機械学習品質マネジメントガイドライン 第2版
付属文書 2:
開発事例リファレンスに関するプレリミナリレポート

Machine Learning Quality Management Guideline, 2nd revision
Annex 2: Preliminary reference report for Application examples

2020 年度版
Revision FY 2020

2021 年 7 月 5 日
July 5, 2021

産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

デジタルアーキテクチャ研究センター テクニカルレポート
Technical Report － Digital Architecture Research Center
DigiARC-TR-2021-03

サイバーフィジカルセキュリティ研究センター テクニカルレポート
Technical Report － Cyber Physical Security Research Center
CPSEC-TR-2021-003

人工知能研究センター テクニカルレポート
Technical Report － Artificial Intelligence Research Center

はじめに

本レポートは、機械学習品質マネジメントガイドライン第 1 版（CPSEC-TR-2020001）及びその英語版（CPSEC-TR-2020002）を元に、産業技術総合研究所内でいくつかの想定事例にガイドラインを模擬適用した際の知見をまとめたものである。

2020 年度は、以下の 5 つの事例にガイドラインを模擬適用した。

1. 物体認識（object detection）（英語）
   車の自動運転のための物体認識アプリケーション。BDD100K というオープンデータを事例として用い、物体認識 AI の品質評価における特に AI Performance とリスク回避性の評価を検討した結果をまとめた。
2. 外観検査（visual inspection）（英語）
   金属製品の傷や凹みなどの欠陥を認識する外観検査アプリケーションの試験構築を行い、ガイドラインに沿った品質評価を実施した。欠陥データの数が少ないため、データ Augmentation による訓練データの被覆性の評価に重点を置いた。
3. 自動運転判断（intelligent wheelchair）（日本語・英語）
   自動運転を行う車椅子の人物認識アプリケーションを想定し、特に安全性のためのリスク回避性の品質評価プロセスに主眼を置いた検討を行い、知見をプロセス管理に用いるアセスメントシートにまとめた。
   なお、このリスクアセスメントシートについては、ガイドライン第 2 版への最低限の対応を行ってある。
4. 価格予測（house price prediction）（英語）
   Regression を行うアプリケーションの品質評価事例として、オープンデータ Kaggle に含まれる住宅価格予測アプリケーションの品質評価をガイドラインに沿って実施した。訓練データに偏りがあるため、これに合致するようにアプリケーションの適用範囲を絞ることに注力した。
5. 郵便番号認識（postal code detection）（英語）
   MNIST の数字認識アプリケーションを郵便番号認識に活用する想定で品質評価を行い、特に Robustness の評価手法について検討した。

About this report

This preliminary report is a summary of our knowledge obtained from application of "Machine Learning Quality Management Guideline" (Japanese and English, 1st revision; CPSEC-TR-2020001 and CPSEC-TR-2020002) to trial development of AI systems for several application domains. The domains chosen for this fiscal year's trials are following:

1. object detection (English)
   Using the open data BDD100K as a case study for object recognition applications for automatic car driving, we examined the quality evaluation of object recognition AI, especially AI performance and risk reduction.
2. visual inspection (English)
   A visual inspection application that recognizes defects such as scratches and dents on metal products was constructed experimentally and evaluated its quality in accordance with the Guideline. Since the number of defect data is small, we focused on evaluating the coverage of the training data by data augmentation.
3. automated driving decision (intelligent wheelchair) (Japanese and English)
   Assuming a human recognition application for an automated wheelchair, we studied the quality assessment process for safety risk avoidance, and created an assessment sheet for process management.
   This risk assessment sheet is minimally adapted to the second edition of the guideline. 4.
4. house price prediction (English)
   As a case study of quality evaluation of an application that performs regression, we performed a quality evaluation of a house price prediction application included in the open data Kaggle, according to the Guideline. Given the bias in the training data, we focused on narrowing down the scope of the application to be consistent with this bias.
5. postal code detection (English)
   We evaluated the quality of MNIST's numeric recognition application for postal code detection, especially the evaluation method for robustness.

Reference guideline for Object Detection and Scene Classification task for Autonomous Driving Vehicle

Contents

1. Purpose of the technical report:

The goal of this report is to create an example implementation to demonstrate how Machine Learning Quality Management (referred as MLQM from here after) guideline can be used to evaluate internal properties of an autonomous driving dataset and the AI solutions created using this dataset. This report can be used as a reference for application of the guideline in evaluation of similar AI based systems.

2. Expected outcomes:

Following outcomes are expected to achieve by applying MLQM guideline to an AI based product:

- Setting out well-defined standards for assessing qualities for the product; these specified standards can be used by 'development entrustee' aka 'service developer' during designing and development stage and by 'development entrusted' aka 'user' while in evaluation stage for measuring the promised quality of the final product
- Creating a clear demonstration of the product's quality, safety and reliability for the final users
- Identifying safety-critical scenarios in advance to reduce risks of accidents from possible erroneous behaviors of AI models

3. Author's role:

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

4. Product specifications:

This section describes the specifications of the final product or the anticipation of client about the AI solution. For the specific problem of object detection and classification for autonomous driving scenarios, the final product constitutes of machine learning models which is built using ***supervised learning***. The tasks performed by the ML models are scene understanding/***scene classification*** and ***object detection*** from images. Hence, 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.

4.1 Model Specifications:

Anticipation of the specifications for the models to be developed are given below. Contender models can include (but not limited to) the architectures given in this section:

- **Type of learning:** Supervised
- **Type of AI model:** Classification and object detection

- **Initial dataset:** The initially chosen dataset to use for training 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.

- **Contender model architectures:**

  i. **Task: Road Object detection**
     - YOLOv3 [Redmon, 2018]
     - YOLOv3 + ASFF [Liu, 2019]
     - YOLOv4 [Bochkovskiy, 2020]
     - YOLOv5 [Jocher, 2021]
     - Faster R-CNN [Ren, 2016]
     - M2Det [Zhao, 2019]
     - EfficientDet-D2 [Tan, 2020]
     - MobileNetv1 [Howard, 2017]
     - MobileNetv2 [Sandler, 2019]

  ii. **Task: Scene classification:**
     - VGG19 [Simonyan, 2014]
     - ResNet50 [He, 2016]
     - InceptionV3 [Szegedy, 2016]

4.2 Safety specifications:

ASIL (Automotive Safety Integrity Level) is being used to specify required safety level for this AI product. ASIL D is specified to be maintained as the requirement level of safety. 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.

ASIL may be similarly expressed as

$$ASIL = Severity * (Exposure * Controllability)$$

In terms of the classifications provided by ASIL(details in Appendix) an 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 **S3**, **E4**, and **C3** classifications. More information about this is written in the appendix.

As consequence, the relation between ASIL D and autonomous car is that the system needs to detect objects to avoid any type of situation that can cause any severe accident (S3 and E4). Regarding C3, it is not one of the aims in this research. To focus on C3, the detection models needs to be trained to detect the street lines where the car goes. For this reason, in this report, the **KPI will be defined using S3 and E4 and assume that C3 is always accomplished.**

4.3 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 report, initial KPI specifications are given below:

4.3.1 Object detection task:

Object detection algorithms are usually evaluated based on metrics such as the mAP or 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 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.

4.3.2 Scene classification task:

Classification models are usually evaluated based on metrics such as confusion matrix, accuracy, precision, recall, specificity and 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 provide more detail information about the model's performance in **high risk cases** with respect to false positives and false negatives. So, **F1 score** can be considered as KPI where class labels are not well-balanced and it is necessary to deal with high risk cases.

5.Proof of Concept (PoC) phase:

5.1 Initial investigation of existing dataset:

For the creation of the classification task and object detection task explained in this reference guide, diverse datasets have been considered. There are a lot of datasets related to autonomous driving, but among those, following datasets are used in primary investigation:

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

Table 1: Dataset comparison

For this research, it is necessary to know the weather and time of day of the images in the dataset. Only BDD100k and Oxford Robotcar Dataset accomplish these requisites. However, Oxford Robotcar Dataset does not include labeling in their images. As a consequence, the best option for this research is BDD100k [Yu, BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning, 2020] with its Github repository [Yu, 2021]. BDD100k has 100,000 of images all of them with 2d bounding boxes. The dataset has been split in 3 for training, testing and validation using 70%, 20% and 10% of the dataset respectively. For this research the default splitting given by the dataset authors has been used. Following information are collected from an initial investigation of the dataset.

- Total images in training: 70,000
- Annotated images in training dataset using json format: 69,863
- Total images in validation: 10,000
- Annotated images in validation dataset using json format: 10,000
- Given attributes and attributes values for 'classification' task:
  - **Weather**: rainy, snowy, clear, overcast, undefined, partly, cloudy and foggy
  - **Scene**: tunnel, residential, parking lot, undefined, city street, gas stations and highway
  - **Time of day**: daytime, night, dawn/dusk and undefined
- This dataset has 10 labels to identify objects: *Bus*, *Light*, *Sign*, *Person*, *Bike*, *Truck*, *Motor*, *Car*, *Rider* and *Train*
- Labels for BDD100k images also include additional information on *Occlusion*, *Truncation*, *Traffic light color*, *Lane direction*, *Lane style* and *Lane type*.

Additional useful information from BDD100k labels is presented below:

LANETYPEs
crosswalk
double other
double white
double yellow
road curb
single other
single white
single yellow

OCCLUDED	TRUNCATED	LANE DIRECTION	LANE STYLE
TRUE	TRUE	parallel	solid
FALSE	FALSE	vertical	dashed

TRAFFIC LIGHT COLOR
red
green
yellow
none

Area type
Alternative
Direct

Table 2: Additional information from BDD100k dataset labels

5.2 Distribution of data:

Basic distribution analysis for the available dataset is given below:

5.2.1 Classification task:

For the classification task using BDD100k, in this guide reference, the solution designer has focused on *scene*, *weather* and *time of day* attributes. These 3 have more priority than the others in terms of the visualization of the images and are more relevant to ASIL D than the others.

The following charts show the distribution of the attribute values of *Scene*, *Weather* and *time of day*

Training dataset statistics:

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:

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

Figure 1: Statistics for BDD 100k-classification task for 3 different attributes in training and validation datasets

It is evident that there are some attribute values that appear more often than others, such as, *daytime* and *clear*. As well, there are attribute values that does not appear often in the dataset, such as, *tunnel*. Such attribute values may need a treatment the number of images is needed to be increased.

5.2.2 Object detection task:

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

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

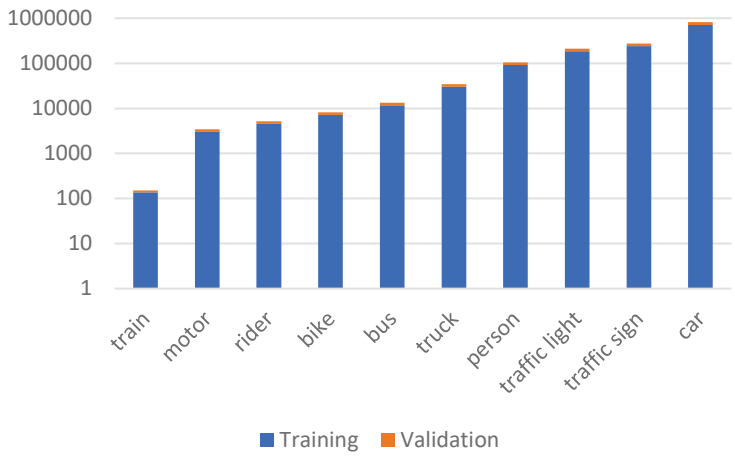


Figure 2: Amount of instances that each label appear in BDD100k dataset

BDD100k has a lot of *car* instances in comparison with other labels. This lack of uniformity in the dataset became a problem to train the detection models and it was one of the issues that the solution designer focused. The main issue was that the detection models were overfitted of *car* instances and could not recognize properly the other labels. Different solutions were created and will be explained in this reference guide later.

5.3 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 training procedure and the trained model performances are summarized below:

5.3.1 Preprocessing steps:

- ✓ Images are auto resized to fit by corresponding networks, 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)

5.3.2 Hyper-parameter specifications:

Tuned hyperparameters of each model are presented below:

Model	No. of iterations	Learning rate	Batch size
YOLOv3	160,000	0.001	8
YOLOv3 + ASFF	7,000,000	0.001	16
YOLOv4	40,000	0.001	32
YOLOv5	980,000	0.01	16
Faster R-CNN	115,000	0.00025	8
M2Det	700,000	variable	1
EfficientDet-D2	300,000	0.079	18
MobileNetv1	120,000	0.01	16
MobileNetv2	130,000	0.01	16

Table 3: Training configurations for object detection models using BDD100k

5.3.3 Validation results:

Following results were obtained from the trained models evaluated on the validation dataset with 10k examples:

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

Table 4: Accuracy measurement of contender object detection models in PoC phase

5.3.4 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 each model:

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
MobileNetv1	https://1drv.ms/u/s!AvkGtmrlCEhDhy1YqWPGTMl1ybee
MobileNetv2	https://storage.googleapis.com/mobilenet_v2/checkpoints/mobilenet_v2_1.4_224.tgz

Table 5: Pretrained weights used for models

5.4 Insights 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 in section 5.1 compared to anticipated/expected domain specifications mentioned in section 4.1. These 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 5.2 reveals the distribution of samples is imbalanced because 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 rarely in real life but carries significance 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 5.3 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.

6. Designing development phase:

6.1 Incorporation of MLQM guideline:

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

- 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 dataset 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

To properly fulfill the above requirements, Machine Learning Quality Management (MLQM) guideline (written by Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology) will be used in this report. The protocols established in the guideline give detail instructions on how to achieve these requirements by setting eight aspects of 'internal quality' as means of quality management. Hence, it is expected that ensuring these eight aspects of 'internal quality' (discussed in MLQM guideline section 1.7 and section 6) in the next stages of development will consequently create the best version of the AI product in terms of required quality, safety , performance and fairness.

6.2 Adjustment of standards of quality management:

In the next stage, AISL (AI Safety Levels, introduced in section 3.1) from MLQM guideline will be used to assess safety and quality levels extensively. However, in the product specification section (4.2) of the report, ASIL (Automotive Safety Integrity Level- risk classification scheme defined by the ISO 26262 - Functional Safety for Road Vehicles standard) is used initially to declare safety requirements. An assumption is made that the requirement provided by the Development entruster is ASIL D (a combination of **S3**, **E4**, and **C3** classifications in terms of Severity, Exposure and Controllability). So, it is necessary to demonstrate that the safety requirements demanded by the Development entruster are fulfilled properly even though protocols provided by MLQM were followed.

In this intermediate stage while shifting from ASIL to AISL, it should be stated which category of ASIL (QM, A, B, C and D; ISO 26262) corresponds to which category of AISL((AISL 4, AISL 3, AISL 2, AISL 1, AISL 0.2,

AISL 0.1, AISL 0). For this report, AISL 0.1 will be considered as the required level of safety for all the internal qualities. In addition to these, AIPL (AI performance level) and AIFL (AI fairness level) are chosen to be AIPL1 and AIFL1 respectively.

7. Internal quality evaluation using MLQM guideline

As discussed in section 6, eight aspects of the internal quality will be evaluated simultaneously throughout the agile development process of AI model generation.

7.1 Sufficiency of requirement analysis:

After initial investigation of the current dataset and insights gained from PoC phase, first characteristics axes of quality management, 'sufficiency of requirement analysis' , will be analyzed in response to the achievement of two external qualities - risk avoidance and AI performance, as mentioned in the MLQM guideline.

7.1.1 Definition of 'Sufficiency of requirement analysis':

According to the guideline, the term 'sufficiency of requirements analysis' 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, mentioned in section 6.1.1.

So, here we analyze the requirements in specific linguistic terms in as much detail as possible. This involves clearly defining the problem domain of the autonomous driving scenario that we are expecting to encounter in real life and wish to solve by using our supervised machine learning models.

7.1.2 Redefining problem domain:

Inspired from the guideline that envisions a concept of feature tree, the problem domain is defined with the following 'attributes' and their corresponding 'attribute values'. The term used here as 'attribute' and 'attribute values' carries the same meaning and significance as defined in section 6.1.2.1 of the guideline.

