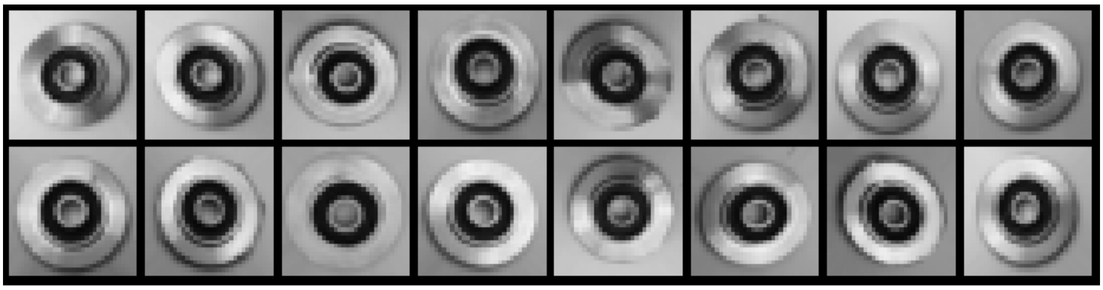


Figure 51. Examples of casting defect data

8.2.1 A-1: Sufficiency of problem domain analysis

According to the general definitions of *sufficiency of problem domain analysis*, in visual inspection applications, the requirements can be materialized to several aspects. For example, from the perspective of algorithm and structures, it is necessary to figure out suitable machine learning algorithms/structure for visual inspection problem, and the constructed visual inspection AI systems should be executed in real world situations. Requirements from the perspective of data are the definition of visual inspection problem domain, data coverage to all possible situations, and definition of high-risk cases. From the perspective of execution, setting KPIs is also needed at the PoC phase.

AI model requirements

If taking the casting defect data for visual inspection, the related problem involves defects detection and classification. Since the dataset contains only two categories of images, such as defect and normal data, it is a standard binary classification problem. Therefore, the general classification models like CNN [22], MLP [23], and ResNet [13] can be applied for visual inspection on casting defect data, and major components involving in these AI models include convolutional layers, max-pooling layers, full-connection components, decision-making layer (namely classification layer). The KPIs like precision, recall and accuracy are also useful for evaluating the constructed AI models' performance.

Data type requirements

Furthermore, there are also important requirements for types of machine learning data. The first is the type of inputs for AI modeling. Different with attributes for problem domain description, input for AI modeling should have concrete values and types (discrete or continuous). For the visual inspection system for the casting defect data, it requires that input are 1-channel grey format images, the targets are binary class labels representing defect or not.

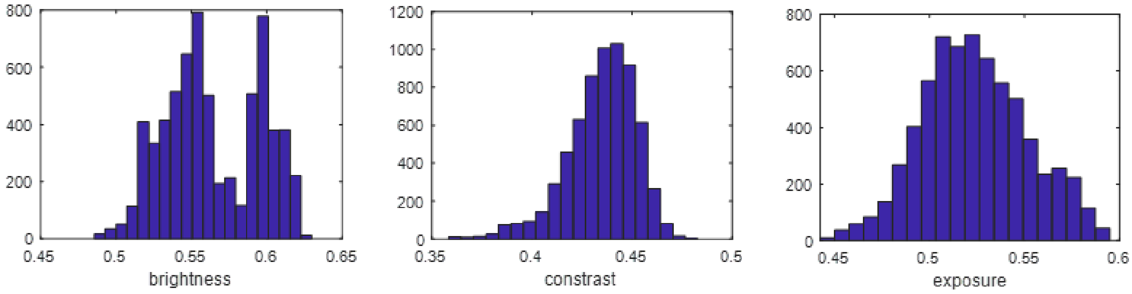
The second is the type of attributes for problem domain description. This requirement is more general, and both discrete and continuous values are acceptable. Usually, discrete values are more suitable to describe problem domain's natural characteristics, e.g., strong light intensity, weak reflection and so on.

Identifying relevant attributes and attribute values

Based on the given casting defect data, we could identify some relevant attributes which would be useful to define the problem domain of the specific AI system. Considering the modeling process of developing a machine-learning-based system on metal casting data, input involve only top-view images of a given casting product. Therefore, some image attributes could be defined. For example, in the casting defect data, we can declare attribute requirements as follows

- Type: defect, free
- Brightness: strong, medium, weak
- Contrast: strong, medium, weak
- Exposure: strong, medium, weak

Considering the casting defect data has only two types, such as defect and free images, here we can describe its problem domain with attributes like brightness, contrast, and an additional attribute exposure [24]. The distributions of those attributes are shown in Figure 52.



(a) brightness (b) contrast (c) exposure

Figure 52. Distributions of the attributes for the casting defect data.

Since three attributes in Figure 52 are continuous, so we can design their domain for vision inspection requirement analysis, as presented in the following table. Meanwhile, with suitable thresholds, we can divide them into strong, medium, and weak cases for vision inspection problem description.

Table 30 Problem domain of the given attributes

Attribute	Brightness	Contrast	Exposure
Domain	0~1	0.4~0.7	0.1~0.9

KPI requirements in the POC phase

Furthermore, in the POC (Proof of concept) trial phase, usually the developers could propose some KPI requirements for the constructed machine learning based visual inspection systems. These KPI requirements can be made out from different perspectives, e.g., models, data, and performance.

8.2.2 A-2: Coverage for distinguished problem cases

For visual inspection applications, the quality *coverage for distinguished problem cases* involves three aspects. Firstly, it involves combination design based on visual inspection described in the previous section. Combinations are realized by attributes, and lead to division of cases (situations) for quality assurance. Secondly, this quality also involves the soundness of cases, which is directly related to the amount of data samples in each case. If a case has few samples, it may be regarded as unsound in visual inspection. Finally, this quality deals with analysis of high-risk cases, e.g., data samples in special situations may lead to incorrect defect recognitions or false defect detection.

Attribute combination and case division

The first content in the study of *coverage for distinguished problem cases* could be the design of data cases based on attribute combinations. As we know, the problem domain of visual inspection applications is constructed based on some description attributes. However, not all of values on each attribute are reasonable, moreover the combination of two attribute values may be also unreasonable even though these values reasonably belong to their attribute domain. In this way, analysis on case division and attribute combination is required in this stage.

For example, taking the casting defect data for experiments, according to the distribution of attributes in the problem domain design stage, we can design case division in the problem domain for AI quality assurance.

Let's assume we divide the problem domain for each attribute (brightness, contrast, exposure) into 10 sub-cases. The cases with different attribute combination are summarized as below.

Table 31. Case division with different number of attribute combinations.

	Amount
One combination:	30 cases
Two combinations:	300 cases
Three combinations:	1000 cases

Corner cases/high-risk cases

After the design of attribute combination and case division, we can obtain a number of data cases. Since this case division is actually the reflection of dividing the problem domain, these cases are more refined in data description. Moreover, we can further find high-risk or unsound data regions, namely unsound cases, based on the case division.

Unsound cases/high-risk cases

It is easily understood that unsound cases are those that should not exist in visual inspection in the real world. For example, if we normalize the values of *brightness* attribute into domain [0,1], a defect image can be completely dark (brightness=0) or completely bright (brightness=1). Such image is unsound. Moreover, with pair-wise attribute combination, too dark or too bright cases cannot have strong contrast, so these cases are also regarded as unsound cases.

High-risk cases are different from unsound cases in that they can exist in the real world for visual inspection. For example, the *blowhole* defect data indeed exists in the real world and causes problems if gets shipped undetected but is hardly detectable by visual inspection. In consequence, this type of defects must be excluded from the target of visual inspection.

Corner case data detection

Similarly, to the execution of traditional software, a dangerous condition in AI system testing may arise in processing data of corner cases, which could generally cause incorrect and unexpected behaviors [25]. For example, when a deep learning(DL)-based autonomous driving system processes corner cases of rainy weather or strong reflection, an incorrect decision may be made and lead to a crash bringing the loss of life and property [26]. Therefore, detecting corner-case samples is important in AI testing. According to the above description of corner cases, we can define the corner case set [27] as follows:

$$\boldsymbol{Corner\ case\ set}: \{\boldsymbol{x} |\ \boldsymbol{DL}(\boldsymbol{x} + \boldsymbol{perturbation}) \neq \boldsymbol{label}(\boldsymbol{x})\} \quad (1)$$

Where, *x* denotes a sample in corner case; its true label is denoted as *label*(*x*); *DL*(*) is the output class based on a given DL model. Through this definition, we see when a small perturbation is added to a corner-case data *x*, where $|perturbation| < \varepsilon$ and $\varepsilon > 0$ is a small value, the class recognized by the DL system will be different with its true label. In this way, a corner case set can include data samples with both incorrect and unexpected behaviors, e.g., boundary adversarial data and incorrectly classified data (outliers), as shown in Figure 53.

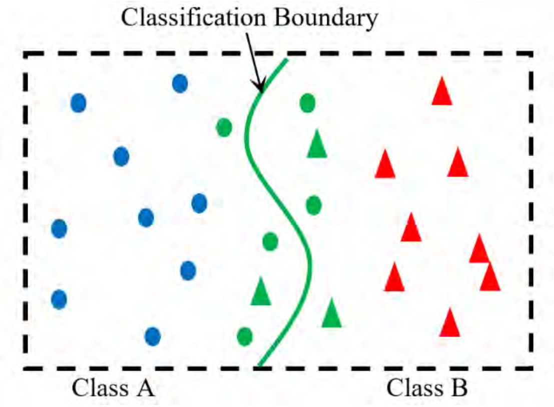


Figure 53. Diagram of corner case.

Two class of data are colored as blue and red. Data of corner cases is colored as green. They include incorrectly classified data as well as data close to the classification boundary, which tend to cause unexpected recognition.

8.2.3 B-1: Coverage of datasets

When constructing a machine learning based visual inspection system, coverage of data contains two perspectives, such as global coverage and local coverage. The global coverage mainly to consider the data's diversity in dataset design, for example training data coverage and testing data coverage. According to the general coverage definition, training data coverage aims to guarantee that enough data in the problem domain for training are available and that no inappropriate learning behavior occurs in visual inspection process due to lack of data. The purpose of testing data coverage is to evaluate the behavior of the constructed visual inspection system in the problem domain as completely as possible. Local coverage is to study the data distribution in each designed case. There are a lot of strategies of calculating the local coverage, e.g. case coverage, attribute-based coverages, neuron-based coverages and surprise coverage.

Case coverage

Directly from the coverage definition, it is easy to define a simple coverage metric from the data amount since data coverage initially describes the amount and diversity of data in problem domain. If case division divides the whole problem domain of visual inspection into refined small cases with attribute combinations, then these cases can reflect the diversity of attributes as well as the diversity of value range. Then, data amount or data percentage can be easily used for describing coverage.

Attribute coverage

Another coverage metric can be defined from the attributes used in problem domain. In this way, we could use the values of the *attributes* to characterize dataset's coverage. Since the internal logic of a DNN is mostly programmed by data, the statistical distribution of original data is very important. The coverage of each feature has great influence on the final output of a DNN model as well as the corner-cases whose output values rarely occur.

For example, given an attribute $x(n)$, the k-multisection coverage measures how thoroughly the given test dataset T covers the range $[low_n,\ high_n]$ (n specifies n-th attribute, $1 \leq n \leq m$, and m is the number of attributes). To quantify this, we divide the range $[low_n,\ high_n]$ into k equal sections (i.e., k-multi-sections), for $k > 0$. We write S_i^n to denote the set of values in the i^{th} section for $1 \leq i \leq k$.

If $x(n) \in S_i^n$, we say the i-th section is covered by the test input x. Therefore, for a given test dataset T and the feature $x(n)$, its k-multisection coverage is defined as the ratio of the number of sections covered by T and the total number of sections, i.e., k in our definition. We define the k-multisection coverage of a feature n as:

$$\boldsymbol{KMCov[x(n), k]} = \frac{|\{\boldsymbol{S_i^n} \mid \exists \boldsymbol{x} \in \boldsymbol{T}\colon \boldsymbol{x(n)} \in \boldsymbol{S_i^n}\}|}{\boldsymbol{k}} \tag{2}$$

Then, we further define the k-multisection coverage of the test set T as:

$$\boldsymbol{KMCov[T, k]} = \frac{\sum_{\boldsymbol{1 \leq n \leq m}} |\{\boldsymbol{S_i^n} \mid \exists \boldsymbol{x} \in \boldsymbol{T} : \boldsymbol{x(n)} \in \boldsymbol{S_i^n}\}|}{\boldsymbol{k * m}} \tag{3}$$

Neuron-based coverage

For neuron-based coverage metrics, the basic one is Neuron Coverage (NC) [28] which was first proposed to drive automated generation of test data. It is simply defined as the percentage of neurons that were activated by at least one input of the test dataset.

For example, assume D be a trained DL model composed of a set N of neurons. The neuron coverage of the input x w.r.t. D is given by

$$\boldsymbol{NC}(\boldsymbol{x}) = \frac{|\{\boldsymbol{n} \in \boldsymbol{N} \mid \boldsymbol{activate}\ (\boldsymbol{n}, \boldsymbol{x})\}|}{|\boldsymbol{N}|} \tag{4}$$

where *activate*(n, x) holds true if and only if n is activated when passing x into D. Several other neuron-based coverage metrics like K-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC), Neuron Activation Coverage (NAC) and Strong Neuron Activation Coverage (SNAC) are also described in the literature [29].

Surprise coverage

Surprise Coverage (SC) [19] is one kind of new coverage metric which is based on the data's surprise. For example, in the literature, distance-based surprise adequacy (DSA) is used to describe data's diversity and novelty and shown to be useful to detect corner case data. Therefore, here we use bucketing to discretize the value space of surprise and define the Distance-based Surprise Coverage (DSC). Given an upper bound of U, and buckets $B = \{b1, b2, \dots b_n\}$ that divide $(0, U]$ into n SA segments, SC for a set of inputs X is defined as follows:

$$\boldsymbol{SC}(\boldsymbol{X}) = \frac{\left|\left\{\boldsymbol{b_i} \middle| \exists \boldsymbol{x} \in \boldsymbol{X} :\ \boldsymbol{SA}(\boldsymbol{x}) \in \left(\boldsymbol{U} \cdot \frac{\boldsymbol{i-1}}{\boldsymbol{n}}, \boldsymbol{U} \cdot \frac{\boldsymbol{i}}{\boldsymbol{n}}\right]\right\}\right|}{\boldsymbol{n}} \tag{5}$$

A set of inputs with high SC is a diverse set of inputs ranging from similar to those seen during training (i.e., low SA) to very different from what was seen during training (i.e., high SA). In situations where an input set for a DL system should not only be diversified but systematically diversified, considering SA should facilitate doing so. Recent results [30] showed that more distant test inputs were more likely to lead to exceptions but might not be as relevant for testing.

8.2.4 B-2: Uniformity of datasets

Generally and also in visual inspection applications, not only we need to prepare sufficient training data for combinations of attribute values with risks which should be avoided by making correct judgments when risk avoidance is strongly sought, but also we must strike a balance between the following goals, one to provide such combinations with sufficient training data even if it entails deviation of the training data distribution from the real world, and the other to provide a training dataset whose distribution closely follows the real world.

Mathematical description

According to the above definitions, a simple way to evaluate uniformity of data could be realized on the assumption of the known distribution of data. Assuming that the real data distribution is $F_0(x)$ and the given training data or testing data has a distribution function $F_1(x)$, the difference between these two distribution functions are a possible indicator to evaluate the evenness. Generally, Kullback-Leibler divergence [31] is good way to calculate this difference, defined as

$$\boldsymbol{DKL(F_0 \| F_1)} = \int_{x \in Cov} \boldsymbol{F_0(x)} \ln \frac{\boldsymbol{F_0(x)}}{\boldsymbol{F_1(x)}} \boldsymbol{dx} \tag{6}$$

Evenness between training and testing sets

While we have known that the uniformity of data actually means the evenness of data, one meaningfull evaluation technique affecting AI quality is the evenness between training data and testing data. By taking the casting defect data as an example, we can also take image brightness and contrast, the two most common description attributes of images, as example and get the distribution of these two attributes on both training and testing datasets.

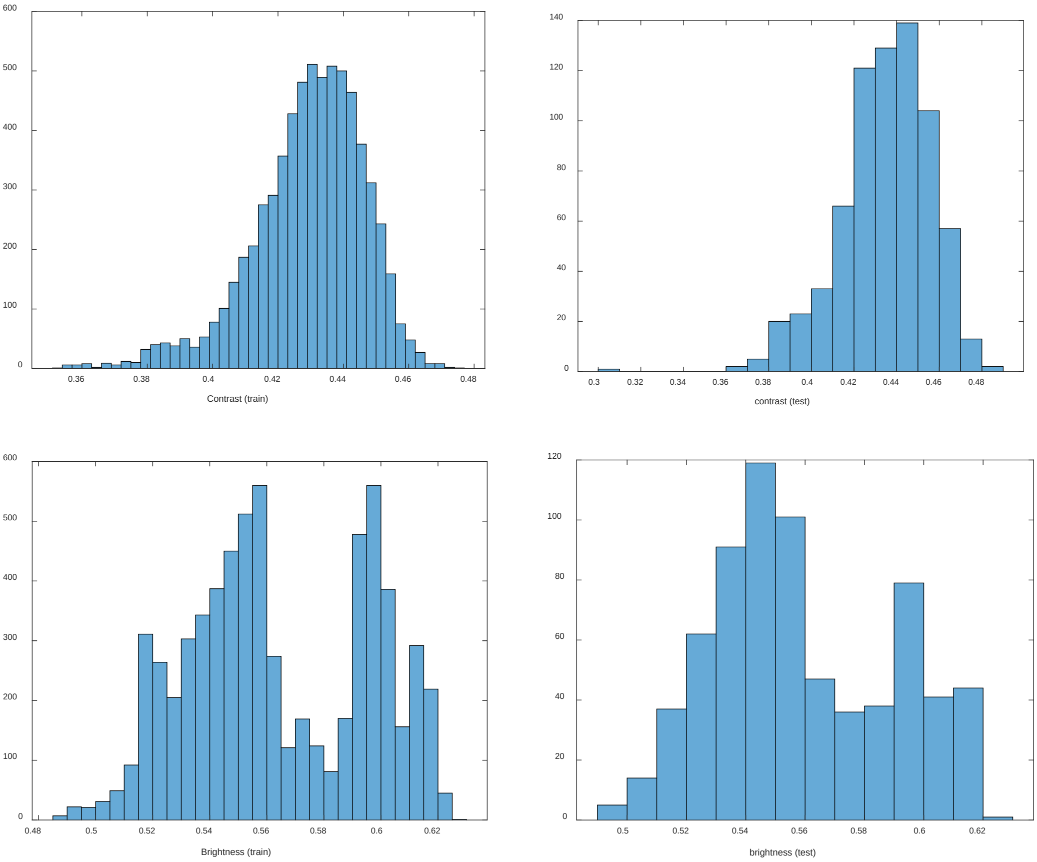


Figure 54. Distribution of attributes in training and testing datasets.

It is seen that the distribution of attributes in training and testing are also uniform, which also make the final machine learning based visual inspection system achieve similar performance.

Uniformity across cases

On the other hand, the uniformity of dataset requires to guarantee the sufficient amount of data for high-risk cases, namely the uniform data amount as common cases. In this way, it illustrates the uniformity across different cases. If data is not even, AI performance will be greatly affected, for example, the data imbalance problem reduce the accuracy in classification applications. While data collected from the real-world is generally unbalanced, subsequently small cases also have unbalanced data coverage. Considering visual inspection problems mainly involve classification, data imbalance/unevenness problem is important to guarantee AI quality.

8.2.5 B-3: Adequacy of data

This internal quality is not discussed in this version of Reference Guide.

8.2.6 C-1: Correctness of trained models

The correctness in visual inspection can be described by correct detection of defects as well as correct recognition of defect types. The visual inspection KPIs proposed in Section 8.2.1, e.g., precision, recall and accuracy, can be used for AI quality evaluation in the correctness analysis.

General correctness metrics

As we know, the major problem involved in visual inspection is the classification of defects into different defect types. Then, the general classification evaluation metrics can be transferred into machine learning based visual inspection systems.

Moreover, metrics based on confusion matrix [32] can also be used for classification evaluation, which aims at models whose output format is event or label, e.g., classification and clustering. The confusion matrix is defined below

Table 32. Confusion Matrix

		Observed Condition	
		Observed Positive	Observed Negative
Predicted condition	Predicted True	TP	FP
	Predicted False	FN	TN

From the above table, four events are defined, namely true positive events (TP), false positive events (FP), false negative events (FN), true negative events (TN). According to the number of these events, several metrics could be defined, including recall, precision, accuracy and so on. Their definitions are presented below.

Accuracy: Accuracy is given by the relation:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FP} + \boldsymbol{FN}} \tag{7}$$

However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}} \tag{8}$$

Recall: Recall can be defined as the ratio of the total number of correctly classified positive examples divided by the total number of positive examples. High recall indicates the class is correctly recognized (a small number of FN).

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}} \tag{9}$$

Precision: To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High precision indicates an example labelled as positive is indeed positive (a small number of FP).

- **High recall, low precision**: This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.
- **Low recall, high precision**: This shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

Since we have two measures (precision and recall) it helps to have a measurement that represents both of them. We calculate an F-measure which uses harmonic mean in place of arithmetic mean as it punishes the extreme values more. The F-measure will always be nearer to the smaller value of precision or recall.

$$\boldsymbol{F\_measure} = \frac{\boldsymbol{2 * Recall * Precision}}{\boldsymbol{Recall + Precision}} \tag{10}$$

By taking the casting defect data as the example, we can analyze the correctness of different models on vision inspection. As we have discussed in the first quality *sufficiency of problem domain analysis*, the vision inspection based on casting defect data is a standard binary classification problem. Here three classification models, CNN, VGG16 [33] and ResNet34, are applied to construct the vision inspection systems. Their training parameters are presented in the following table.

Table 33. Parameters of three models for casting defect data.

	Batch size	# of epochs	Learning rate	Training accuracy (%)
CNN	64	10	0.0002	99.17
ResNet34	64	10	0.0002	99.44

VGG16	64	10	0.0002	92.76

In this table, the correctness of each AI model is described by the classification accuracy, and only the training accuracy is presented. Furthermore, we evaluate these trained models on the testing data and study their performance on the correctness quality as the Guideline requires. Results of correctness analysis are shown below.

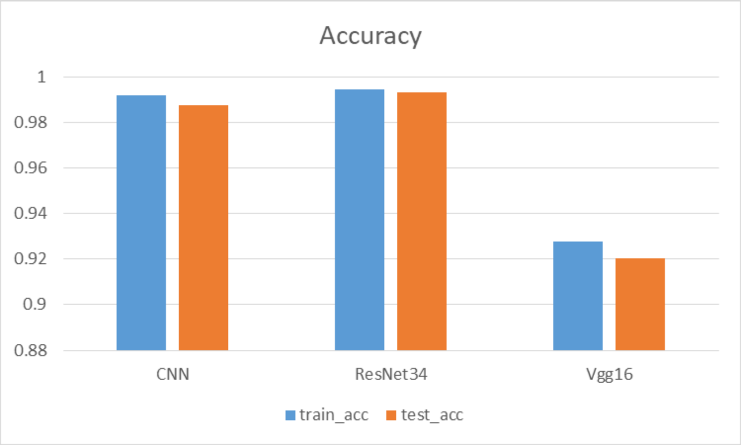


Figure 55. Accuracy of three models on training and testing processes.

Correctness on corner case data detection

According to the definition of *correctness of the trained model*, we can see this quality involves not only the evaluation on the correct behaviors, but also the erroneous behaviors. Therefore, here we propose a new metric that mainly focus on the evaluation of error-inducing data which is namely the corner case data described in *coverage for distinguished problem cases* (Section 8.2.1). To quantitatively evaluate the correctness of AI model on corner case data detection, here we propose a metric as corner-case data coverage as the following form

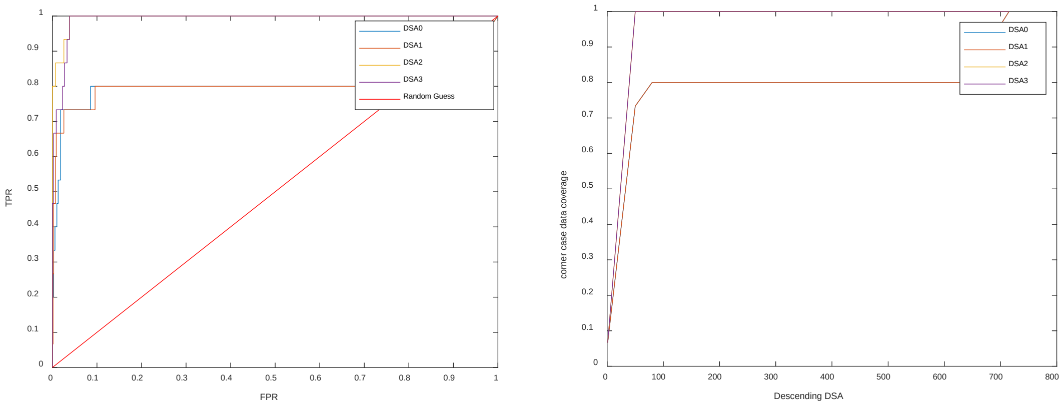
$$\boldsymbol{cov(v_{th}) = \frac{card(\{d|d \in CD,\ DSA(d) > v_{th}\})}{card(CD)} \times 100\%} \tag{11}$$

where, v_{th} is a given threshold; CD represents the dataset of corner case data; $card(\quad)$ is the cardinality; we use DSA for corner case detection here, and $\{d\}$ represents the set consisting of all detected corner case data which have DSA values larger than the given threshold.

❖ Example

For example, by taking the casting defect data as the study case, we can apply some corner case descriptors to detect the corner case data in the original data, e.g., distance-based surprise adequacy (DSA) applied here. Then, a general CNN model with 2 convolutional layers and 2 full-connection layers is constructed for visual inspection.

After modeling and training, the final testing accuracy can reach 97.90% on this casting data. Performances of corner case data detection based on DSA are shown in the following figure.



(a) ROC curves

(b) Corner case data

Figure 56. Performance of corner case data detection.

In Figure 56, four kinds of DSA modifications [27] are considered for performance analysis on corner case detection. In Figure 56 (a), the ROC curves are plotted via values of FPR (false positive rate) and TPR (true positive rate, namely recall) to evaluate performance of corner case detection. In Figure 56 (b), the proposed corner case data coverage is calculated, where X-axis illustrates data are sorted as DSA descending direction. When a specific value *Xt* is chosen at x-axis, it means there are *Xt* points with the largest DSA values are considered to calculate corner case data coverage, namely, to calculate the percentage of erroneous behaviors in visual inspection.

8.2.7 C-2: Stability of trained models

The stability of machine learning based visual inspection systems may involve the following issues. First, the adversarial attack will affect the stability of visual inspection systems. For example, when a small amount of noise is added to a defect image which is input to the trained visual inspection system, the system sometimes behaves significantly differently, namely its stability is destroyed. These noises can be either random noise in nature (e.g., dirty camera lens) or adversarial perturbation caused by malicious attacks. Second, the robustness of AI models is also a factor of stability for the visual inspection systems. For example, when a machine learning based model is overfit, the corresponding system will be sensitive to data, i.e., it will tend to make wrong decision on data far away from the distribution of the training dataset.

Mathematical definition

In software engineering terminology, the standard denotation of robustness [34] is described as: "The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions", which is similar to the stability description above. Then, we can translate this definition in mathematical languages as below.

Definition 1 (Robustness) [35]. Let S be a machine learning system. Let E(S) be the correct output of S. Let δ(S) be the machine learning system with perturbations on any machine learning components such as the data, the learning program, or the framework. The robustness of a machine learning system is a measurement of the difference between E(S) and E(δ(S)):

$$\boldsymbol{r} = \boldsymbol{E}(\boldsymbol{S}) - \boldsymbol{E}\big(\boldsymbol{\delta}\,(\boldsymbol{S})\big) \tag{12}$$

Where, *r* represents a kind of robustness measurement which actually measures the resilience of an ML system's correct output in the presence of perturbations.

A popular sub-category of robustness is called adversarial robustness. For adversarial robustness, the perturbations are designed to be hard to detect. Here, local adversarial robustness is introduced.

Definition 2 (Local Adversarial Robustness) [35]. Let *x* a test input for an ML model *h*. Let *x'* be another test input generated via conducting adversarial perturbation on *x*. Model *h* is δ-local robust at input *x* if for any *x'*:

$$\forall \boldsymbol{x}': \big|\big|\boldsymbol{x} - \boldsymbol{x}'\big|\big|_{\boldsymbol{p}} = \boldsymbol{\delta} \rightarrow \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}') \tag{13}$$

$||\cdot||_p$ represents p-norm for distance measurement.

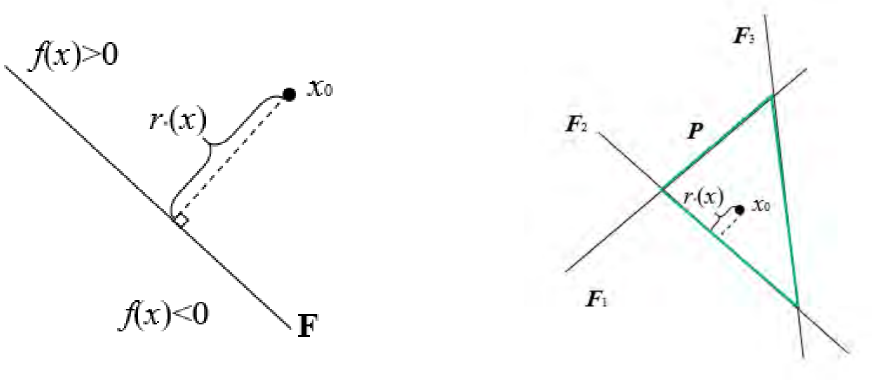
Robustness measurement methodology [36]

According to the definition of robustness, robustness measurement is actually to determine the minimum perturbation causing testing data to make a different decision. In the following Figure 57 (a), for a linear binary classifier, the minimal perturbation to change the DL model's decision-making on a given testing point $\boldsymbol{x_0}$ is the minimum distance from $\boldsymbol{x_0}$ to the hyperplane F, described as the following form

$$\begin{gathered}\mathbf{r}_*(\mathbf{x_0}) = \mathbf{argmin}\,\|\boldsymbol{r}\|_2 \\ \boldsymbol{s.t.\,sign}\big(\boldsymbol{f}(\boldsymbol{x_0}+\boldsymbol{r})\big) \neq \boldsymbol{sign}\big(\boldsymbol{f}(\boldsymbol{x_0})\big)\end{gathered} \tag{14}$$

Similarly, for a multi-classes classifier which is assumed to consist of a set of binary classifiers, as shown in Figure 57 (b). The robustness measurement of DL model to $\boldsymbol{x_0}$ can also be calculated as the minimum distance from $\boldsymbol{x}_0$ to the classification boundary.

Moreover, for an arbitrary classifier which is not linear, the robustness measurement can be calculated via iteration process. Since, at each step of iteration, the differential



(a) Linear binary classifier

(b) Multi-class classifier consisting of multiple binary classifiers.

Figure 57. Robustness of classifiers

part of classifier can be regarded as linear.

Robustness measurement with consideration of corner case data

Moreover, considering that the stability quality needs to avoid overfitting, one more useful application is to consider the corner case data's influence on the robustness analysis of AI model [37]. Different from accuracy analysis, here we consider detecting corner case data in training data and deleting the detected corner case data for model retraining, as described below.

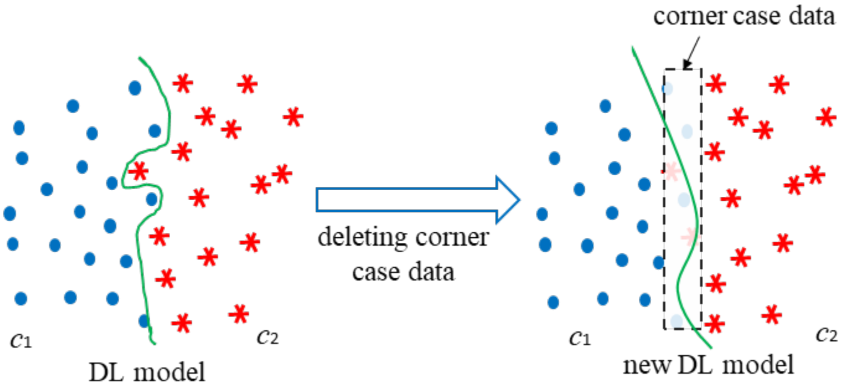


Figure 58. Robustness improvement.

In this way, the trained model will be more robust, since it gets rid of the influence of boundary points (corner case data) and reduces the risk of model overfitting. Taking the casting defect data as an example, a general CNN model is constructed for this vision inspection problem. To measure the robustness of this model, we apply the mentioned methodology above in experiments. Results of robustness measurement are shown in the following table

Table 34. Robustness measurement on the casting data

		L1	L2	L∞
Model 1	Min	0.0010	0.0088	0.3115
	Max	0.7421	7.0115	264.2126
	Mean	0.1890	1.9058	69.4177
Model 2	Min	0.0024	0.0240	0.8842
	Max	0.7653	7.0000	261.8947
	Mean	0.2026	1.9730	72.4405

To improve the robustness of AI models, the mentioned technique is also applied here. For example, we delete top-k corner case data from the original training data then retrain the vision inspection model with the remaining data. Subsequently, the robustness of the retrained model (Model2) is compared with that of the original model (Model1) as shown in Table 34. The robustness measurement (RM) is also computed with L1, L2 and L∞ norm. It is also seen that the retrained model has relatively larger values on these three kinds of RM, implying the conclusion that retraining AI models based on a training set with corner case data removed can improve models' robustness.

8.2.8 D-1: Reliability of underlying software systems

This quality needs to guarantee the dependability of models, data and the execution environment. For example, for given datasets, such as the casting defect data, there are some open-source codes available from websites. If the developers use them directly into the final visual inspection system development, they should be responsible for ensuring sufficient quality.

For the possible methods on dependability evaluation, the conventional quality management methods in software engineering can be also applied in the machine learning based vison inspection systems. However, the following aspects should be considered in this quality assurance.

First, the consistence between the developing environment and the actual operation environment. To develop a program for visual inspection problem, there are many choices for environment selection. To guarantee the final software can be executed successfully, the environment involving operation system, engine, version, is required to be determined. For example, the general deep learning system development, e.g., machine learning based visual inspection systems, can be developed based on Python.

Second, the hardware for visual inspection is an important factor affecting the system's dependability. For example, in visual inspection applications, the inputs are a series of images, so the training process is usually based on GPU in computation, even servers or cloud computing. However, when transferring this system into a real scenario for defect detection, there is no way to guarantee the local platform satisfying these hardware requirements as well. If not, the operation of vison inspection will be paralyzed. Therefore, hardware requirement should be considered in dependability assurance.

Third, memory requirements may also affect the dependability quality. This factor may determine the image size of inputs, the size of parameters, batch size for training and testing. If a large memory is used for training and developing but a small one for testing, then the dependability of the visual inspection systems cannot be guaranteed.

8.2.9 E-1: Maintainability of quality in operation

The term *maintainability of quality in operation* means that internal qualities fulfilled at the time when the operation started is maintained throughout the operation period. Therefore, internal qualities 1) can sufficiently respond to changes in external environments and 2) prevent the quality from deteriorating due to changes in trained machine learning models made for such response.

According to the definition on this new AI quality, two possible topics can be studied in the machine learning based visual inspection systems.

The first one is how to carry out additional learning. For example, in visual inspection, if a new type of defects is found, the machine learning based system should be retrained to adapt itself to this new kind of defects. The additional learning makes the visual inspection systems updated to new situations.

The second one is how to carry out the iterative training. This requirement is mainly applied for adaptive learning or online learning. It makes sense that the trained vison inspection model based on a given defect dataset is not possible to be perfect since the data amount is always finite. To improve the system's performance, we can slightly tune the parameters when new defect images or corner case data are input to the system.

9 Postal code analysis

9.1 Introduction

In this section, we will discuss the problem of *postal code analysis*. This problem refers to identifying written numerical digits which is very similar to popular MNIST digit classification problem. In the later sections, we will go in more details about the problem followed by some possible solutions.

This section has been written to accumulate many ideas and analysis done by researchers to come up with a solution to postal code analysis. The complete analysis will be explained here based on MLQM Guideline written by AIST. The goal of this section is not only analyzing postal code problem but also validating the self-sufficiency of the Guideline. The reader of this report can expect to gather knowledge about:

- Details of the postal code analysis problem and analysis of real cases
- Challenges and methods for data management
- Introducing AI to our problem
- Methods and ideas about validating data and AI models
- Thoughts about future maintenance and handling

9.2 Business requirements

9.2.1 Problem definition (use case)

AI model will identify different numerical digits (one at a time) robust to various handwritings to make the post-codes on envelope machine-readable and segregate letters and parcels according to their code. Each digit of a post-code will be identified independently; so, a supporting software will work alongside the AI to draw single digit images for inputs and to accumulate the outputs of the AI. Post office (postal services) has been chosen here as a client of the product.