Some requirements are declared to fulfill before designing the problem domain. The requirements are:

- **Data for all possible classes and objects:** Requirements for a machine learning solution means the expected performance of the solution in real world scenario. In this case, while designing the domain it must include all the classes and objects that are necessary for the final product. *Differences found between the proposed domain and the existing dataset, should be handled in later stages of the quality evaluation procedure.* Specifications provided by user/development entruster, such as, all possible classes/object types), should be considered here.
- **Selecting well-defined feature dimensions:** The attributes and their corresponding attribute values should cover any possible data specific scenario needs to be considered and listed for later analysis like coverage or sufficiency. While deciding the final set of attributes based on the proposed domain and the PoC phase results, we will consider discussing the scopes of attributes and their values that have been considered; what scenarios each attribute value covers.
- **Selecting in-bound and out of bound areas:** 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. Further specifications **provided/considered** while designing the problem domain, especially if some scenarios are intentionally left

behind the scope of the model should be reviewed in sufficiency of requirement analysis phase.

7.1.3 Proposed problem domain:

Next, we present the problem domain with a newly defined set of attributes to cover all perspectives of the target domain in as much details as possible. 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 concerns the solution designer significantly.

The following table shows the attributes and the corresponding values used to correctly identify such cases that the ML model may face considering the initial product specifications and the solution desiner's concerns.

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

Table 6: Problem domain specifications

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,B is the mean value of 3 channels (resolution: 720*1280)

- The range of brightness values is from 0 to 255, 0 being complete black and 255 being complete white.

7.1.4 Example of training data described using the proposed domain:

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

	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 3: Example datapoint described by the new problem domain

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.

7.1.5 Comparison of existing domain with proposed domain

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

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	-

Table 7: Comparison of problem domain of BDD100k and proposed problem domain

7.1.6 Approaches considered to adopt 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.
  - **'City street'** and **'Residential'** can be merged as **'General way'** in 'Road type'
  - **'Overcast'** & **'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.

7.1.7 Re-designing dataset:

Based on the discussions made on section 7.1.5 and 7.1.6, the following domain is obtained from BDD100k for the AI model to be used in training and validation steps.

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						

Table 8: Final problem domain considered in this application

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

So, instead of **3 different tasks** to be learned for classification models, this new domain allows the solution designer to create **6 classification models performing 6 different tasks**.

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 7.1.4) 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 will be made in the future.

7.2 Coverage for distinguished problem cases:

7.2.1 Definition of 'coverage of distinguished problem cases:

MLQM guideline recommends to identify various possible combinations of the attribute values presented in the requirement 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 attribute 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.

7.2.2 Steps required for evaluation:

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.

7.2.3 Example of evaluation process:

The following example is used for demonstration of the evaluation process. For the purpose of 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. Let's consider the following attributes and their corresponding values to assess the complexity of the situation.

7.2.3.1 Considered domain details:

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)**

7.2.3.2 Summary of assessment 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.
- Total number of groups to check combinations: **21**
- Total number of combinations: **542**

An example of presence of data for one combination **'Lighting + Signal'** among these 542 number of combinations is presented below:

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

Table 9: Presence of data for combination 'Lighting + Signal' across combined attribute values

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

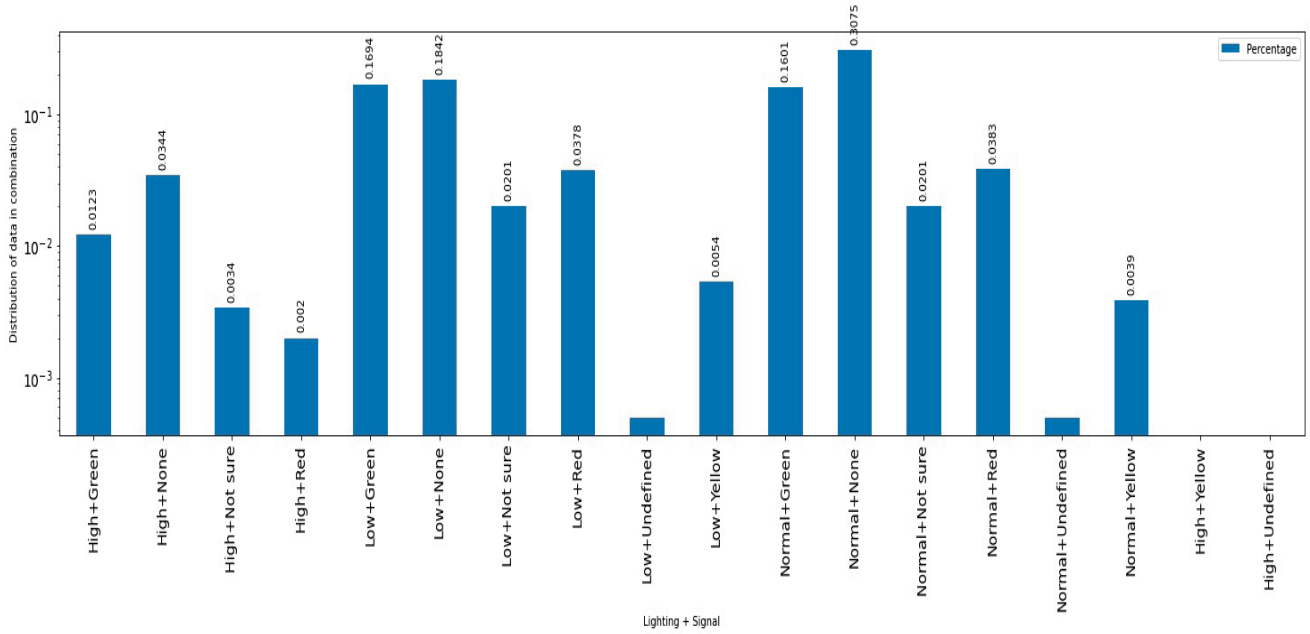


Figure 4: Distribution of data for possible case in combination 'Lighting + Signal'

It can be very time-consuming and exhaustive process for a solution designer to check the results of this kind of analysis for ensuring the presence of enough data in possible combinations separately in this type of pair-wise analysis. So, in the next step, instead of doing pair wise analysis, it can be considered to calculate distribution for possible combinations taking all 7 attributes and all values(**~54k**). Since python scripts can be written to calculate this distribution, the process will be automatic.

7.2.4 Identifying special cases:

The solution designer can also 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 requires 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, it is 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).

7.2.4.1 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. Following are 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.

Case	Primary Condition 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

Table 10: Unsound cases that are not expected to be present in data

7.2.4.2 Safety critical cases/Risky cases:

Following are some risky cases identified by the solution designer. These cases are considered risky because in model performance may degrade in these scenarios and wrong decisions made by the 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 easily reduce the number of combinations **'with enough data'** to a manageable value. This evaluation can also 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

7.3 Coverage of dataset

7.3.1 Definition of 'Coverage of dataset':

In the 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.

7.3.2 Steps required for evaluation:

Following approaches can be taken to ensure if enough amount of good data with unbiased distribution is available for each important scenario, specially the cases identified in section 7.2.4.

- 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.

7.3.3 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 specially test/validation dataset. But for now, let's continue the evaluation of coverage of dataset for specific scenarios on the same dataset used in 7.2.3.2. Following are the results obtained after investigating the coverage for various scenarios mentioned in previous section:

7.3.3.1 Presence of data in unsound cases:

Following is the presence of data 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

Table 11: Data coverage in unsound cases

As mentioned earlier, the training data should not contain such unsound cases. The inclusion of this kind of data may happen due to different reason, 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.

7.3.3.2 Presence of data in risky cases:

Results of data coverage are summarized below for high risk cases mentioned in 7.2.4.2.

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

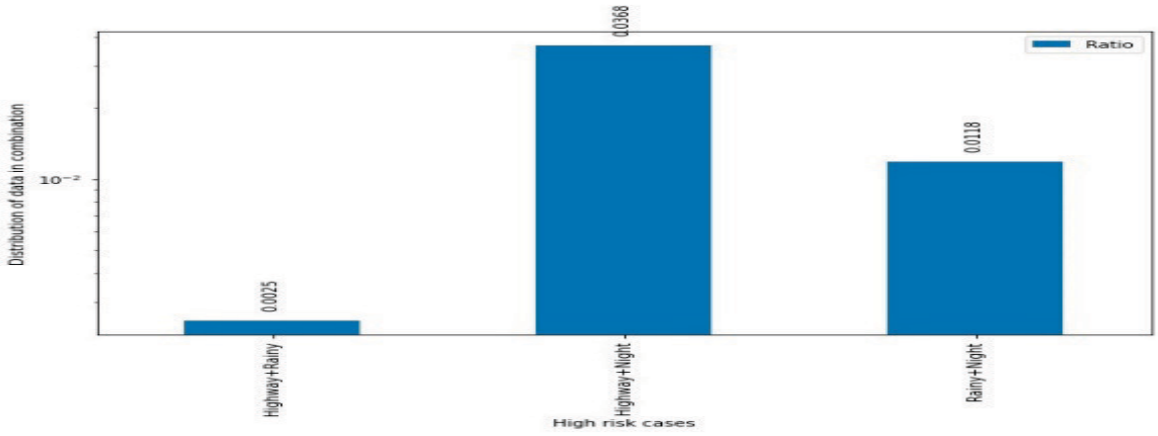


Figure 5: data coverage across some high risk cases

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

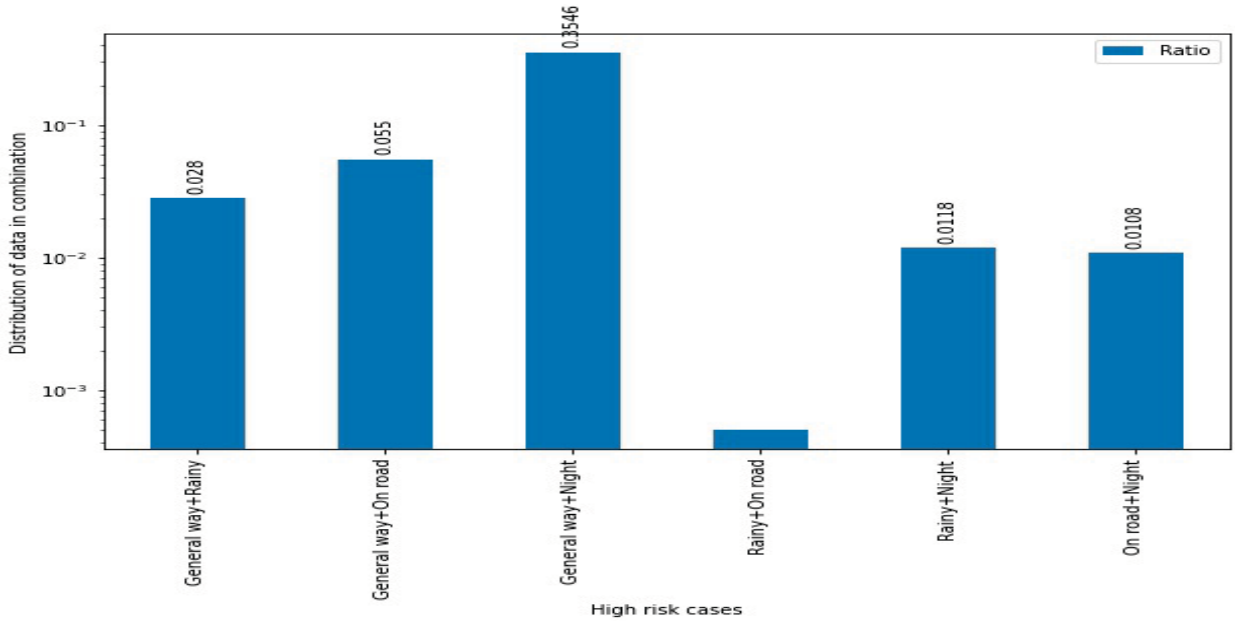


Figure 6: data coverage across some high risk cases

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

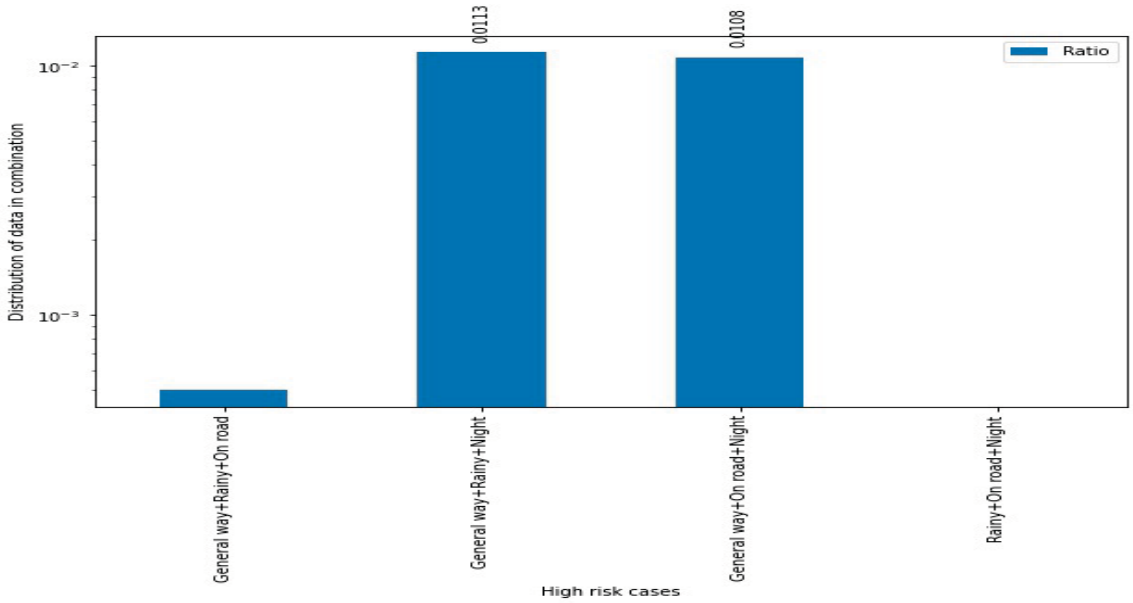


Figure 7: data coverage across some high-risk cases

7.3.4 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. From results presented in 7.3.3, rare images are identified that do not occur in the dataset that often but extremely important for autonomous driving scenarios. Since it is needed to take acceptable number of images into the training and more examples from the rare case scenarios are needed, 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. The images from BDD100k dataset are all of them obtained from BDD videos. Additionally, each image in BDD100k has an ID pointing which video was used in the BDD video dataset. As 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 can be used but with different timestamp. But from data coverage investigation, it is found that such cases rarely happen. So, first an unclear image due to water on the windshield is identified in the video dataset. Next, two frames are extracted: from 1 second before and after the original sampled 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, as consequence, the extracted images can be included in the dataset for training or validation. In this way, the amount of images of the combinations that have low available images can be increased.

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

Figure 8: consecutive frames of '02701fba-809c39f3.mov' from BDD video dataset.

7.4 Uniformity of dataset

7.4.1 Definition of 'Uniformity of dataset'

MLQM guideline tells us that 'Uniformity of dataset' is a concept contrary to 'coverage' mentioned in 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 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". Here the primary concern is to evaluate the balance between 'coverage' and 'uniformity'. MLQM guideline states that it is necessary to consider to which one receives priority, coverage or overall uniformity, and how to strike a balance between them.

7.4.2 Steps required for evaluation

The priority mentioned in 7.4.1 will have a significant effect on the models' performance. Depending on the given requirements, the solution designer may choose to:

- 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.
- 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.

7.4.3 Example evaluation process

We will 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 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 describes 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*

7.4.3.1 Evaluating general distribution of dataset

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

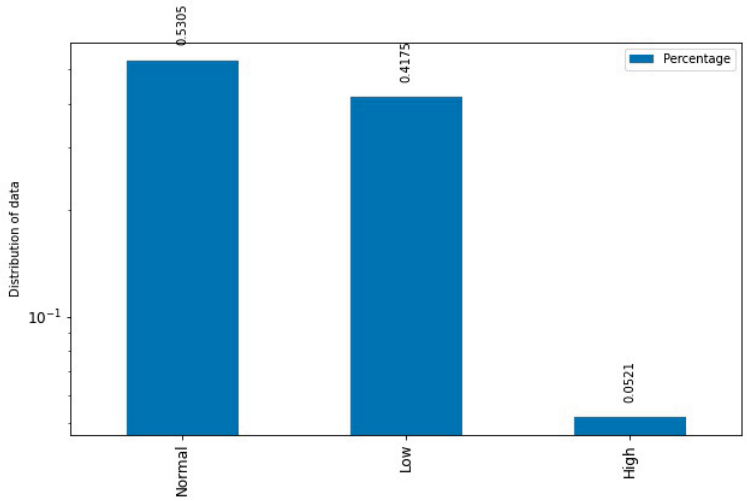


Figure 9: Distribution of data across attribute values of attribute 'Lighting'

The following graph shows the distribution of data for 'Road type' attribute across attribute values.