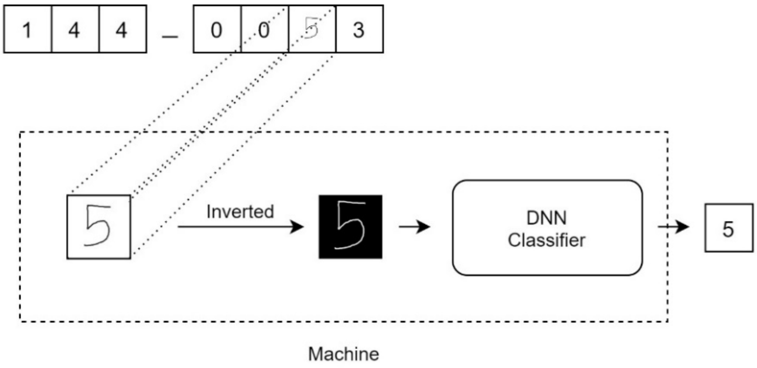


Figure 59. An overview of the workflow of Postal Code Analysis machine.

As shown in Figure 59, first, the machine takes images of each number of the code sequentially, preprocesses and inverts color. Preprocessing may include centering the number, adding more contrast etc. Then the images are passed to a trained classifier which outputs the prediction. This is a simple description of *postal code detecting machine*. In the following section, we will discuss what specifications are needed for each part of the machine.

9.2.2 Background

Postal code is a shorthand reference to help in specifying addresses for the postal service. In most countries the ZIP/postal code contains enough information to identify the city name and the state/region which not only minimizes the necessity of abundant addresses but also indicates possibility of automatic sorting of posts in quick time. This will save a lot of time and manpower of any posts company or government organization.

9.2.3 Purpose/objective

The report will mainly focus on the AI implementation of handwritten postal code detection. But there will be an input system to gather and prepare model inputs and an output system to use the model outputs effectively. The pre-processing of images and post-processing of predictions will also be defined and develop in the development phase. But if we concentrate only on the AI model, the objective can be defined as follows.

- A trained AI model will identify handwritten numerical digits from images.
- The model will consider various disturbances in the input images (selected and ranged by the developers and approved by the stakeholders).
- The AI model needs to be less complex, hence fast operation.

9.2.4 Stakeholders of the product

The automated handwritten postal code detection system has the following stakeholders:

- Postal agency (including investors and authority; the user of the system)
- Employees (operators of the system)
- Users/customers (users of postal services)

9.2.5 Initial demands of the stakeholders

- All stakeholders will want accurate outputs from the product.
- The postal agency will demand fast inference time to compete in the economic market efficiently.
- The agency will want the product to be cost effective as well for operation cost reduction.
- From service provider's (employees) point of view, the use of the product must be easy and understandable.
- To make the product user friendly, the AI needs to be capable with image variations for input flexibility; for example, use of different inks for writing or different color paper background etc.
- The postal agency may demand acceptance of a wide variety of handwritings to improve efficiency of processing mails and parcels from as many customers in the market as possible.

9.2.6 Details of the business requirements

Functional requirements

- The AI will detect handwritten digits (0-9).
- It will take fixed size grey scale/RGB images as input.
- If the AI is doubtful about a digit recognition, it should keep a log which will be checked manually later.

Non-functional requirements

- The product system needs to work fast; a postal agency will want the recognition system to output within a second.
- The model should have noise handling capability and acceptability of various styles of handwritings.

Assumptions

- For handwritten digit recognition, the model will receive as clear and noise-free images as possible.

Dependencies

- Cases in which the system has doubts and/or fails will be monitored during the maintenance phase. This helps system updates in operation phase.

Constraint

- Data constraints: To develop the initial product developers need to use only publicly available datasets.

Out-of-scope issues

- Typed fonts are considered out-of-scope for this AI model.

Risk and concerns

- Wrong identification of postal code can mislead the distribution of posts and result in financial and time loss.

9.2.7 Requirements concerning external qualities

Based on the business requirements described above we will set some external qualities to be achieved. Designed following MLQM Guideline, these external qualities are more comprehensible for developers and evaluators.

Safety

- There is no safety concern for human related risk/accident from the product.
- There is safety concern for economic loss from the product which is closely related to model's performance.

Performance

- The model needs to show equal performance in operational phase compared to development phase on defined KPIs.
- The class-wise performance should be equivalent to the overall performance.

Fairness

No fairness issues.

9.2.8 Defining the levels of external qualities

Table 35. The levels of external qualities to be realized.

External Quality	Additional specifications	Assumed severity	Realized level
Safety	AI safety level for human related risk	No physical damage for humans is expected.	AISL 0
	AI safety level for economic risk	Considerable damage to assets; possible to avoid through monitoring by humans.	AISL 1

Performance	AI performance level overall	Strong expectation that the product or service satisfies certain KPIs for the system's operation.	AIPL 2
	AI performance level class-wise	Strong expectation that the product or service satisfies certain KPIs for the system's operation.	AIPL 2
Fairness	AI fairness level overall	no requirement for fairness of the product or service	AIFL 0

9.2.9 Conclusion

This section on business requirements is written from the business owner's point of view. The business demands and external quality of the product are defined using assumption. In real case, a team consists of business or marketing persons and product developers will make all decisions and define realized level to achieve for each external quality. This section will help both developers and business owners to communicate with each other and decide on the product requirements.

9.3 Product specification

In this part, we will describe the specifications of the final product or the anticipation of client about the AI solution. For the specific problem of postal code analysis, the specifications or descriptions can be stated like below.

9.3.1 Data related specification

- **Classification using image data:** Digit identification needs to be done using only image data. We can get the images by cropping boxes from scanned documents. Later these images can be gray scaled or inverted for computational ease.
- **Ink used for writing:** People can use pencil or pen for writing. Pencils can vary at graphite grading where gel pen can spread ink over the document. So, black ball-point pen is the tool to write the numbers.
- **Declaring prohibited patterns of input (handwriting):** Since there are infinite number of hand writings, it is necessary to define some unacceptable patterns which are ambiguous for proper identification. For example, loop of '9' must be closed otherwise the pattern will be similar as '4'.

9.3.2 Model specifications

- Type of learning: Supervised
- Type of AI model: Classification
- Model architecture: CNN
- **Task to perform (identifying numbers):** Postal code consists of only numerical digits. So, the machine needs to be capable of identifying ten digits (from 0 to 9).

9.3.3 KPI specifications

- **Accuracy:** accuracy, recall, precision, F-measure etc.
- **Stability/robustness:** mutational robustness, distance-based surprise adequacy (DSA), etc.

9.4 Introduction of the datasets

9.4.1 Exploring dataset

Postal code detection or digit recognition is a supervised classification problem in the AI region. So, a dataset will be involved for training and testing purpose. To manage the dataset, we can do any of the following:

We can use any open-source dataset. For example,

- The MNIST database [38]
- USPS dataset - Handwritten digits [39]
- ARDIS – the Swedish dataset of historical handwritten digits [40]

We can use multiple datasets combined. We also can build our own dataset as per requirements.

Among all of these, MNIST is the most popular dataset of handwritten digits and so it will be used for our analysis throughout this section.

9.4.2 MNIST dataset

MNIST dataset holds 70,000 image data where 60,000 are training data and the rest are for testing. These are gray scaled images having dimensions 28X28. It is written in the description of the dataset that there were 250 different writers involved in making this. The digits were also centered by computing the center of mass of the pixels.

When we choose a dataset to work with or train a model, it is necessary to decide on the targeted region of the solution. For the postal code detection problem, we need to clarify for which country or area of the world the solution will be used. If we know that, we need to build dataset based on that region. Other than dataset, the evaluation procedures defined by MLQM Guideline are universal. We know, MNIST is a US based handwritten dataset [41], so a classifier trained on MNIST will be suitable for the US people.

9.4.3 Sample of input data

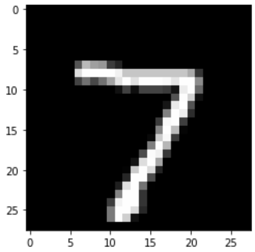


Figure 60. A sample input for Postal Code Analysis machine from MNIST dataset.

A sample of input data (image) for the model is given in Figure 60. It is a gray scaled 28X28 image with color inverted and high contrast. This is one of the images from the test set of MNIST dataset. The digit is well defined, but it is not fully centered from practical point of view. So, for many reasons, we need to analyze the distribution and orientation of the images for later use. This part is referred to as MLQM assessment.

9.5 Quality assurance procedures using MLQM Guideline

This is the section where we will show all our analysis on postal code detection. There are nine quality assessment criteria:

- Sufficiency of problem domain analysis
- Coverage for distinguished problem cases
- Coverage of datasets
- Adequacy of data
- Uniformity of datasets
- Correctness of trained models
- Stability of trained models
- Reliability of underlying software systems
- Maintainability of quality in operation

9.5.1 A-1: Sufficiency of problem domain analysis

Definition

Sufficiency of problem domain analysis means analyzing all possible characteristics of real data that are to be input to the machine learning models. The primary target is to cover all possible cases in real world. We also analyze and define impossible case scenarios if there is any.

Requirements are the first thing we decide before starting with the solution. In postal code detection, the major goal is identifying handwritten digits from images but that is not all. We know there are only 10 digits to recognize, but when it comes to handwritten digits, the variety can be vast. So, defining expected features of the input images and establishing a problem domain is a pre-requisite. For postal code detection, since the solution will be using AI, we will expect the machine to be equivalent to human capability. So, we can say, "Machine needs to identify all the digits which human eyes can identify." Next, we will discuss every requirement step by step.

Data for all possible classes

In the dataset, I need uniform distribution of all the digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) both in train and test set.

Selecting well-defined feature dimensions

There are lots of features both in images and in handwritings which are responsible for creating variances among the same numbers. Here, we have tried to list all of them.

I. **Position of the digit:** This refers to the location of the writing in the image grid.

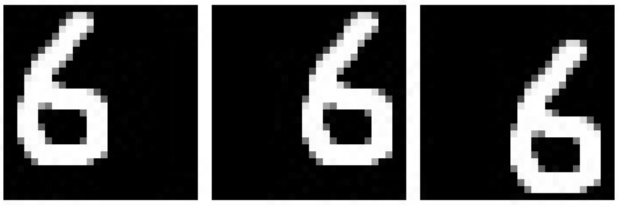


Figure 61. Different positions of the digit.

II. **Area:** It means what is the dimensions of the digit in proportion to the image.

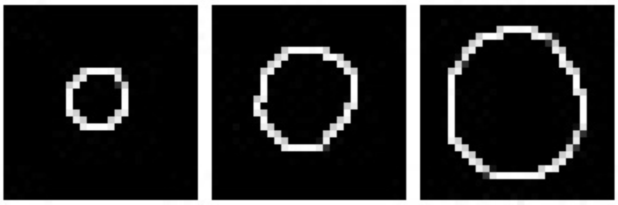


Figure 62. Different areas of the digit.

III. **Length:** It refers to the length of the pen stroke. In the following figure '1' has small length while '8' has large length.

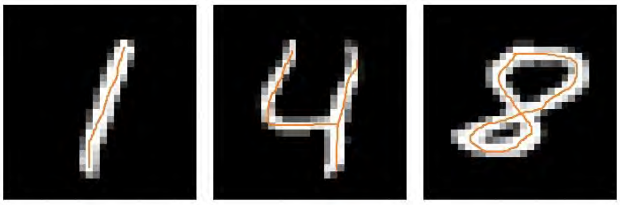


Figure 63. Different length of the digit.

IV. **Brightness/contrast:** It means, how bright or sharp the image is.

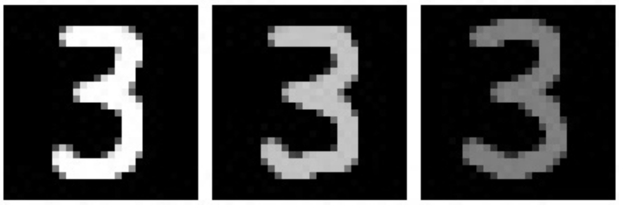


Figure 64. Different contrast of the digit.

V. **Straight/tilted:** It refers to the orientation of the digit.

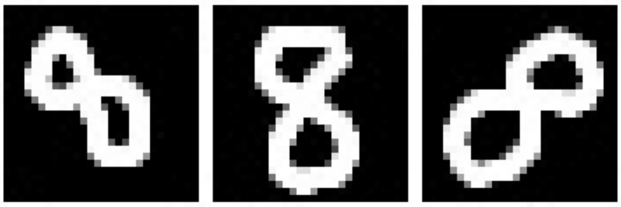


Figure 65. Different tilt of the digit.

VI. **Boldness:** This is self-explanatory. It refers to the thickness of the writing.

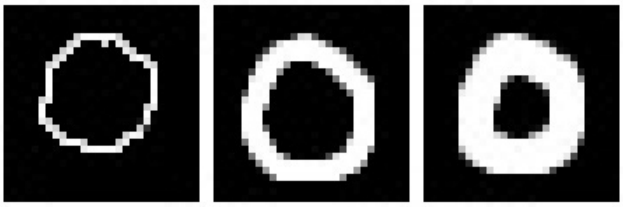


Figure 66. Different boldness of the digit.

VII. **Handwriting:** There are a lot of handwritings proportional to the number of populations and this feature will vary among digits. So, a complete analysis will create several (10) digit specific problem domain. We are presenting some

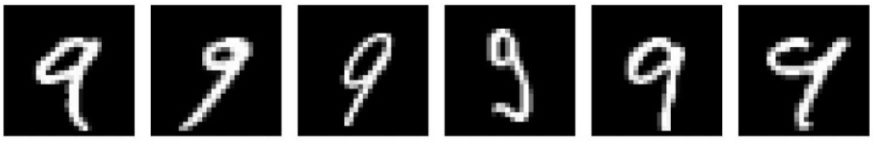


Figure 67. Different types of writing '9'

examples of different writing style of number '9'. In the next step, we will define some features and their values to describe these writing types.

Selecting ranges within and out of bounds

In this section of analysis, we need to define values or a range of values as region of interest for each chosen feature. Here, we will describe five digit-independent features for any handwritten digits (grey scaled images) followed by a digit specific problem domain.

I. **Position of the digit:** The image frame can be divided into four basic regions like Figure 68. This feature can be avoided if we can crop out the digit from any corner of the image.

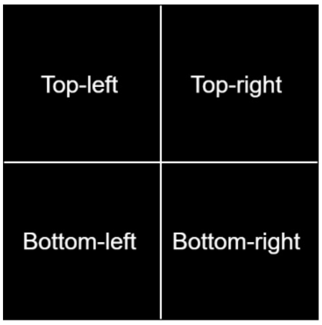


Figure 68. Four regions of an image.

II. **Size of the digit:** We can consider 3 different sizes of digits; Small (0-25%), Medium (26-50%) and Large (51-75% of the image frame), assuming 76-100% ratio of the bounding box to the image does not occur. This feature can also be avoided by cropping out regular size digits or by padding some extra pixels.

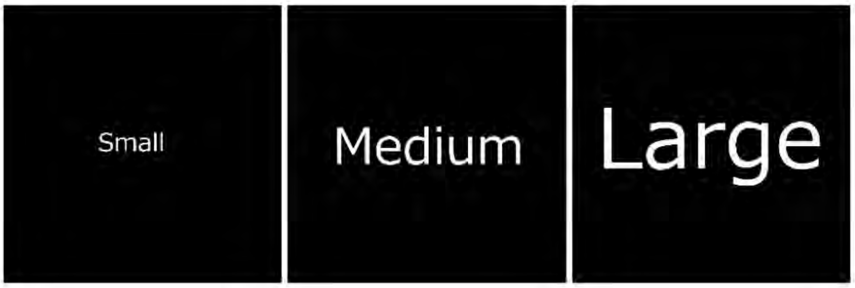


Figure 69. Different size of the digit.

III. **Brightness/contrast:** It is the measure of saturation for both black and white. When the contrast is low, edge detection of the digit becomes tough. We can consider just two variations in contrast.

Figure 70. Different contrast of the digit.

IV. **Straight/tilted:** The categories for this feature can be straight, tilted right and tilted left. There is an opportunity of describing this feature via angles, but it will complicate the dimension and we can leave it to the learning process. Here, we are considering just three variations of orientation.

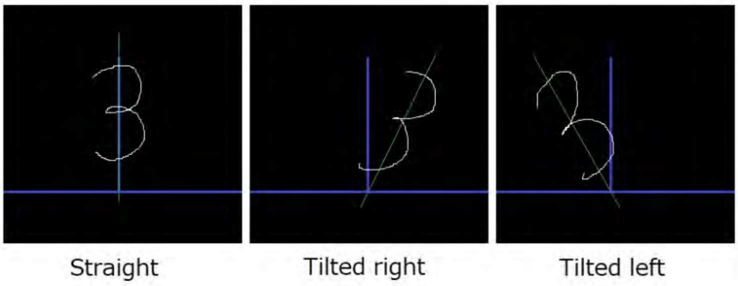
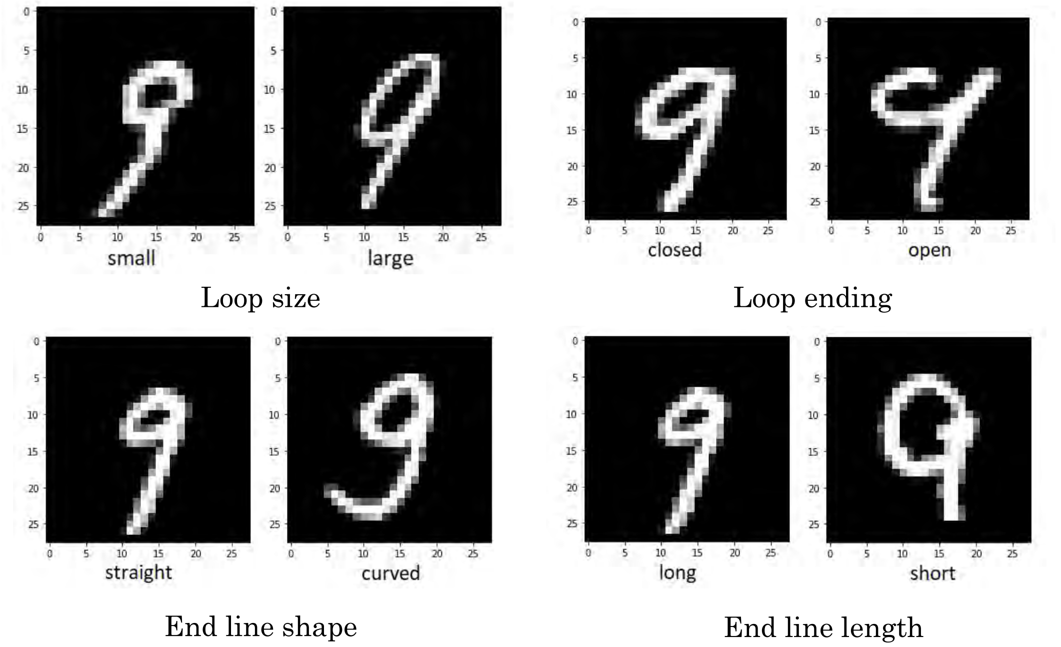


Figure 71. Different tilt of the digit.

V. **Boldness:** This feature depends only on the ink width of the pen. Here, we will consider 3 levels of line thickness. Narrow, medium and wide. We can also choose a numerical range if we consider number of non-zero pixels as approximation to boldness. Here, we have chosen 5-20% pixel ratio digits as accepted or expected.

Figure 72. Different boldness of the digit.

VI. **Handwriting:** This is not a single feature but a common identity for every digit specific problem domain. It holds as much as 10 separate domains varying in feature number as well as feature length. The detailed problem domain for digit '9' is described in this scope. Based on different types of handwriting, we have come up with the following features that can occur while writing '9':



Loop size

Loop ending

End line shape

End line length

Figure 73. Different types of handwriting.

Summary of the problem domain

Here, we will sum up the entire problem domain to visualize briefly. After the above analysis, the domain looks like below:

Table 36. The summary of the problem domain.

Feature dimension	Feature values
Position of the digit	Top-left, top-right, bottom-left, bottom-right (4)
Size of the digit	Small, medium, large (3)
Brightness/contrast	High, low (2)
Straight/tilted	Straight, tilted right, tilted left (3)
Boldness	Narrow, medium, wide (3)
Handwriting (digit specific)	'9': Loop size: small, large (2) Loop ending: closed, open (2) End line shape: straight, curved (2) End line length: long, short (2)

Conclusion

After this stage of quality assurance, the analysis of problem domain is complete. Every possible real-world data can be fit into its dimensions. In the next section, we will focus on cases i.e., combination of attributes or features and check the validity of such combinations.

9.5.2 A-2: Coverage for distinguished problem cases

Definition

Problem cases or combinations of features are necessary to look upon before further data analysis. It is because the later analysis will show data availability for each case and help us find the rare or corner cases. But there is a possibility of impossible cases as well for which confirming data availability is unrealistic. Here, we will discuss such cases for our chosen problem and select the valid cases for data analysis.

Identifying unsound (never to occur) cases

To select the valid/sound cases, it is easy to identify the unsound cases first. An unsound case appears if the defined features depend on each other. For example,

- If it is raining, the ground will be wet (driving dataset).
- If a house doesn't have pool, there will be no question of pool quality (house price dataset).

For postal code detection, the features defined for the problem domain are independent of each other. Hence, all combinations of the features are valid. Based on the described feature space, the number of combinations can be counted. Taking only digit independent features into account,

-

Conclusion

After problem case analysis, we have got many possible cases which cannot be fully analyzed within this example report. We will perform later analysis on 1 or 2 possible cases.

There is only one possible unsound case for this problem which is erroneous annotations of digits in the dataset.

9.5.3 B-1: Coverage of datasets

Definition

Coverage of dataset i.e., data coverage means availability of data points across the feature space. Increasing or fulfilling the coverage criterion is the work of developers. As an evaluator, we need to identify empty spaces in problem domain. In this section we will analyze the coverage of problem domain and give knowledge about required data. Various procedures are described here associated data coverage.

Determining data coverage

Here, we will do coverage analysis on *Area* feature using MNIST dataset. The feature is defined as the area of the minimum parallelogram covering the digits. The mathematical domain of this feature is [0,784] since the size of MNIST image data is

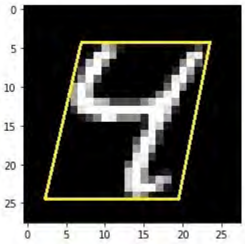
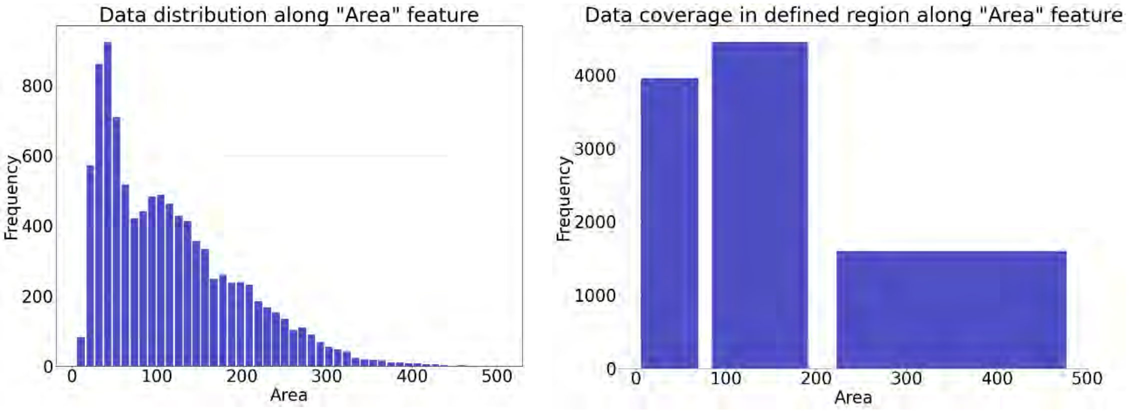


Figure 74. A sample bounding box defining the measurement of *Area* feature.

28X28. According to our defined problem domain, *Area* feature contains three levels of categories.

- Small: digit covers (0-75 pixels) of the image frame
- Medium: digit covers (75-200 pixels) of the image frame
- Large: digit covers (200-500 pixels) of the image frame

Here, we have calculated the *Area* for all test images and got the following distribution.



(a) Data distribution along *Area*

(b) Data coverage over *Area* classes

Figure 75. Distribution of *Area* feature.

We can see, it is not an even distribution but there is data in all defined region of the feature space. To quantify the data coverage, we have got,

- Small: 3954
- Medium: 4447
- Large: 1598

So, the targeted feature dimension has data in all regions. This coverage calculation is like K-multi-section coverage defined in Section 8.2.3.

We can also take the range of areas found in the dataset and compare it with defined values of the feature. It will give a numerical rank to data coverage. To express mathematically, if $x \in dataset, for\ n^{th} feature,$

$$\boldsymbol{Cov}[\boldsymbol{x}(\boldsymbol{n})] = \frac{|\boldsymbol{max}\{\boldsymbol{x}(\boldsymbol{n})\} - \boldsymbol{min}\{\boldsymbol{x}(\boldsymbol{n})\}|}{|\ \boldsymbol{high_n} - \boldsymbol{low_n}|} \tag{15}$$

$$\boldsymbol{Cov}[\boldsymbol{MNIST}('\boldsymbol{size}')] = \frac{|\mathbf{500 - 5.75}|}{|\ \mathbf{500 - 0}|} = \mathbf{0.9885} \tag{16}$$

Coverage of test dataset for *Area* feature has been calculated above. This coverage analysis is described as conventional coverage. Earlier research on *postal code analysis* found some different approaches to calculate data coverage. Such as,

a. Value-level coverage
   - Conventional coverage
   - K-multi-section coverage
   - Boundary coverage

b. Pattern-level coverage

c. Extended variants

Definitions and experimental results of these coverage indicators are reported in Section 8.2.3. There is a method called 'surprise adequacy' which can be used successfully for coverage analysis. We will discuss the application of this method for our problem.

Coverage using surprise adequacy

Surprise Adequacy can be used as a metric to define data coverage/diversity of dataset. DSA is a ratio of distance from test input (x) to its nearest same labeled input (x_a) and distance from input (x_a) to nearest other class input (x_b) [19].

Experiments

Based on the definition of *Distance-based Surprise Adequacy* (DSA) and *Surprise Coverage* (SC), we have experimented with MNIST dataset to calculate the values of DSA of the test dataset. It simply depicts the similarity and difference between test and training data. The model used for this experiment is summarized below.

-

```
ConvNet(
    (layer1): Sequential(
        (0): Conv2d(1, 8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
        (1): ReLU()
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
    (layer2): Sequential(
        (0): Conv2d(8, 24, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
        (1): ReLU()
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
    (drop_out): Dropout(p=0.5, inplace=False)
    (fc1): Linear(in_features=1176, out_features=1000, bias=True)
    (fc2): Linear(in_features=1000, out_features=10, bias=True)
)
```

We have considered different activation layers in CNN for calculating *Activation Traces* (ATs) and plotted the results of accuracy vs. DSA changing.

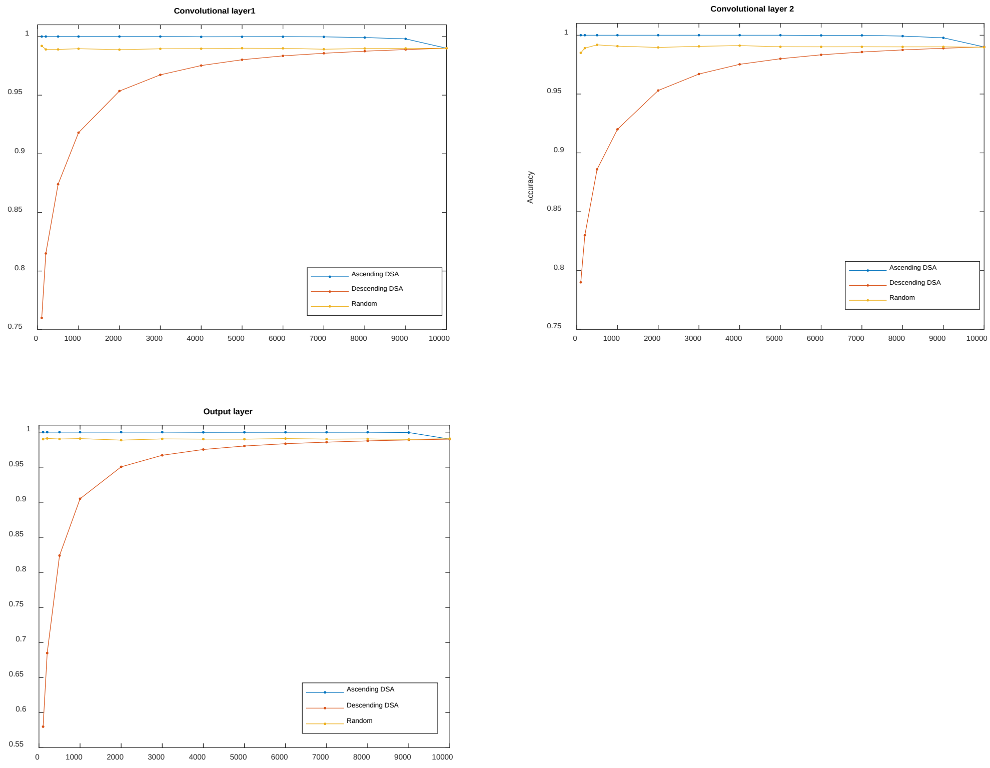


Figure 76. Changes in accuracy with the change in DSA.

Here, the graph is got by reordering the data based on their DSA (ascending and descending). By sequential feeding into the network, we can see the relation between DSA and model accuracy. It is evident that, low surprise data has higher accuracy than high surprise data, i.e., higher the DSA score, higher the probability of model failure will be. We can calculate AI model's SC and accuracy for any feature of our defined problem domain. For example, if *digit area* and *length* are features of the problem domain, we can calculate SC and accuracy along those feature dimensions. Below, we have presented the results got from DSA analysis.

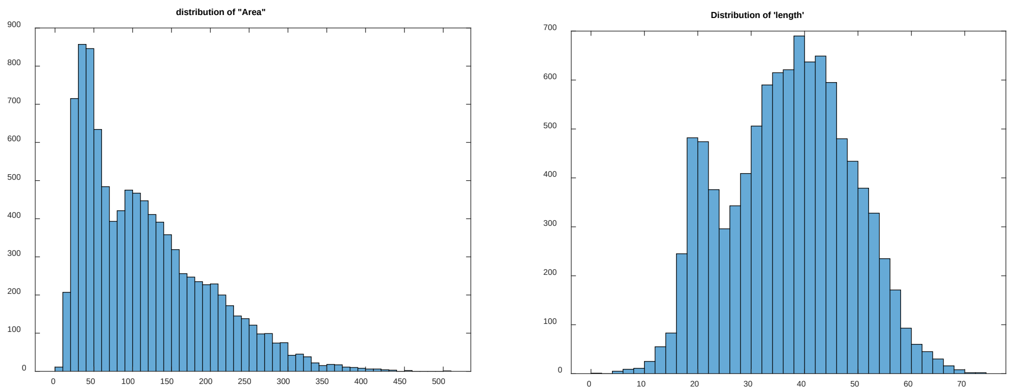


Figure 77. Distribution of MNIST (test dataset) along the features *Area* & *Length*.

Table 37. Divisions of the feature *Area*.

	Small (0, 75)	Medium (75, 200)	Large (200, 500)
Data ratio	39.55%	44.48%	15.97%
SC (layer1)	0.7438	0.7250	0.5687
SC (layer2)	0.7375	0.7312	0.5500
SC (output layer)	0.8063	0.8063	0.6375
Accuracy	0.9901	0.9897	0.9906

Table 38. Divisions of the feature *Length*.

	Small (0, 25)	Medium (25, 50)	Large (50, 75)
Data ratio	19.07%	67.24%	13.69%
SC (layer1)	0.6750	0.7750	0.5938
SC (layer2)	0.6625	0.7813	0.5375
SC (output layer)	0.7000	0.8875	0.6250
Accuracy	0.9911	0.9899	0.9890

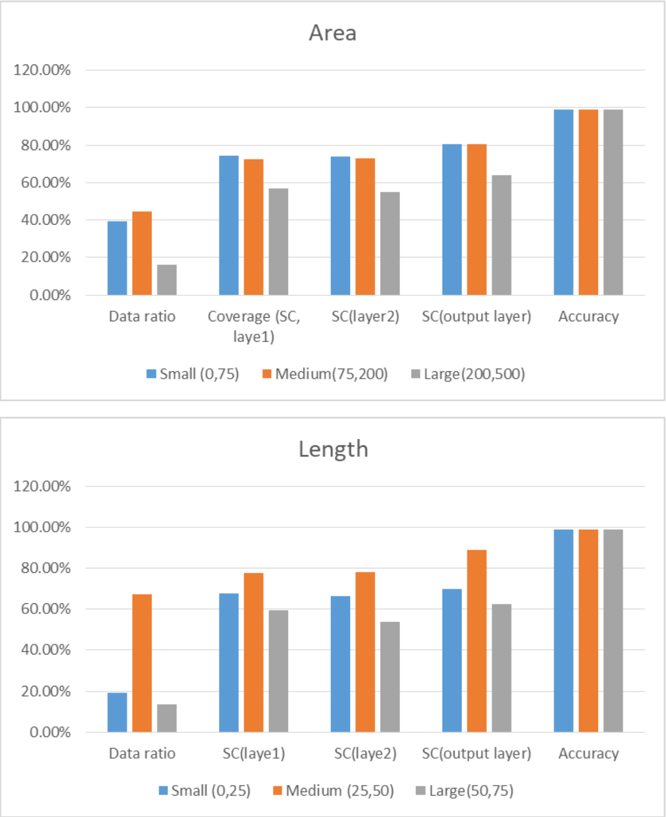


Figure 78. Graphical representation of SC and accuracy for each division of the features *Area* & *Length*.

- We can also combine these two attributes for pair-wise analysis. The divisions of the feature space are shown in Figure 79. As per definition, there are 9 possible segments of the feature space holding unique feature combination.

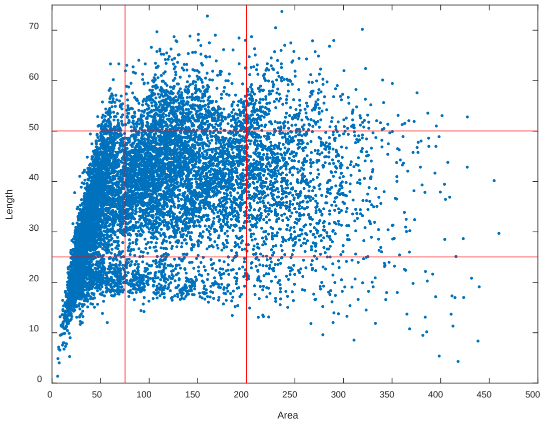


Figure 79. Data (MNIST; test data) distribution across 2-dimensional (*Area*, *Length*) feature space.

Then, we can calculate the data ratio, the surprise coverage (SC) and accuracy for the data of each segment as below,

Table 39. Data coverage analysis within a 2-dimensional problem domain.

Data ratio		Area		
		small	medium	large
Length	small	0.1296	0.2546	0.0113
	medium	0.0438	0.3119	0.0891
	large	0.0172	0.1059	0.0365
SC (layer1)		Area		
		small	medium	large
Length	small	0.6500	0.6500	0.3500
	medium	0.4750	0.6438	0.5625
	large	0.3813	0.5375	0.4438
Accuracy		Area		
		small	medium	large
Length	small	0.9907	0.9898	0.9912
	medium	0.9932	0.9904	0.9854
	large	0.9942	0.9887	0.9973

From the SC analysis, it is hard to relate model accuracy with the test data's SC. But clearly SC has a positive correlation with data ratio which universally means data coverage. So, SC is chosen as a scale of data coverage and used to identify rare/corner cases.

Identifying rare/corner cases

Coverage analysis will reflect the necessity of data points in determined regions of feature space. According to definition, the combinations of features for which there is no data can be called **risky case**. In real world,

– If a risky case occurs in normal frequency, then dataset preparation is faulty.

– If a risky case occurs in low frequency, then it will be considered as rare/corner case.

For example, from the above coverage calculation, we have found, there is no large-scale digit image present in the training dataset where there is a probability of getting such images in real time operation. So, large-scale digit images are corner cases.

Corner case detection based on DSA

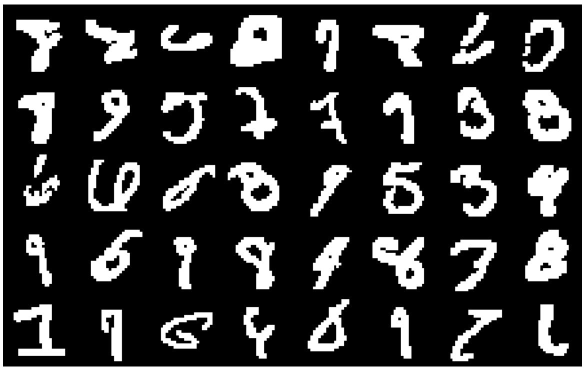
SA can describe the novelty between testing data and training dataset. For an individual testing data point, SA describes its difference/similarity to the whole training data. Therefore, SA can be considered a useful metric to capture the corner cases. The set of corner cases is defined as follows.

$$\boldsymbol{Corner\ cases}: \{\boldsymbol{x} \mid \boldsymbol{class}(\boldsymbol{x}+\boldsymbol{pert}) \neq \boldsymbol{class}(\boldsymbol{x}), |\boldsymbol{pert}| < \boldsymbol{\varepsilon}\} \tag{17}$$

Here, we can still use the DSA as a measurement, and get the variation of Accuracy vs. that of DSA. Then, we can further analyze the relationship between SA and corner cases.