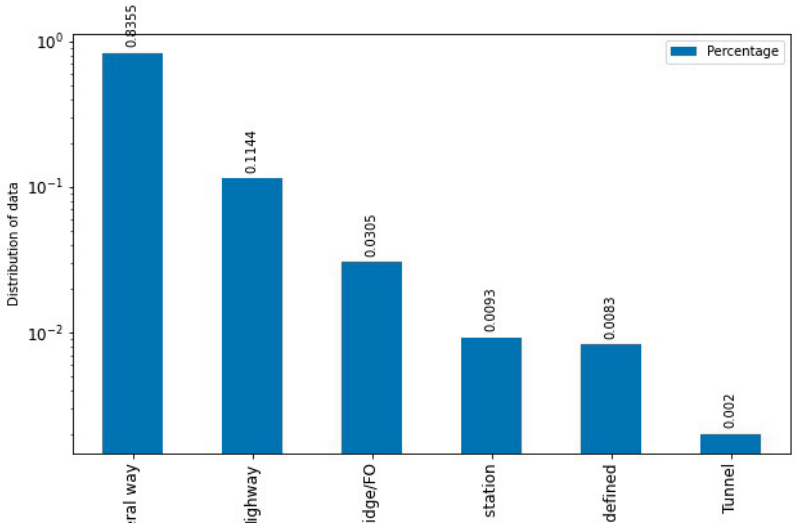


Figure 10: Distribution of data across attribute values of attribute 'Road type'

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

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 12: Comparison of frequency of occurrence for attribute values of 'Lighting'

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

Table 13: Comparison of frequency of occurrence for attribute values of 'Road type'

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.

7.4.3.2 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:

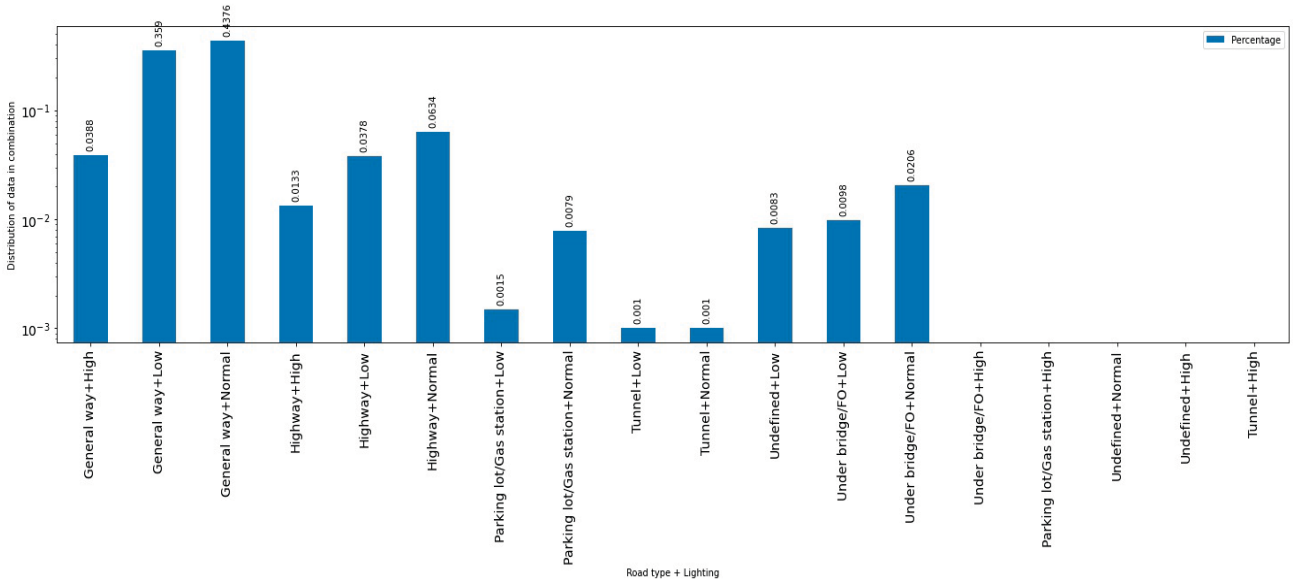


Figure 11: Distribution of data across combinations of attribute values of Road type+ Lighting

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

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

Table 14: Comparison of frequency of occurrence for combined attribute values

It should be noted that with availability of actual data form operation environment and adequate time, these evaluation procedures should be executed on original 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 reference reports. This version of the report does not include detail experiment or exact evaluation procedures in most cases. Rather it discusses how the evaluation process and model development process can be carried out for object detection task. In next version of the report, more detailed experiments and evaluation procedures are expected to be included. Also, similar results for the classification 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 Correctness of the trained model

7.5.1 Definition of 'Correctness of the trained model'

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).

7.5.2 Decisions from PoC phase:

Using the results shown in section 5.3.3, it has been decided to only continue researching 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% of mAP. As consequence, the solution designer focused on the best models trying to improve the accuracy of those ones. Regarding the FPS, it is possible to improve the FPS of the models, however, improving the FPS decreases accuracy, as far as we know. As consequence, we have decided to focus only on improving accuracy and leave the improvement and study of FPS for future work. As consequence, Yolov3, M2Det, EfficientDet-D2, MobileNetv1 and MobileNetv2 are removed from the research 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 of this research to improve their accuracy to achieve enough mAP to accomplish SADL D.

7.5.3 Evaluation procedure for correctness of object detection models

Through the research, it has been developed different approaches to improve the accuracy.

7.5.3.1 Removing noise information in the dataset

The below image shows how some bounding boxes overlap making difficult the training process. This issue brings noise to the training process of each detection model reducing the accuracy obtained.

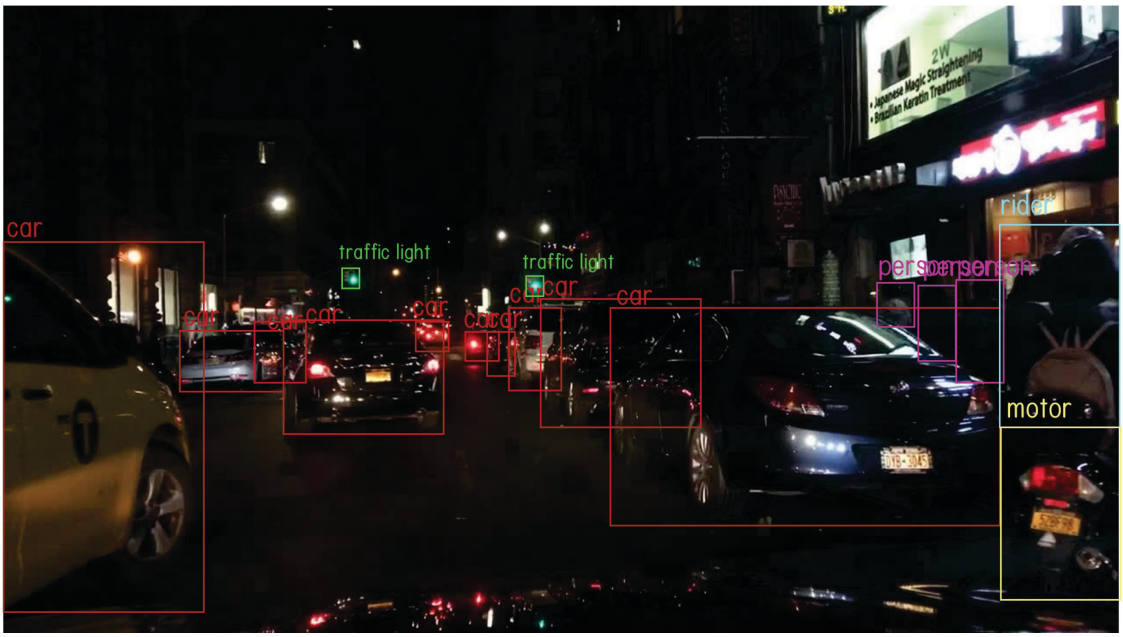


Figure 12: Image example of overlapping bounding boxes

After checking with Toyota and their necessities, the best solution to increase the accuracy was to reduce the amount of boxes and make the detection models learn about objects that are close to the car. As a result, it has been tested different approaches trying to remove the overlapping bounding boxes. The name of this process is called **Reducing Boxes**.

Due to it is impossible to know the whole time the distance of an object from a single image, it has been created 6 different measures to remove the bounding boxes. The idea is if a bounding box fits inside the created artificial window, then that bounding box is removed from the training dataset. After all images have been checked and all the bounding boxes that fit inside the artificial window has been removed, then a new dataset without the problematic bounding boxes is created 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 13: Different artificial windows sizes used to remove bad bounding boxes

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 15: mAP comparison after reducing boxes mechanism

As a result, the table before shows the results of removing the bounding boxes and re-training each detection model selected. As a result, there is a mAP increment when removing bounding boxes smaller of 10x10, 20x20 and 30x30 about 1.7%, 7.1% and 10.2%, 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 in order to detect overlapping annotations and check if the mAP increases.

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 16: Label mAP result after reducing boxes mechanism is performed

The previous table shows the mAP results per label of using *Reducing Boxes* algorithm on Yolov4. This result shows how removing the overlapping boxes helps the detection model to improve the accuracy of all labels.

7.5.3.2 Specific training for the detection models

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

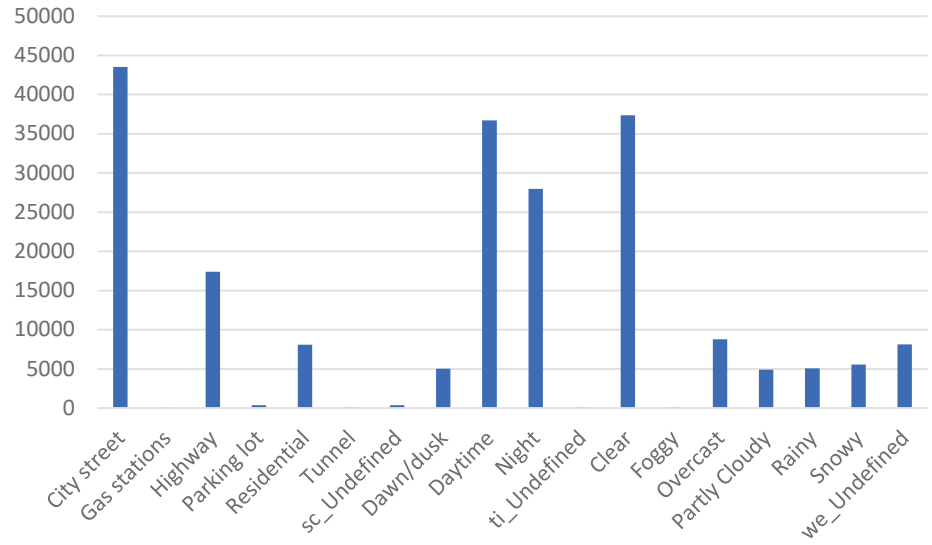


Figure 14: Number of images per attribute value

For training a detection model is necessary to have at least 20,000 images. Figure 14 shows the number of images per attribute value. As consequence, 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)*. We are showing the data of *city street*, *daytime* and *night.*

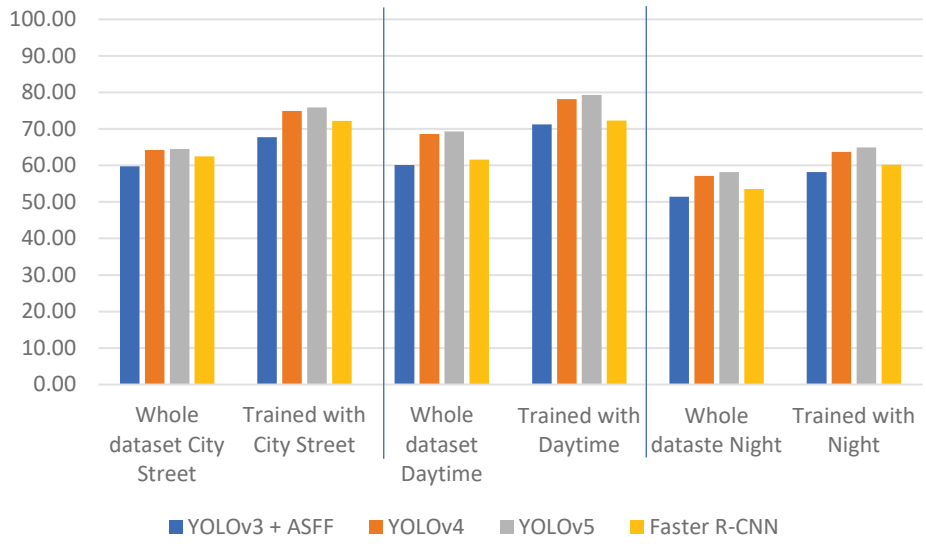


Figure 15: mAP comparison of specific training using attribute values to the whole dataset of attribute value used for specific training

Figure 15 shows the mAP comparison of training each detection model only with images with *city street*, *daytime* or *night* attribute values, 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%.

Summarizing, with this experiment it is demonstrated that there are situations that is better to use specific dataset for training instead of having a dataset with diverse situations. Using this idea, a system could use multiple detection models and use them depending on the situation, such as, a model that works in *city street* attribute value when the car is inside the city, or two models that one works with *daytime* and the other when is *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, it has been used the whole dataset, that means, the *reducing boxes* algorithm has not been used. For this reason, we want to join specific training with reduction boxes mechanism in the future. Additionally, in the future, test combinations of attributes values will be tested, such as, having a model that works on *daytime* and when the car is in a *city street*.

7.6 Stability of the trained model

7.6.1 Definition of 'stability of the trained model

According to section 1.7.6 of the MLQM guideline, the term "stability of the trained model" means that a machine-learning component shows a reaction to input data which is not included in learning datasets sufficiently similar to data in learning datasets. Ensuring this quality includes evaluating a model's generalization ability, evaluating a model's reaction to corner cases/rare cases, evaluating the model's performance on adversarial examples. Assessing 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.

7.6.2 Steps required for evaluation

Steps that can be executed by the solution designer to evaluate this inter 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 close proximity of our 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. Used evaluation techniques and their effectiveness needs to be described beforehand.

- **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. In section 1.7.2, the guideline mentions that these evaluation techniques can be done directly by numerical analysis or indirectly by test method management. Evaluation procedures of measuring robustness and the level of robustness needs to be described to reflect the response of the model to adversarial examples.

7.6.3 Evaluating generalization capability

For this section, a different dataset has been used to evaluate the performance of the detection models used after they have been trained using BDD100k. In this case, NuImage [Caesar, 2019] dataset has been used. NuImage has the following properties:

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

Table 17: NuImage dataset description

This dataset contains 93476 images divide it by train, test and validation data:

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

Table 18: NuImage dataset images distribution

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

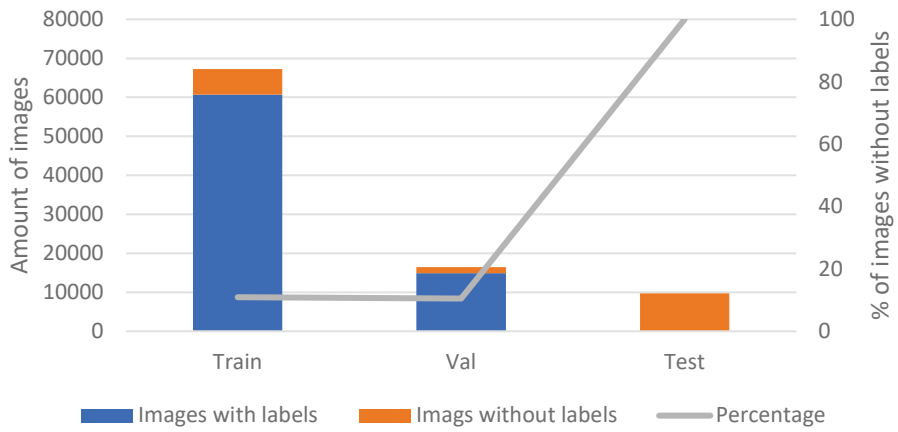


Figure 16: NuImage distribution of images that have and not have annotation

So, it is possible to only use 60,668 and 14,884 images for training and validating the detection models, respectively. Regarding the amount of annotations:

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

Table 19: NuImage label distribution and (in green) labels used for converting to BDD100k labeling

BDD100k is a dataset with 10 labels while NuImage has 23 labels. As consequence, the solution designer has reduced the 23 labels to the ones that can match the detection models trained by BDD100k. Not all labels can be used, so it has been highlighted in green color on the previous table the labels used to convert NuImage annotation in BDD100k annotation. This is necessary to evaluate the capabilities of the detection models because it is necessary to have the ground truth of NuImage annotation. As consequence, only the following BDD100k labels can be detected: *bike*, *bus*, *car*, *motor*, *person* and *truck*.