We have used the CNN described in the upper section. The testing accuracy is 99%, and there are in total 100 images that are incorrectly classified. The accuracy vs DSA graph in Figure 76 shows that not all of outliers have high DSA, also some correct images can have high DSA. Since, DSA ranking based on output layer shows minimum accuracy at higher DSA values, it can detect the most incorrectly classified images as corner cases. The following input images have the largest DSA based on output layer.

Table 40. Images with the largest DSA.

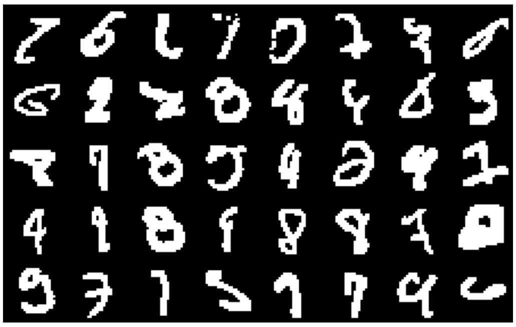
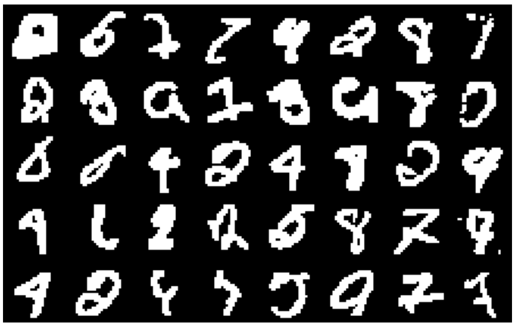
Images	Labels
	Actual label [8 2 6 2 7 8 6 0 8 9 5 7 7 7 3 8 6 6 0 8 9 5 3 4 9 5 9 8 4 8 3 8 1 7 6 9 0 9 7 6] Predicted label (13) [7 7 4 0 9 7 6 7 7 9 7 2 3 9 5 8 6 6 8 0 9 5 3 9 1 6 9 9 4 8 7 8 1 9 5 4 6 1 8 1]

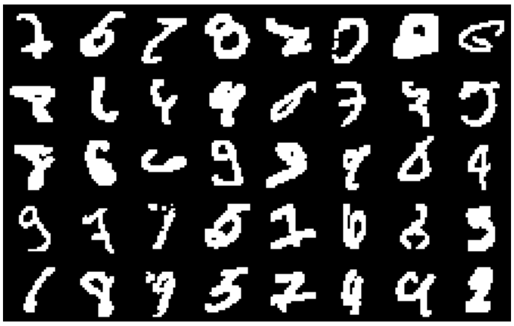
Among these high DSA input images, 27 images have been predicted incorrectly which are corner cases for the model. Still there are 13 correctly predicted images, therefore we need an improved metric to identify corner cases.

There are three proposed modifications on DSA calculation [42]. To improve corner case detection metric based on DSA, we have used those modifications and compared their results and output to select the best DSA metric for our problem space.

We have applied these new definitions of DSA and from model prediction on high DSA inputs, we got corner case data for each modification. In the following table, we have compared three modified DSA's performance in corner case detection. For activation trace calculation, we have used output layer only.

Table 41. Images with large DSA1, DSA2, or DSA3.

	Images	Labels
DSA1		True label [7 5 6 7 0 7 3 0 6 2 2 8 8 9 0 5 8 7 8 5 9 2 4 1 4 9 8 9 8 8 7 2 9 7 7 5 7 7 9 6] Predicted label (8) [8 6 1 1 7 2 7 8 5 8 7 0 8 4 6 3 7 9 0 7 4 2 9 2 9 9 8 9 8 9 3 0 5 3 1 5 9 7 4 4]
DSA2		True label [2 5 7 7 4 2 8 7 2 3 9 1 3 9 8 0 0 0 4 2 4 8 2 4 4 6 2 2 5 8 7 7 4 2 9 7 5 9 7 7] Predicted label (9) [0 6 2 8 9 8 9 1 8 8 9 2 8 9 7 7 6 8 9 2 4 7 0 9 9 1 8 2 0 8 2 7 4 0 4 7 7 0 2 3]

DSA3		True label [7 5 7 8 2 0 2 6 8 6 9 4 0 7 3 5 8 6 6 9 9 9 0 4 9 7 7 5 1 6 8 5 6 8 9 5 7 9 9 2] Predicted label (3) [2 6 8 0 7 7 0 5 7 1 4 9 8 3 7 7 7 6 4 3 3 8 6 9 5 3 1 0 2 6 2 3 1 9 7 5 2 4 4 8]

From the results, we can see that DSA3 is the most successful in identifying inputs where model shows erroneous behavior. So, we can use DSA3 definition on adversarial data to get the most corner cases.

Feature deletion

Based on the coverage results, we need to design data properly. In this process, we may find some features or some ranges of values of features which we can exclude from the problem domain.

For example, we can do feature-based data mutation testing to get model behavior on dataset structure. Any unchanged output means the mutation doesn't affect model's performance; hence we can omit the feature or values of feature from our feature space.

Conclusion

This MLQM criterion is for data evaluation, describing data coverage over problem domain. From this analysis, we can define learning capability from training dataset and performance measuring capability from test dataset. To improve data coverage, we will discuss in later section some possible methods to follow to fill in the rare cases. In the following section, we will analyze the distribution of data i.e., uniformity/evenness of dataset.

9.5.4 B-2: Uniformity of datasets

Definition

This MLQM criterion covers computation of dataset distribution across the defined feature space. While coverage analysis was searching data in every corner of the feature space, Uniformity analyzes data density in those regions. In this section, we will compute as well as visualize dataset distribution to decide upon its uniformity.

Determining data distribution from visual representation

Let's say we want to analyze data distribution on boldness feature from visual perspective.

Measuring thickness of handwritten digits in images can be hard. But if we consider the fact that bolder digits will have more pixels, we can roughly estimate the distribution of the numbers for boldness. For this we can consider drawing contours instead of bounding boxes.

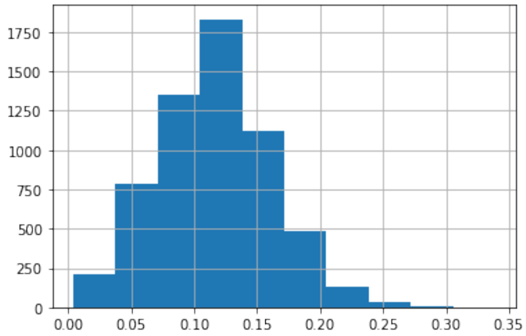


Figure 80. Data distribution based on digit boldness.

In the distribution graph shown in Figure 80, x-axis holds ratio of pixels constructing digits. So, we can see that there is a normal distribution along x-axis up to 20%. The distribution is not completely uniform, but the dataset holds good amount of data for our selected range of *boldness*.

Computing data evenness

According to the above coverage analysis, a simple indicator could be proposed to evaluate the evenness of data. We can use TPCov idea for calculating uniformity.

When considering the evenness property, the above coverages could be developed as new indicators. For example, the simplest value coverage could be developed as top *p*% coverage (TPCov), which implies that the ratio of region having *p*% of data points with highest density and the original coverage, so definitions are expressed as below.

$$\boldsymbol{TPCov}[\boldsymbol{x}(\boldsymbol{n})] = \frac{|\{\boldsymbol{S}|\boldsymbol{density}(\boldsymbol{S}) = \boldsymbol{p}\%\}|}{\boldsymbol{high_n} - \boldsymbol{low_n}} \tag{18}$$

Here, S is the coverage portion for top p% data.

Now, using the TPCov to evaluate evenness, the indicator is defined as the difference between TPCov and *p*%, as

$$\boldsymbol{EI_{err}} = |\boldsymbol{TPCov} - \boldsymbol{p}\%| \tag{19}$$

If the value of $\mathrm{EI}_{\mathrm{err}}$ is low, it implies that the data is evenly distributed, as shown in Figure 81.

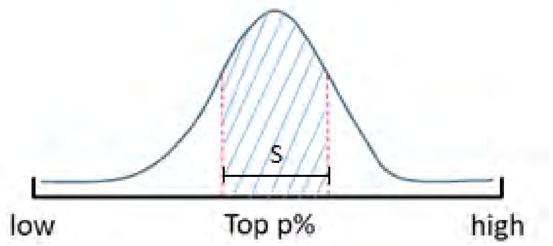


Figure 81. An approximate data distribution to aid visualization of evenness computation.

If we consider different values of p%, we could further compare values of coverage indicators with p%. In this way, we can refer to the *area under the curve* (AUC) to measure the performance of evenness, defined as

$$\boldsymbol{AUC = area\{coverage(p): p \in (0, 100\%)\} = \int_0^1 Coverage(p)dp} \tag{20}$$

Reducing biasness of data collection

To increase rare case density, the data gathering procedure will be biased. This can harm ML model's average performance. So, we need an optimum level of biasness towards data collection.

Data design

Before ending this section, since it is the last internal quality for data evaluation, there must be a description of research about how to improve data quality i.e., explaining various data design process. Data design means preparing dataset for machine learning models keeping the defined problem domain in mind. The above problem domain can be rephrased as feature space. In this part, we will discuss some methods to follow for creating or gathering enough data points for every region of that feature space.

To do so, we can follow either of the following two.

- Build a new dataset
- Work with an existing dataset

Reason for data design

It seems, data design is the task for developers of an AI, but for the following instance, an AI evaluator may need to design one.

- Say, the requirement analysis of the developer is different from that of the evaluator. Then the evaluator will need to design a proper test dataset for evaluation.
- On the other hand, if both developer and evaluator work on same requirements, then

evaluator can adopt the dataset building process from the developer for his evaluation.

Build a new dataset

If we have enough manpower or technology, we can build dataset from scratch. If we do so,

– We can easily ensure the involvement of all features that we have already described.
– Also, we can decide the number of data points for each case.
– In this way, the later analysis of *coverage* and *uniformity* will be for nothing but visual representation.

Work with an existing dataset

Building new dataset is currently out of our scope. So, for this section, we will adopt MNIST, the most popular hand-written digit-classification dataset, to demonstrate MLQM workflow. Now, the challenges of using predefined dataset are:

– We cannot expect data in all feature dimensions. We need to check the coverage in later section.
– We also cannot ensure uniform distribution across the feature space.
– Later, from *coverage* and *uniformity* analysis, we can find the actual distribution of the dataset and identify the cases where we will need *data augmentation*.

Procedures of data design

When we use existing dataset for our problem, we will not get data points well distributed across the defined feature space. So, in this section, we will define some of the methodologies to augment data or increase data coverage.

Data augmentation

Data augmentation methods will be described in detail focusing the coverage for every region of the feature space. For example, here we will describe a possible augmentation process for the brightness/contrast feature.

- **Data augmentation by varying contrast:** This is one of the feature dimensions where very few or no data point of MNIST dataset will lie upon. To add new data points for this feature, we can simply *darken some of the given data*. A few examples are shown here.

Feature deletion by adding external method

In some feature dimension, we may not find a suitable data augmentation method for increasing data coverage. So, alternatively, we may describe some external methods to handle that specific characteristic feature. Thus, we can exclude that feature from our problem domain which not only will decrease the number of features but also the number of training or testing data. For example, we have described an external method for centering digits in any image frame.

Handling *position of the digit* by external method: We can easily get the bounding box of digits for inverted images like MNIST. Then we can center the box in the frame and thus can eliminate the requirement *position of the digit*. Here are some examples from MNIST test set.

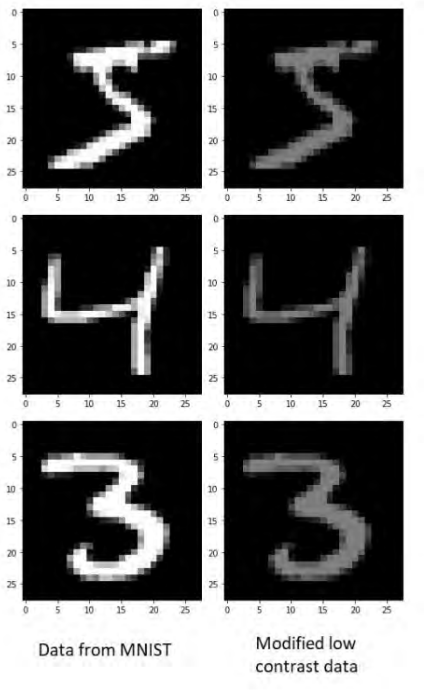


Figure 82. Modification of contrast

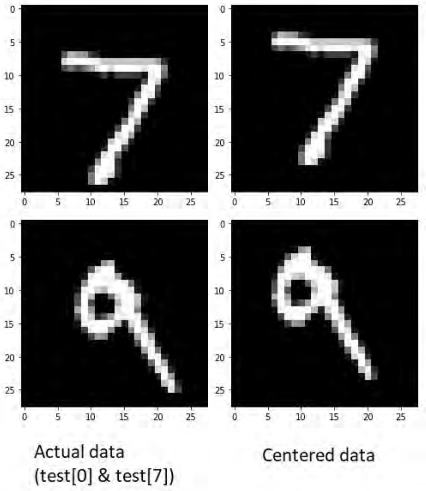


Figure 83. Centering images.

Machine learning is a data driven process, so proper data management is mandatory. In this section, we have discussed some procedures to that purpose but there can be some features that cannot be described in similar way. For those features, building manual dataset is an option otherwise, developer cannot train models for that characteristic feature as well as evaluator won't be able to build a complete test set for evaluation. In that case, the feature will be out of scope of the ML models.

Conclusion

Uniformity is more important criterion for training dataset than testing. It inherently deals with output biasness of machine learning models. By fulfilling this criterion, a model can achieve greater performance, corner case accuracy and avoiding risk factors.

9.5.5 B-3: Adequacy of data

This internal quality is not discussed in this version of the Reference Guide.

9.5.6 C-1: Correctness of trained models

Definition

Accuracy is the primary measurement of model's correctness which evaluates its performance. In the following section, we will define some useful correctness measurement metrics or key performance indicator (KPI) followed by their actual use to describe a trained model's output.

Different accuracy measures and key performance indicators (KPI)

Since postal code analysis is a classification problem, the commonly used performance metric is confusion matrix. This matrix can be visualized in the following way.

Table 42. The confusion matrix of binary classification.

		Predicted output	
		Positive	Negative
Actual output	Positive	TP	FN
	Negative	FP	TN

Confusion matrix: This is a simple confusion matrix for binary classification. There are four possible output behavior.

TP = when predictor predicts positive correctly

FP = when predictor predicts positive incorrectly

FN = when predictor predicts Negative incorrectly

TN = when predictor predicts Negative correctly

Based on this matrix, we can define some useful and popular performance measures.

Accuracy: This is the measure of correct prediction by the model which can be defined as,

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}} \tag{21}$$

Recall: This is a ratio of correct positive prediction to the total number of positive data which can be defined as,

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \tag{22}$$

Precision: This is a ratio of correct positive prediction to the total number of predicted positives which can be defined as,

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} \tag{23}$$

F-measure/F-score: This is the harmonic mean of precision and recall which can be defined as,

$$\boldsymbol{F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision}} \tag{24}$$

For example, we have chosen a CNN to train on MNIST dataset for *postal code analysis* problem. The model architecture is given below.

Table 43. The model for *postal code analysis*.

Architecture	Conv(24,24,24)+ReLU MaxPooling(12,12) Conv(8,8,64)+ReLU MaxPooling(4,4) Flatten() FC(1000)+ReLU FC(10)+Softmax
Number of trainable parameters	1,074,098
Train accuracy	99.26%
Test accuracy	99.50%

KPI measures other than accuracy is calculated separately for all classes using test set and presented in the table below.

Table 44. KPIs for all classes.

	0	1	2	3	4	5	6	7	8	9
Recall	0.998	0.999	0.998	0.998	0.995	0.990	0.987	0.992	0.996	0.995
Precision	0.994	0.996	0.995	0.994	0.994	0.994	0.999	0.997	0.995	0.991
F-measure	0.996	0.998	0.997	0.996	0.994	0.992	0.993	0.995	0.995	0.993

The test accuracy of the trained model is very good, but we can get in depth behavior of the model if we see recall, precision, and F-measure. From the results of recall, the classifier has the highest accuracy in predicting '1' and the lowest accuracy in predicting '6'. From the results of precision, the classifier gives the lowest wrong prediction for '6' and the highest wrong prediction for '9'. From the results of F-measure, it can be said that the overall performance is best for digit '1' and worse for digit '5'.

Defining models' behavior & finding corner cases

Until now, we are deciding on corner cases by observing data coverage or data distribution. But actual corner cases can be identified from models' behavior on test data. Those input data for which a model outputs wrong prediction can be called unidentified cases for that model. If we can separate input data for which similar models will output incorrectly, we can have corner cases for that specific solution. There is a study which prioritizes input data for which models' show erroneous behavior [43].

Conclusion

Accuracy is only one aspect of models' performance, but in this scope, we will evaluate a model from various perspectives. For example, the accuracy of identifying positive and negative separately, identifying corner cases based on models' behavior and so on. In the next section, we will define and evaluate the robustness i.e., stability of ML models.

9.5.7 C-2: Stability of trained models

Definition

Stability is one of the most important characteristics of an ML model which determines model behavior due to perturbation both in input data and model. Stability is defined conventionally as follows [34]. "The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions."

Let 'S' be a machine learning system. Let E(S) be the correctness of 'S'. Let δ(S) be the machine learning system with perturbations on any machine learning components such as the data, the learning program, or the framework. The robustness of a machine learning system is a measurement of the difference between E(S) and E(δ(S)):

$$\boldsymbol{r} = \boldsymbol{E}(\boldsymbol{S}) - \boldsymbol{E}\big(\boldsymbol{\delta}(\boldsymbol{S})\big) \tag{25}$$

Robustness thus measures the resilience of an ML system's correctness in the presence of perturbation. Here, we will define some robustness measures followed by experimental results for *postal code analysis*.

Robustness can be defined for two basic components of an AI: data and model. In the following section, we have considered these two measures separately.

Robustness due to perturbation in data

Related to data, this robustness measure depends on adversarial examples. These examples can be created within proximity of actual test data. The vector distance between original data and adversarial data is considered the measure of robustness. There are various metrics for calculating robustness/stability.

- **Local Adversarial Robustness:** Let *x* be a test input for an ML model *h*. Let *x'* be another test input generated via conducting adversarial perturbation on *x*. Model *h* is δ-local robust at input *x* if for any *x'*,

$$\forall \boldsymbol{x}' : \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|_{\boldsymbol{p}} = \boldsymbol{\delta} \rightarrow \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}') \tag{26}$$

$||*||_p$ represents p-norm for distance measurement. The commonly used p cases in machine learning testing are 0,1 and 2.

- **Global Adversarial Robustness:** Let *x* be a test input for an ML model *h*. Let *x′* be another test input generated via conducting adversarial perturbation on *x*. Model *h* is δ-global robust if for any *x* and *x′*,

$$\forall \boldsymbol{x}, \boldsymbol{x}': ||\boldsymbol{x} - \boldsymbol{x}'||_{\boldsymbol{p}} = \boldsymbol{\delta} \rightarrow \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{h}(\boldsymbol{x}') \leq \boldsymbol{\epsilon} \tag{27}$$

Based on the above definitions of adversarial robustness, we could use the value of δ as the robustness measurement directly. However, to evaluate different model's robustness performance, we may need to generalize these robustness metrics into relative ones. Here, assuming the input data is normalized into $[0,1]^d$ where d is the dimensionality, then the relative robustness metrics could be defined as below,

$$\boldsymbol{r_1} = \frac{\boldsymbol{\delta_1}}{\boldsymbol{0.5 \times d}}, \quad ||\boldsymbol{x} - \boldsymbol{x}'||_1 = \boldsymbol{\delta_1} \tag{28}$$

$$\boldsymbol{r_2} = \frac{\boldsymbol{\delta_2}}{\sqrt{\boldsymbol{0.5 \times d}}}, \quad ||\boldsymbol{x} - \boldsymbol{x}'||_2 = \boldsymbol{\delta_2} \tag{29}$$

$$\boldsymbol{r_\infty} = \frac{\boldsymbol{\delta_\infty}}{\boldsymbol{0.5}}, \quad ||\boldsymbol{x} - \boldsymbol{x}'||_\infty = \boldsymbol{\delta_\infty} \tag{30}$$

There are several studies on robustness (δ) measurement. For example, CNN-Cert [44] and Fast-Lin [45].

We can use the measured δ value to calculate relative robustness formulated above. For example, taking MNIST data as an example, the results of CNN-Cert and Fast-Lin have been used to calculate relative robustness metrics.

Table 45. Relative robustness metrics.

		Certified lower bounds (δ)		Relative robustness (r) ($\times 10^{-2}$)	
	L_p norm	CNN-Cert	Fast-Lin	CNN-Cert	Fast-Lin
MNIST 4 layers 5 filters 8680 hidden nodes	L_∞ L_2 L_1	0.0491 0.1793 0.3363	0.0406 0.1453 0.2764	9.82 0.91 8.58	8.12 0.73 7.05
MNIST 4 layers 20 filters 34720 hidden nodes	L_∞ L_2 L_1	0.0340 0.1242 0.2404	0.0291 0.1039 0.1993	6.80 0.63 6.13	5.82 0.52 5.08

MNIST	L_∞	0.0305	0.0248	6.10	4.96
5 layers	L_2	0.1262	0.1007	0.64	0.51
5 filters	L_1	0.2482	0.2013	6.33	5.14
10680 hidden nodes					

Robustness due to perturbation in model

Related to model, this robustness measure depends on model perturbation. There are various methods for calculating model level robustness.

- **Parameter Robustness:** Let w be the parameter of an ML model h. Let w' be another parameter generated via adding some slight perturbation on w. Model h is δ_w-local robust on parameter perturbation if for any w',

$$\forall \boldsymbol{w}': \left|\left|\boldsymbol{w} - \boldsymbol{w}'\right|\right|_{\boldsymbol{p}} = \boldsymbol{\delta_w} \rightarrow \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}'(\boldsymbol{x}) \tag{31}$$

Figure 84 shows the example of parameter perturbations in AI model's robustness evaluation. Compared with the adversarial robustness which aims at seeking for the minimum distance $\delta_{\min}$ as the robustness metric, the parameter robustness utilizes the maximum distance $\delta_{w\ \max}$ as the robustness measurement. Moreover, program-level robustness has an advantage of no generation of adversarial samples and their certification.

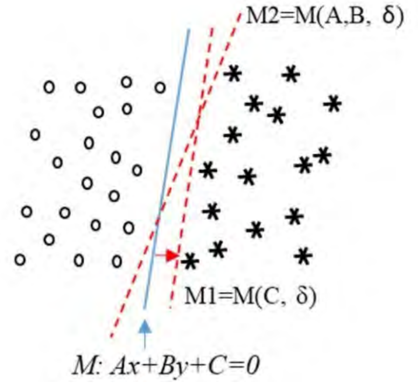


Figure 84. Changes in a linear classifier due to parameter perturbation.

- **Mutation Robustness:** To measure the software mutational robustness, we formalize it with respect to a software program P (a member of the set of all software programs Ƥ), a set of mutation operators M (where each $m \in$ M is a function mapping Ƥ →Ƥ), and a test suite T: Ƥ →{true, false}. A program P is said to pass the test suite if and only if T(P) = true. Given a program P, a set of mutation operators M, and a test suite T such that T(P) = true, we define the *software mutational robustness*, written *MutRB*(P, T,M), to be the fraction of all direct mutants P' = *m*(P), ∀*m* ∈ M, which both compile and pass T,

$$\boldsymbol{MutRB(P,T,M)} = \frac{|\{\boldsymbol{P'} \mid \boldsymbol{m} \in \boldsymbol{M}.\boldsymbol{P'} = \boldsymbol{m(P)} \cap \boldsymbol{T}\,(\boldsymbol{P'}) = \boldsymbol{true}\}|}{|\{\boldsymbol{P'}|\boldsymbol{m} \in \boldsymbol{M}.\boldsymbol{P'} = \boldsymbol{m(P)}\}|} \tag{32}$$

This measurement can be transferred in MLQM. For example, if we take the results of DeepMutation [46] as an example, where three studied DL models A, B and C are tested on MNIST dataset.

Table 46. Mutation robustness.

	Model A	Model B	Model C
Architecture	Conv(6,5,5)+ReLU MaxPooling (2,2) Conv(16,5,5)+ReLU MaxPooling (2,2) Flatten() FC(120)+ReLU FC(84)+ReLU FC(10)+Softmax	Conv(32,3,3)+ReLU Conv(32,3,3)+ReLU MaxPooling(2,2) Conv(64,3,3)+ReLU Conv(64,3,3)+ReLU MaxPooling(2,2) Flatten() FC(200)+ReLU FC(10)+Softmax	Conv(32,3,3)+ReLU Conv(32,3,3)+ReLU MaxPooling(2,2) Conv(64,3,3)+ReLU Conv(64,3,3)+ReLU Maxpooling(2,2) Flatten() FC(200)+ReLU FC(200)+ReLU FC(10)+Softmax
Trainable parameter	107,786	694,402	698,402
Training Acc.	97.40%	99.30%	99.50%

For each mutation operator, 20 DL mutants were created to acquire mutation score. The mutation score in DeepMutation and MutRB are complementary to each other or Mutation_score + MutRB=1. Below, we have shown mutation scores presented in the literature followed by calculated MutRB scores in separate tables.

Table 47. Mutation score (%).

	0	1	2	3	4	5	6	7	8	9
Model A	7.22	8.75	9.03	6.25	8.75	8.19	8.75	9.17	9.72	9.03
Model B	1.59	3.29	8.29	7.44	5.49	4.02	8.17	3.66	5.85	8.41
Model C	8.33	7.95	8.97	9.74	9.74	9.62	9.62	8.97	9.74	7.56

Table 48. MutRB score (%).

	0	1	2	3	4	5	6	7	8	9
Model A	92.78	91.25	90.97	93.75	91.25	91.81	91.25	90.83	90.28	90.97
Model B	98.41	96.71	91.71	92.56	94.51	95.98	91.83	96.34	94.15	91.59
Model C	91.67	92.05	91.03	90.26	90.26	90.38	90.38	91.03	90.26	92.44

There are some studies which focus on creating adversarial examples targeting model failure. By adding a scale of perturbation in the input data, we can also adopt these studies for robustness measurement. Here, we have listed some of those studies.

- Deepxplore: Automated Whitebox Testing of Deep Learning Systems [28]
- Guiding Deep Learning System Testing Using Surprise Adequacy [19]
- TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing [47]

Conclusion

Robustness measures indicate models' performance for unknown inputs or under new environmental conditions. By the defined KPI for robustness, we can ensure whether a trained model for postal code analysis meets the requirement level of stability set by the client/user. Also, this is the last internal quality which will be evaluating the AI solution. The following MLQM internal properties are for evaluating other parts of the machine which will support the AI before or during operation either in public or in a controlled environment.

9.5.8 D-1: Reliability of underlying software systems

Definition

According to MLQM Guideline, the term *reliability of underlying software systems* means software components used for machine learning training stage as well as prediction/interface should operate correctly when they are executed in response to training data and trained ML model respectively. AI solutions are seldom built from scratch. It often consists of numerus components like software components (e.g., image processing software, python packages and libraries etc.). The following AI components should be described in details and their quality need to be assured.

Program & open-source libraries

Python language has been used for developing this AI. It uses various open-source packages which should be version compatible with each other. So, the list of used packages and their versions should be provided by the developer. It is better to describe similarly versions of all programs used even when they are not packages nor open-source.

Table 49. List of used packages and their versions.

Programing language	Version
Python	3.6.12
Package	Version
NumPy	1.18.5
TensorFlow	2.3.1
Pillow (PIL fork)	8.0.1

Image processing unit

This refers to the image capturing of the camera and processing for creating input images comparable to dataset.

For simplicity, we will be taking images of the post codes with a camera which produces 3 channel (RGB) 2-dimensional images. These images will be subject to an algorithm or program flow that converts them into analogous inputs for the trained AI model.

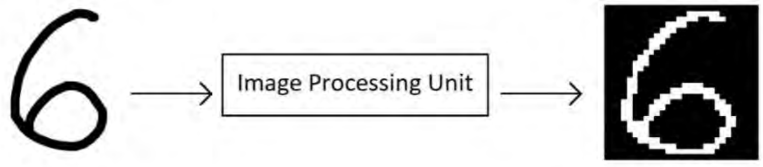


Figure 85. Modification of images by image processing unit.

As an example, we have taken a general image of a digit (6); written on a white paper with black ink. We have written a program that converts this image to a black n white (28, 28) image which is equivalent to MNIST handwritten digit dataset.

Inside data preprocessing, python programing language has been used and open-source package Pillow has been used as image processing tool.

Unit for external methods

In this part of the device, we define algorithm for external method to omit certain problem domain characteristics described in Section 9.5.2 on *coverage for distinguished problem cases*. We need to ensure the correctness of the algorithm. We have already described an external method to remove a defined feature, *position of the digit*. The developed algorithm for that task will be put after data preprocessing and before feeding to the model.

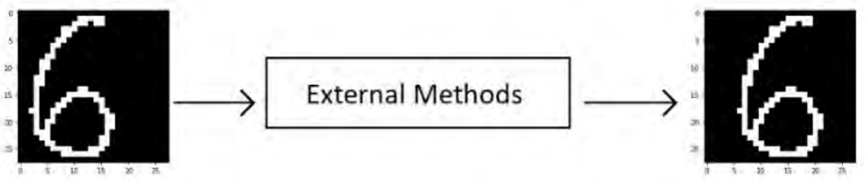


Figure 86. Modification of images by *external methods* unit.

This unit can be consisting of one or more external methods which will be executed on the input image sequentially. The algorithm for handling *position of the digit* feature has also been written using python programing language with the help of open source packages; NumPy and Pillow.

Usage of memory

We need to define a minimum and maximum usage of memory when the AI device is in operation.

- **Model architecture and weights:** Hierarchical Data Format (HDF) file (.h5) is one of the file formats AI developers use to store trained AI networks and their weights. In Section 9.5.5 on *correctness of the trained model*, we have reported results of a trained CNN. The saved network has 1,074,098 parameters and takes about 12.3MB space on the hard drive.
- **Input data:** Input data means camera image data, cropped images, converted images via preprocessing and external methods. Though it is possible to quantify the converted image size, the original image will take the largest space on the hard drive. So, the memory usage by the input data can be defined after complete design of the machine, the camera, the quality, and resolution of the image etc.
- **Codes/algorithm:** Different algorithms, written by programing languages are part of the workflow of the machine. These codes do not take much space on hard drive. For example, the algorithm for the external method described above takes about 4KB space.
- **Dataset for retraining:** We need to keep a space for holding dataset for possible re-training phase at least the size of the actual dataset. For example, MNIST dataset takes about 52.4 MB space.
- **RAM and GPU for network training:** If we need to train model in between operations, we need to define machine specification for that task. For example, our described model training using MNIST dataset can be done in reasonable time with 8 GB of RAM and no GPU.

Combining all memory or machine requirements, we can design the memory allocation of the device.

Time cost

Time is an issue when applying in real world, it reduces machine's efficiency. We need to eliminate any unnecessary time loss in any stage of the machine as well as direction for improving to faster algorithms.

As an example, for postal code analysis, the most time-consuming task of the entire workflow is the classification task because majority of the computation occurs in this part. So, batch execution will be faster and efficient than sequential execution with the cost of bigger memory requirement.

Software security

Security is a major concern when machine operates online. For the themes that the solution designer should consider while building the application sets with AI/Machine learning functionality, see the chapter on security in the Guideline (Chapter 9 in the second edition).

Difference between training and operational environment

Training environment and actual operational environment are often different especially in applications of machine learning, also behaviors of numerical calculations often change. In such cases, it is necessary to evaluate differences between these environments and define how the outcome of the machine will be affected by this.

For example, ML models train from random initialization to optimized loss function. This process consists of numerous computation which is done implicitly by the machine. So, it is very common to differ the results of KPIs in different machines. We need to evaluate the whole system in various machines so that we can set probable performance deviations due to change in machine environment.

Conclusion

Above program components should be defined and assured for quality and security before the solution goes into operational phase.

9.5.9 E-1: Maintainability of quality in operation

Definition

Machines are very much reliable, but they break down too. So, maintenance is a periodical task and a part of every machine in operational phase. According to MLQM Guideline, “this section describes technologies to maintain internal qualities satisfied at the commencement of operation throughout the operation period”.

Task flow during maintenance

Here, we will discuss various *in-operation* machine failures and procedures to overcome them. To demonstrate the possible crisis during operational phase, we will be using contrived dataset based on restricted feature dimension. If we consider *Area* feature as one of the feature dimensions, we can see train and test set have different distribution and different ranges of values.

From Figure 87, *Area* values in train set ranges from 22.75 ~ 315.125 where in test set ranges from 5.75 ~ 505.25. If we consider the train set has defined range of problem domain, then we will have a test set having same range of values. We have taken a subset of original test dataset which represents the expected test set (say test_in1) and another subset of test dataset which represents operational inputs lies beyond the defined region (say test_out1).

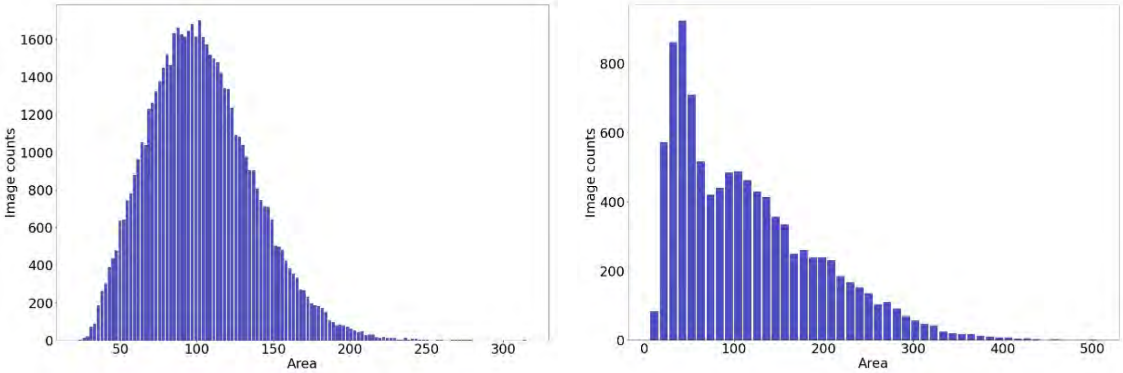


Figure 87. Data distribution based on Area feature of train and test set respectively.

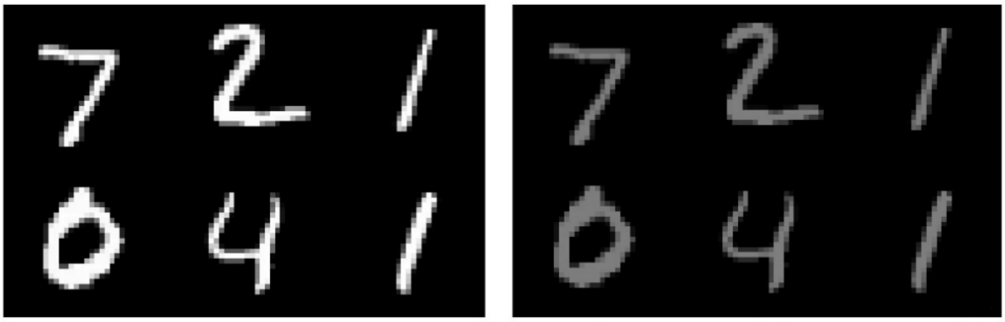


Figure 88. Example inputs from test_in2 and test_out2 respectively.

We will consider *contrast* feature as well. Since all the images from the test set (say test_in2) are of high contrast, we have used one of the data design procedures to build a similar test dataset (10,000 images; say test_out2) to represent possible operational inputs.

Accuracy (KPI) monitoring

Periodically, we need to evaluate the trained model with diverse test inputs (test_in1). The test set should contain data for all possible classes as well as combinations of cases inside problem domain. For example, here we have evenly class distributed test_in1 having full coverage across *Area* feature (Figure 89).

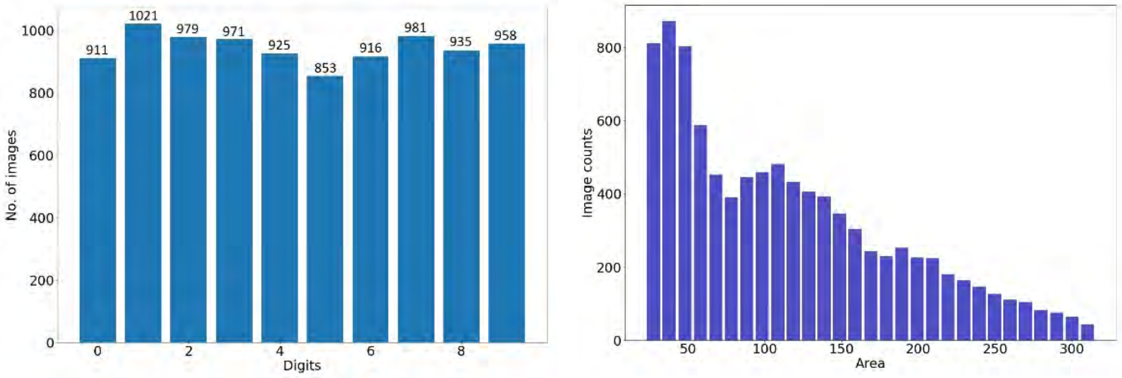


Figure 89. Class distribution and data distribution of test_in1 set.