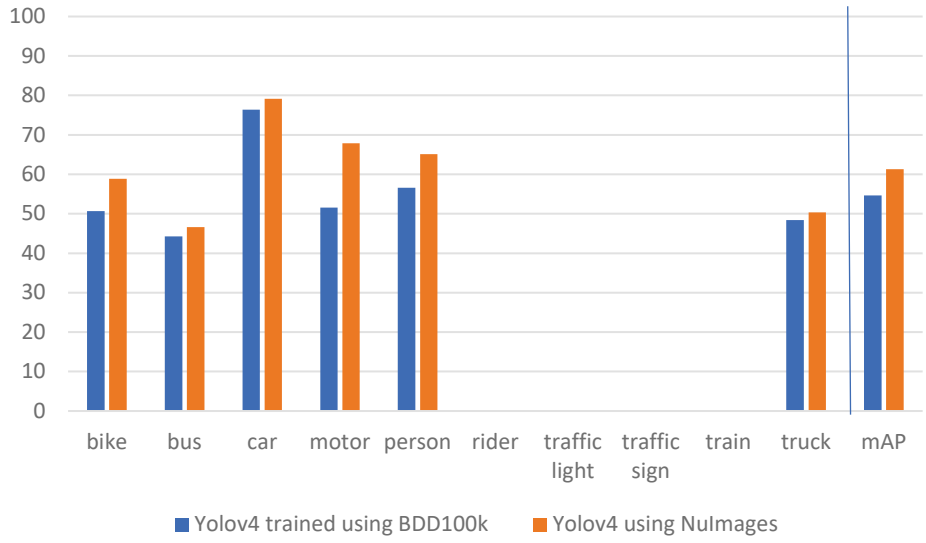


Figure 17: Accuracy comparison of Yolov4 on NuImage validation dataset after being trained by BDD100k and NuImage training datasets.

To compare the results, it has been trained 2 different models of Yolov4. 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 know how effective the detection model capability performs 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 1 shows this comparison. The labels *car*, *bus* and *truck* achieve similar accuracy, 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*.

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

7.6.4 Evaluating robustness to adversarial images

To evaluate the robustness of the object detection models it has been decided to use Surprise Adequacy 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:

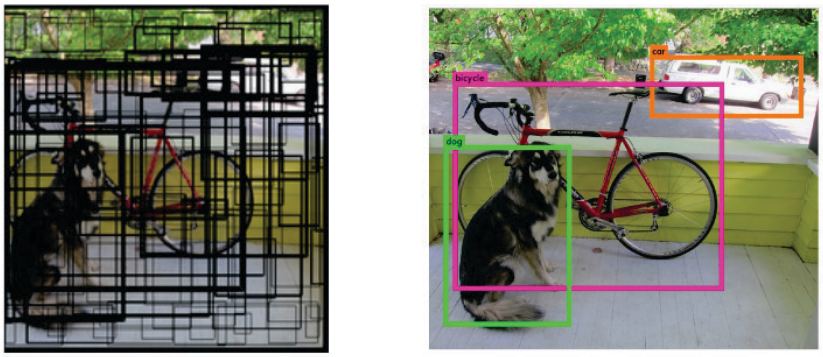
- Fast Gradient Method (FGSM)**Invalid source specified.**
- Basic Iterative Method (BIM-a, BIM-b)**Invalid source specified.**

- Jacobian-based Saliency Map Attack (JSMA)**Invalid source specified.**
- Optimization-based attack (Opt)**Invalid source specified.**

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

7.6.4.1 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 and it is not working with all. 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 NN, 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 making only the boxes with higher label score are kept. For the creation of adversarial examples is not possible to use non-max suppression because it is a selection method and not a layer. As consequence, the generation of adversarial examples using Yolo neural networks is an issue that we tried to resolve.



Yolo result output of last 3 layers

Yolo result after performing non-max suppression

Figure 18: Example of Yolo last layer performance

7.6.4.2 Attempted approaches:

The solution designer attempted to adapt FGSM attack for YOLOv4 architecture being unsuccessful due to 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.

7.6.4.3 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.

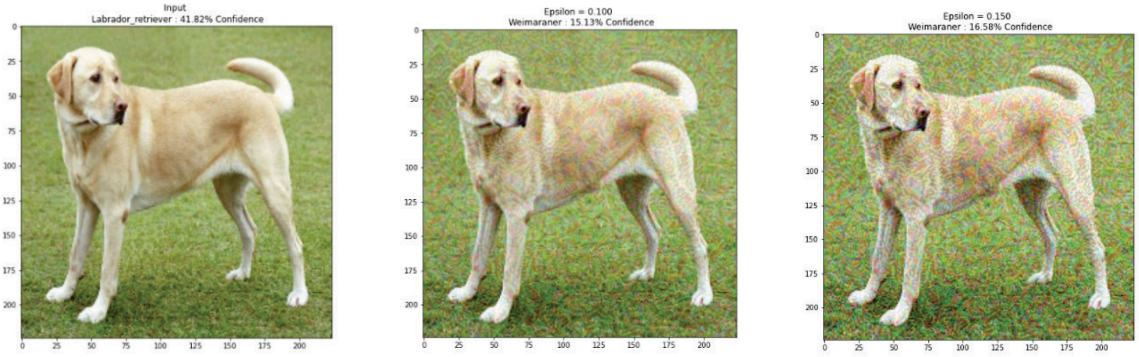


Figure 19: Example from successful implementation of FGSM attack on MobileNet

- **Trying different attacks** can be a solution to the challenge. However, all the methods that generate adversarial examples to do Surprise Adequacy use output label confidence and, as consequence, they only work if the NN has one last output layer and not multiple last output layers. So, defining new attacks different that the ones mentioned might be necessary. For example, using ***1-pixel change*** **Invalid source specified.** method to detect the ***corner cases*** that has a high percentage to cause a bad labelling. If it has a high percentage, it can be removed from the training data. After that, we can retrain the detection model and check how it is working.

7.6.4.4 Adapting adversarial attacks using a different NN

In order 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. Adversarial examples are malicious attempts to perturb the dataset adding different noise or attacks to the images in the dataset. In this case, it has been used MobileNetv2 that has been trained using BDD100k and Tensorflow-Keras framework. After training MobileNetv2, the trained model obtains an accuracy of 84.49% but with an FPS of 4.5. As consequence, 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 the adversarial attacks, such as, FGSM. After the adversarial examples are generated, Surprise Adequacy is used to examine the corner cases. This corner cases have been assumed that will 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.

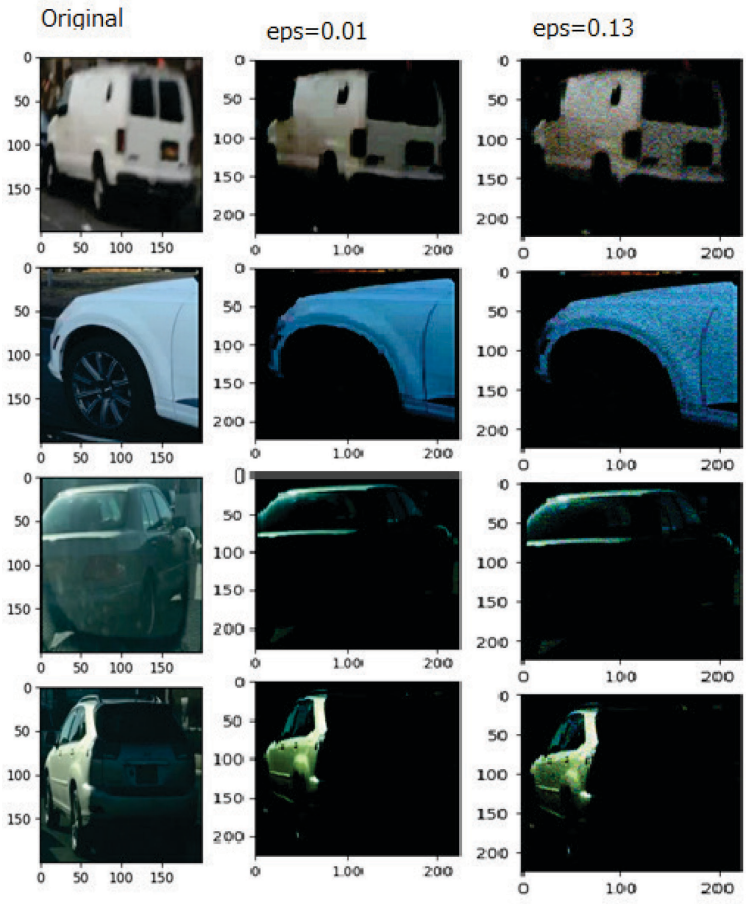


Figure 20: Data generated using FGSM attack on BDD100k dataset for eps= 0.01 & 0.13

7.6.4.5 Adapting surprise adequacy to work with MobileNetv2

We have talked about Surprise Adequacy, but we did not explain what it means and do. Surprise Adequacy 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** to follow the activation of the neurons. Let $N=\{n_1, n_2, \ldots\}$ be a set of neurons that constitutes a neural network N, and let $X=\{x_1, x_2, \ldots\}$ be a set of inputs. The activation value of a single neuron n with respect to an input x is defined as $\alpha_n(x)$. For an ordered set of neurons, let $N \subseteq \boldsymbol{N}$, $\alpha_N(x)$ denote a vector of activation values, each element corresponding to an individual neuron in N: the cardinality of $\alpha N(x)$ is equal to $|N|$. $\alpha N(x)$ is the Activation Trace (AT) of x over the neurons in N. Therefore, for the set of inputs X, we can define the Activation Traces as $A_N(X)=\{\alpha_N(x) | x \in X\}$.

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. In case you want to know more information, please read the following papers**Invalid source specified.**.

DSA has been defined using the Euclidean Distance between the AT of a new input *x* and ATs observed during training:

$$dist_a = \left\| a_{N(\mathrm{x})} - a_N(X_a) \right\|$$

$$dist_b = \| a_N(X_a) - a_N(X_b) \|$$

$$DSA(x) = \frac{dist_a}{dist_b}$$

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 1 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.

$$x_b = \underset{c_{x_i} \in \{C - c_x\}}{\operatorname{argmin}} \; x e^{-x^2} \| a_N(x) - a_N(x_i) \|$$

$$dist_b = \| a_N(x) - a_N(x_b) \|$$

DSA_2 compares the testing data *x* to data of all classes.

$$m = \frac{1}{k} \sum_{i=1}^{k} x_i \, , \{ x_i | c_{x_i} = c_s \}$$

$$dist_a = \left\| a_{N(\mathrm{x})} - a_N(m_a) \right\|$$

$$dist_b = \| a_N(x) - a_N(m_b) \|$$

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, ($c_b \in \{C - c_x\}$).

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:

$$m = \frac{1}{k} \sum_{i=1}^{k} x_i \, , \{ x_i | c_{x_i} = c_s \; \& \; x_i \in N_k(x) \}$$

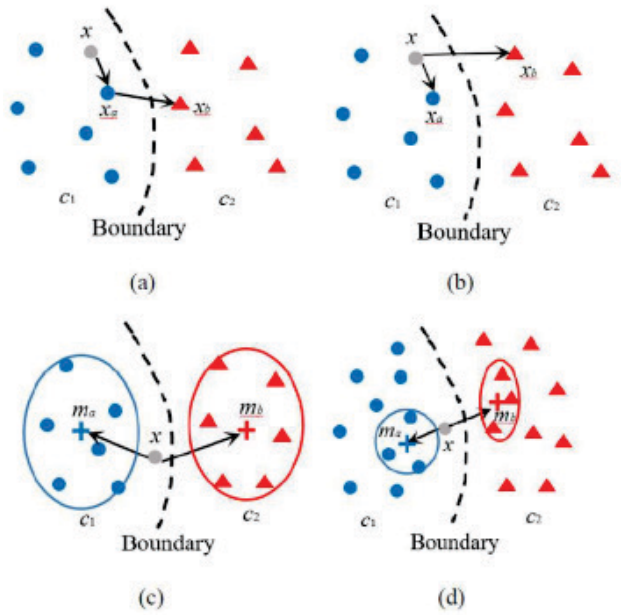


Figure 20: Diagram of four types of DSA: the original DSA_0 (a), DSA_1(b), DSA_2(c) and DSA_3(d).

For the experiments, we define that we want to remove from the BDD100K dataset the corner cases that has more than 1 in their distance. The idea is 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). The following graph shows the distribution of bounding boxes in the validation data that has been used for the Surprise attack:

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

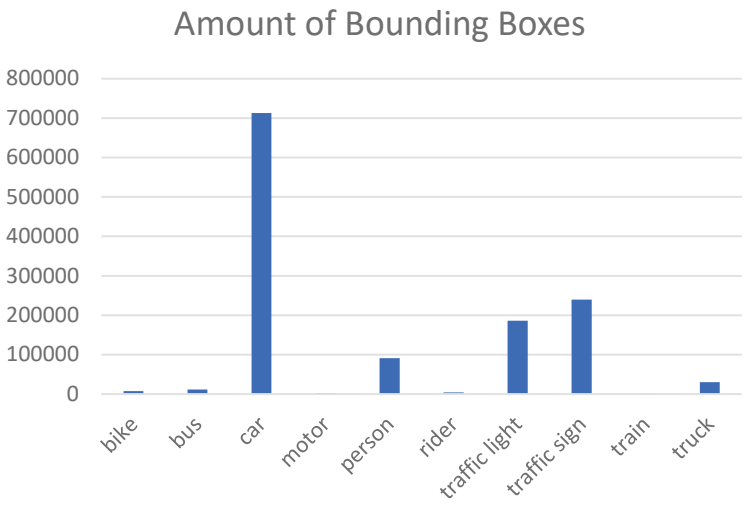


Figure 21: BDD100k label distribution

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.

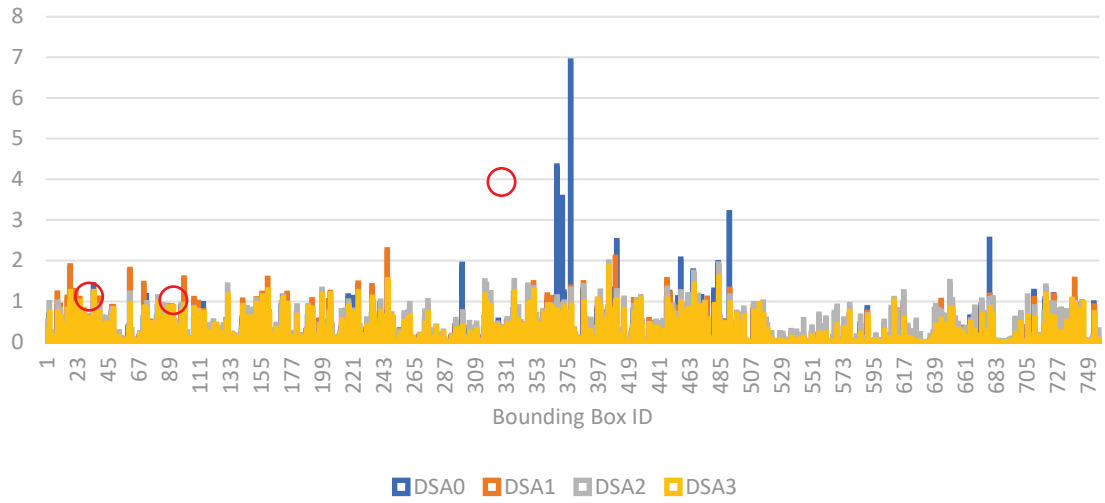


Figure 22: Surprise Adequacy calculation of the first 750 bounding boxes of bike in the training data

Figure 22 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. As an example, it has been marked with red circle 3 corner cases in Figure 22 and they are presented in Figure 23.

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 23: Bike corner case examples detected using Surprise Adequacy

As a result, checking all labels of BDD100k:

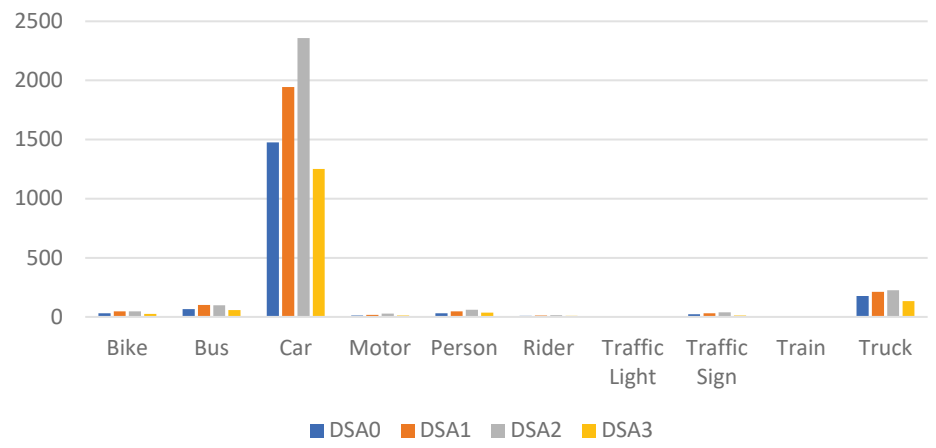


Figure 24: Amount of corner cases detected by labels

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.

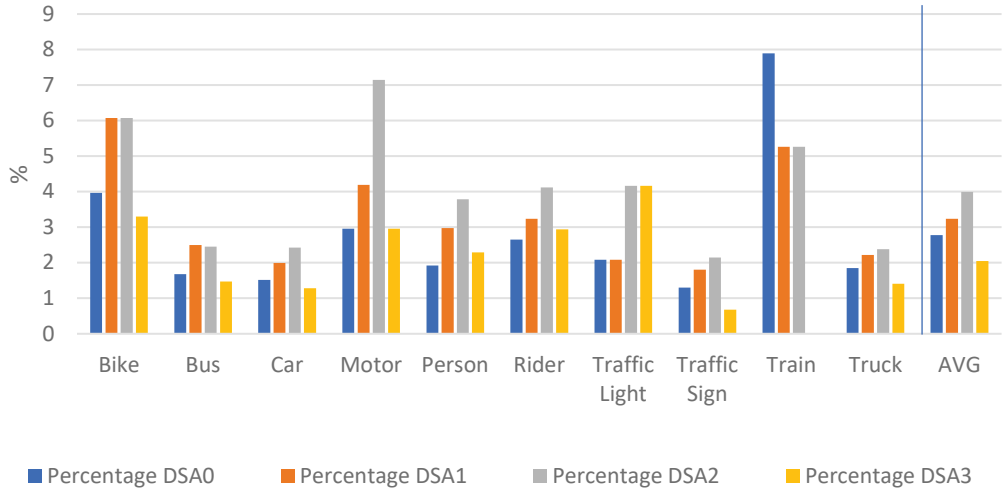


Figure 25: Percentage of corner cases in BDD100k dataset

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*. On overall, DSA_3 is not detecting very well corner cases for BDD100K, such as, it cannot detect any corner case for *Train* label.

In order to use the corner case detections, the solution designer trained 6 different MobileNetv2:

- 1 model per each DSA detection removing the corner cases detected in the training dataset
- 1 model removing all corner cases independently which DSA detected
- 1 model without removing any corner case of the training dataset

It has been used a MobileNetv2 that has been previously trained to identify images from Imagenet dataset. In this way, the training is faster and only the last layers need to be re-trained.

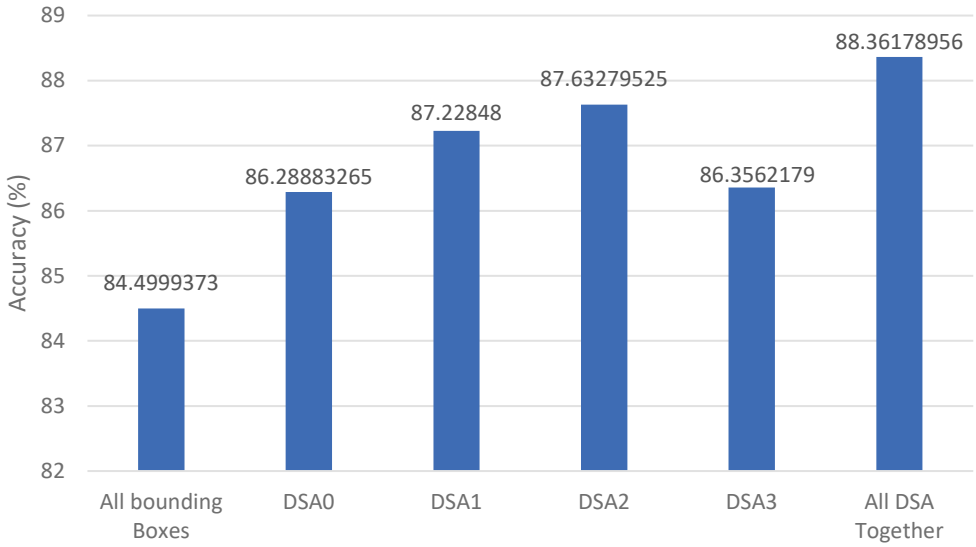


Figure 23 Model accuracy after removing corner cases using MobileNetv2

Multiple things to say about Figure 23. Without removing any image when training (*All images*), 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 more quantity of 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% regarding 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 (SADL D). For this reason, it is used Yolov4 to check if the corner cases detected by Surprise Adequacy and MobileNetv2 can be used for other detection methods.

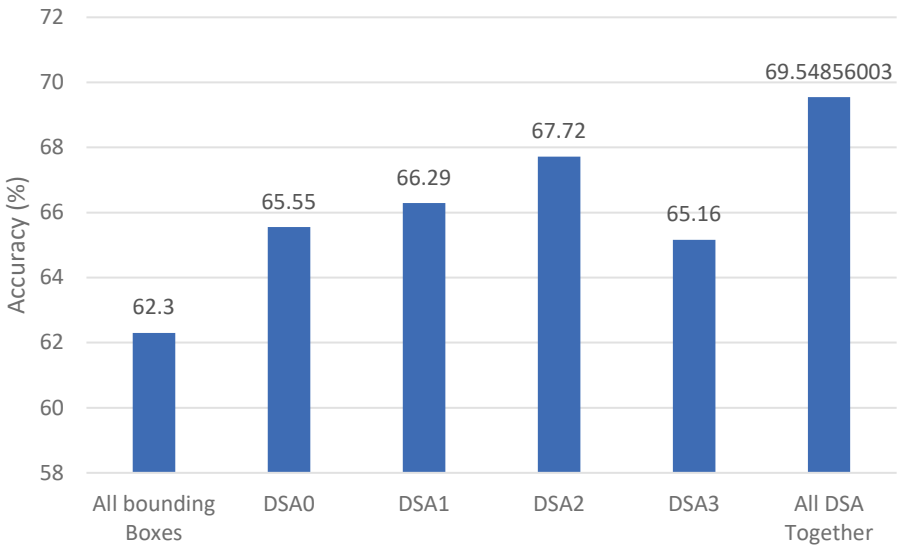


Figure 24 Model accuracy after removing corner cases using Yolov4

Figure 24 shows the result of removing the corner cases detected using MobileNetv2 from training dataset and then re-training Yolov4. As consequence, there is an increment of accuracy in all corner cases removed. The increment is similar to the one obtained on MobileNetv2 results. This demonstrates that Surprise Adequacy can detect corner cases using a different Neural Network and it can be used for another Neural Network as well. The removing of the corner cases increases the robustness of the dataset and the detection models as well.

It is necessary to remark 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 accuracy should be reduced due to the training process is losing information because the corner cases have been removed from the dataset. However, in BDD100k, removing the corner cases make the detection models to be more confidence when they are detecting cars. This definition is made because the boundary among *car* labels and others is removed.

7.6.4.6 1-Pixel Change

In the previous section it has been used FGSM attack to generate adversarial examples. However, in this section, it is presented a new adversarial example generator. 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

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{\frac{1}{p}}$$

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.

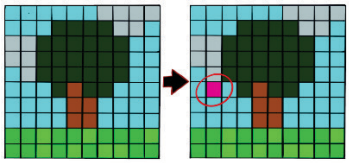


Figure 25: L_0 image example

2) L_2 distance measures the standard Euclidean (rootmean-square) distance between x and x' images. The L_2 distance can remain small when there are many small changes to many pixels.

Figure 26: L_2 image example

3) L_∞ distance measures the maximum change to any of the coordinates:

$$\|x - x'\|_\infty = max\left(|x_1 - x'_1|, \ldots, |x_n - x'_n|\right)$$

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.

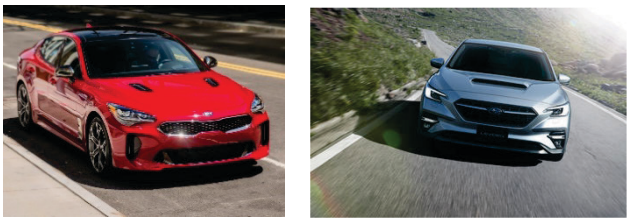


Figure 27: L_∞ image example

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

$$MSR(A, f) = max\{|A - A'|_p \, f(A) \; = \; f(A')\}$$

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.

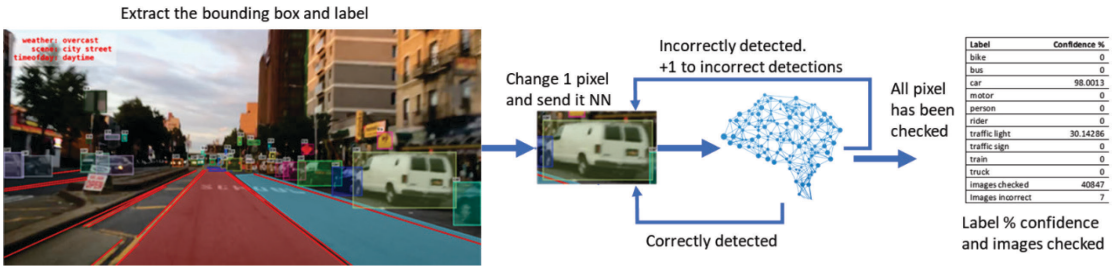


Figure 28 1-pixel attack method using BDD100k

Figure 28 shows how 1-pixel attack is working with BDD100k training dataset. First, each bounding box of the dataset is extracted. After, the extracted bounding box is treated as an image were the method modifies 1 pixel. Then, it is changed 1 pixel of the image, that is the bounding box only. So, the modified image is sent to the detection model to determine if the detection model 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.

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 29: 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, 1-pixel attack force the detection model to check 60,000 images gathering the results of the 60,000 images.

Using the bounding box of Figure 28 and Yolov4 as the detection model, the confidence percentage of image after using 1-pixel attack shows that the detection model recognizes the image as *car* or *traffic light*. However, when it 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, it has been checked 40,847 images and in 7 of them the detection models made an incorrect detection being only 0.017% affected by noise. As a result, this bounding box can support noise and is good to be used for training the model.

As an instance, if it is used Figure 29 by 1-pixel attack and Yolov4 as detection model, the detection model can recognize it as *truck*, *car*, *bus*, *motor* and *traffic sign*. The confidence of recognizing the image as *car* is bigger than 60% causing an incorrect training process for the detection model due to this image is a corner case between *truck* and *car*. In addition, if this image has noise, the system fails about 79% of the times to recognize it properly. As consequence, this image is a clear example of how 1-pixel attack can recognize corner cases and bad bounding boxes to be used for the training process.

1-pixel change can detect the images that are creating noise in the NN and detect the corner cases. However, it is a long process if the user wants to examine all the bounding boxes in the dataset. For example, for the experiments, it has been used the ABCI server (powerful server machine) for 8 hours. In that time 1-pixel attack achieves 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. As consequence, this 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, such as, retraining the models.

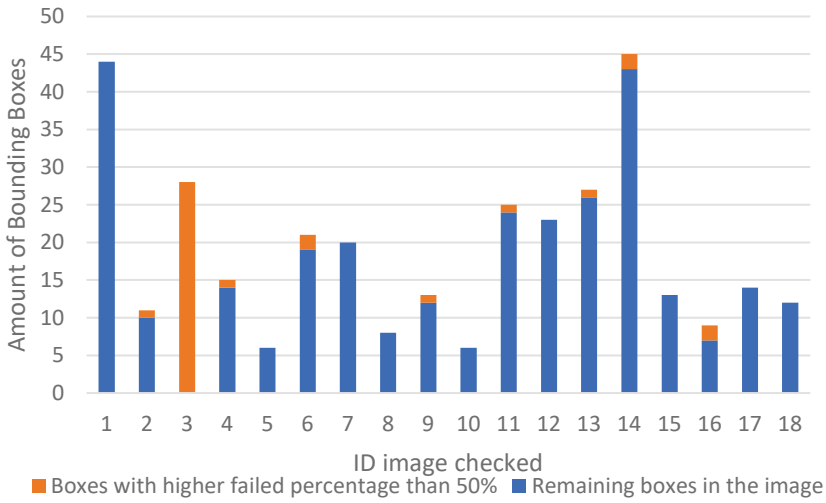


Figure 30: ABCI experiment using 1-pixel attack to detect incorrect bounding boxes

Figure 30 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 bad bounding boxes to be used for the training process. Additionally, it is highlighted that image 3 adds only noise to the detection model. Unfortunately, due to time, the solution designer could not check all bounding boxes in the BDD100k dataset and as consequence, it could not re-train the detection models removing the corner cases and check the accuracy obtained. Nevertheless, there are two ways to resolve this issue. One is to improve the algorithm, such as, adapting 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 after, use 1-pixel change to classify the corner cases detected. In this way, 1-pixel change only checks a few bounding boxes compared to check all bounding boxes and prioritize the corner cases that has worst failed percentage. Additionally, using 1-pixel change with Surprise Adequacy helps to determine which labels are close to the corner case thanks of using the confidence obtained on 1-pixel change examination.

7.7 Dependability of underlying software system

7.7.1 Definition of 'Dependability of underlying software system':

In the MLQ guideline, the term "dependability of underlying software system" 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:

- 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

7.7.2 Correctness of algorithms

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

For example, in this research, some additional algorithms were designed for various purposes. Such as- robustness evaluation (1-pixel change), data annotations (annotation tool) and data label conversion (the conversion of NuScene 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 produced expected outputs with no error or warning involved.

7.7.3 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.

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

Table 20: List of open-source packages used in this research

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.

7.7.4 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 the use of 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. At this current state, all the research has been made using testing environments and not operational environments that is the hardware objective. For now, the testing environment has been set to be similar. However, hardware requirement should be considered in dependability assurance.

7.7.5 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: AI developers generally use Hierarchical Data Format (HDF) file (.h5) to store trained AI networks and their weights. 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 hard drive compared to data and trained weights.

Input data: Memory required for the training and the validation dataset can be very large depending on number of samples and 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 came 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.

7.7.6 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 split seconds after correctly identifying a life-threatening test case. So, the lower the inference time, the better the dependability of the AI product. Inference timing should be recorded for all the candidate models and evaluation should be done to check if the inference procedure is taking minimum 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.

7.8 Maintainability of quality during operation

7.8.1 Definition of 'maintainability of quality'

According to section 7.8 of the reference guide, maintainability of quality is the process to update the system developed or incorporate new data to improve the detection models either for the training or validation dataset. As consequence, 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 beginning of de development is maintained throughout the whole research project. There are two operational patterns to update the system developed. One is to return to the development phase and change the whole process adding the new changes and deploying the whole research from the development phase. The other one is to update the necessary software components to keep update the system. So, update the necessary components only is shorter than the other pattern but the risk of getting incorrect dependencies and worst detection models is higher.

Regarding the reference guide, there are four tasks of monitoring the maintainability:

1. Accuracy monitoring
2. Model output monitoring
3. Input data monitoring
4. KPI monitoring

7.8.2 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.

7.8.3 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 mechanism: 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.

7.8.4 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 is necessary to reply 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, as consequence, the solution designer has decided to use a standard definition of security levels when defining autonomous driving machine learnings. In particular, it has being selected ASIL (Automotive Safety Integrity Level) more concrete ASIL 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, it has been used the following attributes from BDD100k:

- **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.