Secondly, test_in2 is the complete test dataset having even distribution of digits and high contrast images. Table 50 shows the results of accuracy monitoring.

Table 50. The results of accuracy monitoring.

Test set	No. of images	Accuracy
test_in1	9,450	99.21%
test_in2	10,000	99.20%

Continuous data collection and labeling

During operation, the model always gets new/unknown inputs. It is necessary to store these data to analyze current data distribution in problem space. This work consists of data gathering and labeling. In most of the cases, labeling is a manual process having high cost. In this phase, a new dataset will be built upon operational inputs. For example, complete test set together with test_out2 can be the set of operational inputs.

Analyzing novelty of model input

From the set of operational inputs, we need to measure data novelty by analyzing coverage and distribution. We can also use distance or similarity-based methods to identify novelty so that we may estimate model's performance based on previous robustness measures. Thus, we can build a novel dataset.

Here, we have taken test_out1 and test_out2 dataset described as outside the scope of problem domain. We will consider them as example of novel datasets for this step of maintenance. From Table 51, we can understand the effect of novel data on our trained model.

Table 51. The effect of novel data.

Test set	No. of images	Accuracy
test_out1	550	99.09%
test_out2	10,000	97.66%

Analyzing re-training necessity

Although we train the best model, showing the best performance, it will deteriorate with time. This happens because of changes in inputs or changes in environment. From the above table, we can see, test_out1 has small but test_out2 has significant negative effect on the trained model. So, it is necessary to re-train the model. However, we need to be careful with re-training so that the model doesn't forget its previous learning.

For example, we have retrained the same model with training images (60,000) and their low contrast versions (60,000); evaluated on test_in2 and test_out2. The results are in the following table.

Table 52. The results of retraining.

Test set	No. of images	Previous accuracy	Latest accuracy
test_in2	10,000	99.20%	99.40%
test_out2	10,000	97.66%	99.40%

Model output monitoring

Model output needs to be analyzed to check whether it gives finite and valid results. Aside from wrong prediction, a model can also output unidentifiable numbers (i.e., NaN value). So, the model should be validated on this especially, after re-training.

For example, in [47], a tool is described to find input that results Nan value. First, it chooses an image from input corpus and mutates it by adding noises. Then it passes the image through an ML model and computes the output values and activation vector. If any of the output values are Nan, the program halts; otherwise activation vector is compared from the previous runs using nearest neighbor algorithm to determine new coverage and add that mutated image to input corpus. The newest element is drawn from the corpus for the next run.

Creating additional dataset

Creating a new or additional dataset and re-training the model is the only way to expand the solution space of an AI problem (postal code analysis). For example, if US postcode classifier is brought to Japan, it needs to retrain itself using Japanese handwritten digits to obtain equivalent performance.

Conclusion

Maintenance of machine learning technology helps improving its models both in accuracy and robustness. It also helps to build a long-lasting model, real world distribution of data, also corner cases. Therefore, implementing maintenance procedures can develop better machine learning solution gradually.

10 House price analysis

10.1 Introduction

Our goal is to create an example implementation to demonstrate how Machine Learning Quality Management (MLQM) Guideline rules can be used to assess internal properties of a house price dataset. Here we will discuss about house price problem, a regression problem which refers to predict the price of house from the Kaggle datasets. This report can be used as a reference for application of the Guideline in evaluation of similar AI based systems.

We expect to achieve the following outcomes by applying MLQM Guideline to an AI based product:

- Detailed analysis of the house price problem and breaking down genuine cases
- Different techniques for data management and feature reduction
- Introducing machine learning algorithm to our interest
- Examining product's quality, safety, and reliability for the end user.

10.2 Details of business requirement

10.2.1 Use case

Prediction of house prices is expected to help people plan to buy a house; the model shows them the price range in the future, then they can plan their finance well. Input of the model will be the feature of the houses. As output this model will give the price of houses based on their facilities.

10.2.2 Background

House price prediction can help the estate developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are several factors that influence the price of a house such as its location, physical condition, and the year of its building. To solve this issue an estate agency wants a machine learning model that predicts the price of houses with the minimum prediction error.

10.2.3 Purpose/objectives

- Getting an estimate of the price of a certain house based on its features.

- Providing both sellers and buyers with a tool for house price estimation.

10.2.4 Stakeholders of the product

Stakeholders of the product considered for this Reference Guide are:

- Property buyers and sellers

- Real estate agency, who provides the price prediction service

10.2.5 Initial demands of the stakeholders

- Property buyers and sellers and the real estate agency demand that the product will give proper estimation of the house prices in the specific region.
- Property buyers and sellers and the real estate agency expect that the product will accept wide range of attributes which impact house prices.
- Property buyers and sellers and the real estate agency expect that product should be simple to use and comprehend.

10.2.6 Details of the business requirements

Business requirements for the product to be developed are discussed in detail below

Functional requirements

Functional requirements of the final AI system are given below:

- The AI model will estimate house price based on provided information about house feature.

Non-functional requirements

Non-functional requirements of the deployed AI system are given below:

- The AI model performs well with houses in the State of Iowa; houses outside the region are considered out of scope.
- Houses of a wide price range; from cheap to expensive are considered.
- In case a value is missing for a certain feature, the AI model will continue prediction.
- The AI model should be robust in estimating the price of the houses whose feature values are very uncommon.

Dependencies

- There are no dependencies issues for this product.

Constraints

- Data constraint: owners' personal information will not be included in the feature list though it can influence the price of the property.

Risk and concerns

- Wrong prediction can mislead the distribution in the market.

10.2.7 Requirements concerning external qualities

The expected level of quality requirements in terms of three major external qualities for the AI software to be developed are given below:

Safety

- There is no risk of physical injury related to this product.
- There is a possibility of economical loss if the price prediction of new and good house is not accurate.

Performance

- The final AI system should satisfy agreed upon thresholds of the KPI measures by the stakeholders and the developers.
- Balance in accuracy, precision is expected for overall system.

Fairness

- There are no identifiable requirements for fairness of the product or service.

10.2.8 Defining the levels of external qualities

Table 53. The levels of external qualities to be realized.

External quality	Additional specification	Assumed severity	Realized level
Safety	AI safety levels for human-related risks	No physical damage is expected	AISL 0
	AI safety level for economic risk	Minor loss of profit; possible to avoid through monitoring by humans	AISL 0.2
Performance	AI performance level in general	KPIs will be identified beforehand but thresholds for each KPI may deviate based on other factors and best efforts will be	AIPL 1

		provided	
Fairness	AI Fairness Level for overall accuracy	No Identifiable requirement for the product or service	AIFL 0

10.2.9 Conclusion

This section described from the business standpoint the purpose behind this solution including the user's needs and expectations and high-level constraints that could impact a successful deployment. It also derived business demands for an AI based product that predicts house prices. It will help the developers to evaluate the internal qualities in the following sections.

10.3 Product specification

Details that can be proposed for product specification can be stated as:

10.3.1 Model specifications

- Type of learning: Supervised
- Type of AI model: Regression
- Model architecture: Simple deep learning
- Task to perform: Price prediction

10.3.2 Data related specifications

- Data related specifications: List of attributes to consider/attributes to ignore/specified values for certain features

10.3.3 KPI specifications

- Accuracy: LRMSE, MSE, RMSE etc.

10.4 Introduction of the datasets

10.4.1 Exploring dataset

Here we have chosen ‘House price’ dataset for a regression problem for predicting the sale-price of a house. We will use Kaggle dataset [48] for the analysis of the problem. We also can build our own dataset as per requirement.

10.4.2 Sample of input data

Now, 'GarageQual' is a simple input data for the model. Garage quality is one of the features for house price analysis. We have seen values available in our dataset for this attribute. The description of the dataset for 'GarageQual' shows that the total number of data points is 1379 and the number of unique values is 5. There are different types of garage qualities, Gd (Good), TA (Typical/Average), Fa (Fair), Po (Poor), NA (No Garage). Typical or average number of data is high that is 1311 others are nan-value.
Feature name: GarageQual,
dtype: object (description of a single attribute)

• count	1379
• unique	5
• top	TA
• freq	1311

10.5 Quality assurance procedures using MLQM Guideline

After initial investigation of the current dataset, next we explore each of the eight characteristics axes (internal qualities) of quality management in response to the achievement of two external qualities - risk avoidance and AI performance, as mentioned in the first edition of MLQM Guideline.

- A-1: Sufficiency of requirement analysis
- A-2: Coverage for distinguished problem cases
- B-1: Coverage of datasets
- B-2: Uniformity of datasets
- B-3: Adequacy of data
- C-1: Correctness of the trained model
- C-2: Stability of the trained model
- D-1: Dependability of underlying software systems
- E-1: Maintainability of qualities in use

10.5.1 A-1: Sufficiency of problem domain analysis

Definition

The sufficiency of problem domain analysis deals with analysis of risk factors in conventional software and test requirements analysis to include those risk factors when a black-box test is conducted. It is required to fully examine data design as "sufficiency of data design" in order to secure sufficient training data and test data with respect to various situations systems need to respond to. More specifically, the number and details of combinations of attribute values focused on the stage between the preparation of training data to the test process is examined at this stage.

General process or structure of analysis for sufficiency of problem domain analysis is as follows.

- Define the problem domain and check whether we have data for all ranges
- Identify the corner cases in our problem domain
- Select important features by applying different types of feature selection method
- Set up acceptable ranges of variations for the selected features

Defining problem domain

For the house price problem, we have 79 features and 1460 data points. Primarily, we can say this is our problem domain which has 79 dimensions.

Data for all possible price ranges

We need to see if we have data for all possible price ranges. For example, according to the United State Census in 2020 [49], only 0.6% of 906,967 occupied houses in Iowa cost $1,000,000 or more and their absence in the dataset is probably harmless. In contrast, 8.4% of the houses cost less than $50,000 and should be represented in the dataset.

Selecting well-defined feature dimensions

The attributes and their corresponding attribute values should cover any possible data specific scenarios that need to be considered and listed for later analysis like coverage or sufficiency.

Here we have 79 features, which we need to analyze to determine any features to exclude and see if there are any new features to add. For feature reduction we can apply different methods including PCA, correlation matrix, backward elimination, etc. For inclusion of some new features, we need a lot of human effort.

Example:

Problems like Kaggle: House Price have lots of features; both necessary and redundant. Decreasing this feature space will simplify the evaluation of dataset quality as per MLQM Guideline. So, we should not do any dimension reduction to the original dataset but feature selection. This will eliminate the unnecessary features only, which results in smaller feature space with explainable attributes.

Filtering by correlation matrix:

For filtering, first we separated the numerical data. Then we calculated the correlation matrix of the numeric dataset.

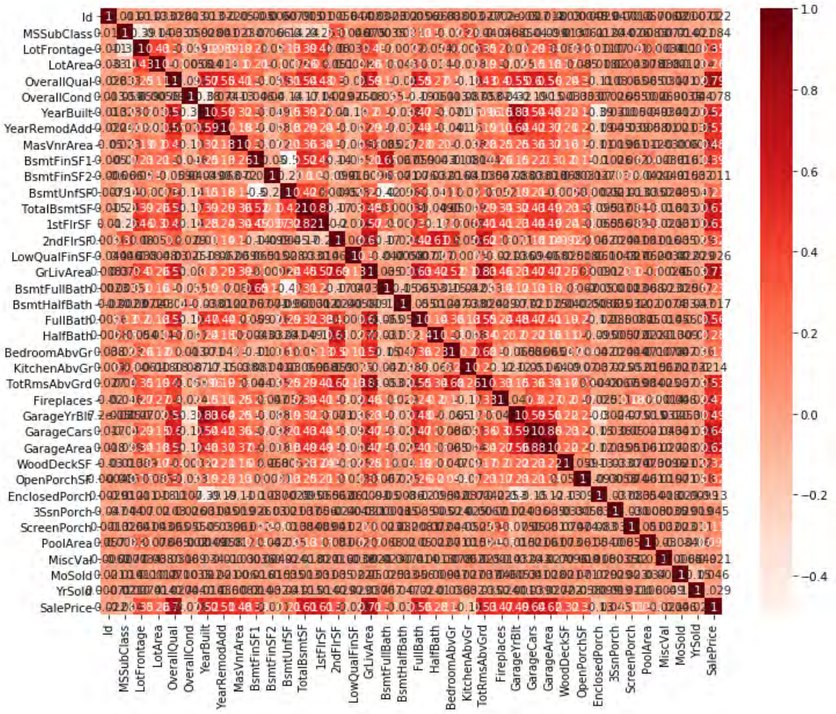


Figure 90. Correlation matrix for original numeric dataset (38 X 38).

Now, we have 38 numerical attributes from there we will select the attributes which have >=0.5 correlation factor with the 'SalePrice'. After the selection, we got 11 attributes and the reduced correlation matrix looks like below.

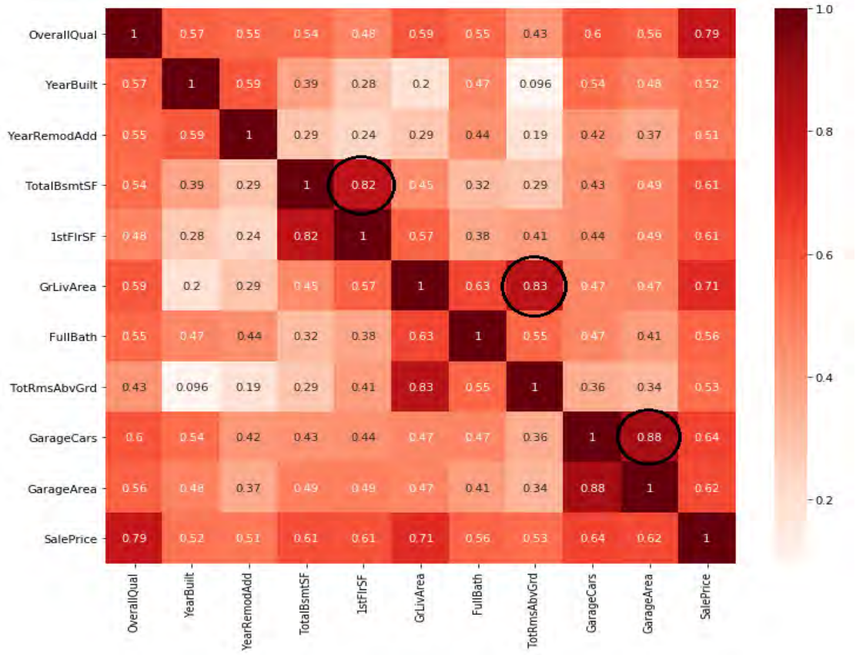


Figure 91. Correlation matrix for selected features (11 X 11).

This is much simpler than the previous. Still there are some attributes which are highly correlated (>=0.8) with each other. We have searched for those attributes and took only one of them for our feature space which have higher correlation with 'SalePrice' than the other. Thus, our selected attributes reduced to only 8 and 'Saleprice' is one of them. The selected numeric attributes are below.

'OverallQual', 'YearBuilt', 'YearRemodAdd', 'TotalBsmtSF', 'GrLivArea', 'FullBath', 'GarageCars', 'SalePrice'

In this way, the number of numerical attributes reduces from 36 to only 7. Similar reduction can be done for categorical attributes.

Even though all those seven attributes are worth analyzing, they are too many for our analysis here. Instead, we use just the following two features for well-defined feature dimensions

'GrLivArea': This attribute is total ground living area in square feet. It is a very common information for a house and an important one.

'ExterQual': Similar to the previous attribute, we have listed the categories found in our dataset for this non-numerical attribute.

Selecting ranges within and out of bounds

The acceptance of variations in the selected features that are to be considered in our problem domain should be declared specifically. User requirements should be prioritized here. As the designer we define the range of values which we should include and exclude. Take, for example, a feature called 'GrLivArea'. Sale price is closely related to it. The United States Census in 2019 [50] shows that houses whose square footage falls in 1,000-1,499 square feet are most frequent nationwide and the median of square footage per person is 700 square feet. Houses in Iowa have in average 1,550 square feet [51], In Des Moines, the state capital, new zoning laws make it harder to build small houses (less than 1,100 square feet) and easier to build family homes upwards of 1800 square feet. But we have old houses from 1930 whose sizes are in low ranges. So here we choose acceptance variation for our ground live area to be from 300 to 5000 square feet.

We have another feature called 'ExterQual', a non-numerical feature. It has attributes called Ex = Excellent, Gd = Good, TA = Typical/Average, Fa = Fair. As a solution designer I will keep this as our acceptance of variation.

Identifying unsound cases

Any combination of attributes which looks like impossible need to be excluded from our analysis. For example, in some cases the house has a pool, but it does not have the area sufficient for a house with a pool.

Conclusion

Finally, we have selected two feature 'GrLivArea' and 'ExterQual' for our analysis from 79 feature and these two features are well explained. Now the problem domain analysis is complete, and our definitions of their ranges cover all the relevant area. Every possible real data can be fit on this dimension. We can say our requirement is fulfilled.

10.5.2 A-2: Coverage for distinguished problem cases

Definition of 'coverage for distinguished problem cases':

The term *coverage for distinguished problem cases* means that sufficient requirements analyses are made concerning the situations where machine learning based systems are used in real world and their analysis results cover all possible situations.

General process or structure of analysis for coverage for distinguished problem cases:

- Choose the dataset for the defined problem
- Apply different types of augmentation or annotation rules to add new features if necessary.
- Check if we have enough data for our potential ranges

Data management in each feature dimension

From the above defined problem domain, the numbers of all possible combinations of attribute values need to be calculated.

- Here, we need to design our dataset that fits our defined problem domain. As it is very difficult to make our own dataset, we choose our dataset from available data that match our problem domain; for here we are choosing Kaggle house price analysis data. First, we need to check if the dataset is like our problem domain. For similarity check we can check coverage or distribution in the *coverage of datasets* part.
- If our dataset is as expected that is fine. If we do not have enough available data, then we need to do some augmentation. If we as the solution designer want to include new features into the existing data set which have no relation with the existing attributes, we need to give some human effort like annotation.
- Since house price is a discreate dataset numerical augmentation is quite impossible.
- Now we need to check the distribution of the sale prices for checking whether we have data for all possible price ranges. Here, we will estimate Iowa state house prices, and this is our distribution of sale prices for Iowa.

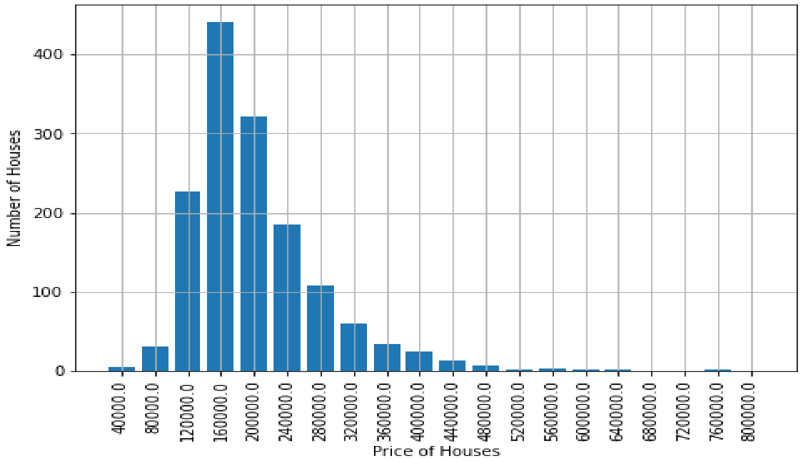


Figure 92. Distribution for sale prices.

When we design the dataset, we find a huge number of combinations of features. As a solution designer we need to identify important and less important combinations. From problem domain analysis if we want to combine some of the less important features into one useful feature, we will introduce numerical method for that purpose. This will help reducing the dimensions and complexity of feature space. So that, solution designer can eliminate some combinations that are less important or inappropriate or risky.

Conclusion

Kaggle House Prices is a regression problem, and its feature space is vast. Neither adding new feature nor data augmentation nor feature deletion is feasible. This data is not friendly for our data design. If we could define or make our own dataset then the problem would be solved. But since we don't have enough manpower and time to do so, we will use this dataset for the analysis.

10.5.3 B-1: Coverage of datasets

The term *coverage of datasets* means that a sufficient amount of data is given to cases covered by establishing the standard distribution as described in the previous paragraph without any part being missed or overlooked in response to possible input corresponding to those cases.

General Process or structure of analysis for coverage of datasets:

- Check coverage for our selected combinations
- Identify rare or corner cases
- Try to find out features or some ranges of values we can exclude from the problem domain

Coverage for each combination

For each described combination of features to be considered, data coverage should be calculated and matched against previously set standards of coverage. This depicts the range of the training dataset as well as scope of the test dataset.

- We need to see distribution of our dataset over the feature space covered by combinations of ranges in all feature dimensions in the problem domain we defined. We should check how well our dataset covers each combination.
- A complete distribution of training data over both feature dimensions is presented in the following table.

Table 54. Distribution of the training data over the feature space.

	ExterQual			
GrLivArea	Ex	Gd	TA	Fa
5001-6000	1	0	0	0
4001-5000	2	1	0	0
3001-4000	2	5	6	1
2001-3000	25	103	68	0
1001-2000	21	359	630	5
0-1000	1	20	202	8

- The defined range of 'GrLivArea' is between 300 to 5000 square feet. From this table we can see that we have data for all ranges of values of that feature, which means the dimension has full data coverage.

Identifying rare/corner cases

For certain combination of cases, there can be lack of data points, but they may be extremely important cases. These are considered rare or corner cases. Decisions need to be made about the necessity of generating or gathering data with such cases.

- We must check whether each region is covered by some data or not. For example, the region where 'GrLivArea' ranges from 5000 to 6000 has only one data point, hence this can be considered as rare or corner case.

Feature deletion

In some cases, data points may be rare, but it has very little effect on the whole system. During model evaluation phase, we can identify such rare cases by fully excluding them from training but not from testing. If the outcome is similar, then the feature can be said unnecessary.

- From the table we can see that in 'GrLivArea' in the region of 5000 to 6000 we have only one data point. We do not have much coverage in that region. We can delete or reduce the range but that will create limitation of the model.

Conclusion

In this problem we have chosen two features. They have not enough coverage in some regions and our model will fail in those regions. If we could add more data for those regions, that would be good for our model. But at this time we cannot do so and the dataset is not suitable for training and testing the model for those regions. So, we must say that coverage of the dataset is not enough for our selected problem. Here our selected data set cannot pass this internal quality test.

10.5.4 B-2: Uniformity of datasets

Definition

A concept contrary to *coverage* mentioned earlier is *uniformity* of data in relation to the overall assumed input data. When each situation or case in datasets is extracted in accordance with the frequency of its occurrence in whole data to be input, data is considered as "uniform".

General Process or structure of analysis for uniformity of dataset:

- Check the distribution of our selected feature
- Try to compare between expected and actual distribution
- Seeing the distribution, we can update our problem domain if it is necessary
- Finally, we will make the decision

Enough data for each case

We need to make sure that there are enough data for every possible case scenario.

The distribution of the data points should be measured and analyzed against expected distribution. The distribution is expected to follow real world distribution.

Example: An expected uniform distribution of a training dataset with respect to the selected attributes in the defined region of interest is shown in Figure 93.

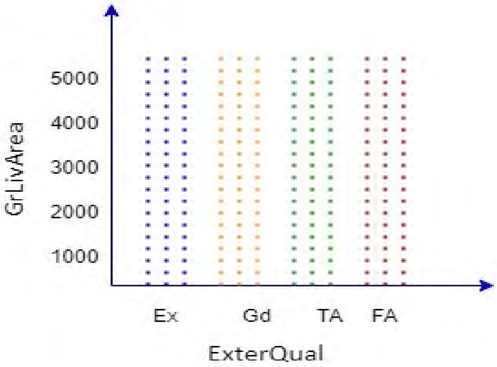


Figure 93. Expected distribution on houses for the selected feature space.

The actual distribution is shown in Figure 94.

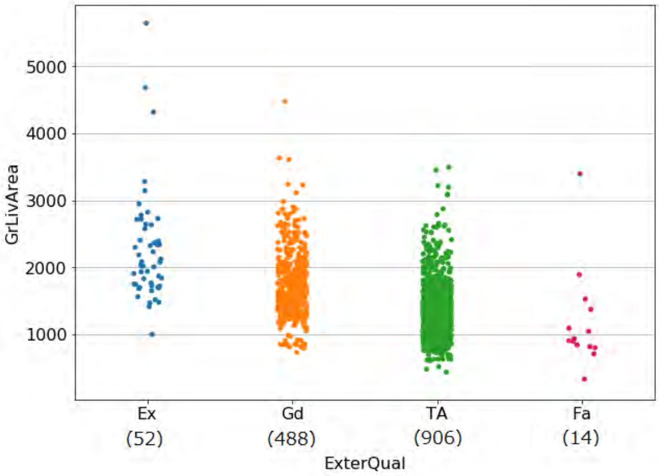


Figure 94. Actual distribution of training data in the selected feature space.

In this distribution graph, we can see there are no 'Ex' houses below 1000 sq ft and the number of houses between 4000 to 5000 sq ft is very small. Similarly, 'Gd' houses in that range are even less and of 'TA' houses there are none in that range.

There are a few houses with 'Fair' external quality which are comparatively smaller houses in our defined range of problem domain. Also, the 'Ground Living Area' of the houses are mostly from 1000 to 3000 sq ft.

Conclusion

Here our expected distribution and actual distribution do not match. So, we must say that the distribution is not uniform. It can be said that problem domain is well covered, but data points are not uniformly distributed. Our selected problem is failing in this quality testing measurement.

10.5.5 C-1: Correctness of trained models

Definition of 'correctness of the trained model'

The term *correctness of trained models* means that a machine-learning component reacts to specific input data included in learning datasets (consisting of training data, test data and validation data) as expected.

General Process or structure of analysis for correctness of the trained model:

- Select KPI for the performance checking
- Check the performance of the model by using the KPI
- Try to find out the cases where model performance is failing
- Make the decision after all analysis

Selecting specific method

The method of evaluation should be described both for convergence against training data and achievement against test data.

Example: First, we need to select KPI for the performance checking. There are different types of KPI for the evaluation.

- Mean Square Error
- Root Mean Squared Error
- Mean Absolute Logarithmic Error

Mean Square Error: In statistics, Mean Square Error (MSE) is defined as mean or average of the square of the difference between actual and estimated values. This is used as a model evaluation measure for regression models and the lower value indicates the better fit. By using Mean Square Error, we have found error result 3.211.

$$\boldsymbol{Mean\ Square\ Error} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}} \qquad (33)$$

Root Mean Squared Error: (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of the best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results. By using RMSE, we have found error result 1.791

$$\boldsymbol{Root\ mean\ Squared\ Error} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\frac{(\boldsymbol{\hat{y}_i} - \boldsymbol{y_i})^{\mathbf{2}}}{\boldsymbol{n}}} \qquad (34)$$

Mean Absolute Logarithmic Error: It is the difference between the measured value and "true" value. Since sales price distribution is scattered, we use logarithmic error. By using mean absolute logarithmic error, we have found error results 0.7653.

$$\boldsymbol{Mean\ Absolute\ Logarithmic\ Error} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\boldsymbol{log\ y_i - log\hat{y}_i}| \tag{35}$$

From the analysis we have seen different measurement by using different types of KPI. Now we will compare **performances of models trained with datasets of different sizes** using mean absolute logarithmic error:

First, we trained and evaluated a fully connected **dense layer** with data from different regions of the dataset coverage table. The results are listed in the table below.

Table 55. Mean absolute logarithmic errors.

	ExterQual							
GrLivArea	Ex		Gd		TA		Fa	
	Data	Err.	Data	Err.	Data	Err.	Data	Err.
5001-6000	1	-	0	-	0	-	0	-
4001-5000	2	-	1	-	0	-	0	-
3001-4000	2	-	5	134.567	6	8.69171	1	-
2001-3000	25	8.95012	103	6.51191	68	18.5707	0	-
1001-2000	21	40.3462	359	1.44147	630	1.33367	5	46.6369
0-1000	1	-	20	4.42439	202	3.72669	8	41.8109

Here, the error is mean absolute logarithmic error obtained from cross validation of available data with 'fold = 4'. We plotted the error results against the number of training data.

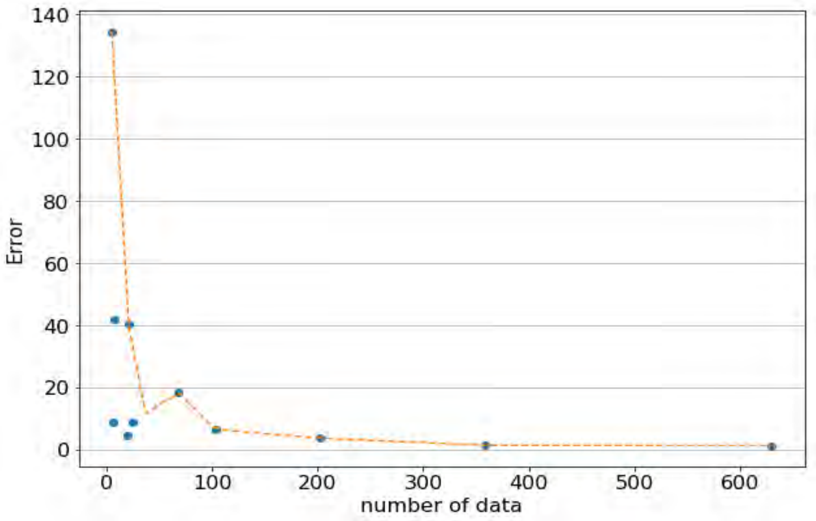


Figure 95. Error vs. number of training data interpolation

The curve is like exponentially decaying which is as expected. Error is inversely proportional to amount of data.

$$\mathbf{Error} \propto \frac{\mathbf{1}}{\boldsymbol{number\ of\ data}} \tag{36}$$

Performance comparison for different distribution of data

In the above case, the categorical attribute was inactive during training; it was redundant. Now, we want to see how different distributions of attribute affect the performance of a model. For this analysis, we have made sub-groups of data like below.

Table 56. Distribution of data along GrLivArea.

	ExterQual			
GrLivArea	Ex	Gd	TA	Fa
5001-6000	1			
4001-5000	3			
3001-4000	14			
2001-3000	196			
1001-2000	1015			
0-1000	231			

Here, we could have taken all 1460 data points but then distribution of 'GrLivArea' would have effects too on model performance. From Figure 96 showing the distribution plot of this attribute below, we can see most of the data lies between 1001-2000 sq ft. By taking only this range, we will minimalize the effect of the distribution of data samples with respect to GrLivArea on training and evaluation.

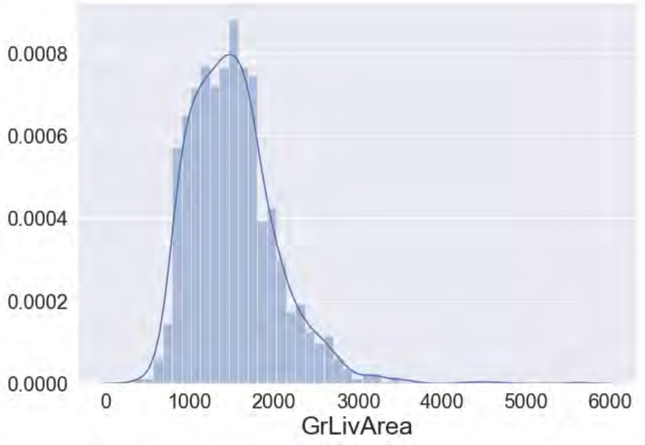


Figure 96. Distribution of data samples with respect to GrLivArea.

So, we took the sub-group with best number of data samples; *<u>'1001-2000 sq ft GrLivArea'</u>*. The distribution of the selected sub-group for attribute 'ExterQual' is given in Figure 97.

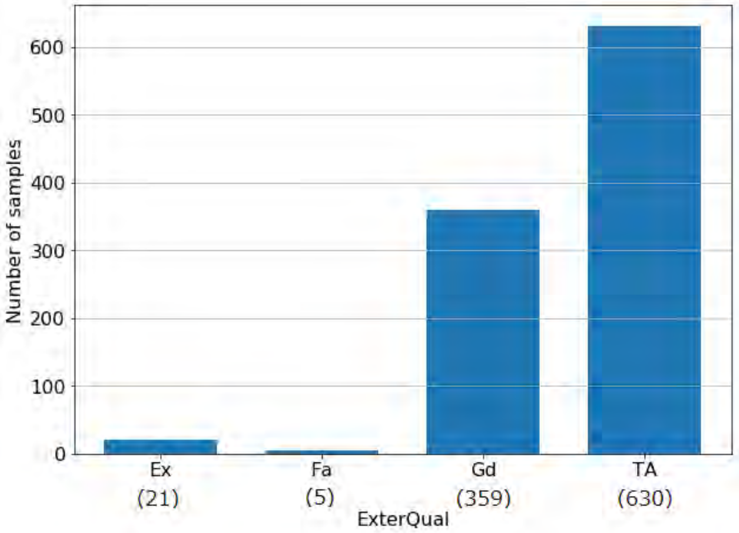


Figure 97. Distribution of sub-group of training data (GrLivArea: 1001-2000 sq ft) for ExterQual

To understand the effect of data distribution, we took three different combination of categories and then trained and evaluated using the same previous model. Obtained results along with the combinations taken has been summarized in Table 57.

Table 57. The results of training with data from different combinations of categories.

Combination of categories	Number of rarest data	Error
TA + Fa (630 + 5)	5	1.28569
TA + Ex (630 + 21)	21	1.18049
TA + Gd (630 + 359)	359	0.91479

Here, "Number of rarest data" indicates the number of data available for the rarest category in a certain combination.

Here we can see that when the number of rarest data is low, error rate is become high on the other hand when the number of rarest data is high error become low. So, these results indicate that *having rare cases in dataset is necessary but we need enough amount of data samples to represent those cases.*

We have measured the corner case seeing the data distribution. From now a model can identify corner cases by measuring the performance of the model.

Conclusion

Prediction accuracy can be shown by KPI. Here we are using mean absolute logarithmic error for the KPI measurement. Our model fails in some cases because we do not have enough coverage in some regions. From the above analysis we can say the more we have data coverage the less error we get. We can also say that the more uniform the data are the better the performance is.

10.5.6 C-2: Stability of trained models

Definition

The term *stability of trained models* means that a machine-learning component shows a reaction to input data which is not included in learning datasets sufficiently like data in learning datasets. The predictability of behaviors of the machine-learning component improves by eliminating unpredictable behaviors caused by low generalization capabilities or adversarial examples. Stability is strongly related to the machine learning lifecycle so that it needs to be evaluated and enhanced mainly in these phases in order to achieve the stability goals

General process or structure of analysis for stability of the trained model:

- Select different model for the performance checking
- Do some parameter tuning and check the loss curve
- Finally, make the decision

Example:

We have some iterative training phase to avoid over-fitting of training datasets through separating training datasets and validation datasets. We will monitor the whole training process after evaluating the impact of minimal changes in input on output. Now, we have chosen a full connected model for ‘House price analysis’ problem. The model architecture is given in Table 58.

Table 58. The architecture of the model used for *house price analysis*.

Architecture	FC(128)+ReLU FC(256)+ReLU FC(64)+ReLU FC(10)+ ReLU
Number of trainable parameters	88,449

When the batch size is 32 and epochs is 300; our training loss: 0.0212, validation loss: 0.0514. Figure 98 shows the loss curve for training and testing.

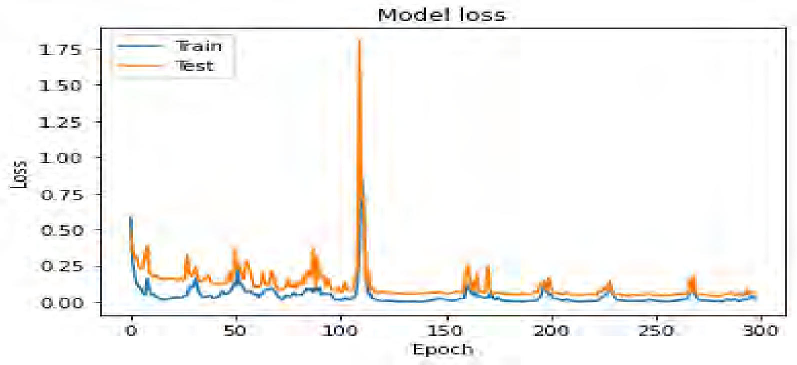


Figure 98. Loss curve of simple training and testing.

In Figure 99, we have the same model architecture but doing some hyperparameter tunning we get our training loss: 0.0451 and validation loss: 0.1237 when epochs: 100. We can see from the loss curve that training and testing curve associate with each other.

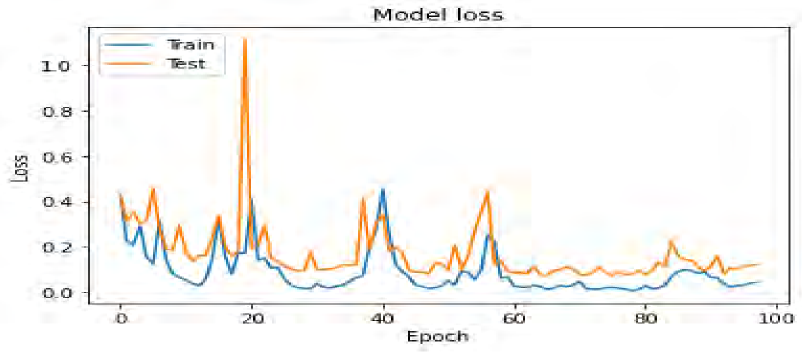


Figure 99. Loss curve of training and testing with hyperparameter tuning.

After changing the model architecture, we will check if there any changes impact on output.

Here is our model architecture:

Table 59. The architecture of the model.

Layer (type)	Output shape	Parameter
dense_84 (Dense)	(None, 19)	5776
dense_85 (Dense)	(None, 19)	380
dense_86 (Dense)	(None, 19)	380
dense_87 (Dense)	(None, 19)	380
dense_88 (Dense)	(None, 1)	20

Trainable params: 6,936

The training loss: 0.0241 and validation loss: 0.0494, epochs: 100

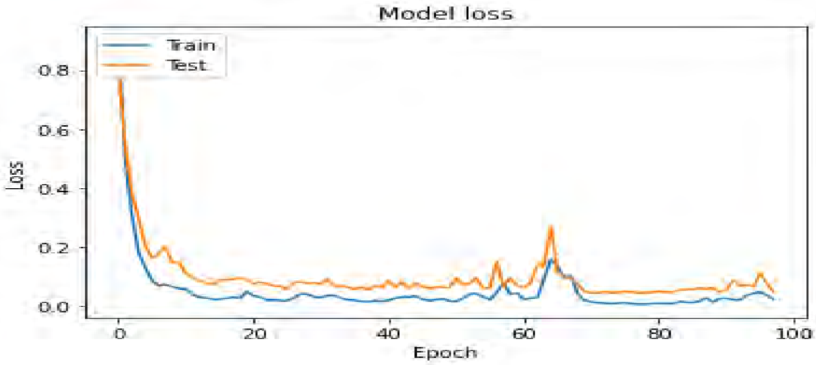


Figure 100. Loss curve of training and testing with another model.

Here we can see from those output results that differences in model parameter and architecture do not give any difference in the result of the output. We can see that training loss and validation loss is corelated with each other, so overfitting problem is not happening for our problem.

Here are some research paper links [52] [53] for measuring stability.

Conclusion

The internal quality *stability of the trained model* can be improved by measuring the model's generalization ability, evaluating the model's reaction to corner cases or rare cases, and evaluating the model's performance on adversarial examples. The quality concerns robustness, that is, the model's performance for unknown inputs or under new environmental conditions.

10.5.7 D-1: Reliability of underlying software systems

Definition

The term *reliability of underlying software systems* means that training software components used in the machine learning training stage and prediction/inference software component used when they are executed operate correctly in response to given training data or trained machine learning models. In addition to the correctness as algorithms, the general quality requirements for software such as fulfillment of memory resource constraint and time constraint, and software security are included here.

General process or structure of analysis for reliability of underlying software systems:

- Language
- Framework
- Usage of memory
- Metaparameters

- Hardware
- Software security

Language

We can work with Python, R, Java, Julia, and Scala for ML. Python language has been used for developing this AI. It uses various open-source packages which should be version compatible with each other. So, the list of used packages and their versions should be provided by the developer.

Table 60. The list of packages and their versions.

Programing language	Version
Python	3.6.12
Package	Version
NumPy	1.18.5
TensorFlow	2.3.1
Pandas	1.1.5
Matplotlib	3.3.2

Framework

ML models can be run on a variety of frameworks, all of which run at different speeds. The most popular are Keras, TensorFlow, Caffe, Theano, Microsoft CNTK, PyTorch and scikit-learn. Each framework is different from the next and was created to suit different needs. TensorFlow, Keras and Theano run neural networks very fast, AWS is generally robust, and scikit-learn is best for tabular data. Some frameworks allow us to pay to get faster results. Here for this project, we use TensorFlow and Keras.

Usage of memory

It shows you how much computer memory your model is using, and how much is available, so we need to define a minimum and maximum usage of memory when the AI device is in operation combining data storage, model parameters, codes/algorithms and others.

- **Model architecture and weights:** Here we use Hierarchical Data Format (HDF) file (.h5), a format sometimes used with Keras, to store trained AI networks and their weights. The saved network has 88,449 parameters and takes about 2 MB space on the hard drive.
- **Input data:** The general meaning of input is to provide or give something to the computer, in other words, when a computer or device is receiving a command or signal from outer sources, the event is referred to as input to the device. So, the memory usage by the input data can be defined after complete design of the machine.

- **Code/algorithm:** An *algorithm* in machine learning is a procedure that is run on data to create a machine learning *model*. Machine learning algorithms *learn* from data or are *fit* on a dataset. There are many machine learning algorithms. Different algorithms written in different programing languages are part of the workflow of the machine. Such code does not take much space on hard drive.
- **Dataset for retraining:** We need to keep a space for holding dataset for possible re-training phase at least the size of the actual dataset. For example, the house price dataset takes about 449.88 KB space.

Metaparameters

Metaparameters are values input to our ML algorithm that tell it how to behave (thereby influencing the training/predicting time of our algorithm). Not all models have the same metaparameters.

- **Learning rate (η, eta):** As our learning rate increases, the computational time of our model decreases.
- **Number of features:** As the number of features in our model increases, the computational time of your model also increases. (In NNs, this can be number of layers; in KNNs, the value of k; etc.)
- **Number of rounds/epochs:** If we increase the number of rounds or epochs for a machine learning model, it will take longer to train (but the prediction time is the same).
- **Objective:** Some ML models are adaptable for different objectives. Different objectives have different train and prediction times (usually regression takes longer in binary than in count).
- **Early stopping:** Some ML implementations allow you to stop training your model early (automatically) if your model performs well enough on a validation dataset. Adding this early-stopping functionality will never hurt run time.
- **Others:** Every ML model has different metaparameters which can influence training time that must be attended to.

Hardware

Changing the hardware for the model runs on is an expensive though simple way to make your model run faster. There are three main processing units: CPUs, GPUs, and TPUs.

Tensor Processing Units (TPUs) are proprietary property of Google that can be accessed through Google Cloud and are constantly being improved. They run fast for neural networks. They are the chosen processing unit of DeepMind. TPUs have higher input/output operations per Joule than any existing GPU.

GPUs are faster than CPUs. If anyone increase the number of processing units your model runs on, model will train and predict faster. Most image recognition algorithms run especially fast on GPUs. Some GANs for image generation only run on GPUs. There are some cases where the bit length of the CUDA matrix calculation is different from each other.

Example[10]: In some version of TensorFlow or Chainer, 32 bits is enough, and 64 bits is too much under the consideration of precision of matrix calculation and consumption of the memory or resources of calculation. Some versions of Quadro (nvidia) don't support the 32bit. In such case, no effect for the acceleration of calculation for 32 bits with the expensive Quadro GPU. At that time, 32 bit was supported with the GeForce so people tended to use GeForce instead of Quadro. On the other hand, Intel released the instruction set and driver software specialized for AI/ML. So, the provider of the framework (TensorFlow, Pytorch or so) tends to support both of the implementation (CUDA 32bit and Intel driver). Sometimes 16 bits is enough for some business solution and GPU was not used in another solution. In the house price problem, we can check soundness of our problem by using different environment like using GPU then we can compare difference between them by reproduce the result. Our model training using house price dataset can be done in reasonable time with 2 GB of RAM and no GPU. The best configuration of H/W or S/W will change frequently according to the technical progression so the designer should search the current technical information and decide the best configuration with good balance.

Software security

Security is a major concern when machine operates online. For the themes that the solution designer should consider while building the application sets with AI/Machine learning functionality, see the chapter on security in the Guideline (Chapter 9 in the second edition).

Conclusion

It takes not only code, algorithm, data etc., but also the components around to construct a complete application when we talk about AI solution. We listed and identified some of the most important components of the 'house price prediction' machine in the above section.

10.5.8 E-1: Maintainability of quality during operation

Definition

The term *maintainability of quality during operation* means that internal qualities

[10] Based on observations as of ca. 2019.

satisfied at the commencement of operation are maintained throughout operation. This concept means that internal qualities can fully respond to changes in operational environments outside the system and that any change in trained machine learning models do not cause unnecessary deterioration of quality.

It is required to continuously monitor behaviors of machine learning based systems and machine learning components for the purpose of checking if the quality fulfilled at the commencement of operation is maintained throughout the operation period.

General Process or structure of analysis for Maintainability of quality during operation:

- Accuracy (KPI) monitoring
- Model output monitoring
- Input data monitoring

Accuracy (KPI) monitoring

Accuracy monitoring directly measure the accuracy of trained machine learning models. This monitoring is divided into some patterns in accordance with the method of collecting correct answers to inference results of trained machine learning models required for calculating the accuracy.

Model output monitoring

Model output monitoring is further categorized into a case where each output inference is checked by experts as in the case of medical diagnostic and a case where all inferences are checked altogether after a certain period.

Input data monitoring

Input data monitoring refer to the monitoring of results of inferences made by a trained machine learning model and the monitoring of its input data. The monitoring methods can be done human monitoring in case of house price prediction data. We need to check how to handle cases where quality deteriorates.

Conclusion

For this internal quality we did not show any particular results, but maintenance of machine learning technology helps improving its models both in accuracy and robustness.

11 Automated guided vehicle

This section presents just business requirement description of an automated guided vehicle as an example of such description developed with more detailed specifications of its usage and environmental conditions than the other example.

11.1 Product name

Functional Safety Transport Vehicle DRC-X

11.2 Use case

The following use cases are assumed for this transport vehicle.

1. The service provider uses special software to input map information and destinations for the facility in advance.
2. The user loads the cargo in the cargo bed, selects a destination if necessary, and starts the vehicle.
3. The destination can be selected according to the schedule or by the user using the screen of the control unit. The system may also receive commands from an external system that manages the transport.
4. After a warning tone is emitted to notify people around the vehicle that the vehicle is about to move, the vehicle will start moving and automatically travels to the destination. In this case, real-time monitoring is not required.
5. An upper limit is set for the traveling speed of the transport vehicle. The upper limit is automatically changed according to the traveling conditions.
6. Acceleration and deceleration should be such that there is no concern about stability, such as a collapse of cargo.
7. The transport vehicle travels while bypassing obstacles. If an obstacle approaches in the direction of travel and a collision is foreseen, the vehicle will slow down by reducing the upper speed limit. However, if the vehicle passes near a human, it will slow down regardless of the above. If it is unclear whether the obstacle is a human or a non-human, it assumes that it is a human.
8. The vehicle shall stop when it comes close enough to an obstacle to collide with it. If the obstacle is far enough away after the vehicle detects the closeness and

stops, the vehicle shall emit a warning sound after a certain time and start running with the upper speed limit restricted to a low speed, and then gradually release the restriction.

9. Even if the obstacle remains close to the vehicle after a certain period has passed after the vehicle detects the closeness and stops, if there is a direction in which the vehicle can proceed without any obstacles, the vehicle will emit a warning sound to notify the people around it that the vehicle is about to move, limit the speed to a very low speed, and attempt to escape by proceeding in a direction without any obstacles. After escaping, the vehicle will stop at once and resume normal driving after emitting a warning sound to notify people around the vehicle that the vehicle is about to move.
10. When the vehicle arrives at the destination, the vehicle will stop, and a warning tone will be emitted to notify users that the vehicle has arrived. The user shall unload the luggage from the vehicle.
11. An emergency stop switch is located on the outside of the vehicle, and pressing the switch will stop the vehicle reliably. At this time, the vehicle automatically notifies the service provider of the warning. Even if the emergency stop switch is returned to its original position, the vehicle will remain stationary until the service provider releases it.
12. The vehicle automatically moves to the standby position according to the user's instructions, the service provider's remote commands, or the schedule.
13. The service provider shall conduct periodic inspections.
14. The developer shall supply replacement consumable parts and perform repair services over a period of time.
15. If the vehicle detects an abnormality, such as a complete loss of position or a malfunction, the vehicle will stop, and a warning will be sent to the service provider.
16. If the vehicle is stopped for a long period of time, or if there is any other abnormality, the service provider will visit the site or act remotely.
17. The service provider can obtain onboard camera images and location information remotely and can give direct driving instructions. However, driving instructions that ignore proximity judgments of obstacles must be given in presence, not remotely.
18. When disposing of the device, remove the battery and entrust disposal to an industrial waste disposal company.

11.3 Business requirements

11.3.1 Background

As the working population declines, there is a growing need for automated transport vehicles to meet the manpower-saving needs of commercial facilities, factories, and logistics sites. To make use of such vehicles in more various workplaces, it is desirable to use a system that does not require a dedicated lane and shares the same activity space with humans. In this case, safety is a critical issue, and not only to avoid harmful collisions, but also to control behavior that may causes surprise and anxiety. This is in contradiction to the availability perspective, where transportation should be achieved by moving as fast as possible. Therefore, it is desirable that a system prioritizes speed only when doing so is safe and causes no anxiety to people.

11.3.2 Purpose/objectives

To ensure safety, including the risk of failure.

For the purpose of reducing the feeling of uneasiness in the surrounding people by changing the behavior of the automated guided vehicle, object detection AI is used to determine if there is any human in the environment.

11.3.3 Stakeholders of the product

Stakeholders of the product considered for this Reference Guide are:

- Developer (manufacturer of the vehicle)
- Service provider (who sets up the system to suit the workplace, and performs maintenance and remote monitoring)
- Users (who use the vehicles in the workplace, and do not have to be the same on the sending and receiving sides).
- Pedestrians
- Industrial waste companies

11.3.4 Initial demands of the stakeholders

- Developer: wants to collect operational data by the vehicle to be used for future development.
- Service provider: wants to operate a useful system with little manpower.
- Users: want high-speed transfer services with a safety premise.
- Pedestrians: want the system to be safe and secure.
- Industrial waste companies: want to dispose of waste without danger.

11.3.5 Details of the business requirements

Business requirements for the product to be developed are discussed in detail below

Assumptions

- The workplace of the transport vehicle is an indoor walkway in a large facility such as a warehouse or factory.
- There is no possibility of the vehicle entering stairs, escalators, or steps because of obstacles such as roadblocks.
- The same walkway may be used by trolleys, people, and other transport vehicles.
- There are no animals, infants, or other beings in the driving environment that could maliciously or unintentionally jump into the transport vehicle.
- All safety-related obstacles are at a height that can be detected by the 2D Lidar of the transport vehicle.
- Environmental conditions, such as lighting, are suitable for the sensors.
- The cargo to be loaded is not at risk of collapse or leakage.
- The object detection AI requires real time processing power of 30 fps or more with computing resources equivalent to NVIDIA GeForce RTX2060 Mobile. For this purpose, a network model based on DarkNet or a similar architecture is applied.
- The object detection AI model uses a generic trained network model.

Dependencies:

- The system must have a communication method to call the service provider in case of abnormalities.
- If the system depends on an external system to specify the destination, the API must be additionally supported.

Constraints

- The maximum speed is 6 km/h, based on the provisions for electric wheelchairs in the Road Traffic Law.
- In accordance with the JIS D 6802 standard for automatic guided vehicles, the vehicle shall continuously emit a warning sound, and when turning or backing up, the vehicle shall notify people around the vehicle with a blinker or backing up sound.

Functional requirements

- There shall be no collision causing harm to pedestrians and also no minor harm such as pressing or scratching.
- An emergency stop button shall be located on the vehicle's body in an easy-to-press position, and the vehicle should completely stop when the button is pressed.
- Even if the object detection AI results are incorrect, there shall be no collision

causing harm to obstacles or passengers.

- The vehicle's 2D Lidar and driving control system shall be constructed as a safety-related system to detect single faults.
- The vehicle shall be equipped with 2D Lidar for obstacle detection, estimate its own position, and autonomously determine its direction of travel.
- Depending on the driving direction and speed, the vehicle determines the area to deal with obstacles and the upper speed limit for each area, and for each obstacle detected in a closer area, the vehicle decelerates gradually to stop before a collision.
- The vehicle is equipped with a camera capable of acquiring color images and depth information for object detection AI.
- The object detection AI performs instance segmentation on surrounding objects. The label output should be grouped in three categories: "human", "non-human", and "unknown".
- In general, the label output of a training dataset is classified into various categories. Output stage of a network should have additional filters to gather them into the above three categories.
- Labels are assigned to objects that entered into the obstacle response area based on the output of the object detection AI and the point cloud information from Lidar.
- When passing near an obstacle, if the object is not certain to be a "non-human", the vehicle should slow down and pass through even if there is no fear of collision.
- The learning dataset for object detection AI should include not only persons but also expected objects in the environment (desks, chairs, boxes, carts, printers or copiers, shelves, etc.).
- The training dataset must be validated by an organization other than the dataset creator.
- For objects that exist over a wide area in the same plane such as floors, walls, ceilings, the decision should be made by plane extraction using point cloud data, not by object detection AI.

Non-functional requirements

- Service provider shall immediately take action in case of abnormalities.
- Users should receive training and become proficient in the operation of the vehicle.
- The user should have accident insurance in case of emergency.
- There shall be no unreasonable invasion of privacy by information collected by sensors and cameras.

Out-of-scope issues

- It is not assumed that anyone will be riding on the transport vehicle.

- It is not assumed that the transport vehicle is used in a situation so crowded that it makes difficult for the vehicle to drive.

Risks and concerns

- Risk of harm caused by abnormal behavior as a result of multiple simultaneous failures of safety-related systems.
- Risks of harm caused by falling from steps or colliding with people or obstacles that are difficult to detect with 2-D LiDAR.
- Risks of traffic congestion or blockage due to traffic concentration of similar vehicles.
- Risks of spreading hazards such as dirt, illness, or fire as a result of driving.
- Risks of information leakage, disguised driving commands, or abnormal behavior caused by cyber-attacks.

11.3.6 Requirements concerning external qualities

The expected level of quality requirements in terms of three major external qualities for the AI software to be developed are given below.

Safety

- The motor power of this product may cause physical damage resulting in serious injury or death due to a collision.
- Crashes into the environment may result in economic loss, such as destruction of equipment.
- Improper handling of charging equipment may result in fire or other hazards.

Performance

- The transport vehicle system should satisfy the thresholds of KPI indicators agreed upon by stakeholders.
- The entire system should have a good balance between processing speed and safety.
- The 2D Lidar should support functional safety equivalent to SIL2 and be able to set a safety protection area equivalent to a travel distance of at least 3 seconds at maximum speed. In addition, it must be able to acquire point cloud information for a distance of at least three times that distance.
- The vehicle needs to be equipped with an electromagnetic brake that activates the braking state when the power is shut off.
- The object detection AI shall be capable of processing at least 30 frames per second.
- The label output of the object detection AI shall be less than 10% "non-human" for humans and more than 50% "non-human" for non-humans.
- The object detection AI should use training data that can determine that a person in a wheelchair is also a human.

Fairness

There are no identifiable requirements for fairness of the product or service.

11.3.7 Defining the levels of external qualities

The functional safety level of this product except for the object detection AI is supposed to achieve SIL 2 defined in IEC61508.

Based on MLQM Guidelines, the following are the levels of external quality required for the object detection AI.

External Quality	Additional specification	Assumed severity	Realized level
Safety	AI Safety Level for human-related risks	No physical damage is expected	AISL 0
	AI Safety Level for economic risks	Minor loss of profit; impossible to avoid through monitoring by humans	AISL 1
Performance	AI Performance Level in general	KPIs will be identified beforehand but thresholds for each KPI may deviate based on other factors and best efforts will be provided	AIPL 1
Fairness	there are no identifiable requirements for fairness of the product or service		AIFL 0

11.4 Conclusion

This case is described from a business perspective, including the needs and expectations of the user, the objectives behind the solution, and the high-level constraints that may affect the success of the deployment. Here, we have derived the business requirements for automated transfer vehicles. This report is a snapshot on the PoC exit phase and help developers to assess the internal quality in the next steps.

Appendices

A. Business requirement description

In the beginning of each example application, hypothetical business requirements are derived for a fictional AI product. Developers of a product usually receive this kind of formal requirements from the entities that want the product to be developed (service providers/owners). The authors of the Reference Guide set the requirements themselves in this section just as an example so that it can be used to evaluate the internal qualities throughout the entire product development process in the next phase.

In many cases, the presented document with the given business requirements may not reflect actual scenarios. Following are some clarifications about the presented business requirements that can help the readers understand its purpose, applicability, and limitations.

A.1 Choice of stakeholders

Business requirements (BRs) are supposed to come from the needs of the stakeholders of the related product. So, identifying all the stakeholders for the product can lead us to different perspectives of the BRs.

However, MLQM Guideline helps the stakeholders who are primarily concerned with safety, performance, and fairness related requirements. While choosing relevant stakeholders, the reference guide examples have the same intention. For this reason, the most relevant stakeholders are mentioned whose demands have a direct impact on the functional and nonfunctional requirements of the product.

A.2 Recursive adjustments

It is acknowledged that business requirements for a product/service are usually set after multiple iterations of discussion and adjustments between the service providers (entity who wants the product/service to be developed) and developers (entity who will develop the product). Even though the iterative process is followed while writing the business requirement descriptions, only the finalized business requirements are mentioned for simplicity and limited scopes of the reference guide.

A.3 Choice of perspective

Usual business requirement documents cover various perspectives that are related to every phase of product life cycle. For example, some key elements of a business requirement description (BRD) such as constraints and dependencies often contain non-technological perspectives. They may be related to financial or schedule related requirements. The presented BRDs do not mention them since they are outside the scope of the reference guide. In real cases, such requirements are included in BRDs and maintained accordingly.

A.4 Requirements of additional systems

The entire product/service often has elements other than the AI module. Monocular cameras capturing video streams and systems that extracts images from the videos can be such examples for autonomous vehicles. Usually, BRDs contain requirements related to those additional parts as well. However, the reference guide is concerned about evaluating qualities of the AI part of the system only. Hence, in the presented BRDs, only the requirements that are closely related to the development of AI systems are included.

A.5 Choice of format

In real cases, there is practice of writing business requirements, functional requirement specifications (FRS) and software requires specifications (SRS) individually. For simplicity, the presented BRDs may include elements from traditional FRS and SRS. Here, the goal is not to standardize the structures of these documents, rather it is to exemplify how MLQM Guideline can be incorporated with such documents and how it can assist service providing entities to express their expected quality goals more explicitly.

After realizing the expected quality requirements of a product, it should be evident which aspects are mandatory during development and where achieving the best possible solution is enough. This realization can be distinctively expressed using external quality levels mentioned in MLQM Guideline. For this reason, at the end of BRDs, requirements are expressed in terms of external quality axes of the Guideline to make them clear and comprehensive for the developers.

It is not mandatory to identify external quality levels in a business requirement document. However, this practice can undoubtedly straighten up expectations of quality goals to achieve for both service providers and developers.

B. Surprise adequacy

Surprise adequacy [19] is a mechanism that study the quality of the data in a dataset. For this purpose, surprise adequacy defines an adequacy criterion that quantitatively measures behavioral differences of validation to the training data. For the measurement, surprise adequacy uses *activation trace* (**AT**) to follow the activation of the neurons. Let $N=\{n_1, n_2, . . .\}$ be a set of neurons that constitutes a neural network N, and let $X=\{x_1, x_2, . . .\}$ be a set of inputs. The activation value of a single neuron n with respect to an input x is defined as $a_n (x)$. For an ordered set of neurons, let $N \subseteq \boldsymbol{N}$, $a_N(x)$ denote a vector of activation values, each element corresponding to an individual neuron in N: the cardinality of $aN(x)$ is equal to $|N|$. $aN(x)$ is the activation trace of x over the neurons in N. Therefore, for the set of inputs X, we can define the activation traces as $A_N(X)=\{a_N(x) | x \in X\}$.

Regarding surprise adequacy, it is computed the activation traces of all training data ($A_N(T)$). After that, surprise adequacy computes the activation trace of a new input x from the validation data ($A_N(x)$). The result is obtained comparing the $A_N(x)$ to $A_N(T)$. There are different mechanisms to compare them, but for this reference guide we are only using *distance-based surprise adequacy* (**DSA**). We do not discard to use different comparison mechanisms for surprise adequacy in the future. It is not the objective of this reference guide to explain in more detail surprise adequacy, activation traces and DSA.

DSA has been defined using the Euclidean Distance between the AT of a new input x and ATs observed during training:

$$\boldsymbol{dist_a} = \|\boldsymbol{a_N}(\boldsymbol{x}) - \boldsymbol{a_N}(\boldsymbol{X_a})\| \tag{37}$$

$$\boldsymbol{dist_b} = \|\boldsymbol{a_N}(\boldsymbol{X_a}) - \boldsymbol{a_N}(\boldsymbol{X_b})\| \tag{38}$$

$$\boldsymbol{DSA}(\boldsymbol{x}) = \frac{\boldsymbol{dist_a}}{\boldsymbol{dist_b}} \tag{39}$$

Additionally, we have defined 3 more DSA measurements, that we have given the name of DSA_1, DSA_2 and DSA_3, being DSA_0 the original DSA explained before. These new DSAs are used to detect the corner cases in a dataset. Figure 101 represents the differences among these DSAs. DSA_1 compares x's novelty in its belonging class and its class novelty to other classes. $dist_a$ is the same as DSA_0.

$$\boldsymbol{x_b} = \underset{c_{x_i}\in\{C-c_x\}}{\mathbf{argmin}}\ \boldsymbol{xe^{-x^2}}\ \|\boldsymbol{a_N}(\boldsymbol{x}) - \boldsymbol{a_N}(\boldsymbol{x_i})\| \tag{40}$$

$$\boldsymbol{dist_b} = \|\boldsymbol{a_N}(\boldsymbol{x}) - \boldsymbol{a_N}(\boldsymbol{x_b})\| \tag{41}$$

DSA$_2$ compares the testing data *x* to data of all classes.

$$\boldsymbol{m} = \frac{\mathbf{1}}{\boldsymbol{k}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{k}} \boldsymbol{x_i}, \{\boldsymbol{x_i}|\boldsymbol{c_{x_i}} = \boldsymbol{c_s}\} \tag{42}$$

$$\boldsymbol{dist_a} = \|\boldsymbol{a_N}(\boldsymbol{x}) - \boldsymbol{a_N}(\boldsymbol{m_a})\| \tag{43}$$

$$\boldsymbol{dist_b} = \|\boldsymbol{a_N}(\boldsymbol{x}) - \boldsymbol{a_N}(\boldsymbol{m_b})\| \tag{44}$$

where, m$_a$ represents the center of class c$_a$ (c$_a$ =c$_x$); m$_b$ is the nearest center point of class c$_b$, (c$_b$ ∈ {C – c$_x$}).

DSA$_3$ compares the center of the neighborhood of data *x* to the k-nearest neighborhood. dist$_a$ and dist$_b$ are the same as DSA2, but:

$$\boldsymbol{m} = \frac{\mathbf{1}}{\boldsymbol{k}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{k}} \boldsymbol{x_i}, \{\boldsymbol{x_i}|\boldsymbol{c_{x_i}} = \boldsymbol{c_s}\ \&\ \boldsymbol{x_i} \in \boldsymbol{N_k}(\boldsymbol{x})\} \tag{45}$$

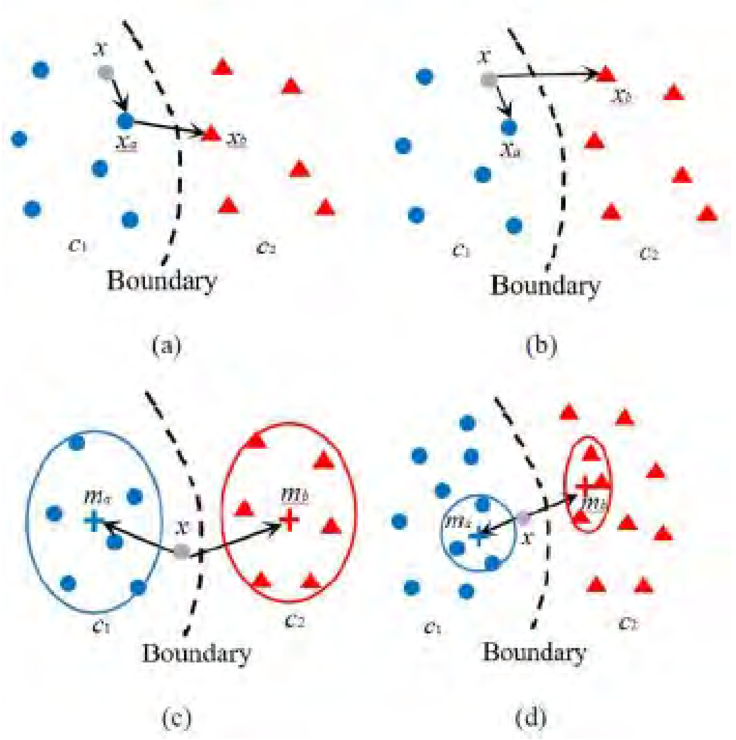


Figure 101. Diagram of Four Types of DSA:
the Original *DSA_0 (a), DSA_1 (b), DSA_2 (c) and DSA_3 (d)*

For the experiments in the main chapters, corner cases that has more than 1 in their distance are removed from the BDD100k dataset. The idea is to detect and remove the corner cases from the training dataset and re-train all detection models: Yolov3 + ASFF, Yolov4, Yolov5, Fast R-CNN and MobileNetv2.

C. 1-pixel change

In the literature, there are three widely used distance metrics for generating adversarial examples, all of which are L_p norms.

The L_p distance is written $||x - x_0||_p$, where the *p*-norm $||\cdot||_p$ is defined as :

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{\frac{1}{p}} \tag{46}$$

1. L_0 distance measures the number of coordinates *i* such that $x_i \neq x'_i$. Thus, the L_0 distance corresponds to the number of pixels that have been altered in an image.

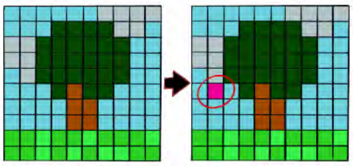


Figure 102. L₀ Image Example

2. L_2 distance measures the standard Euclidean (root mean-square) distance between x and x' images. The L_2 distance can remain small when there are many small changes to many pixels.

Figure 103. L₂ Image Example

3. L_∞ distance measures the maximum change to any of the coordinates:

$$\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty = \mathbf{max}(|\boldsymbol{x_1} - \boldsymbol{x'_1}|, \dots, |\boldsymbol{x_n} - \boldsymbol{x'_n}|) \tag{47}$$

FGSM uses L∞ to generate adversarial examples. The definition of FGSM also explains its name: It is the gradient of the loss function, and because of the L∞ bound on the perturbation magnitude, the perturbation direction is the sign of the gradient.

Figure 104. L∞ image example

For evaluating robustness, the maximum safe radius (MSR) is often used. MSR for the image *A* and the trained model (classifier) *f* is the distance such that

$$\boldsymbol{MSR}(\boldsymbol{A}, \boldsymbol{f}) = \mathbf{max}\{|\boldsymbol{A} - \boldsymbol{A}'|_{\boldsymbol{p}} | \boldsymbol{f}(\boldsymbol{A}) = \boldsymbol{f}(\boldsymbol{A}')\} \tag{48}$$

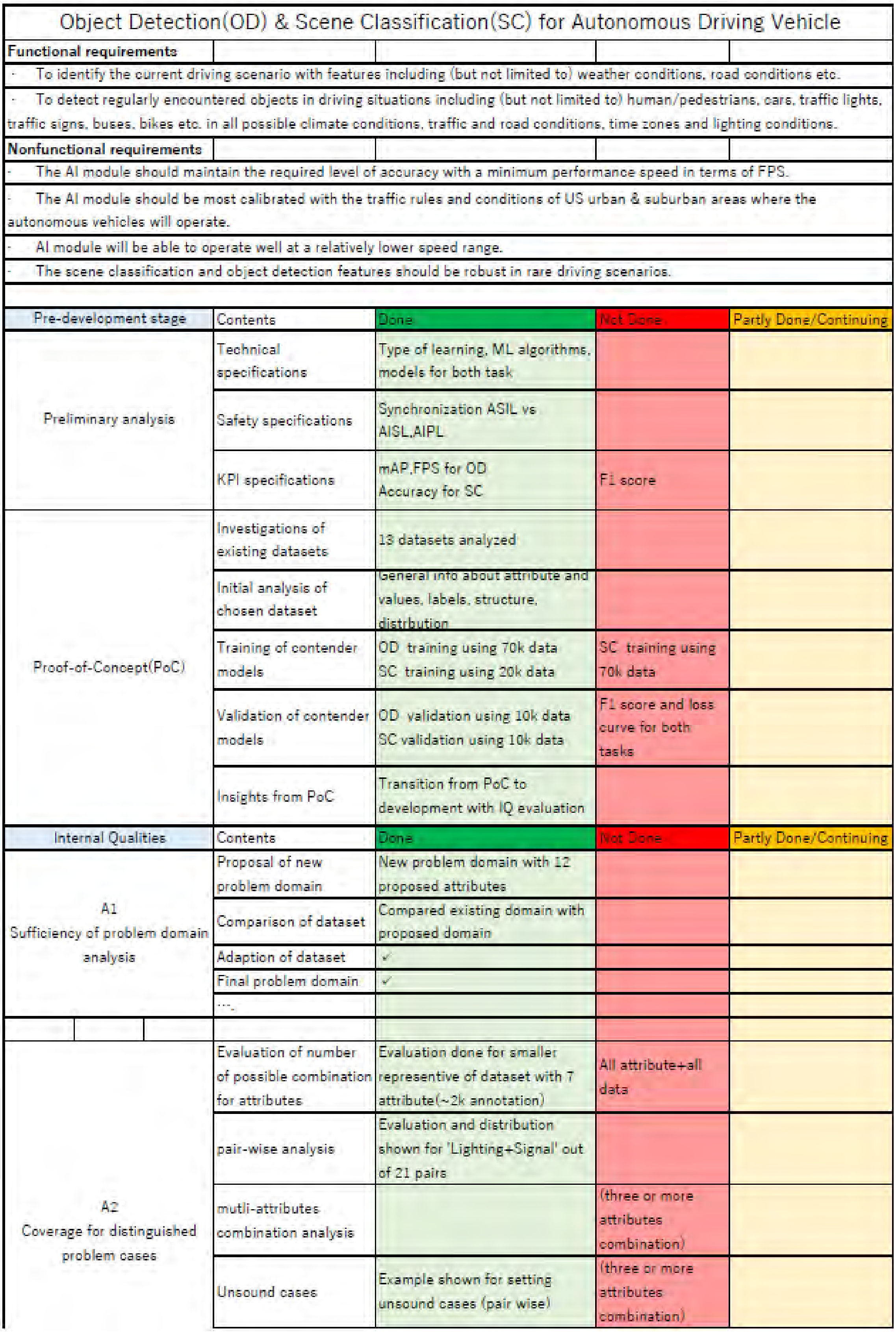
In other words, MSR is the distance to the nearest adversarial example.

Regarding the BDD100k, the solution designer estimated the MSR for each label. Nevertheless, due to the research already has FGSM that is L∞, the solution designer tries to generate adversarial examples of L_0 and/or L_2. It has been tried to develop a method for L_2 without success, however it was possible to generate adversarial examples using L_0. This means, it could only use *p=0* for the adversarial generation. L_0 is based on adding noise to images changing pixels on images. Therefore, the solution designer developed 1-pixel attack method that is based on L_0.

D. Summaries of assessments

This section presents tables showing summaries of assessments conducted for each of the first four examples given in this Guide. The tables also list assessments that are mentioned but conducted only partially or not performed at this time. That certain assessments were not fully conducted does not indicate that they are of less importance. The reasons why they were not conducted fully vary and are often constraints on availability of data or other resources.