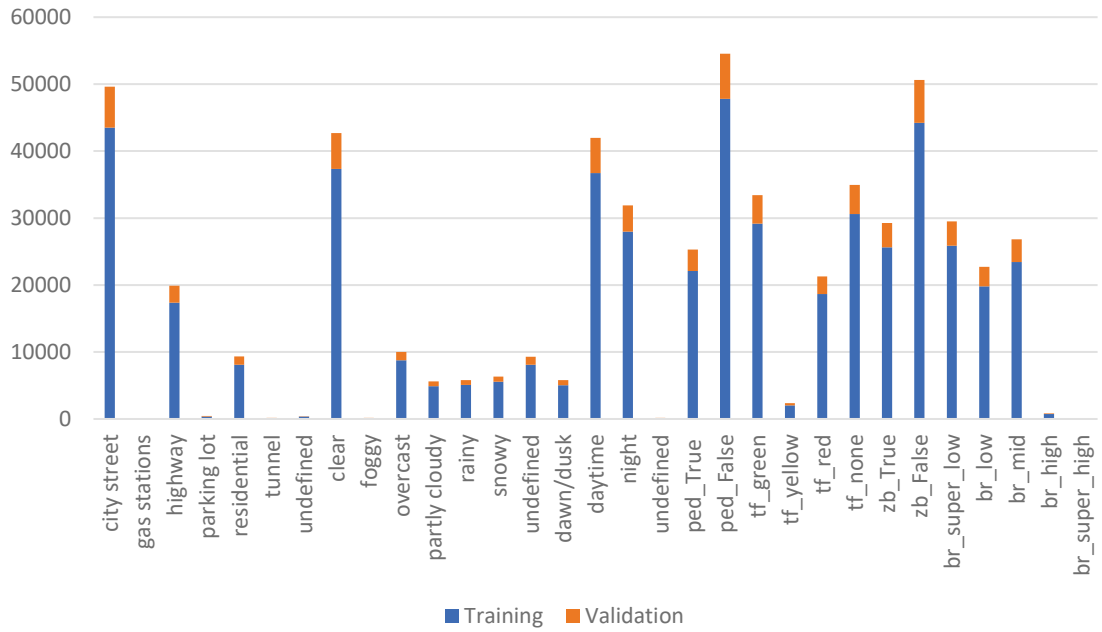


Figure 31: Distribution of Images per attribute values and per training or validation data of BDD100k

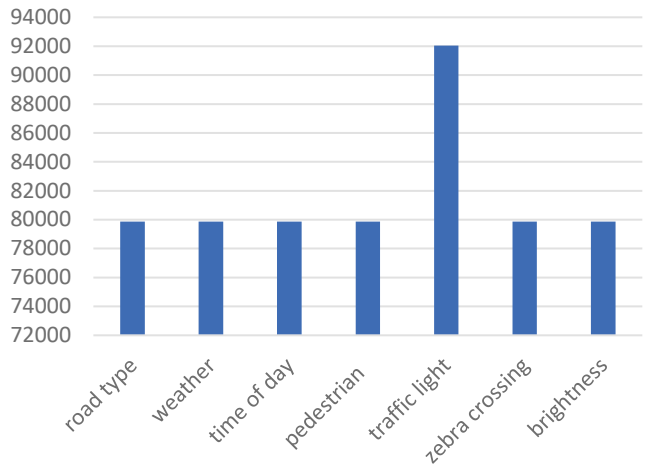


Figure 32: Number of images per attributes on bDD100k

As you can see, there are attributes that appear more 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 if we combine the attribute values in pairs. Doing this, we have 375 combinations.

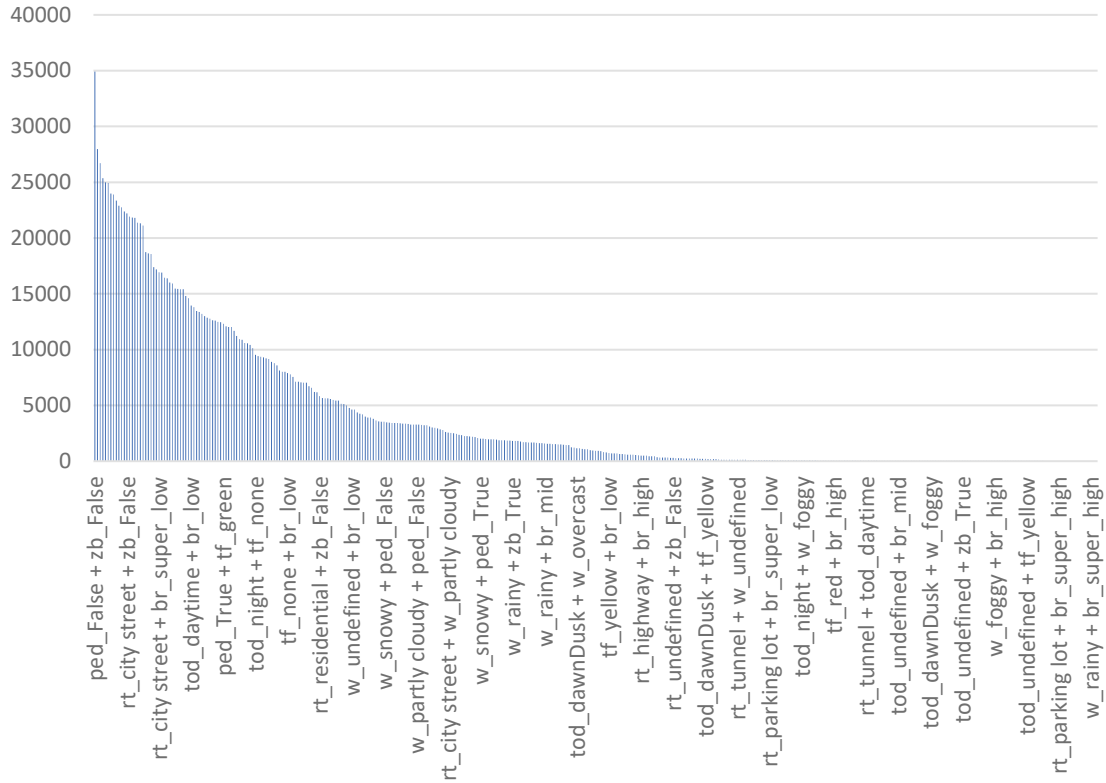


Figure 33: Amount of images per pair-wise combination of BDD100k

Because it is difficult to correctly see the chart before, let's focus on the first 30 combinations:

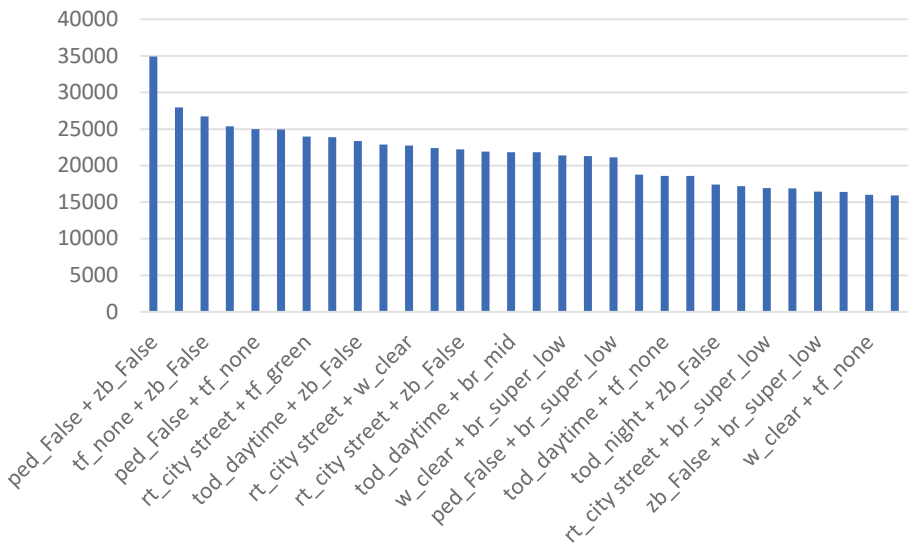


Figure 34: Top 30 amount of images per pair-wise combination of attribute values of BDD100k

Due to there is 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.

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			

Table 21: Amount of pair-wise images when road type, weather and time of day are combined

There are situations that are impossible to know or are unrealistic, such as, inside a tunnel know the weather is partly cloudy (number 63). 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:

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

Table 22:Label accuracy example of the pair-wise combination of residential and foggy attribute values

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 in 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**.

7.8.5 Example: KPI - Residential + foggy

Figure 35: An example of Residential + Foggy attributes

Figure 35 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.

- 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?
    - It is important to achieve ASIL D and avoid any health problem from failure detection.
- 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 increment of accuracy is better than 60% in all labels or at least in almost all of them.
    - Check the detection model trained using images from other datasets, such as NuScene.
- How often will you review progress towards the outcome?
    - Every time a new KPI is check it.
    - 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.

Glossary:

Below is a list of technical terminology from the protocol that is described in the following section.

For definitions not mentioned here, section 2.3 of the MLQM guideline can be checked.

a) *object detection* - The task of Identifying the presence and location of an object in an image.

b) *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.

c) *dataset* - A set of images and annotations used to train and validate a machine learning algorithm.

d) *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.

e) *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. 7

f) *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.

g) *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), and False Negatives (FN, non-existing predictions for existing ground-truth objects).

h) *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 = $TP(TP + FN)$.

i) *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.

j) *FPS (frames per second)* - The number of images that a machine learning algorithm can evaluate in one second.

k) *KPI (key performance indicator)* - A specific target mAP that should be achieved by a machine learning algorithm.

l) *contender model* - A machine learning algorithm that is being considered as an object detection method for an automated driving system.

m) *reference model* - A machine learning algorithm that is a state-of-the-art object detection method.

n) *candidate model* - A machine learning algorithm selected for development from a set of contender models.

Appendix:

Automotive Safety Integrity Level (ASIL):

ASIL may be similarly expressed as

$$ASIL = Severity * (Exposure * Controllability)$$

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

References

Bochkovskiy, A. a.-Y.-Y. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Caesar, H. a. (2019). nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*.

He, K. a. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770--778.

Howard, A. G. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv*.

Jocher, G. a. (2021). ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights \& Biases logging, PyTorch Hub integration. *Zenodo*.

Liu, S. a. (2019). Learning spatial fusion for single-shot object detection. *arXiv preprint arXiv:1911.09516*.

Redmon, J. a. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S. a. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 1137--1149.

Sandler, M. a. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv*.

Simonyan, K. a. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C. a. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818--2826.

Tan, M. a. (2020). Efficientdet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10781--10790.

Yu, F. a. (2020). BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu, F. a. (2021, 2 22). Retrieved from https://github.com/bdd100k/bdd100k

Zhao, Q. a. (2019). M2det: A single-shot object detector based on multi-level feature pyramid network. *Proceedings of the AAAI conference on artificial intelligence*, 9259--9266.

Reference for machine learning based Vision Inspection

Contents

1 Purpose and background

In manufacturing factories, the vision inspection is a significantly meaningful topic. Defects are an unwanted thing, especially on the appearance of products or parts. For removing this defective product all industry have their quality inspection department. But the main problem is this inspection process is carried out manually. It is a very time-consuming process and due to human accuracy, this is not 100% accurate. This can because of the rejection of the whole order. So it creates a big loss in the company. Therefore, there is a growing need for using machine-learning based methodology for the automated inspection. Here, the goal of this report is to create an example implementation to demonstrate how the Artificial Intelligence Quality Management (AIQM) guideline can be used to assess interior properties of AI models on vision inspection. Here we will discuss about vision inspection problem based on some publicly available industrial datasets. This report can be used as a reference document for other vision inspection applications involving evaluation of AI based systems.

2 Expected outcomes

In this reference, the following outcomes are expected to be achieved via applying the AIQM guideline on vision inspection applications

- Procedures of quality control referring to vision inspection tasks.
- Methodology on AI modeling and testing on vision inspection
- Useful AI algorithms and KPIs in vision inspection applications
- Reference on AI quality assurance for machine-learning based systems in vision inspection

3 Author's role

Here, we play as a 'service developer'. We consider the guideline as a 'technical starting point' for developing the service in vision inspection. The essential intention of this report is to introduce a total investigation of the vision inspection issue and answer for it as indicated by the predefined AIQM appraisal standards. As a creator of this model issue, our objective will investigate each part of it and attempt to arrange them on the characterized structure of the rule. It will help us to analyze the vision inspection problem and finding some incite from this problem. We follow the guideline from early development stage to ensure quality standards are maintained throughout the process.

4 Introduction of possible datasets

There are a number of vision inspection applications in real world, which could usually provide the public usage in website. Here, the report presents brief description of two datasets in hand for vision inspection, as well as some example samples as below

❖ Example dataset 1

Magnetic Tile Defect dataset (https://github.com/abin24/Magnetic-tile-defect-datasets) is one dataset used for defect detection, and its related AI model training and evaluation are also discussed. The Magnetic Tile Defect dataset contains totally 2,688 static images, as shown in Figure 4.1. For each image, its category label and the region of defect are both provided, and the defect regions are shown in images named as GroundTruth "GT" in Figure 4.1.

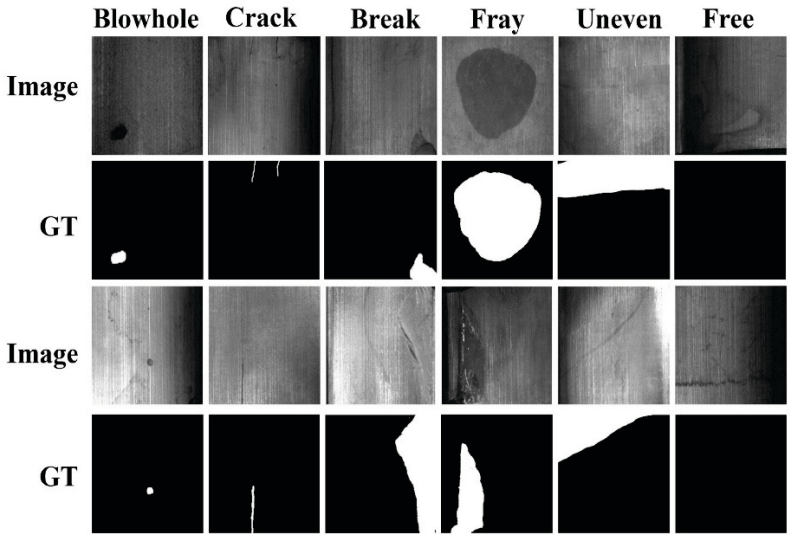


Figure 4.1. Examples of Magnetic Tile Defect data

In Figure 4.1, there are totally 6 categories assigned, including “Blowhole” (230), “Crack” (114), "Break" (170), "Fray" (64), "Uneven" (206), and "Free" (1904). Among them, only the images belonging to "Free" are normal, and the images belonging to the other five categories are determined as defect data.

- Example dataset 2

Casting defect is an undesired irregularity in a metal casting process. There are many types of defect in casting like blow holes, pinholes, burr, shrinkage defects, mould material defects, pouring metal defects, metallurgical defects, etc. The Casting Defect data is from Kaggle (https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product). The dataset contains total 7,348 image data. These all 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 contains 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

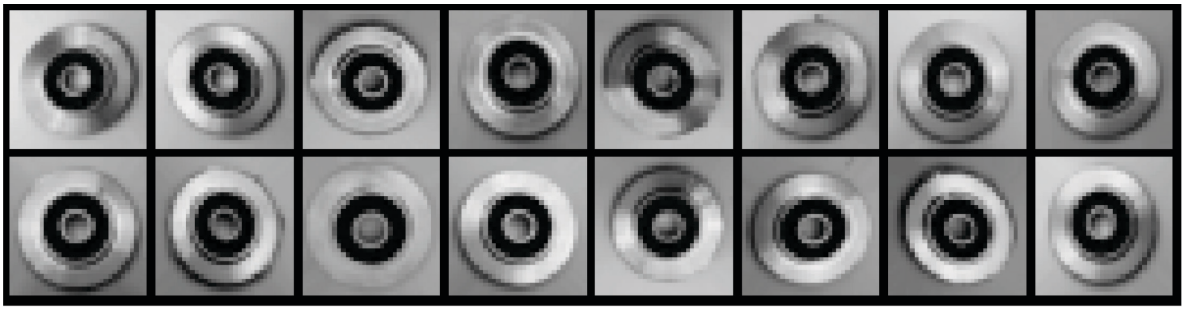


Figure 4.2. Examples of Casting Defect data

5 Quality assurance procedures using AIQM guideline

After initial investigation of these current datasets, the report explores eight internal properties in quality management of AI models for vision inspection applications, as mentioned in the AIQM guideline.

5.1 Sufficiency of requirements analysis

5.1.1 General definition

- **What is meaning of this specific AI quality?**

"Sufficiency of requirements analysis" means that usages of a target machine learning based system are analyzed sufficiently and every requirements for the system is captured (described in Section 1.7.1).

Requirements analysis is important especially in the early stage of development of a conventional software which is used for a safety application. The main purpose of requirements analysis for the development of machine learning based systems in this guideline are as follows.

① Sufficient identification of cases in which risk management is needed, mainly in applications required "safety".

② Sufficient identification of attributes of objects for which inequality is not allowed, mainly in applications required "Fairness".

③ Sufficient analysis of real world in order to verify that training datasets and test datasets are comprehensive and appropriate extractions of the real world, to all requirements including "AI performance".