D.1 Autonomous driving vehicle



Object Detection(OD) & Scene Classification(SC) for Autonomous Driving Vehicle

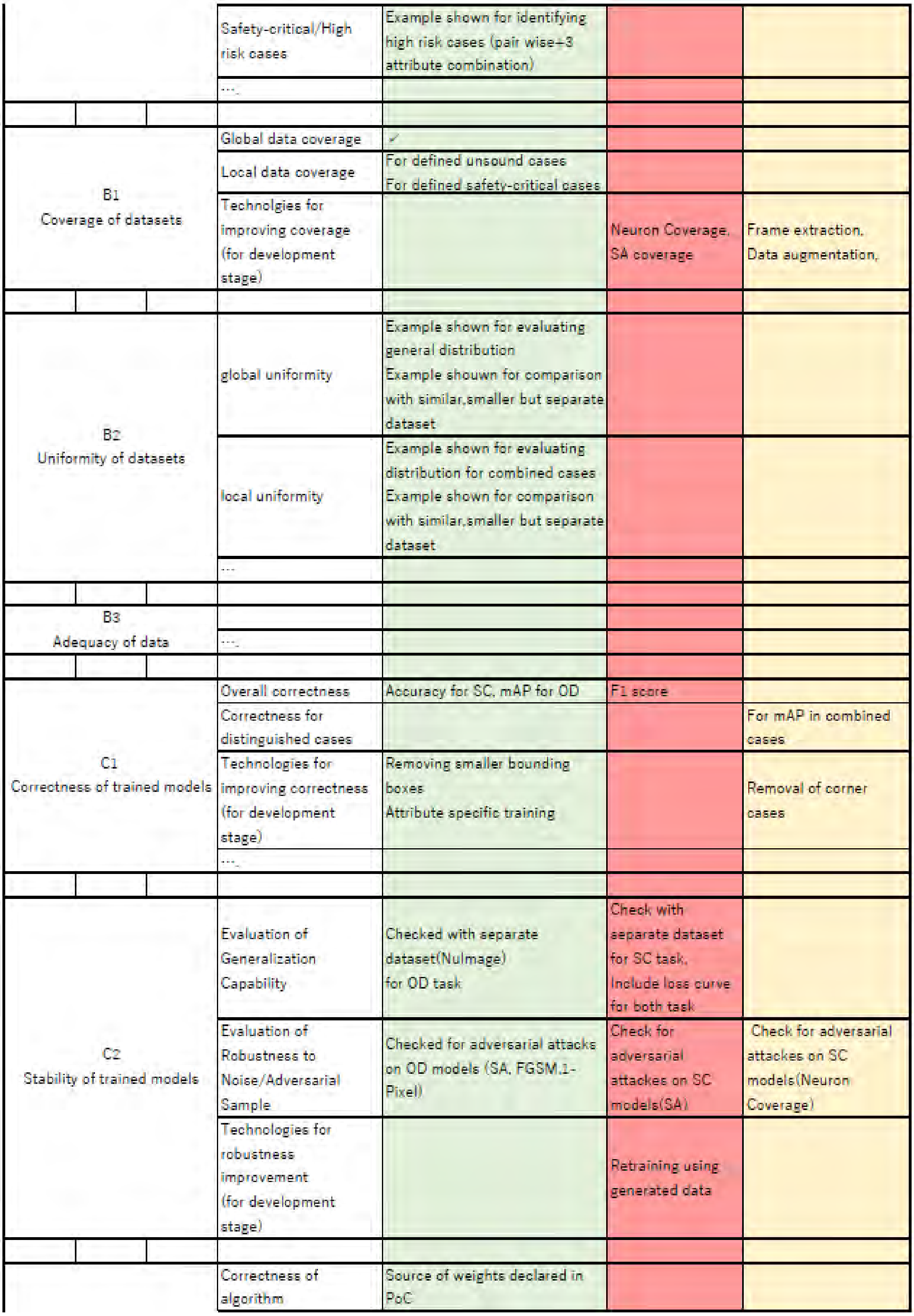
Functional requirements

- To identify the current driving scenario with features including (but not limited to) weather conditions, road conditions etc.
- To detect regularly encountered objects in driving situations including (but not limited to) human/pedestrians, cars, traffic lights, traffic signs, buses, bikes etc. in all possible climate conditions, traffic and road conditions, time zones and lighting conditions.

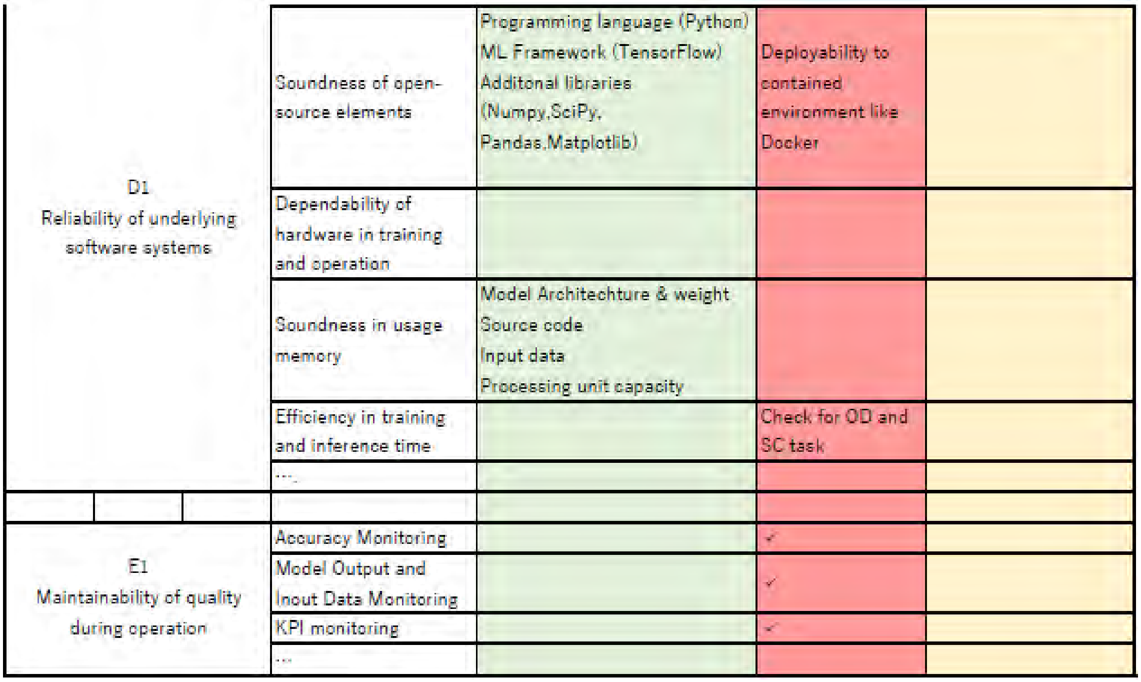
Nonfunctional requirements

- The AI module should maintain the required level of accuracy with a minimum performance speed in terms of FPS.
- The AI module should be most calibrated with the traffic rules and conditions of US urban & suburban areas where the autonomous vehicles will operate.
- AI module will be able to operate well at a relatively lower speed range.
- The scene classification and object detection features should be robust in rare driving scenarios.

Pre-development stage	Contents	Done	Not Done	Partly Done/Continuing
Preliminary analysis	Technical specifications	Type of learning, ML algorithms, models for both task		
	Safety specifications	Synchronization ASIL vs AISL,AIPL		
	KPI specifications	mAP,FPS for OD Accuracy for SC	F1 score	
Proof-of-Concept(PoC)	Investigations of existing datasets	13 datasets analyzed		
	Initial analysis of chosen dataset	General info about attribute and values, labels, structure, distrbution		
	Training of contender models	OD training using 70k data SC training using 20k data	SC training using 70k data	
	Validation of contender models	OD validation using 10k data SC validation using 10k data	F1 score and loss curve for both tasks	
	Insights from PoC	Transition from PoC to development with IQ evaluation		
Internal Qualities	**Contents**	**Done**	**Not Done**	**Partly Done/Continuing**
A1 Sufficiency of problem domain analysis	Proposal of new problem domain	New problem domain with 12 proposed attributes		
	Comparison of dataset	Compared existing domain with proposed domain		
	Adaption of dataset	✓		
	Final problem domain	✓		
	….			
A2 Coverage for distinguished problem cases	Evaluation of number of possible combination for attributes	Evaluation done for smaller representive of dataset with 7 attribute(~2k annotation)	All attribute+all data	
	pair-wise analysis	Evaluation and distribution shown for 'Lighting+Signal' out of 21 pairs		
	mutli-attributes combination analysis		(three or more attributes combination)	
	Unsound cases	Example shown for setting unsound cases (pair wise)	(three or more attributes combination)	

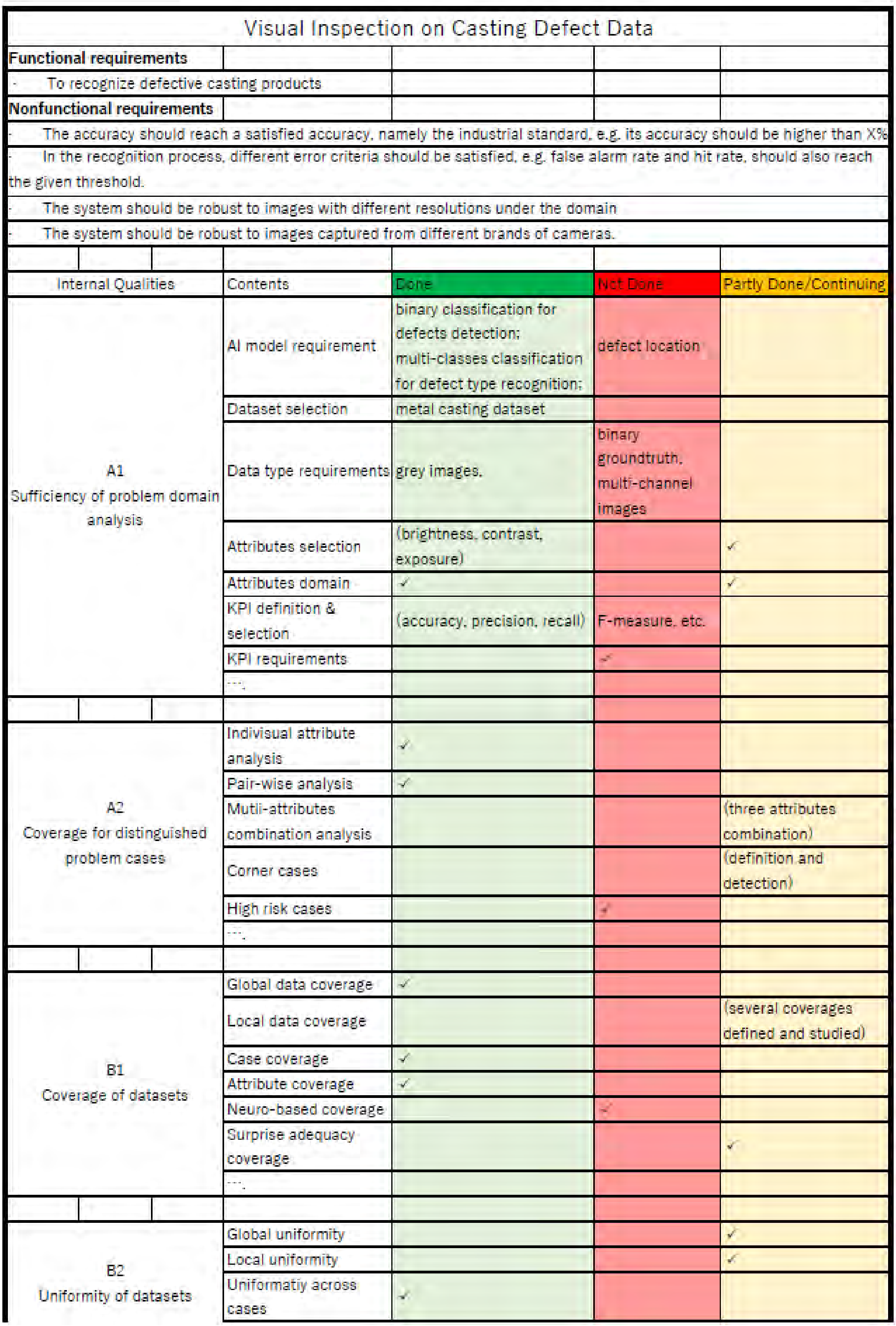


	Safety-critical/High risk cases	Example shown for identifying high risk cases (pair wise+3 attribute combination)		
	・・・			
B1 Coverage of datasets	Global data coverage	✓		
	Local data coverage	For defined unsound cases For defined safety-critical cases		
	Technolgies for improving coverage (for development stage)		Neuron Coverage, SA coverage	Frame extraction, Data augmentation,
B2 Uniformity of datasets	global uniformity	Example shown for evaluating general distribution Example shouwn for comparison with similar,smaller but separate dataset		
	local uniformity	Example shown for evaluating distribution for combined cases Example shown for comparison with similar,smaller but separate dataset		
	・・・			
B3 Adequacy of data				
	・・・			
C1 Correctness of trained models	Overall correctness	Accuracy for SC, mAP for OD	F1 score	
	Correctness for distinguished cases			For mAP in combined cases
	Technologies for improving correctness (for development stage)	Removing smaller bounding boxes Attribute specific training		Removal of corner cases
	・・・			
C2 Stability of trained models	Evaluation of Generalization Capability	Checked with separate dataset(NuImage) for OD task	Check with separate dataset for SC task, Include loss curve for both task	
	Evaluation of Robustness to Noise/Adversarial Sample	Checked for adversarial attacks on OD models (SA, FGSM,1-Pixel)	Check for adversarial attackes on SC models(SA)	Check for adversarial attackes on SC models(Neuron Coverage)
	Technologies for robustness improvement (for development stage)		Retraining using generated data	
	Correctness of algorithm	Source of weights declared in PoC		



D1 Reliability of underlying software systems	Soundness of open-source elements	Programming language (Python) ML Framework (TensorFlow) Additonal libraries (Numpy,SciPy, Pandas,Matplotlib)	Deployability to contained environment like Docker	
	Dependability of hardware in training and operation			
	Soundness in usage memory	Model Architechture & weight Source code Input data Processing unit capacity		
	Efficiency in training and inference time		Check for OD and SC task	
	….			
E1 Maintainability of quality during operation	Accuracy Monitoring		✓	
	Model Output and Inout Data Monitoring		✓	
	KPI monitoring		✓	
	…			

D.2 Visual inspection of metal casting



Visual Inspection on Casting Defect Data

Functional requirements

- To recognize defective casting products

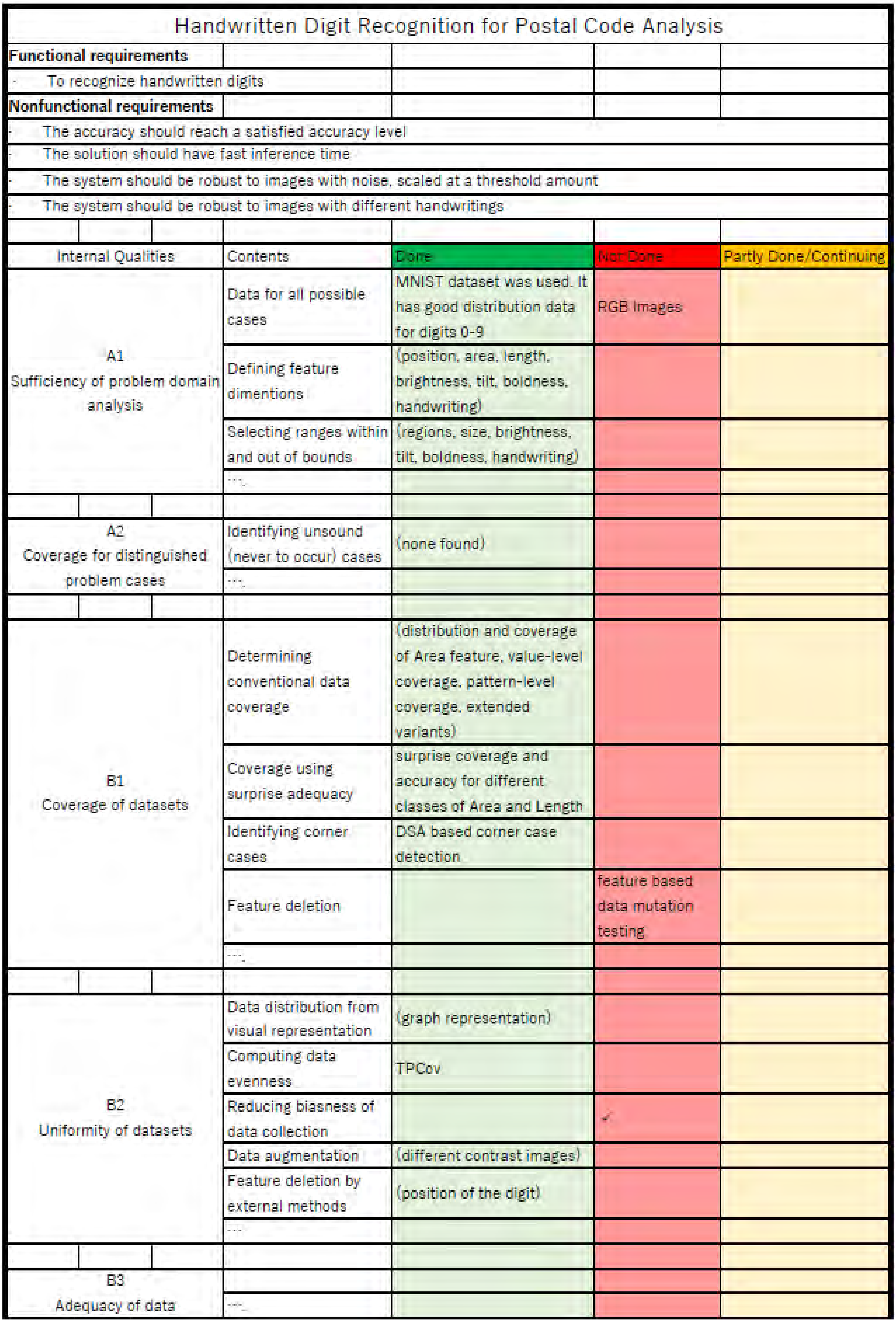
Nonfunctional requirements

- The accuracy should reach a satisfied accuracy, namely the industrial standard, e.g. its accuracy should be higher than X%
- In the recognition process, different error criteria should be satisfied, e.g. false alarm rate and hit rate, should also reach the given threshold.
- The system should be robust to images with different resolutions under the domain
- The system should be robust to images captured from different brands of cameras.

Internal Qualities	Contents	Done	Not Done	Partly Done/Continuing
A1 Sufficiency of problem domain analysis	AI model requirement	binary classification for defects detection; multi-classes classification for defect type recognition;	defect location	
	Dataset selection	metal casting dataset		
	Data type requirements	grey images,	binary groundtruth, multi-channel images	
	Attributes selection	(brightness, contrast, exposure)		✓
	Attributes domain	✓		✓
	KPI definition & selection	(accuracy, precision, recall)	F-measure, etc.	
	KPI requirements		✓	
	---.			
A2 Coverage for distinguished problem cases	Indivisual attribute analysis	✓		
	Pair-wise analysis	✓		
	Mutli-attributes combination analysis			(three attributes combination)
	Corner cases			(definition and detection)
	High risk cases		✓	
	---.			
B1 Coverage of datasets	Global data coverage	✓		
	Local data coverage			(several coverages defined and studied)
	Case coverage	✓		
	Attribute coverage	✓		
	Neuro-based coverage		✓	
	Surprise adequacy coverage			✓
	---.			
B2 Uniformity of datasets	Global uniformity			✓
	Local uniformity			✓
	Uniformatiy across cases	✓		

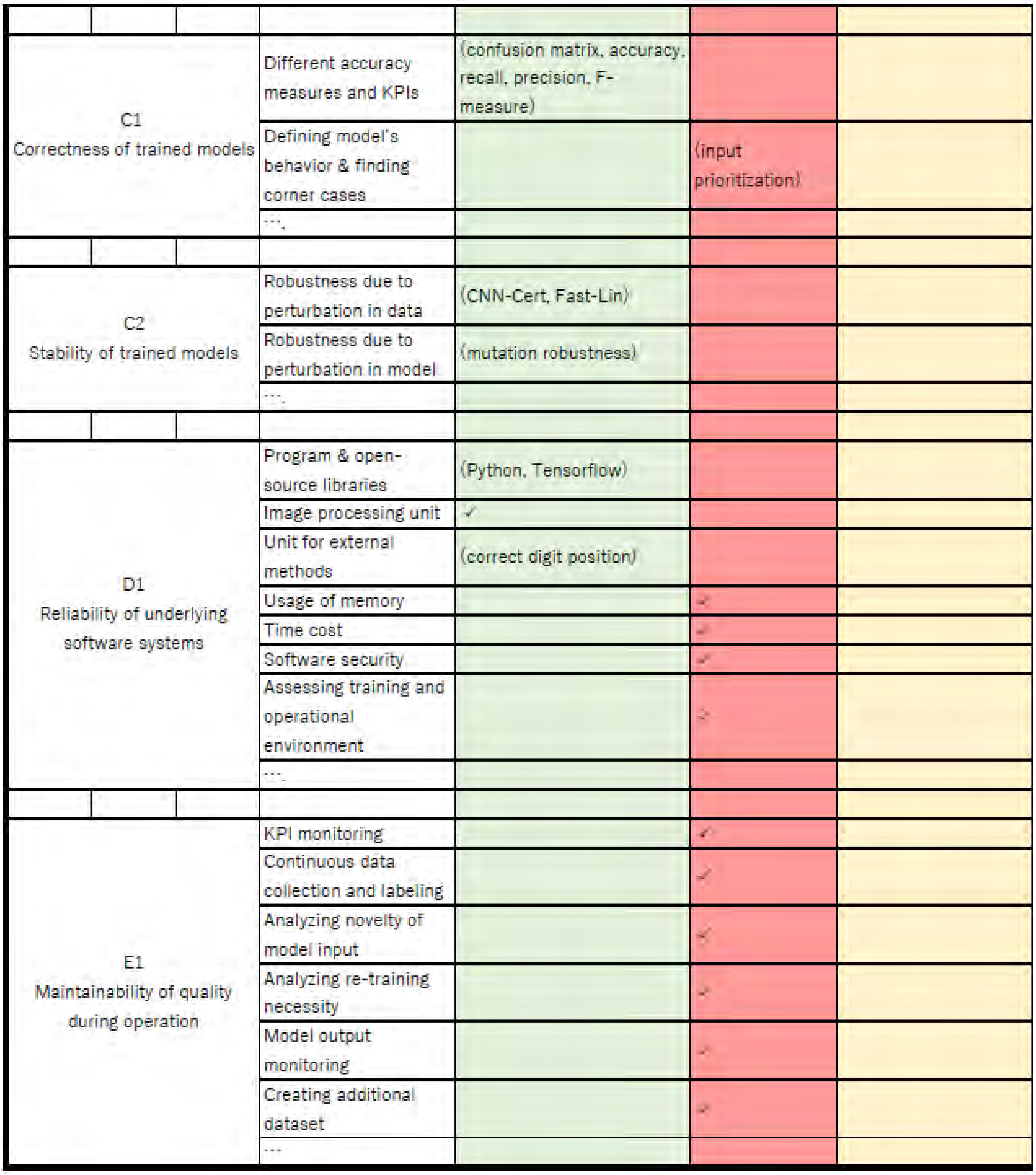
	···			
B3 Adequacy of data	Data augmentation			✓
	Adversarial generation		✓	
	Metamorphic generation		✓	
	Corner cases generation		✓	
	···			
C1 Correctness of trained models	Overall correctness	(accuracy)		
	Correctness of positive events w.r.t. total observed posititve events	(recall)		
	Correctness of positive events w.r.t. total predicted posititve events	(precision)		
	Correctness of cases			✓
	Correctness of corner case detection	✓		
	···			
C2 Stability of trained models	Robustness measurement	(local robustness measurement)		✓
	Robustness improvement			(with consideration of corner cases)
	Adversarial testing		✓	
	Mutation testing		✓	
	···			
D1 Reliability of underlying software systems	Reliability evaluation on data		(casting defect data)	
	Reliability evaluation on platform		(Python, TensorFlow, PyTorch)	
	Reliability evaluation on excution enviroment		(Windows, MacOs)	
	Reliability evaluation on phsyical opertation environment		✓	
	···			
E1 Maintainability of quality during operation	Maintainability evaluation		✓	
	Online training		✓	
	Risk management		✓	
	···			

D.4 Postal code analysis



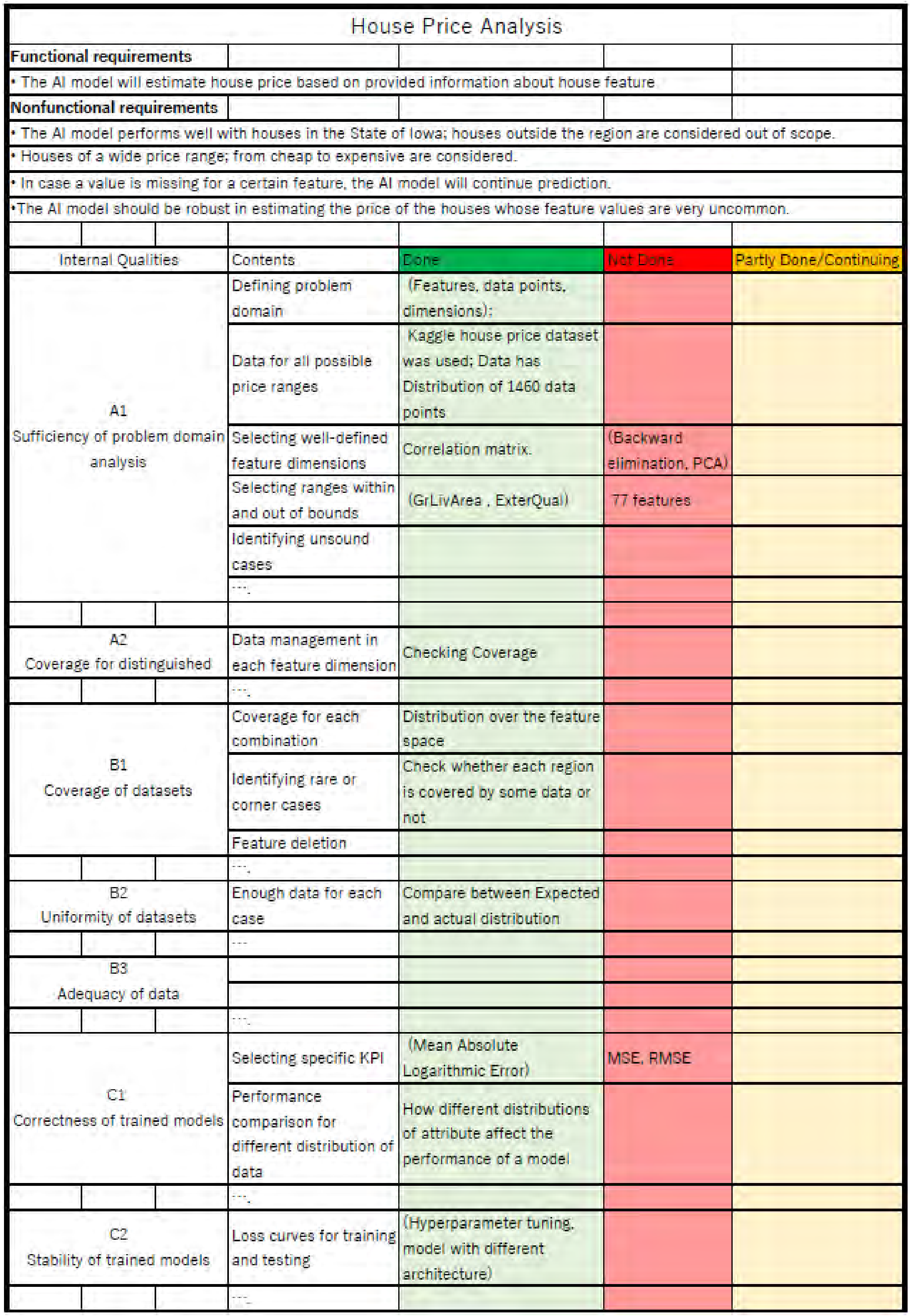
Handwritten Digit Recognition for Postal Code Analysis

Functional requirements				
· To recognize handwritten digits				
Nonfunctional requirements				
· The accuracy should reach a satisfied accuracy level				
· The solution should have fast inference time				
· The system should be robust to images with noise, scaled at a threshold amount				
· The system should be robust to images with different handwritings				
Internal Qualities	Contents	Done	Not Done	Partly Done/Continuing
A1 Sufficiency of problem domain analysis	Data for all possible cases	MNIST dataset was used. It has good distribution data for digits 0-9	RGB Images	
	Defining feature dimentions	(position, area, length, brightness, tilt, boldness, handwriting)		
	Selecting ranges within and out of bounds	(regions, size, brightness, tilt, boldness, handwriting)		
	···.			
A2 Coverage for distinguished problem cases	Identifying unsound (never to occur) cases	(none found)		
	···.			
B1 Coverage of datasets	Determining conventional data coverage	(distribution and coverage of Area feature, value-level coverage, pattern-level coverage, extended variants)		
	Coverage using surprise adequacy	surprise coverage and accuracy for different classes of Area and Length		
	Identifying corner cases	DSA based corner case detection		
	Feature deletion		feature based data mutation testing	
	···.			
B2 Uniformity of datasets	Data distribution from visual representation	(graph representation)		
	Computing data evenness	TPCov		
	Reducing biasness of data collection		✓	
	Data augmentation	(different contrast images)		
	Feature deletion by external methods	(position of the digit)		
	···.			
B3 Adequacy of data				
	···.			



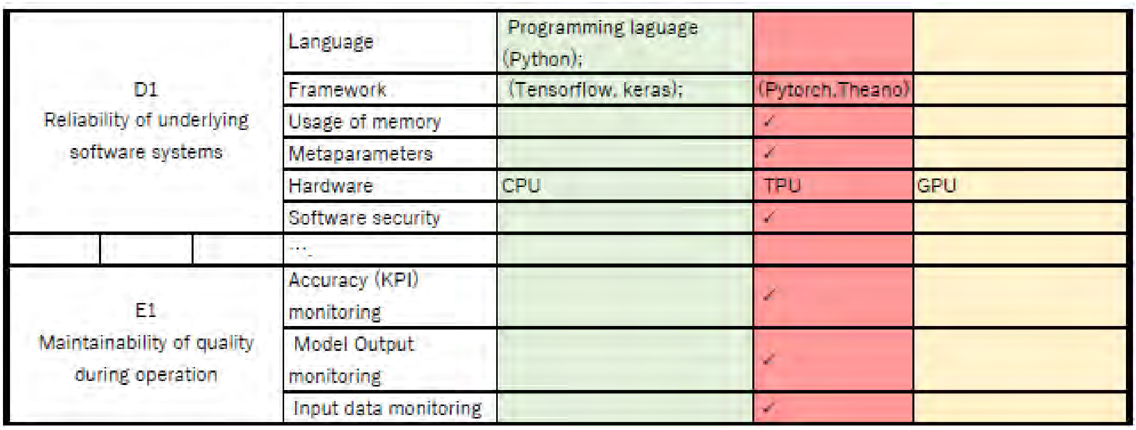
C1 Correctness of trained models			Different accuracy measures and KPIs	(confusion matrix, accuracy, recall, precision, F-measure)		
			Defining model's behavior & finding corner cases		(input prioritization)	
			···.			
C2 Stability of trained models			Robustness due to perturbation in data	(CNN-Cert, Fast-Lin)		
			Robustness due to perturbation in model	(mutation robustness)		
			···.			
D1 Reliability of underlying software systems			Program & open-source libraries	(Python, Tensorflow)		
			Image processing unit	✓		
			Unit for external methods	(correct digit position)		
			Usage of memory		✓	
			Time cost		✓	
			Software security		✓	
			Assessing training and operational environment		✓	
			···.			
E1 Maintainability of quality during operation			KPI monitoring		✓	
			Continuous data collection and labeling		✓	
			Analyzing novelty of model input		✓	
			Analyzing re-training necessity		✓	
			Model output monitoring		✓	
			Creating additional dataset		✓	
			···			

D.5 House price analysis



House Price Analysis				
Functional requirements				
• The AI model will estimate house price based on provided information about house feature				
Nonfunctional requirements				
• The AI model performs well with houses in the State of Iowa; houses outside the region are considered out of scope.				
• Houses of a wide price range; from cheap to expensive are considered.				
• In case a value is missing for a certain feature, the AI model will continue prediction.				
•The AI model should be robust in estimating the price of the houses whose feature values are very uncommon.				
Internal Qualities	Contents	Done	Not Done	Partly Done/Continuing
A1 Sufficiency of problem domain analysis	Defining problem domain	(Features, data points, dimensions):		
	Data for all possible price ranges	Kaggle house price dataset was used; Data has Distribution of 1460 data points		
	Selecting well-defined feature dimensions	Correlation matrix.	(Backward elimination, PCA)	
	Selecting ranges within and out of bounds	(GrLivArea , ExterQual)	77 features	
	Identifying unsound cases			
	---.			
A2 Coverage for distinguished	Data management in each feature dimension	Checking Coverage		
	---.			
B1 Coverage of datasets	Coverage for each combination	Distribution over the feature space		
	Identifying rare or corner cases	Check whether each region is covered by some data or not		
	Feature deletion			
	---.			
B2 Uniformity of datasets	Enough data for each case	Compare between Expected and actual distribution		

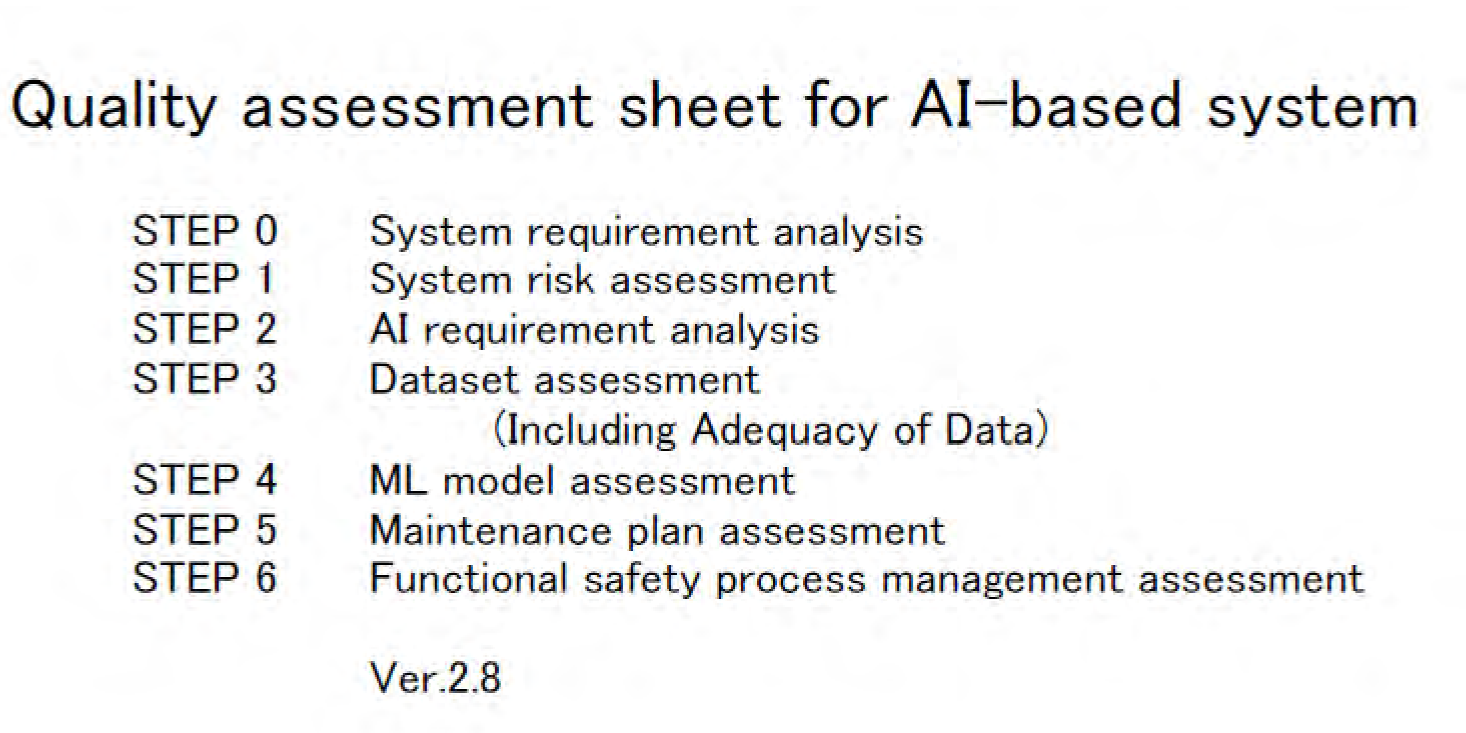
B3 Adequacy of data				
	---.			
C1 Correctness of trained models	Selecting specific KPI	(Mean Absolute Logarithmic Error)	MSE, RMSE	
	Performance comparison for different distribution of data	How different distributions of attribute affect the performance of a model		
	---.			
C2 Stability of trained models	Loss curves for training and testing	(Hyperparameter tuning, model with different architecture)		
	---.			



D1 Reliability of underlying software systems	Language	Programming laguage (Python);		
	Framework	(Tensorflow, keras);	(Pytorch,Theano)	
	Usage of memory		✓	
	Metaparameters		✓	
	Hardware	CPU	TPU	GPU
	Software security		✓	
	...			
E1 Maintainability of quality during operation	Accuracy (KPI) monitoring		✓	
	Model Output monitoring		✓	
	Input data monitoring		✓	

E. Quality assessment sheets

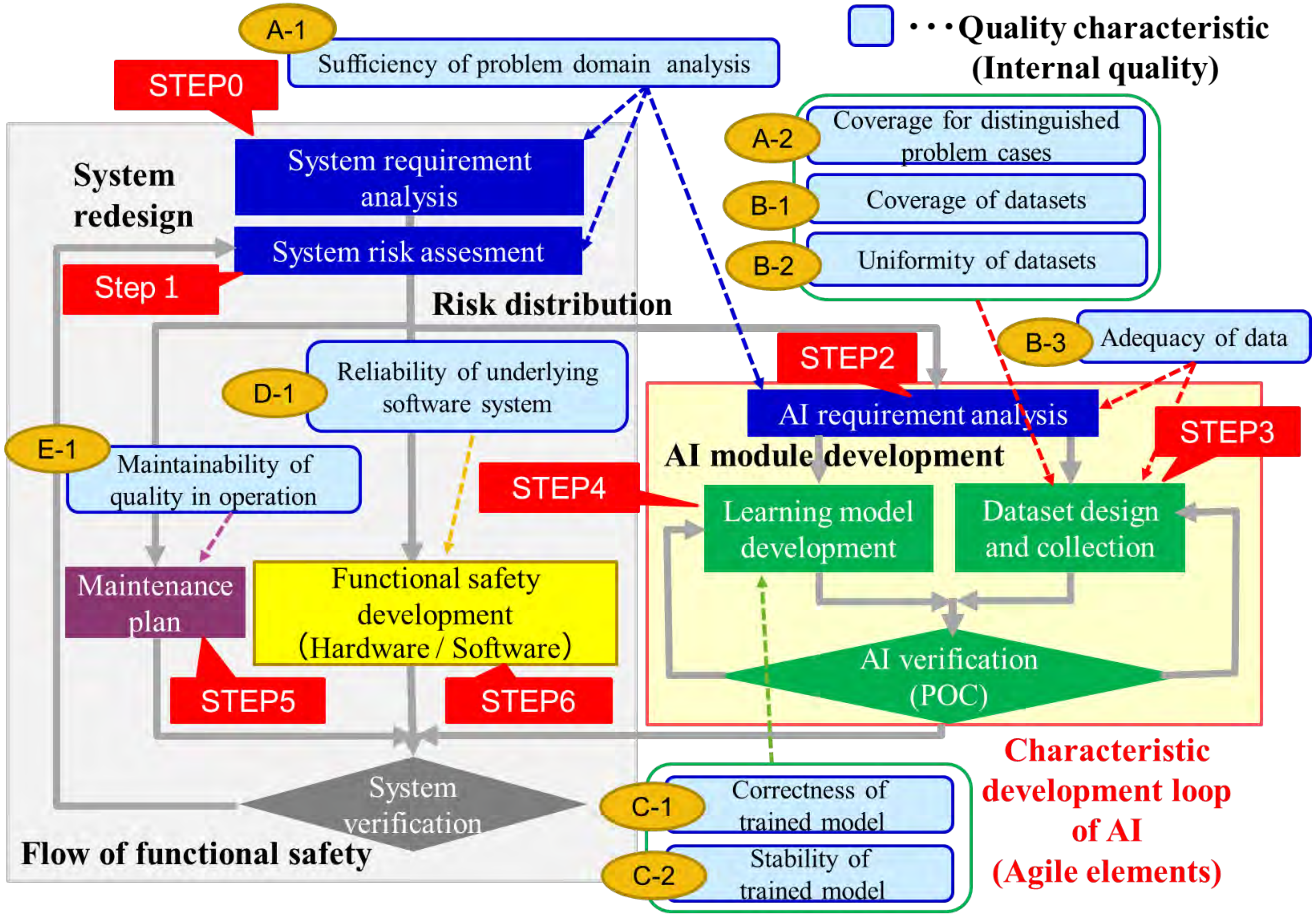
A complete, blank set of the quality assessment sheets presented in Chapter 6 are shown below. It also includes examples of using some of the sheets. For further description of how to use the sheets, see Chapter 6.



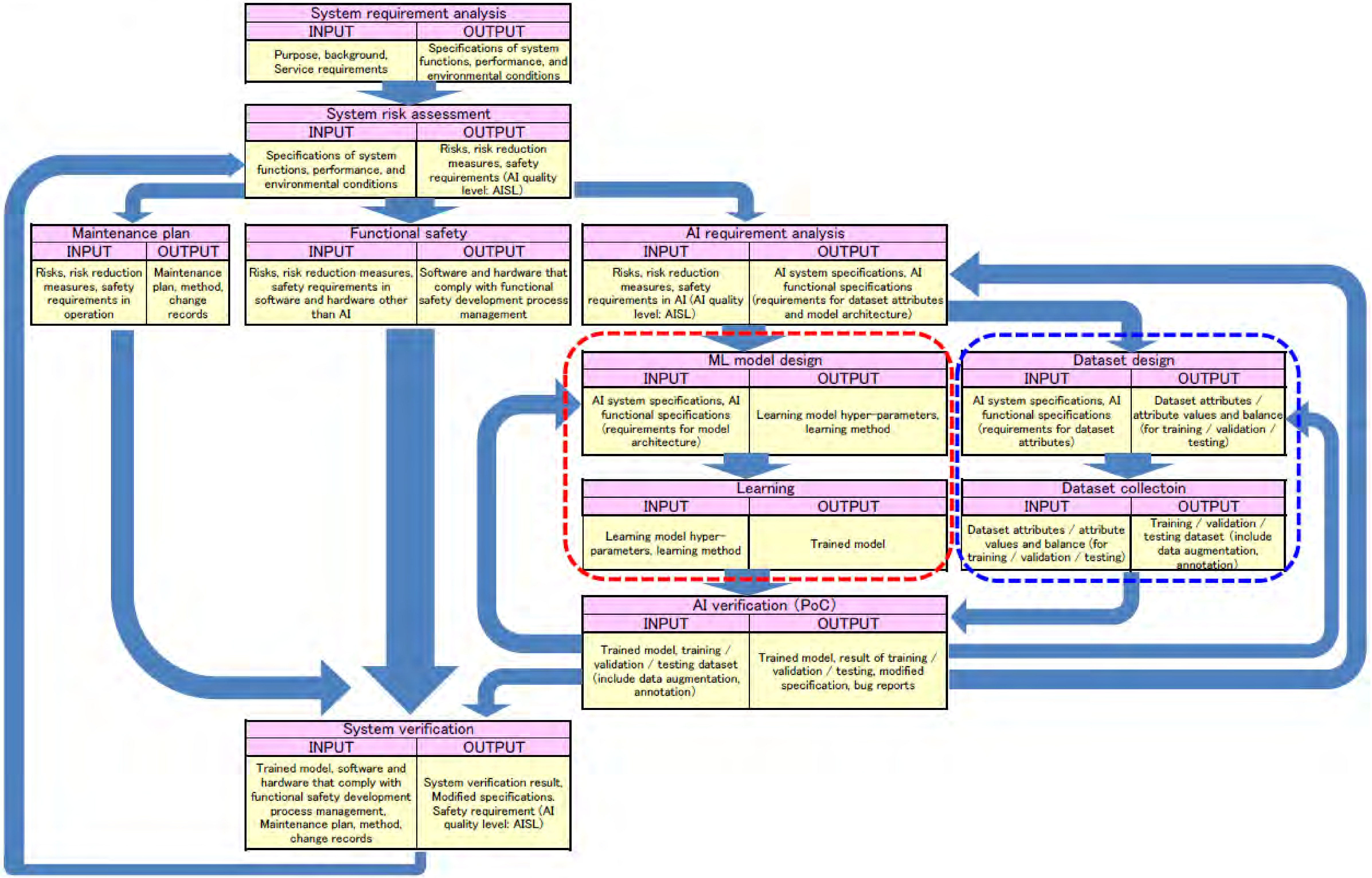
Quality assessment sheet for AI-based system

STEP 0	System requirement analysis
STEP 1	System risk assessment
STEP 2	AI requirement analysis
STEP 3	Dataset assessment (Including Adequacy of Data)
STEP 4	ML model assessment
STEP 5	Maintenance plan assessment
STEP 6	Functional safety process management assessment

Ver.2.8


A-1
Sufficiency of problem domain analysis
・・・Quality characteristic
(Internal quality)
STEP0
System redesign
System requirement analysis
System risk assesment
Step 1
Risk distribution
A-2
Coverage for distinguished problem cases
B-1
Coverage of datasets
B-2
Uniformity of datasets
B-3
Adequacy of data
STEP2
AI requirement analysis
STEP3
D-1
Reliability of underlying software system
E-1
Maintainability of quality in operation
STEP4
AI module development
Learning model development
Dataset design and collection
Maintenance plan
Functional safety development (Hardware / Software)
STEP5
STEP6
AI verification (POC)
System verification
Flow of functional safety
C-1
Correctness of trained model
C-2
Stability of trained model
Characteristic development loop of AI
(Agile elements)

Input / output in the development flow



System requirement analysis

INPUT	OUTPUT
Purpose, background, Service requirements	Specifications of system functions, performance, and environmental conditions

System risk assessment

INPUT	OUTPUT
Specifications of system functions, performance, and environmental conditions	Risks, risk reduction measures, safety requirements (AI quality level: AISL)

Maintenance plan

INPUT	OUTPUT
Risks, risk reduction measures, safety requirements in operation	Maintenance plan, method, change records

Functional safety

INPUT	OUTPUT
Risks, risk reduction measures, safety requirements in software and hardware other than AI	Software and hardware that comply with functional safety development process management

AI requirement analysis

INPUT	OUTPUT
Risks, risk reduction measures, safety requirements in AI (AI quality level: AISL)	AI system specifications, AI functional specifications (requirements for dataset attributes and model architecture)

ML model design

INPUT	OUTPUT
AI system specifications, AI functional specifications (requirements for model architecture)	Learning model hyper-parameters, learning method

Learning

INPUT	OUTPUT
Learning model hyper-parameters, learning method	Trained model

Dataset design

INPUT	OUTPUT
AI system specifications, AI functional specifications (requirements for dataset attributes)	Dataset attributes / attribute values and balance (for training / validation / testing)

Dataset collectoin

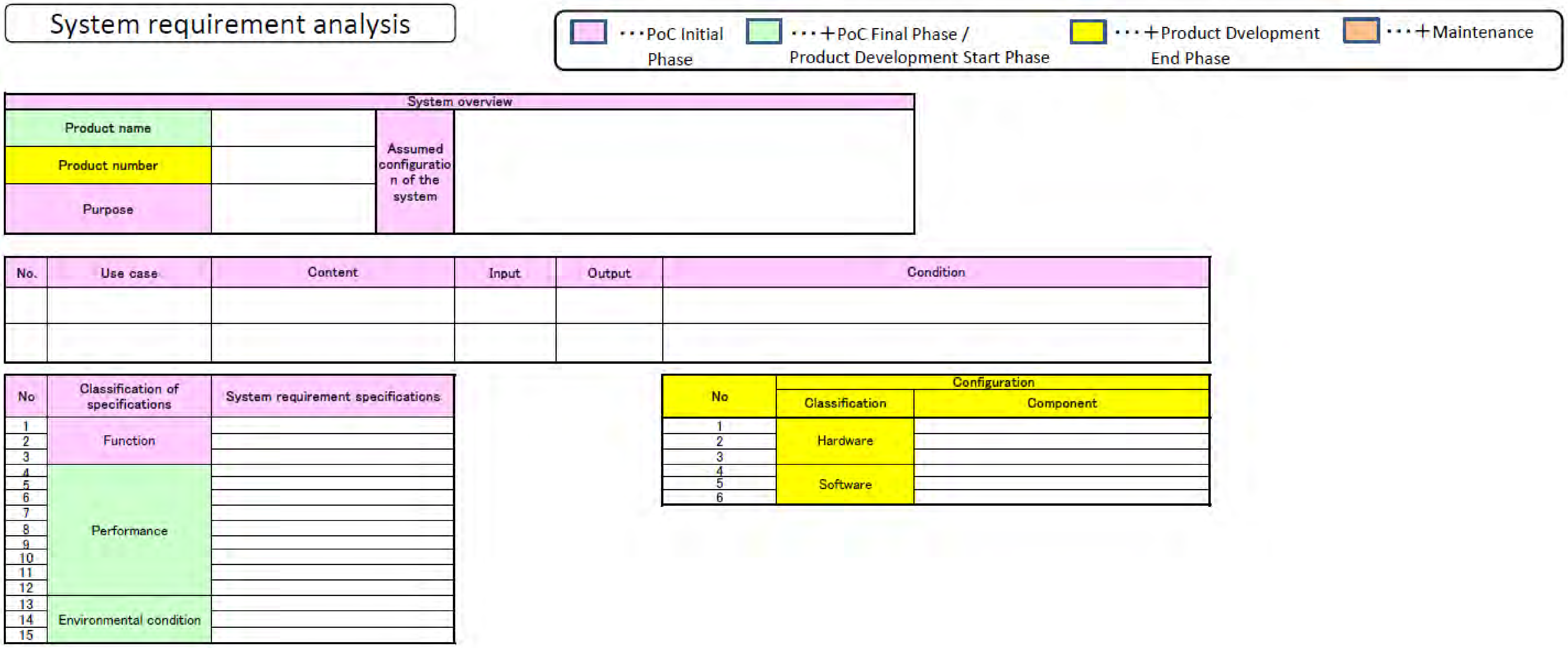
INPUT	OUTPUT
Dataset attributes / attribute values and balance (for training / validation / testing)	Training / validation / testing dataset (include data augmentation, annotation)

AI verification (PoC)

INPUT	OUTPUT
Trained model, training / validation / testing dataset (include data augmentation, annotation)	Trained model, result of training / validation / testing, modified specification, bug reports

System verification

INPUT	OUTPUT
Trained model, software and hardware that comply with functional safety development process management, Maintenance plan, method, change records	System verification result, Modified specifications, Safety requirement (AI quality level: AISL)



System requirement analysis

🟪 ･･･PoC Initial Phase
🟩 ･･･＋PoC Final Phase / Product Development Start Phase
🟨 ･･･＋Product Dvelopment End Phase
🟧 ･･･＋Maintenance

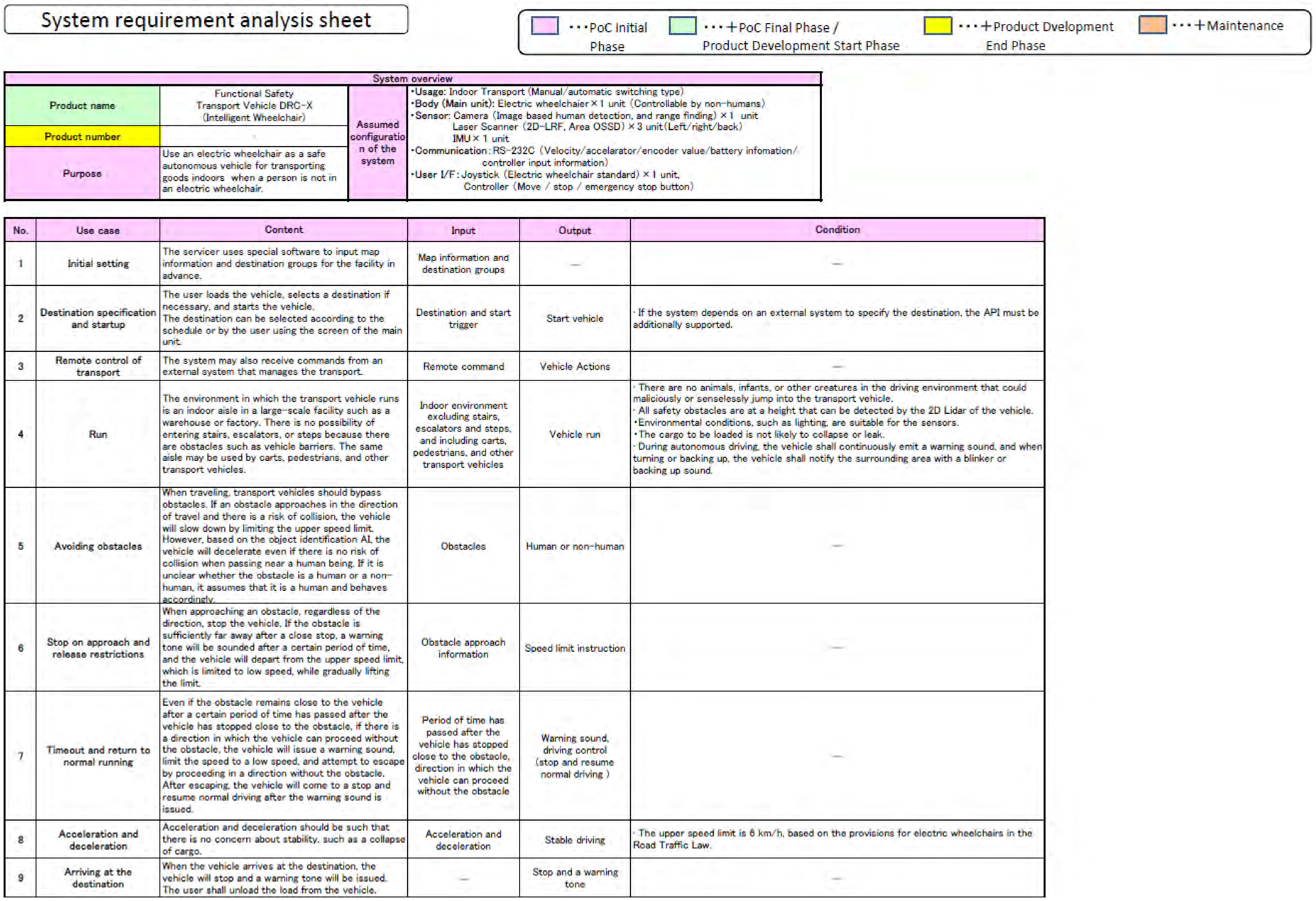
System overview			
Product name		Assumed configuration of the system	
Product number			
Purpose			

No.	Use case	Content	Input	Output	Condition

No	Classification of specifications	System requirement specifications
1	Function	
2		
3		
4	Performance	
5		
6		
7		
8		
9		
10		
11		
12		
13	Environmental condition	
14		
15		

No	Configuration	
	Classification	Component
1	Hardware	
2		
3		
4	Software	
5		
6		

An example of using the system requirement analysis sheet



System requirement analysis sheet

🟪 ･･･PoC Initial Phase 🟩 ･･･+PoC Final Phase / Product Development Start Phase 🟨 ･･･+Product Dvelopment End Phase 🟧 ･･･+Maintenance

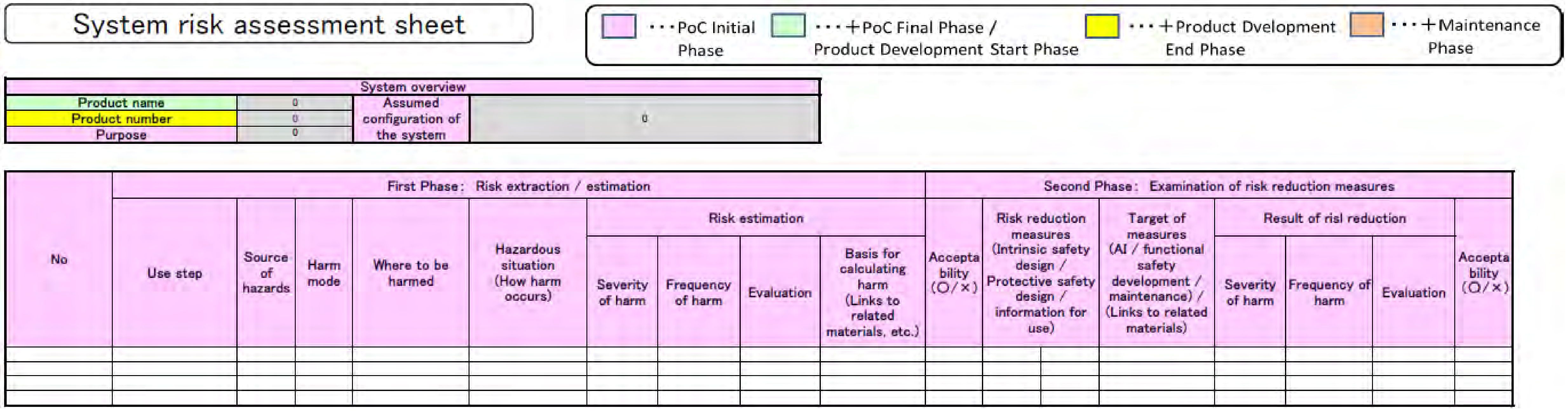
System overview			
Product name	Functional Safety Transport Vehicle DRC-X (Intelligent Wheelchair)	Assumed configuratio n of the system	･Usage: Indoor Transport (Manual/automatic switching type) ･Body (Main unit): Electric wheelchair × 1 unit (Controllable by non-humans) ･Sensor: Camera (Image based human detection, and range finding) × 1 unit Laser Scanner (2D-LRF, Area OSSD) × 3 unit(Left/right/back) IMU × 1 unit ･Communication：RS-232C (Velocity/accelarator/encoder value/battery infomation/ controller input information) ･User I/F：Joystick (Electric wheelchair standard) × 1 unit, Controller (Move / stop / emergency stop button)
Product number			
Purpose	Use an electric wheelchair as a safe autonomous vehicle for transporting goods indoors when a person is not in an electric wheelchair.		

No.	Use case	Content	Input	Output	Condition
1	Initial setting	The servicer uses special software to input map information and destination groups for the facility in advance.	Map information and destination groups	—	—
2	Destination specification and startup	The user loads the vehicle, selects a destination if necessary, and starts the vehicle. The destination can be selected according to the schedule or by the user using the screen of the main unit.	Destination and start trigger	Start vehicle	· If the system depends on an external system to specify the destination, the API must be additionally supported.
3	Remote control of transport	The system may also receive commands from an external system that manages the transport.	Remote command	Vehicle Actions	—
4	Run	The environment in which the transport vehicle runs is an indoor aisle in a large-scale facility such as a warehouse or factory. There is no possibility of entering stairs, escalators, or steps because there are obstacles such as vehicle barriers. The same aisle may be used by carts, pedestrians, and other transport vehicles.	Indoor environment excluding stairs, escalators and steps, and including carts, pedestrians, and other transport vehicles	Vehicle run	· There are no animals, infants, or other creatures in the driving environment that could maliciously or senselessly jump into the transport vehicle. · All safety obstacles are at a height that can be detected by the 2D Lidar of the vehicle. ･Environmental conditions, such as lighting, are suitable for the sensors. ･The cargo to be loaded is not likely to collapse or leak. · During autonomous driving, the vehicle shall continuously emit a warning sound, and when turning or backing up, the vehicle shall notify the surrounding area with a blinker or backing up sound.
5	Avoiding obstacles	When traveling, transport vehicles should bypass obstacles. If an obstacle approaches in the direction of travel and there is a risk of collision, the vehicle will slow down by limiting the upper speed limit. However, based on the object identification AI, the vehicle will decelerate even if there is no risk of collision when passing near a human being. If it is unclear whether the obstacle is a human or a non-human, it assumes that it is a human and behaves accordingly.	Obstacles	Human or non-human	—
6	Stop on approach and release restrictions	When approaching an obstacle, regardless of the direction, stop the vehicle. If the obstacle is sufficiently far away after a close stop, a warning tone will be sounded after a certain period of time, and the vehicle will depart from the upper speed limit, which is limited to low speed, while gradually lifting the limit.	Obstacle approach information	Speed limit instruction	—
7	Timeout and return to normal running	Even if the obstacle remains close to the vehicle after a certain period of time has passed after the vehicle has stopped close to the obstacle, if there is a direction in which the vehicle can proceed without the obstacle, the vehicle will issue a warning sound, limit the speed to a low speed, and attempt to escape by proceeding in a direction without the obstacle. After escaping, the vehicle will come to a stop and resume normal driving after the warning sound is issued.	Period of time has passed after the vehicle has stopped close to the obstacle, direction in which the vehicle can proceed without the obstacle	Warning sound, driving control (stop and resume normal driving)	—
8	Acceleration and deceleration	Acceleration and deceleration should be such that there is no concern about stability, such as a collapse of cargo.	Acceleration and deceleration	Stable driving	· The upper speed limit is 6 km/h, based on the provisions for electric wheelchairs in the Road Traffic Law.
9	Arriving at the destination	When the vehicle arrives at the destination, the vehicle will stop and a warning tone will be issued. The user shall unload the load from the vehicle.	—	Stop and a warning tone	—

10	Emergency stop	An emergency stop switch is located on the outside of the vehicle, and pressing the switch brings the vehicle to a definite stop. Even if the emergency stop switch is returned to its original position, the vehicle will remain stationary until the servicer releases it.	Pressing emergency stop switch	Stop	—
11	Moving to the standby position	The vehicle automatically moves to the standby position according to the user's instructions, the servicer's remote commands, or the schedule.	User's instructions, the servicer's remote commands, or the schedule	Move to the standby position	—
12	Periodic inspections.	The servicer shall conduct periodic inspections.	—	—	—
13	Parts replacement and repair	The developer shall supply replacement consumable parts and perform repair services over a period of time.	—	—	—
14	Detecting an abnormality and warning	If the vehicle detects an abnormality, such as a complete loss of position or a malfunction, the vehicle will stop, and a warning will be sent to the servicer.	Detect an abnormality (eg. complete loss of position or a malfunction)	Stop, and a warning are sent to the servicer.	･The system must have a means of communication to call the servicer in case of abnormalities.
15	Servicer visit or remote maintenance	If the vehicle is stopped for a long period of time, or if there is any other abnormality, the servicer will visit the site or take action remotely.	Stopped for a long period of time, or other abnormality	—	—
16	Remote driving of vehicles	The servicer can remotely obtain images and location information from the vehicle's onboard camera, and can directly instruct the vehicle to drive.	Instruct the vehicle to drive	Camera images and location information	—
17	Driving in presence	Driving instructions that ignore proximity judgments of obstacles must be given in presence, not remotely.	Driving instructions that ignore proximity judgments of obstacles	Driving while ignoring the proximity judgment of obstacles	—
18	End of life cycle	When disposing of the device, remove the battery and entrust disposal to an industrial waste disposal company.	—	—	—

No	Classification of specifications	System requirement specifications
1	Function	· The vehicle's 2D Lidar and driving control system shall be constructed as a safety-related system and shall detect a single fault.
2		· The vehicle shall be equipped with a two-dimensional Lidar for obstacle detection, estimate its own position, and autonomously determine the direction of travel.
3		· The vehicle shall set an upper speed limit based on the distance to the obstacle, and if there is a threat of collision, the vehicle shall decelerate in stages and stop before the collision.
4		· The vehicle is equipped with a camera capable of acquiring color images and seismic intensity information for object detection AI. The detection results are in three levels: "human," "non-human" and "unknown".
5		· When passing near an obstacle, if the vehicle is not convinced that it is "non-human," it will slow down and pass even if there is no threat of collision.
6	Performance	—
7	Environmental condition	· Servicer shall take prompt action in case of abnormalities.
8		· Users should receive training and be familiar with the operation of the equipment.
9		· The user should have accident insurance in case of emergency.
10		· There shall be no unreasonable invasion of privacy by information collected by sensors and cameras.
11		· It is not assumed that people will be riding in the transport vehicle.
12		· It is not assumed that the transport vehicle will be so crowded that it will be difficult to drive.

No	Configuration	
	Classification	Component
1	Hardware	Electric wheelchair (DRC-X)
2		LRF
3		Camera
4	Software	YOLOv3
5		ROS
6		ubuntu 18.04 LTS



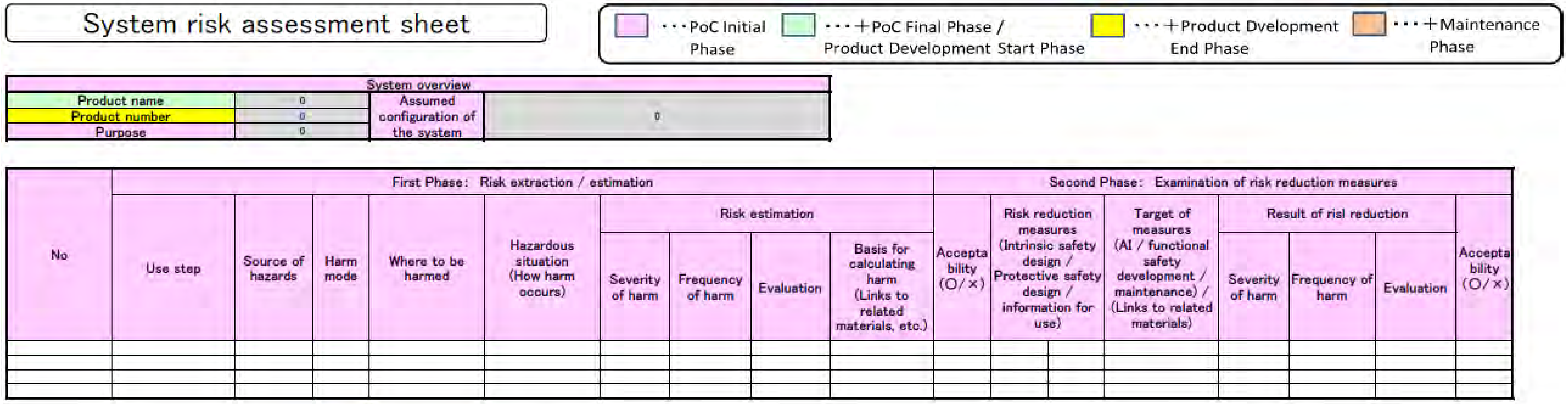
System risk assessment sheet

🟪 ・・・PoC Initial Phase 🟩 ・・・＋PoC Final Phase / Product Development Start Phase 🟨 ・・・＋Product Dvelopment End Phase 🟧 ・・・＋Maintenance Phase

System overview			
Product name	0	Assumed configuration of the system	0
Product number	0		
Purpose	0		

No	First Phase： Risk extraction / estimation									Second Phase： Examination of risk reduction measures						
						Risk estimation							Result of risl reduction			
	Use step	Source of hazards	Harm mode	Where to be harmed	Hazardous situation (How harm occurs)	Severity of harm	Frequency of harm	Evaluation	Basis for calculating harm (Links to related materials, etc.)	Acceptability (○/×)	Risk reduction measures (Intrinsic safety design / Protective safety design / information for use)	Target of measures (AI / functional safety development / maintenance) / (Links to related materials)	Severity of harm	Frequency of harm	Evaluation	Acceptability (○/×)

An example of a risk map for the system risk assessment sheet

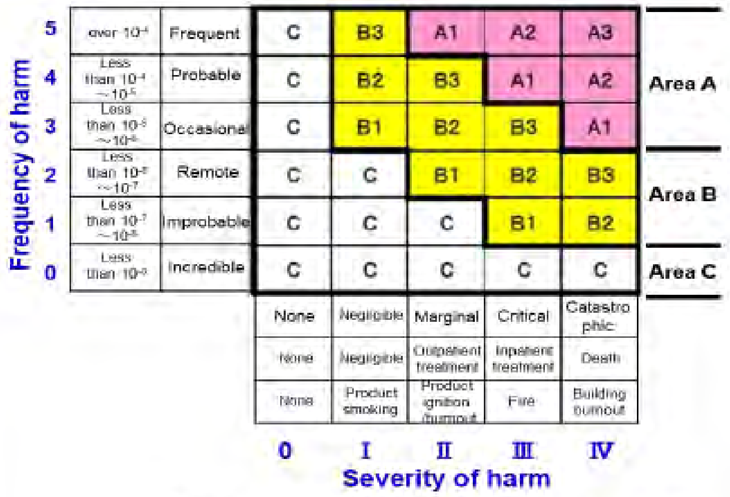


System risk assessment sheet

🟪 ･･･PoC Initial Phase 🟩 ･･･＋PoC Final Phase / Product Development Start Phase 🟨 ･･･＋Product Dvelopment End Phase 🟧 ･･･＋Maintenance Phase

System overview			
Product name	0	Assumed configuration of the system	0
Product number	0		
Purpose	0		

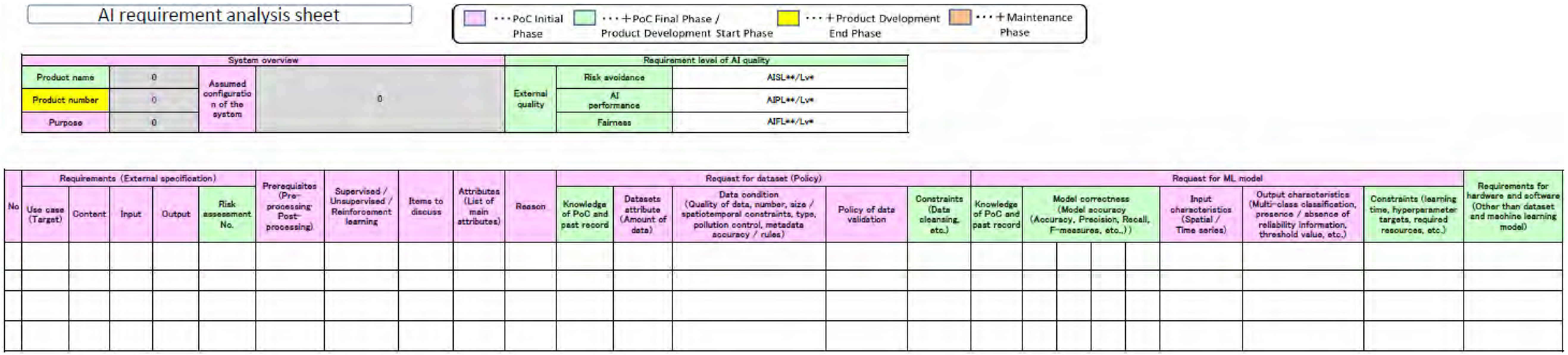
No	First Phase： Risk extraction / estimation									Second Phase： Examination of risk reduction measures						
	Use step	Source of hazards	Harm mode	Where to be harmed	Hazardous situation (How harm occurs)	Risk estimation				Acceptability (O/×)	Risk reduction measures (Intrinsic safety design / Protective safety design / information for use)	Target of measures (AI / functional safety development / maintenance) / (Links to related materials)	Result of risl reduction			Acceptability (O/×)
						Severity of harm	Frequency of harm	Evaluation	Basis for calculating harm (Links to related materials, etc.)				Severity of harm	Frequency of harm	Evaluation	



Frequency of harm									
5	over 10^{-4}	Frequent	C	B3	A1	A2	A3	Area A
4	Less than 10^{-4} ～10^{-5}	Probable	C	B2	B3	A1	A2	
3	Less than 10^{-5} ～10^{-6}	Occasional	C	B1	B2	B3	A1	
2	Less than 10^{-6} ～10^{-7}	Remote	C	C	B1	B2	B3	Area B
1	Less than 10^{-7} ～10^{-8}	Improbable	C	C	C	B1	B2	
0	Less than 10^{-8}	Incredible	C	C	C	C	C	Area C
			None	Negligible	Marginal	Critical	Catastrophic	
			None	Negligible	Outpatient treatment	Inpatient treatment	Death	
			None	Product smoking	Product ignition /burnout	Fire	Building burnout	
		Severity of harm	0	I	II	III	IV	

* This risk map is an example of a risk definition and is not limited to this format.

* "Applying the R-Map Method to Product Safety and Risk Management, Japan", which is referred to as one of the risk management methods from ISO13077:2013(en).

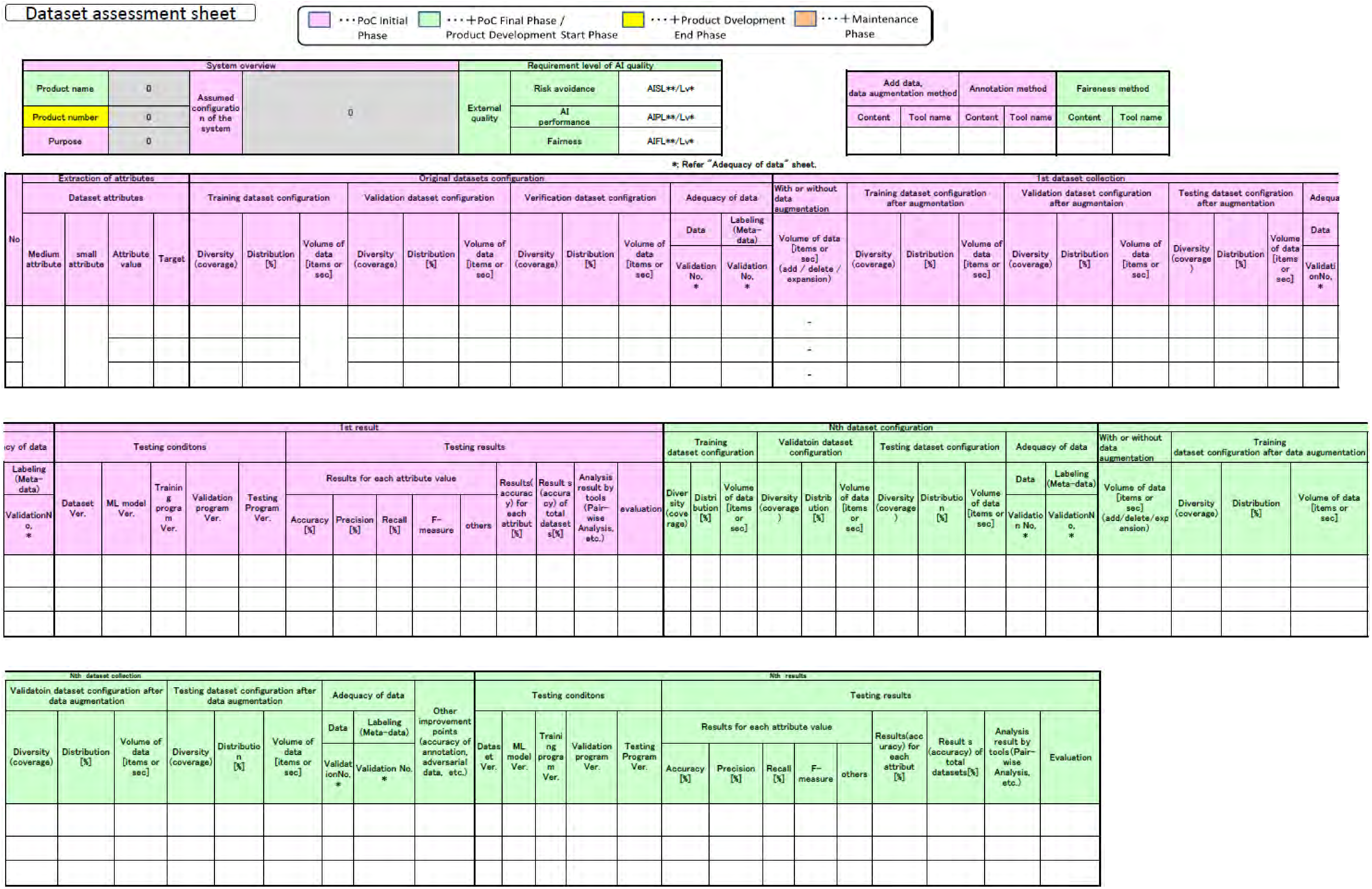


AI requirement analysis sheet

🟪 ・・・PoC Initial Phase 🟩 ・・・＋PoC Final Phase / Product Development Start Phase 🟨 ・・・＋Product Dvelopment End Phase 🟧 ・・・＋Maintenance Phase

System overview				Requirement level of AI quality		
Product name	0	Assumed configuration of the system	0	External quality	Risk avoidance	AISL**/Lv*
Product number	0				AI performance	AIPL**/Lv*
Purpose	0				Fairness	AIFL**/Lv*

No	Requirements (External specification)					Prerequisites (Pre-processing·Post-processing)	Supervised / Unsupervised / Reinforcement learning	Items to discuss	Attributes (List of main attributes)	Reason	Request for dataset (Policy)					Request for ML model									Requirements for hardware and software (Other than dataset and machine learning model)
	Use case (Target)	Content	Input	Output	Risk assessment No.						Knowledge of PoC and past record	Datasets attribute (Amount of data)	Data condition (Quality of data, number, size / spatiotemporal constraints, type, pollution control, metadata accuracy / rules)	Policy of data validation	Constraints (Data cleansing, etc.)	Knowledge of PoC and past record	Model correctness (Model accuracy (Accuracy, Precision, Recall, F-measures, etc.))				Input characteristics (Spatial / Time series)	Output characteristics (Multi-class classification, presence / absence of reliability information, threshold value, etc.)	Constraints (learning time, hyperparameter targets, required resources, etc.)	