"Sufficiency of requirement analysis" is always required not only for machine learning based systems but also equipment and services controlled by software.

5.1.2 Contents in vision inspection

- **What should be done for the developer in the evaluation process**
  - Problem formulation;
  - Problem domain definition;
  - Problem evaluation;

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

5.1.3 Possible approaches and experiments

The two goals of this internal quality aspects described in this paragraph are

- Clarifying what is required for machine learning components; and
- Clarifying the limited scope which machine learning components have to deal with.

Combining the definition of "sufficiency of requirement analysis", the detailed contents in vision inspection for requirement analysis can include the following aspects.

5.1.3.1 Product specification

This paragraph should include requirements from development entrusted for the machine learning based vision inspection systems (products). These requirements can be stated from the technical requirements in details:

- Type of learning: Supervised learning
- Machine learning components: Segmentation, Classification
- Available AI models: UNet, VAE, Fast RCNN, CNN, MLP, ResNet and etc.
- Task to perform: Defects detection and classification
- KPI specifications: Precision, Recall, Accuracy, and etc.

❖ Example 1

If taking the Magnetic Tile Defect data for vision inspection, the related problem can involve segmentation and classification. This detailed machine learning components is depended on the selection of data used in modeling. That is to say, the final machine learning based systems can be determined based on the requirements of product.

For example, if taking the original defect images as the inputs, and the groundtruth images as the outputs, a segmentation model for detecting defects in magnetic tile images can be constructed. The selected machine learning model can be chosen as UNet, VAE, R-CNN and etc.

If taking original defect images as the inputs, and the defect types as the outputs, a classification model for recognizing the defect types of magnetic tile defect images can be constructed. The selected machine learning model can be traditional CNN models, VGG or others.

Then, the KPIs like Precision, Recall and Accuracy can be used to evaluate the performance of constructed machine learning based models in Magnetic Tile Defect detection.

❖ Example 2

If taking the Casting Defect data for vision inspection, the related problem only involve 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 model like CNN, MLP, ResNet and etc. can be applied for vision inspection on Casting Defect data. The KPIs like Precision, Recall and Accuracy are also useful for evaluating the constructed AI models' performance.

5.1.3.2 Problem domain

If product specification describe the requirements on model/algorithm selection in machine learning based vision inspection systems, the problem domain describe the requirements on data selection.

- **Problem domain design**

Inspired from the guideline, the final machine learning based systems are required to be applied in vision inspection for all possible situations. Actually, these situations can be described by some characteristics in natural language, also called 'attributes' in the document. Based on these description attributes, we can envision a concept of feature tree according to their corresponding 'attribute values', then define a problem domain describing all of these situation involved in vision inspection. The term used here as 'attribute' and 'attribute values' carries the same meaning and significance as defined in section 6.1.2.1 of the guideline.

❖ Example 1

Taking the Magnetic Tile Defect data for example, we can declare some attribute requirements to design the vision inspection problem domain. Attributes and attribute values can be defined from the following points of view

- Defect type: Blowhole, Crack, Break, Fray, Uneven, Free
- Light intensity: strong, medium, weak
- Reflection strength: strong, medium, weak

According to the above requirements, we can present some results for showing this problem domain

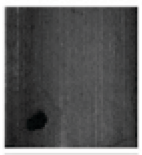
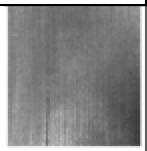
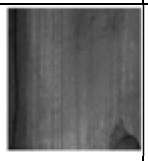
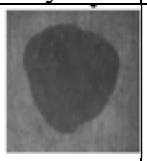
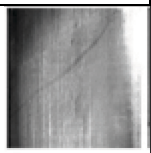
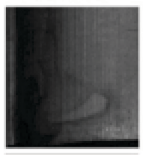
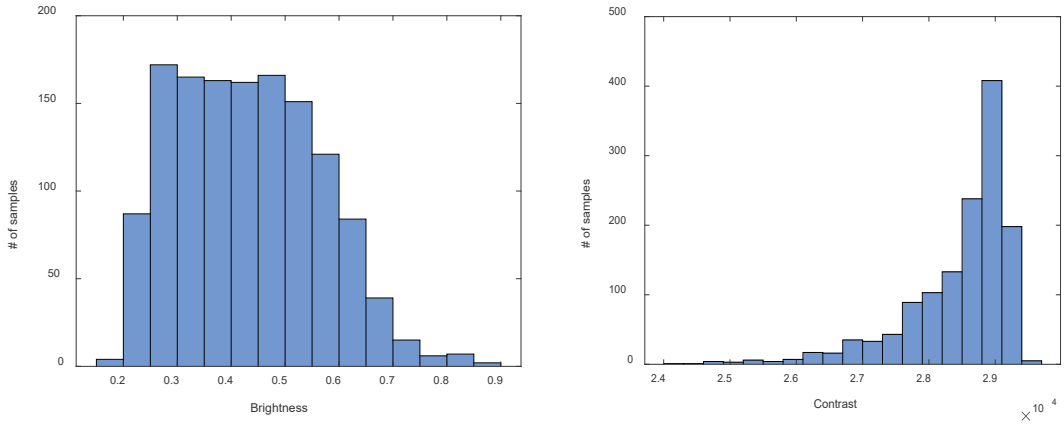
Type	Blowhole	Break	Crack	Fray	Uneven	Free
Image						
#	115	85	57	32	103	952

Figure 5.1.1. Images and number of each defect type in the dataset

Moreover, the light intensity can be represented by transforming the RGB format images into HSL format images where the value of L represents the lightness/brightness, as shown in Figure 5.1.2(a).

Considering the recognition of reflection strength is not easy, here we can consider another quality of image, namely image contrast to replace the reflection strength, as shown in Figure 5.1.2(b).



(a) (b)

Figure 5.1.2. Domain of two description attributes. (a) light intensity (Brightness); (b) reflection strength (Contrast).

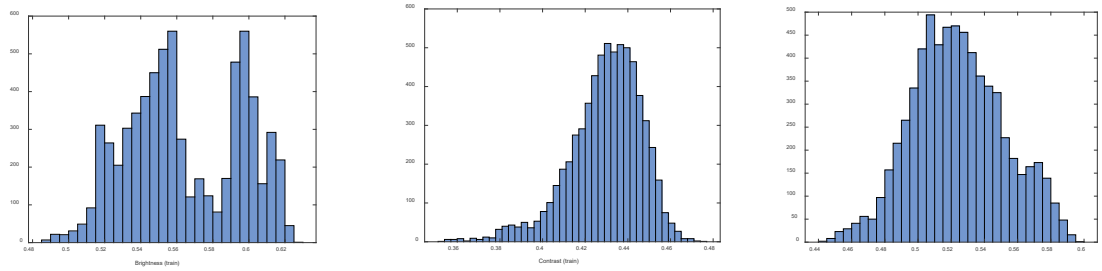
Since the values of brightness and contrast are continuous, then we can give thresholds to divide their domain into three parts according to real vision inspection requirements, and corresponding to the strong, medium, weak cases of light intensity and reflection strength respectively.

❖ Example 2

If taking the Casting Defect data for example, we can also declare similar attribute requirements as the following points

- Type: Defect, Free
- Brightness: strong, medium, weak
- Contrast: strong, medium, weak
- Exposure fusion: 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 similar attributes like brightness, contrast and an additional attribute exposure fusion. The domain of these attributes are shown in Figure 5.1.3.



(a) (b) (c)

Figure 5.1.3. Domain of attributes for the Casting Defect data. (a) Brightness; (b) Contrast; (c) Exposure fusion.

Similarly, three attributes in Figure 5.1.3 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 5.1.1. Problem domain of the given attributes

Attribute	Brightness	Contrast	Exposure fusion
Domain	0.4~0.7	0~0.5	0.2~0.7

- **Data type requirements**
  - Type of inputs for AI modeling
  - Type of attributes for problem domain description

Furthermore, data type is also an important requirement for machine learning data. The first item is the type of inputs for AI modeling. Different with attributes for problem domain description, the inputs for AI modeling should have concrete values and types (discrete or continuous).

- For example, the AI based vision inspection system for the Magnetic Tile Defect data, it requires inputs are 3- channels RGB format images, the targets should be 1-channel binary format groundtruth images and discrete defect type labels.
- For the vision inspection system for the Casting Defect data, it requires inputs are 1-channel grey format images, the targets are binary class labels representing defect or not.

The second item is the type of attributes for problem domain description. This requirement is more general, 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.

5.1.3.3 KPI requirements at 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 vision inspection systems. These KPIs can be expected from perspectives of models, data and performance. For example, taking the Magnetic Tile data for vision inspection, the following policies can been formulated by the developers.

- The inputs should contain only those whose images has been identified as "Crack", "Break", "Fray", "Uneven" and "Free", and that images belonging to "Blowhole" will not be used.
- The training dataset consists of 1230 images, or about 50% of each image in the above categories, and the test dataset consists of 245 images, or about 10% of each image in the above categories.
- The Variational Auto Encoder (VAE) model can be considered as its learning tool.

- As a completion condition of the POC trial phase, we defined Precision and Recall to achieve 0.7 or greater for each of the test data sets.

According to these requirements, some experiments based on the constructed vision inspection model are implemented. First, by taking 50% of data for training, namely 672 samples, and 10% of data for testing, namely 134 samples, the performance of testing data is shown below

Table 5.1.2 Results of Precision and Recall of testing data

(%)	Blowhole	Break	Crack	Fray	Free	Uneven
Precision	72.73	83.33	100	100	92	90.91
Recall	72.73	83.33	62.5	50	97.87	76.92

It is seen that two cases are not satisfied the requirement that Precision and Recall values should exceed 70%.

On the other hand, by setting 70% of data as training data, namely 940 samples, 20% of data for testing, namely 268 samples. The performance of testing data on the trained model is shown below

Table 5.1.3 Results of Precision and Recall of testing data

(%)	Blowhole	Break	Crack	Fray	Free	Uneven
Precision	94.74	85	100	83.33	93.88	100
Recall	81.82	80.95	80	100	98.40	83.33

Conclusion: With more data for training and testing, the precision and recall can be improved, exceeding 70% on all cases, satisfying the requirements in the POC phase.

5.1.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Examine and record the cause of major risks of quality deterioration.
- Based on the examination results, design data and reflect it in necessary attributes.

5.2 Sufficiency of data design

5.2.1 General definition

- **What is meaning of this specific AI quality?**

On the premise of the sufficiency of problem domain analysis, sufficient examinations of data design to collect and sort out sufficient training data and test data are required as "sufficiency of data design" in response to various situations which systems have to respond to.

In an extremely-simple system, it is enough to mention that corresponding data to all "combinations of situations" identified in the problem domain analysis described in the previous section included in training datasets and test datasets. However, if a situation in which a system is envisioned to be used is complex, the number of possible combinations becomes enormous. Therefore, it is not practical to cover all combinations with datasets.

5.2.2 Contents in vision inspection

- **What should be done for the developer in the evaluation process**
  - Combination analysis;
  - Case soundness analysis;
  - Specific cases;

For vision inspection applications, the quality "sufficiency of data design" involves three aspects. For example, firstly it involves the combinations design which is on the basis of described vision inspection in the previous section. The combinations are realized by attributes, and lead to cases (situations) division 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 vision inspection. Finally, this quality deals with the analysis of high-risk cases, e.g. data samples in special situations may lead to incorrect defect recognitions or false defect detection.

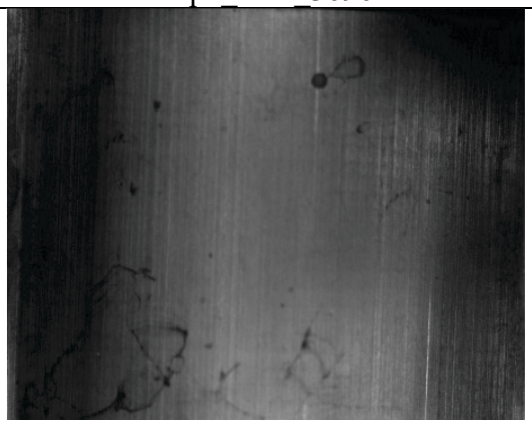
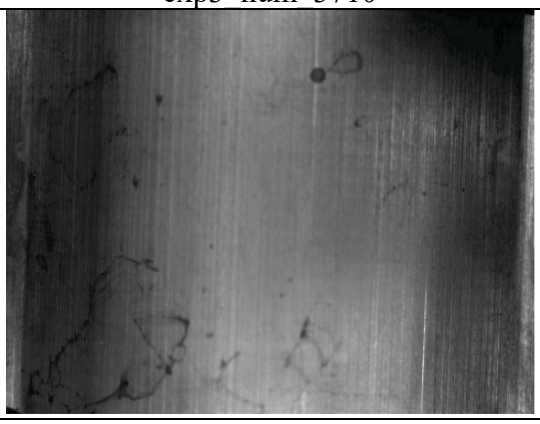
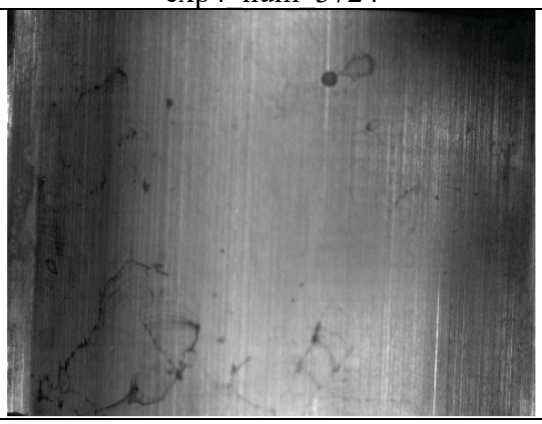
5.2.3 Possible approaches and experiments

According to the definitions of the quality "sufficiency of data design", we can figure out some related contents and techniques for vision inspection, such as data augmentation to guarantee the combination analysis, attribute combination and case division, risk cases and related analysis.

5.2.3.1 Data augmentation

In order to secure sufficient training data and test data with respect to various situations, the vision inspection systems requires to respond to enough amount of data in the defined problem domain.

For example, in the vision inspection application based on the Magnetic Tile data, the low amount of samples is a key problem affecting the performance. Instead of collecting data from real scenarios, data augmentation is an available technique to satisfy the quality analysis of "sufficiency of data design". Here, on the basis of the first AI quality analysis on problem domain requirement, it will be possible for the designers to generate some realistic data for vision inspection. Firstly, we need to explore the problem domain of the original Magnetic Tile data.

exp1_num_3667	exp2_num_3696
	
exp3_num_3710	exp4_num_3724
	
exp5_num_3746	exp6_num_3789

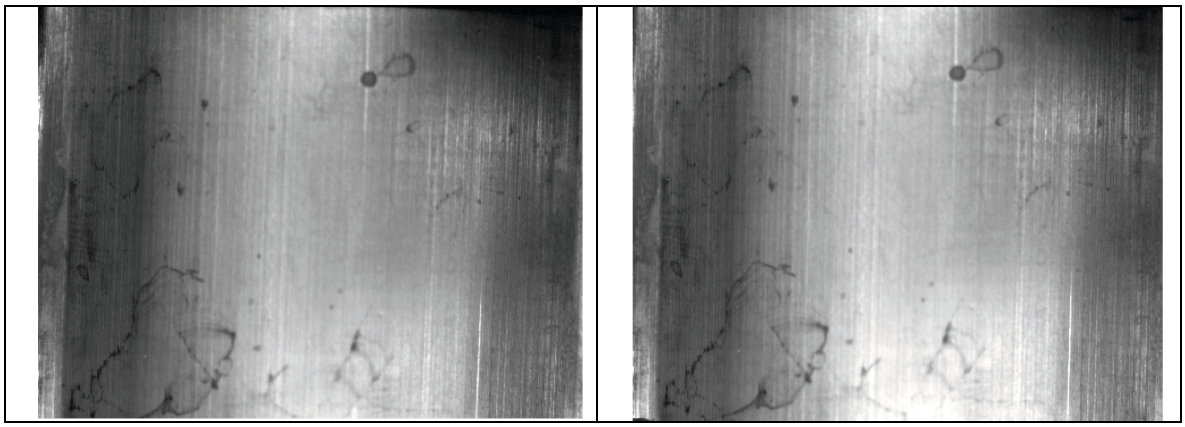


Figure 5.2.1 Examples from the original dataset

For example, Figure 5.2.1 shows some real image from the original dataset. It is found that six figures in class of Blowhole are actually collected from one object, but with different description attribute, namely the brightness. This phenomenon implies that it is meaningful to create some new images on the given problem domain. For example, given a sample, then changing its description attribute within the problem domain, the new generated samples will certainly belongs to the problem domain for vision inspection. Here, we apply four kinds of transformation to generate realistic data in the problem domain of the Magnetic Tile Defect data, such as rotation, transpose, and brightness adjustment to realize data augmentation.