Dataset assessment sheet

🟪 ･･･PoC Initial Phase 🟩 ･･･＋PoC Final Phase / Product Development Start Phase 🟨 ･･･＋Product Dvelopment End Phase 🟧 ･･･＋Maintenance Phase

System overview				Requirement level of AI quality		
Product name	0	Assumed configuration of the system	0	External quality	Risk avoidance	AISL**/Lv*
Product number	0				AI performance	AIPL**/Lv*
Purpose	0				Fairness	AIFL**/Lv*

Add data, data augmentation method		Annotation method		Faireness method	
Content	Tool name	Content	Tool name	Content	Tool name

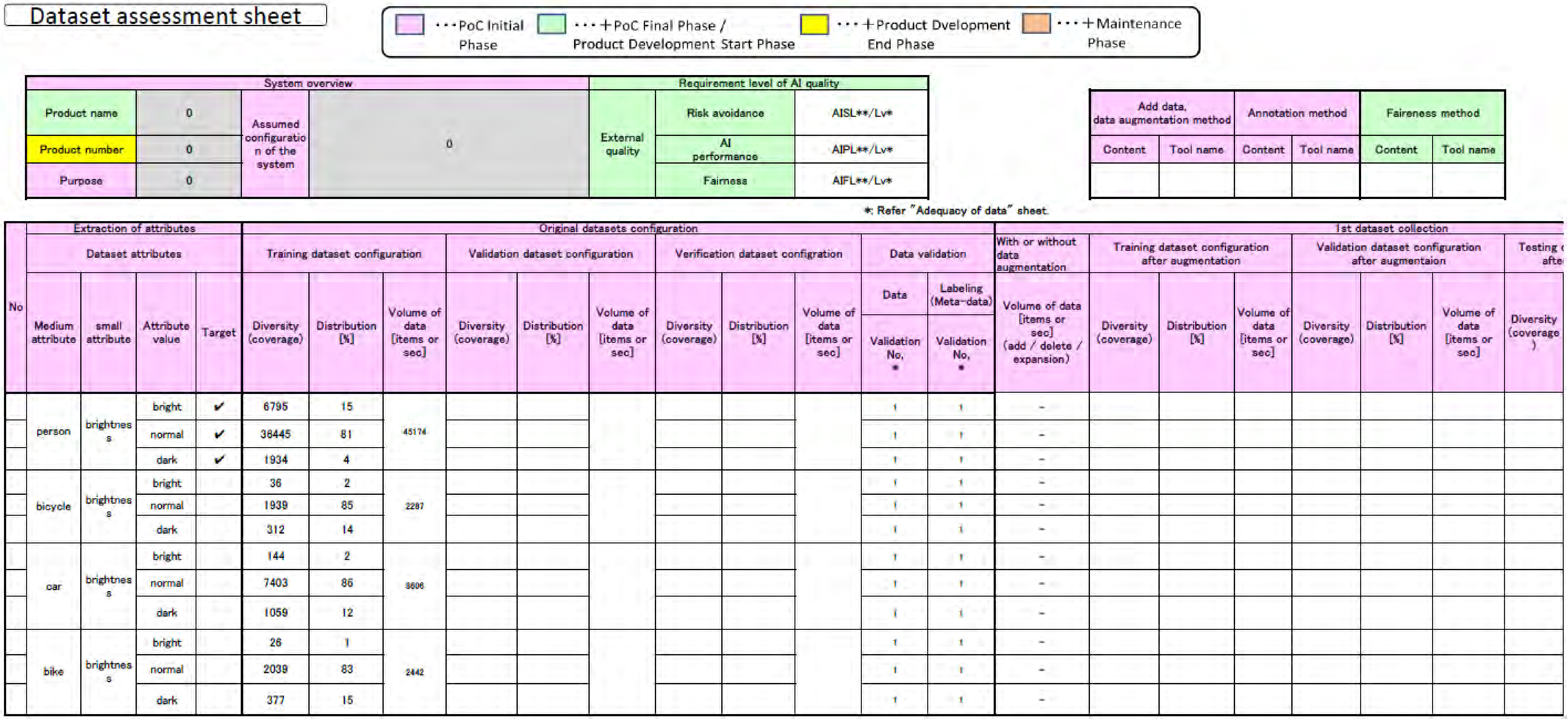
*: Refer ″Adequacy of data″ sheet.

No	Extraction of attributes				Original datasets configuration											1st dataset collection										
	Dataset attributes				Training dataset configuration			Validation dataset configuration			Verification dataset configration			Adequacy of data		With or without data augmentation	Training dataset configuration after augmentation			Validation dataset configuration after augmentaion			Testing dataset configration after augmentation			Adequa
	Medium attribute	small attribute	Attribute value	Target	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Data	Labeling (Meta-data)	Volume of data [items or sec] (add / delete / expansion)	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Data
														Validation No. *	Validation No. *											Validati onNo. *
																-										
																-										
																-										

	1st result														Nth dataset configuration														
icy of data	Testing conditons					Testing results									Training dataset configuration			Validatoin dataset configuration			Testing dataset configuration			Adequacy of data		With or without data augmentation	Training dataset configuration after data augumentation		
Labeling (Meta-data)	Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Results for each attribute value					Results(accuracy) for each attribut [%]	Result s (accuracy) of total datasets[%]	Analysis result by tools (Pair-wise Analysis, etc.)	evaluation	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Data	Labeling (Meta-data)	Volume of data [items or sec] (add/delete/expansion)	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]
ValidationNo. *						Accuracy [%]	Precision [%]	Recall [%]	F-measure	others														Validation No. *	ValidationNo. *				

Nth dataset collection									Nth results													
Validatoin dataset configuration after data augmentation			Testing dataset configuration after data augmentation			Adequacy of data		Other improvement points (accuracy of annotation, adversarial data, etc.)	Testing conditons					Testing results								
Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Data	Labeling (Meta-data)		Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Results for each attribute value					Results(accuracy) for each attribut [%]	Result s (accuracy) of total datasets[%]	Analysis result by tools(Pair-wise Analysis, etc.)	Evaluation
						ValidationNo. *	Validation No. *							Accuracy [%]	Precision [%]	Recall [%]	F-measure	others				

An example of using the dataset assessment sheet



Dataset assessment sheet

🟪 ･･･PoC Initial Phase 🟩 ･･･＋PoC Final Phase / Product Development Start Phase 🟨 ･･･＋Product Dvelopment End Phase 🟧 ･･･＋Maintenance Phase

System overview				Requirement level of AI quality		
Product name	0	Assumed configuration of the system	0	External quality	Risk avoidance	AISL**/Lv*
Product number	0				AI performance	AIPL**/Lv*
Purpose	0				Fairness	AIFL**/Lv*

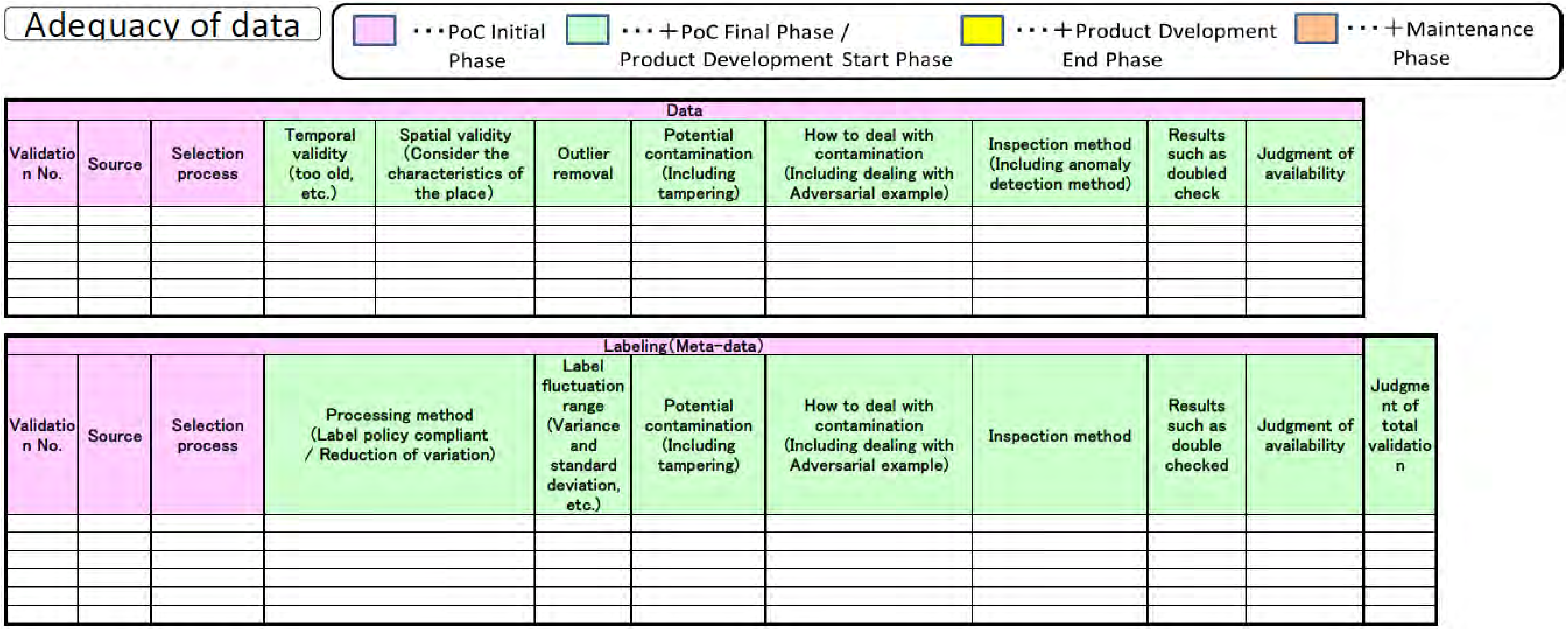
Add data, data augmentation method		Annotation method		Fairness method	
Content	Tool name	Content	Tool name	Content	Tool name

*: Refer "Adequacy of data" sheet.

No	Extraction of attributes				Original datasets configuration											1st dataset collection							
	Dataset attributes				Training dataset configuration			Validation dataset configuration			Verification dataset configration			Data validation		With or without data augmentation	Training dataset configuration after augmentation			Validation dataset configuration after augmentaion			Testing afte
														Data	Labeling (Meta-data)								
	Medium attribute	small attribute	Attribute value	Target	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Validation No, *	Validation No, *	Volume of data [items or sec] (add / delete / expansion)	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)
	person	brightness	bright	✔	6795	15	45174							1	1	-							
			normal	✔	36445	81								1	1	-							
			dark	✔	1934	4								1	1	-							
	bicycle	brightness	bright		36	2	2287							1	1	-							
			normal		1939	85								1	1	-							
			dark		312	14								1	1	-							
	car	brightness	bright		144	2	8606							1	1	-							
			normal		7403	86								1	1	-							
			dark		1059	12								1	1	-							
	bike	brightness	bright		26	1	2442							1	1	-							
			normal		2039	83								1	1	-							
			dark		377	15								1	1	-							

				1st result														Nth dataset configuration										
dataset configration r augmentation		Data validation		Testing conditons					Testing results									Training dataset configuration			Validatoin dataset configuration			Testing dataset configuration			Data validation	
Distribution [%]	Volume of data [items or sec]	Data	Labeling (Meta-data)	Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Results for each attribute value					Results(accuracy) for each attribut [%]	Result s (accuracy) of total datasets[%]	Analysis result by tools (Pair-wise Analysis, etc.)	evaluation	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Data	Labeling (Meta-data)
		ValidationNo, *	ValidationNo, *						Accuracy [%]	Precision [%]	Recall [%]	F-measure	others														Validation No, *	ValidationNo, *

Nth dataset collection													Nth results													
With or without data augmentation	Training dataset configuration after data augumentation			Validatoin dataset configuration after data augmentation			Testing dataset configuration after data augmentation			Data validation		Other improvement points (accuracy of annotation, adversarial data, etc.)	Testing conditons					Testing results								
Volume of data [items or sec] (add/delete/expansion)	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Diversity (coverage)	Distribution [%]	Volume of data [items or sec]	Data	Labeling (Meta-data)		Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Results for each attribute value					Results(accuracy) for each attribut [%]	Result s (accuracy) of total datasets[%]	Analysis result by tools (Pair-wise Analysis, etc.)	valuatio
										ValidationNo, *	Validation No. *							Accuracy [%]	Precision [%]	Recall [%]	F-measure	others				

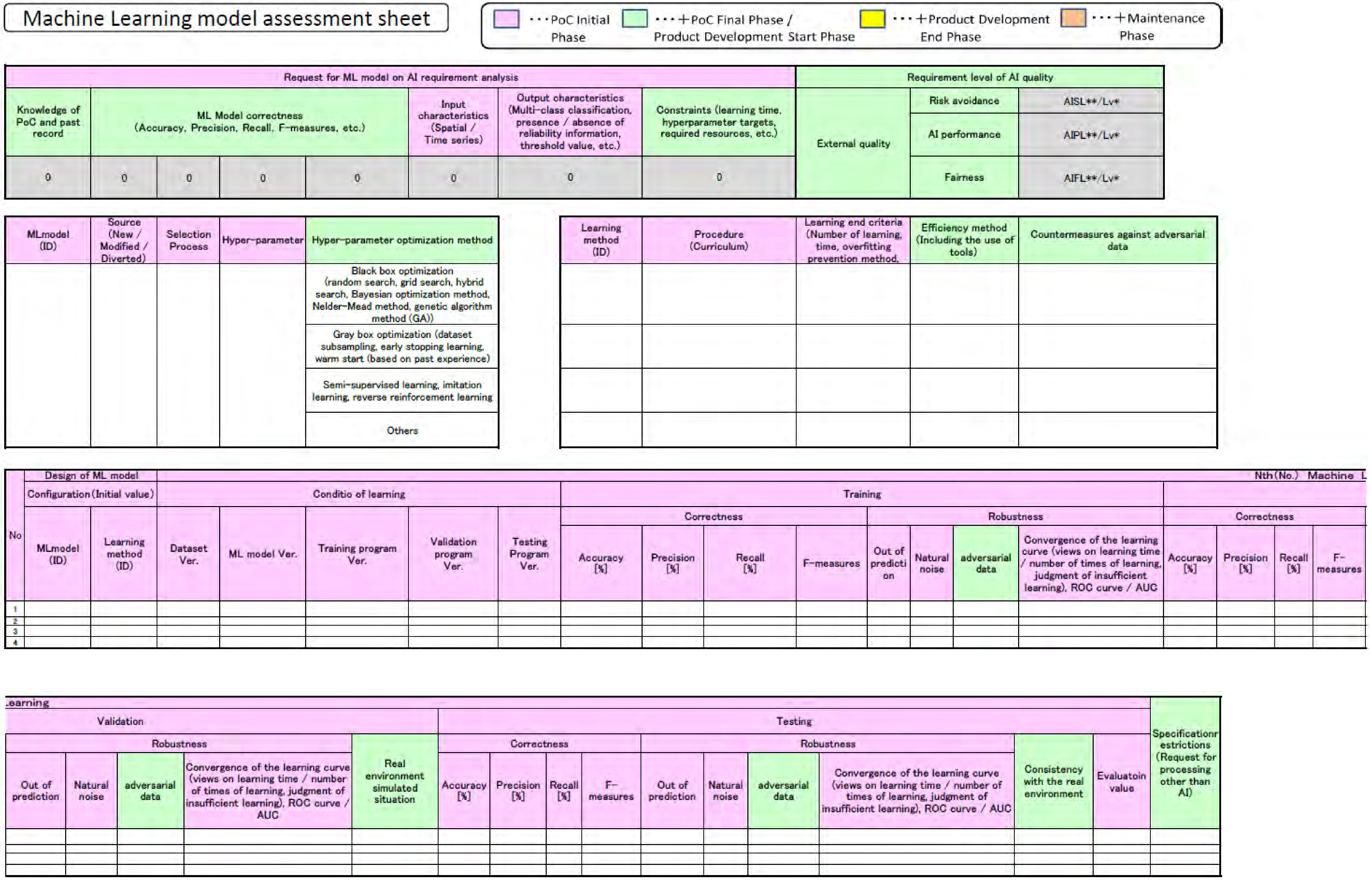


Adequacy of data

🟪 ･･･PoC Initial Phase 🟩 ･･･＋PoC Final Phase / Product Development Start Phase 🟨 ･･･＋Product Dvelopment End Phase 🟧 ･･･＋Maintenance Phase

Data										
Validation No.	Source	Selection process	Temporal validity (too old, etc.)	Spatial validity (Consider the characteristics of the place)	Outlier removal	Potential contamination (Including tampering)	How to deal with contamination (Including dealing with Adversarial example)	Inspection method (Including anomaly detection method)	Results such as doubled check	Judgment of availability

Labeling(Meta-data)											
Validation No.	Source	Selection process	Processing method (Label policy compliant / Reduction of variation)	Label fluctuation range (Variance and standard deviation, etc.)	Potential contamination (Including tampering)	How to deal with contamination (Including dealing with Adversarial example)	Inspection method	Results such as double checked	Judgment of availability	Judgment of total validation



Machine Learning model assessment sheet

🟪 ・・・PoC Initial Phase
🟩 ・・・＋PoC Final Phase / Product Development Start Phase
🟨 ・・・＋Product Dvelopment End Phase
🟧 ・・・＋Maintenance Phase

Request for ML model on AI requirement analysis								Requirement level of AI quality		
Knowledge of PoC and past record	ML Model correctness (Accuracy, Precision, Recall, F-measures, etc.)				Input characteristics (Spatial / Time series)	Output characteristics (Multi-class classification, presence / absence of reliability information, threshold value, etc.)	Constraints (learning time, hyperparameter targets, required resources, etc.)	External quality	Risk avoidance	AISL**/Lv*
									AI performance	AIPL**/Lv*
0	0	0	0	0	0	0	0		Fairness	AIFL**/Lv*

MLmodel (ID)	Source (New / Modified / Diverted)	Selection Process	Hyper-parameter	Hyper-parameter optimization method
				Black box optimization (random search, grid search, hybrid search, Bayesian optimization method, Nelder-Mead method, genetic algorithm method (GA))
				Gray box optimization (dataset subsampling, early stopping learning, warm start (based on past experience)
				Semi-supervised learning, imitation learning, reverse reinforcement learning
				Others

Learning method (ID)	Procedure (Curriculum)	Learning end criteria (Number of learning, time, overfitting prevention method,	Efficiency method (Including the use of tools)	Countermeasures against adversarial data

No	Design of ML model															Nth(No.) Machine Learning																			Specificationrestrictions (Request for processing other than AI)
	Configuration(Initial value)		Conditio of learning					Training								Validation									Testing										
								Correctness				Robustness				Correctness				Robustness				Real environment simulated situation	Correctness				Robustness				Consistency with the real environment	Evaluatoin value	
	MLmodel (ID)	Learning method (ID)	Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Accuracy [%]	Precision [%]	Recall [%]	F-measures	Out of prediction	Natural noise	adversarial data	Convergence of the learning curve (views on learning time / number of times of learning, judgment of insufficient learning), ROC curve / AUC	Accuracy [%]	Precision [%]	Recall [%]	F-measures	Out of prediction	Natural noise	adversarial data	Convergence of the learning curve (views on learning time / number of times of learning, judgment of insufficient learning), ROC curve / AUC		Accuracy [%]	Precision [%]	Recall [%]	F-measures	Out of prediction	Natural noise	adversarial data	Convergence of the learning curve (views on learning time / number of times of learning, judgment of insufficient learning), ROC curve / AUC			
1																																			
2																																			
3																																			
4																																			

Maintenance plan assessment

🟪・・・PoC Initial Phase 🟩・・・＋PoC Final Phase / Product Development Start Phase 🟨・・・＋Product Dvelopment End Phase 🟧・・・＋Maintenance Phase

・Consists of "Maintenance plan" and "Maintenance results".

■Maintenance plan

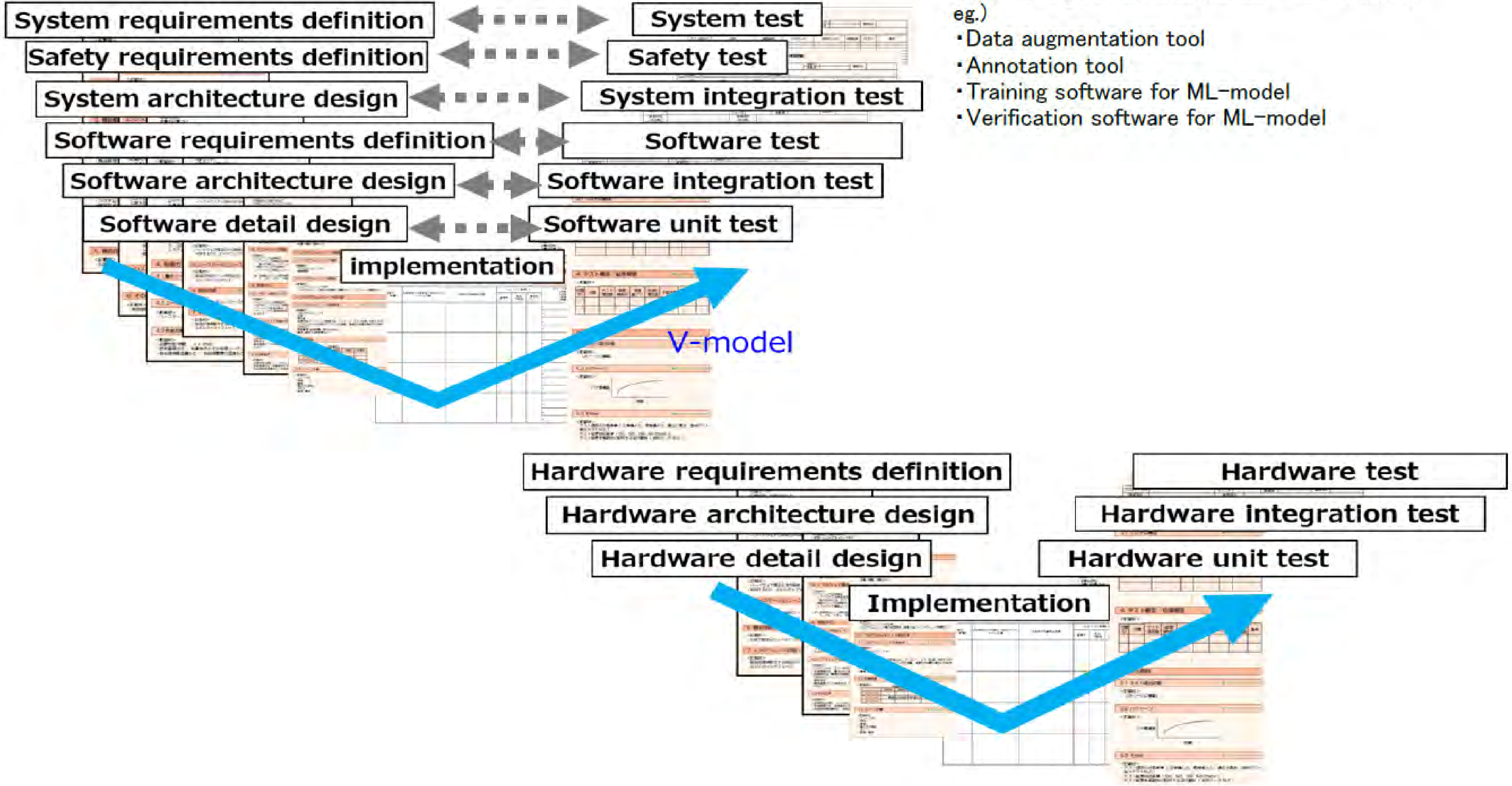
No.	Change target (What)			Change method (How)	Confirmation method (How)	Changes (Impact range)						
	Maintenance target	Purpose of change (Responding to changes in the environment / Specification change / addition / deletion /Bug handling / measures against abnormalities)	Change conditions How to detect changes (Max / Minimum / Threshold / Duration) Change timing	Means of change / procedure (Fine tuning, New learning)	Degreasing prevention confirmation method after change	System requirements analysis	System RA	AI requirements analysis	Dataset design and collection	Learning model	Maintainance plan	Functional safety
1	Machine learning model	Responding to changes in the environment										
2		Specification change (Change / Add / Delete)										
3	dataset	Responding to changes in the environment										
4		Specification change										
5	Functional safety (Soft / Hard)	Responding to changes in the environment										
6		Specification change										
7	(By human) Operation method	Responding to changes in the environment										
8		Specification change										

■Maintenance results

Ver.	Change factor (Why) Specification change management form /Defect management table registration number	Changes (Impact range)							Change result	
		System requirements analysis	System RA	AI requirements analysis	dataset Design and collection	Learning model	Maintainance plan	Functional safety	Judgment	Details (Link to other assessmentsheets)
1.00	Defect management table -001	—	—	—	○	○	—	—	OK	Dataset assessment sheet・ ML model assessment sheet Functional safety assessmentsheet
	Defect management table -002	—	—	—	—	○	—	○	OK	
1.01										

Functional Safety Process Management

■Use procesure and documents of Functional Safety process management


System requirements definition
System test
Safety requirements definition
Safety test
System architecture design
System integration test
Software requirements definition
Software test
Software architecture design
Software integration test
Software detail design
Software unit test
implementation
V-model
Hardware requirements definition
Hardware test
Hardware architecture design
Hardware integration test
Hardware detail design
Hardware unit test
Implementation

Except of target AI elements (datasets and ML-model)
eg.)

- Data augmentation tool
- Annotation tool
- Training software for ML-model
- Verification software for ML-model

References

[1] National Institute of Advanced Industrial Science and Technology (AIST), "Machine Learning Quality Management Guideline 2nd Edition," AIST, 2022.

[2] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," 15 June 2018. [Online]. Available: https://www.sae.org/standards/content/j3016_201806/.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 8 April 2018. [Online]. Available: https://arxiv.org/abs/1804.02767.

[4] S. Liu, D. Huang and Y. Wang, "Learning Spatial Fusion for Single-Shot Object Detection," 21 November 2019. [Online]. Available: https://arxiv.org/abs/1911.09516.

[5] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 23 April 2020. [Online]. Available: https://arxiv.org/abs/2004.10934.

[6] G. R. Jocher, "Yolov5," 22 June 2020. [Online]. Available: https://github.com/ultralytics/yolov5.

[7] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 4 June 2015. [Online]. Available: https://arxiv.org/abs/1506.01497.

[8] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai and H. Ling, "M2Det: A Single-Shot Object Detector based on Multi-Level Feature Pyramid Network," vol. 33, no. 1, pp. 9259-9266, 17 July 2019.

[9] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10778-10787.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 17 April 2017. [Online]. Available: https://arxiv.org/abs/1704.04861.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 13 January 2018. [Online]. Available:

https://arxiv.org/abs/1801.04381.

[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 4 September 2014. [Online]. Available: https://arxiv.org/abs/1409.1556.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 2818-2826.

[15] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects, ISO 26262-9:2018 Road vehicles — Functional safety — Part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses, 2018.

[16] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2020, pp. 2633-2642.

[17] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan and T. Darrell, "GitHub Repository of BDD100k," 21 September 2020. [Online]. Available: https://github.com/bdd100k/bdd100k.

[18] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beibom, "nuScenes: A multimodal dataset for autonomous driving," 26 March 2019. [Online]. Available: https://arxiv.org/abs/1903.11027.

[19] J. Kim, R. Feldt and S. Yoo, "Guiding Deep Learning System Testing Using Surprise Adequacy," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, 2019, pp. 1039-1049.

[20] T. Ouyang, V. S. Marco, Y. Isobe, H. Asoh, Y. Oiwa and Y. Seo, "Corner Case Data Description and Detection," in *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, IEEE, 2021, pp. 19-26.

[21] R. Dabni, "Casting Product Image Data for Quality Inspection, Version 2," 3 July 2020. [Online]. Available: https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product.

[22] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series," in *The Handbook of Brain Theory and Neural Networks*, 1995.

[23] H. Taud and J. F. Mas, "Multilayer Perceptron (MLP)," in *Geomatic Approaches for Modeling Land Change Scenarios*, Springer, Cham., pp. 451-455.

[24] T. Mertens, J. Kautz and F. Van Reeth, "Exposure Fusion," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, IEEE, 2007, pp. 382-390.

[25] Y. Tian, K. Pei, S. Jana and B. Ray, "Deeptest: Automated Testing of Deep-neural-network-driven Autonomous Cars," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 303-314.

[26] The Verge, "A Google Self-driving Car Caused a Crash for the First Time," 29 2 2016. [Online]. Available: http://www.theverge.com/2016/2/29/11134344/googleselfdriving-car-crash-report.

[27] T. Ouyang, V. S. Marco, Y. Isobe, H. Asoh, Y. Oiwa and Y. Seo, "Improved Surprise Adequacy Tools for Corner Case Data Description and Detection," *Applied Sciences,* vol. 11, no. 15, 2021.

[28] K. Pei, Y. Cao, J. Yang and S. Jana, "Deepxplore: Automated Whitebox Testing of Deep Learning Systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ACM, 2017, pp. 1-18.

[29] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang and Z. Chan, "Deepgini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 177-188.

[30] S. Poulding and R. Feldt, "Generating Controllably Invalid and Atypical Inputs for Robustness Testing," in *2027 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2017.

[31] F. Pérez-Cruz, "Kullback-Leibler Divergence Estimation of Continuous Distributions," in *2008 IEEE International Symposium on Information Theory*, IEEE, 2008, pp. 1666-1670.

[32] M. Sokolova, N. Japkowicz and S. Szpakowicz, "Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation," in *Australasian Joint Conference on Artificial Intelligence*, Springer, Berlin, Heidelberg, 2006, pp. 1015-1021.

[33] D. Theckedath and R. R. Sedamkar, "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks," *SN Computer Science,* vol. 1, no. 2, pp. 1-7, 2020.

[34] IEEE Standards Coordinating Committee, IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990), IEEE Computer Society, 1990.

[35] J. M. Zhang, M. Harman, L. Ma and Y. Liu, "Machine Learning Testing: Survery, Landscapes and Horizons," *IEEE Transactions on Software Engineering,* 2020.

[36] S. M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, "Deepfool: a Simple and Accurate Method to Fool Deep Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574-2582.

[37] T. Ouyang, Y. Isobe, V. S. Marco, J. Ogawa, Y. Seo and Y. Oiwa, "AI Robustness Analysis with Consideration of Corner Cases," in *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, IEEE, 2021, pp. 29-36.

[38] Y. LeCun, C. Cortes and C. J. Burges, "The MNIST Database of Handwritten Digits," [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[39] ub, "USPS Dataset, Version 1," 7 April 2018. [Online]. Available: https://www.kaggle.com/bistaumanga/usps-dataset.

[40] Blekinge Institute of Technology, "ARDIS: The Swedish Dataset of Historical Handwritten Digits," 2 April 2019. [Online]. Available: https://ardisdataset.github.io/ARDIS/.

[41] Wikipedia, "MNIST Database," [Online]. Available: https://en.wikipedia.org/wiki/MNIST_database.

[42] T. Ouyang, V. S. Marco, Y. Isobe, H. Asoh, Y. Oiwa and Y. Seo, "Corner Case Data Description and Detection," [Online]. Available: https://arxiv.org/pdf/2101.02494.pdf.

[43] T. Byun, S. Vaibhav, V. Abhishek, R. Sanjai and C. Darren, "Input Prioritization for Testing Neural Networks," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, IEEE, 2019, pp. 63-70.

[44] A. Boopathy, T.-W. Weng, P.-Y. Chen, S. Liu and L. Daniel, "CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks," [Online]. Available: https://arxiv.org/abs/1811.12395.

[45] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhilon and L. Daniel, "Toward Fast Computation of Certified Robustness for ReLU Networks," [Online]. Available: https://arxiv.org/abs/1804.09699.

[46] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao and Y. Wang, "DeepMutation: Mutation Testing of Deep Learning Systems," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*,

IEEE, 2018, pp. 100-111.

[47] A. Odena, C. Olsson, D. Andersen and I. Goodfellow, "TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing," *Proceedings of Machine Learning Research,* vol. 97, pp. 4901-4911, 2019.

[48] Kaggle, "House Prices - Advanced Regression Techniques," [Online]. Available: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data.

[49] United States Census Bureau, "SELECTED HOUSING CHARACTERISTICS," 2020. [Online]. Available: https://data.census.gov/cedsci/table?tid=ACSDP5Y2020.DP04&g=0400000US19.

[50] United States Census Bureau, "American Housing Survey," 2019. [Online]. Available: https://www.census.gov/programs-surveys/ahs/data.html.

[51] A. Lefton and S. Coelho, "This Is the Average Home Size in Every State," [Online]. Available: https://www.bobvila.com/slideshow/this-is-the-average-home-size-in-every-state-53461.

[52] M. Phillip, T. Rusch, K. Hornik and C. Strobl, "Measuring the Stability of Results from Supervised Statistical Learning," *Journal of Computational and Graphical Statistics,* vol. 27, no. 4, pp. 685-700, 2018.

[53] O. Bousquet and A. Elisseeff, "Stability and Generalization," *Journal of Machine Learning Research,* vol. 2, pp. 499-526, March 2002.

Editors and Authors

Digital Architecture Research Center (DigiARC)/
Cyber Physical Security Research Center (CPSEC)/
Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology

List of authors

Supervising editor: Yoshiki Seo
Introductory chapters, editing: Koichi Konishi
Chapter 2, Chapter 3: Sumaiya Saima Sultana
Chapter 6, Appendix E: Takaaki Namba, Tamao Okamoto
Chapter 7, Appendix C, Appendix D.1:
Sumaiya Saima Sultana, Vicent Sanz Marco, MD Nizam Uddin,
Imrad Zulkar Nyeen
Chapter 8, Appendix D.2: Tinghui Ouyang
Chapter 9, Appendix D.3: Imrad Zulkar Nyeen, Tinghui Ouyang
Chapter 10, Appendix D.4: MD Nizam Uddin, Imrad Zulkar Nyeen
Chapter 11: Kiyoshi Fujiwara, Takaaki Namba
Appendix A: Sumaiya Saima Sultana
Appendix B: Imrad Zulkar Nyeen, Tinghui Ouyang, Sumaiya Saima Sultana,
MD Nizam Uddin

List of members of the Reference Guide Task Force (FY 2020-21)

Kiyoshi Fujiwara, AIST

Shintaro Fukushima, Toyota Motor Corporation

Koichi Konishi, AIST

Vicent Sanz Marco, AIST

Kazumasa Miyake, Sumitomo Electric Industries, Ltd.

Takeshi Miyake, Cyber Creative Institute Co. Ltd.

Yoshihiro Nakabo, AIST

Takaaki Namba, Panasonic Corporation

Imrad Zulkar Nyeen, AIST

Tamao Okamoto, Panasonic Corporation

Tinghui Ouyang, AIST

Yoshiki Seo, AIST

Sumaiya Saima Sultana, AIST

MD Nizam Uddin, AIST

Revision history

Version	Date	Description
1.4	August 23, 2022	Corrected Figure 52.
1.3	July 7, 2022	Incorporated the following changes introduced to version 1.0 of Japanese edition. - Format corrections, including numbering equations. - Replaced text on security with reference to the Guideline. - Added footnotes indicating insufficiencies.
1.2	May 25, 2022	Deleted Figure 90 and a paragraph mentioning it in Section 9.5.9
1.1	April 14, 2022	Corrected Figure 2. Corrected several typos. Added revision history.
1.0	March 29, 2022	Initial version