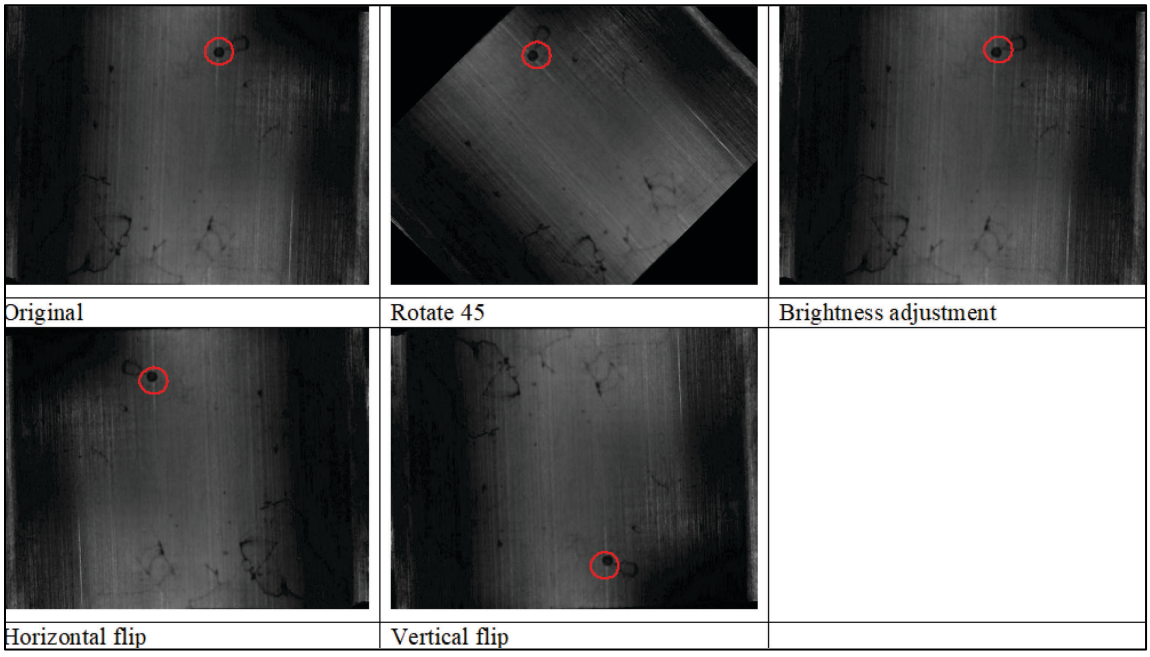


Figure 5.2.2 Examples of data augmentation

The above show images generated based on a given blowhole defect image, which are transformed by 45° rotation, horizontal flip, vertical flip and brightness adjustment in the given problem domain. Similarly, all original images can be used to generate new realistic samples. By implementing the same transformation as the above, totally 5376 new images are generated, their distribution are presented as below

Table 5.2.4. Data distribution after data augmentation

Type	Blowhole	Break	Crack	Fray	Uneven	Free
#	575	425	285	160	515	4760

5.2.3.2 Attribute combination and case division

The second content in the study of "sufficiency of data design" could be the design of data cases based on attribute combinations. As we know, the problem domain of vision 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. For example, the Magnetic Tile Defect data, the "Blowhole" images may be excluded in the PoC phase since its defect regions are too small compared with images of other defect types. 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.

Assuming we divide the problem domain for each attribute (brightness, contrast, exposure fusion) as 10 sub-cases. The cases with different attribute combination are summarized as below.

Table 5.2.5. Case division with different number of attribution combinations

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

5.2.3.3 Corner cases/high-risk case

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, so

these cases are more refined in data description. Moreover, we can further find those high-risk or unsound data region, namely unsound cases on the basis of case division.

- Unsound cases/High-risk cases

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

High-risk cases are different with the unsound cases, they are possible to exist in the real world for vision inspection. For example, in the Magnetic Tile Defect data, the "Blowhole" defect data is reasonable in the vision inspection, however they may lead to high risk of incorrect decision-making. Therefore, we can regard them as high-risk cases, and may be excluded in the PoC phase.

❖ Example 1

Here, taking the Magnetic Tile data for example, if we take two attributes "defect type" and "brightness" for pair-wise analysis, we first can divide the original data into different cases. By setting the threshold of brightness as 0.3 and 0.6, the brightness domain can be divided into 3 cases, and multiple 6 classes of defect types. There as totally 18 cases. Results are presented below

Table 5.2.6. Case division and soundness analysis

brightness	Blowhole	Break	Crack	Fray	Uneven	Free
Weak(<0.3)	21	13	12	11	18	188
Medium(0.3~0.6)	81	61	39	19	67	661
Strong(>0.6)	13	11	6	2	18	103

According to this division, the cases of Crack defect with strong brightness, and Fray defect with strong brightness seem not sound, since they have very low amount for training and testing.

Moreover, with more attribute combinations, we can also count the unsound cases which has no images, or define high-risk cases based on given threshold. Meanwhile, by setting a given accuracy threshold, we can calculated the number of valid cases and the valid ratio, as shown in the table.

Table 5.2.7. Unsound cases in the Magnetic Tile data

	Unsound cases (#=0)	# of cases (Acc>=0.97)	valid coverage (Acc>=0.97)	valid coverage (Acc>=0.96)
Com1	3/30	18/30	0.6667	0.8148(22)
Com2	170/300	94/300	0.7231	0.8077(105)
Com3	895/1000	79/1000	0.7524	0.7905(83)

- Corner case data detection

As well as the execution of traditional software, a dangerous condition in AI system testing is processing data of corner cases which generally cause incorrect and unexpected behaviors. For example, when DL-based autonomous driving system processes corner cases of rainy weather or strong reflection, an incorrect decision may be made to lead to crash bringing the loss of life and property. Therefore, detecting corner-case samples is important in AI testing. According to the above description of corner case, we can define the corner case set as the following

$$Corner\ case\ set: \{x |\ DL(x + pertubation) \neq label(x)\}$$

Where, x is denoted as 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 into a corner-case data x, where $|pertubation| < \varepsilon$ and $\varepsilon > 0$ is a small value, the class recognized by DL system will be different with its true label. In this way, 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 5.2.3

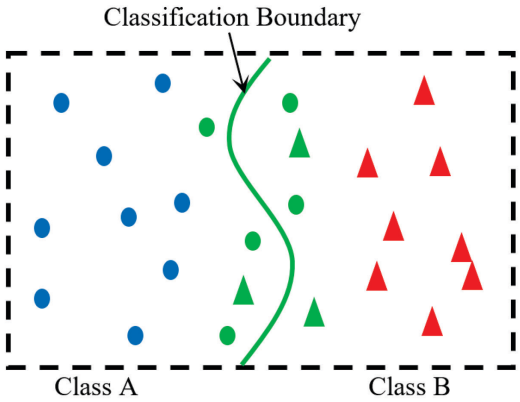
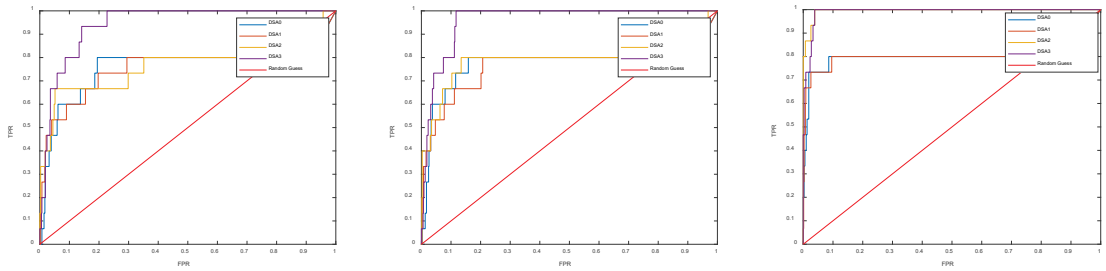


Figure 5.2.3. Diagram of corner case. Two class of data are colored as blue and red. Data of corner case is colored as green, which includes incorrectly classified data as well as data close to classification boundary which is sensitive to cause unexpected recognition.

❖ Example

By taking the Casting Defect data as an example, we can apply some corner case descriptors to detect the corner case data in the original data. For example, here a general CNN model with 2 convolutional layers and 2 full-connection layers is constructed for vision inspection. After modeling and training, the final testing accuracy can reach 97.90% on this casting data. Then, applying distance-based surprise adequacy (DSA) as the corner case descriptors, corner case data detection in the original data is implemented.



(a) (b) (c)

Figure 5.2.4. ROC curves of using DSAs on corner case data detection.

5.2.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Set cases for each of attributes corresponding to major risk factors.
- Set cases corresponding to combinations of composite risk factors.
- Extract attributes of differences in particularly-important environmental factors

5.3 Coverage of dataset

5.3.1 General definition

- **What is meaning of this specific AI quality?**

The term "coverage of datasets" means that a sufficient amount of data is given to cases covered by establishing the standards without any part being missed or overlooked in response to possible input corresponding to those cases.

When conventional software is developed, the details of all characteristics in real world which software operations depend on are captured at any stage from the machine learning

problem domain analysis phase to the implementation phase and reflected in software components in the form of conditional branching or calculation formula.

5.3.2 Contents in vision inspection

- **What should be done for the developer in the evaluation process**
  - Training and testing data coverage;
  - Case coverage;

When constructing a machine learning based vision inspection system, coverage of data contains two perspectives, such as training data coverage and testing data coverage. According to the general coverage definition, training data coverage aims to guarantee enough data in the problem domain for training, and that no inappropriate learning behavior occurs in vision inspection process due to lack of data. The purpose of testing data coverage aims at to evaluate the behaviors of the constructed vision inspection systems in the problem domain as completely as possible.

5.3.3 Possible approaches and experiments

According to the definition of "coverage of dataset", the main purpose of this quality is to guarantee the enough data cover the whole problem domain designed for vision inspection. Therefore, several possible coverage metrics can be applied to analysis.

5.3.3.1 Data coverage

Directly from the coverage definition, it is easily 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 divide the whole problem domain of vision inspection into refined small cases with attribute combinations, then these cases can reflect the diversity of attributes, as well as the diversity of data domain. Then, data amount or data percentage can be easily transferred for describing coverage.

5.3.3.2 Attribute coverage

The other possible coverage metric can be defined from the attributes used in problem domain. In this way, we could use the values of the *attribute* to characterize the data's behaviors. Since the internal logic of a DNN is mostly programmed by data, intuitively, 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 data T covers the range $[low\_n, high\_n]$. 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 set 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:

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

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

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

5.3.3.3 Neuron-based coverage

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

For example, assuming 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

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

where activate (n, x) holds true if and only if n is activated when passing x into D.

This definition determines the coverage of an input independently of the other inputs. One can instead define the additional neurons covered by x that were not covered during training.

More 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 literature.

5.3.3.4 Surprise coverage

Surprise Coverage (SC) 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 also 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 $B = \{b1, b2, \ldots b_n\}$ that divide $(0, U]$ into n SA segments, SC for a set of inputs X is defined as follows:

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

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). We argue that an input set for a DL system should not only be diversified, but systematically diversified considering SA. Recent results also validate this notion by showing that more distant test inputs were more likely to lead to exceptions but might not be as relevant for testing.

5.3.3.5 Example and experiments

Here, taking the Magnetic Tile data for example, if we take two attributes "defect type" and "brightness" for pair-wise analysis, we first can divide the original data into different cases. By setting the threshold of brightness as 0.3 and 0.6, the brightness domain can be divided into 3 cases, and multiple 6 classes of defect types. There as totally 18 cases. Then, the coverage of training data can be calculated, as shown below.

Table 5.3.1. Amount of training data in each case

brightness	Blowhole	Break	Crack	Fray	Free	Uneven
Weak(<0.3)	89	47	45	36	852	80
Medium(0.3~0.6)	257	185	142	56	2294	230
Strong(>0.6)	40	22	13	0	272	44

Table 5.3.2. Data coverage of training data

brightness	Blowhole	Break	Crack	Fray	Free	Uneven
Weak(<0.3)	1.89	1.00	0.96	0.77	18.11	1.70
Medium(0.3~0.6)	5.46	3.93	3.02	1.19	48.77	4.89
Strong(>0.6)	0.85	0.47	0.28	0.00	5.78	0.94

Moreover, the amount and coverage of testing data can also be calculate as below

Table 5.3.3. Amount of testing data in each case

brightness	Blowhole	Break	Crack	Fray	Free	Uneven
Weak(<0.3)	30	20	18	12	233	16
Medium(0.3~0.6)	84	69	29	14	647	63
Strong(>0.6)	7	11	3	4	75	9

Table 5.3.4. Data coverage of testing data

brightness	Blowhole	Break	Crack	Fray	Free	Uneven
Weak(<0.3)	2.23	1.49	1.34	0.89	17.34	1.19
Medium(0.3~0.6)	6.25	5.13	2.16	1.04	48.14	4.69
Strong(>0.6)	0.52	0.82	0.22	0.30	5.58	0.67

5.3.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1

– “Fairness”: AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Examine a source and a method of acquiring test datasets to ensure that no bias is found in applications.
- Extract samples without bias from original data for each case to ensure that no bias is found.
- Record activities carried out to prevent bias from entering.
- Check if sufficient training data and test exist for each analyzed case in the training or validation phase.
- In cases where sufficient training data cannot be acquired for any case, review and relax the standards for coverage and record what should be checked individually by system integration tests in line with the original standards.

5.4 Uniformity of dataset

5.4.1 General definition

– **What is meaning of this specific AI quality?**

Evaluating only the “coverage of each case” as described earlier does not always mean that all datasets are good sampling of the overall environment expressed by input data. When a probability of occurrence differs significantly from one case to another, datasets with big bias are generated only by preparing samples separately for different cases, and performance may be compromised particularly from the viewpoint of AI performance. On the other hand, when performance is required for rare cases whose probability of occurrence is very low, it is impossible to balance the preparation of the practical amount of even data without bias for all input and the preparation of the sufficient amount of data for rare cases. Therefore, the quality “uniformity of data” is also important, which is to make an appropriate compromise contrary to the data coverage described above.

5.4.2 Contents in vision inspection

– **What should be done for the developer in the evaluation process**
  - Evenness between data;
  - Evenness between cases;

According to the above general description, we can see that in vision inspection applications, we need to not only 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 require uniformity strongly which means “the same level of training should be given to all cases artificially or all cases are trained randomly in line with the distribution of extracted training datasets”.

5.4.3 Possible approaches and experiments

5.4.3.1 Evenness between real data and collected data

While, we have known that the uniformity of data actually means the evenness of data, the natural evenness description is to compare the difference between natural distribution and collected data distribution. For example, taking the Casting Defect data in this study and the brightness attribute for evenness analysis, our natural knowledge is most of images are collected in moderate light scenarios, a few collected in dark or bright scenarios. Therefore, the natural brightness domain may abide by the Normal distribution. While, from the distribution of brightness on the collected casting defect images, it is seen that the domain locates in a small range nearly from 0.4 to 0.7, that dark and bright images are indeed not too much, but there exits faultage in the medium brightness zone to destroy the Normal distribution, as shown in the following figure.

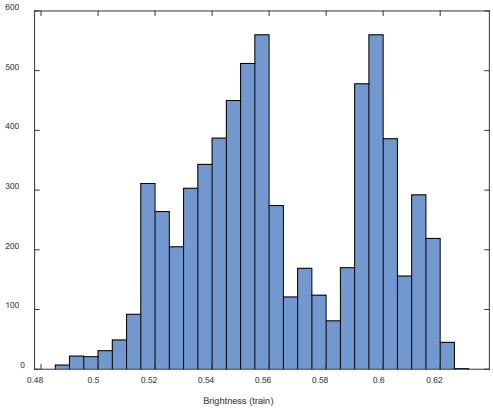


Figure. 5.4.1 Distribution of brightness attribute in casting data.

5.4.3.2 Evenness between training and testing sets

The other evenness metric considering the difference between training data and testing data. For example, with respect to the description attributes of problem domain, what is the relationship between the difference/evenness of data with AI performance.

- ❖ Example 1

Taking the Magnetic Tile data as example, with the division on defect types, we can see the coverage of all kinds of defect types are the same in both training and testing set, which implies the evenness is guaranteed in AI quality assurance.

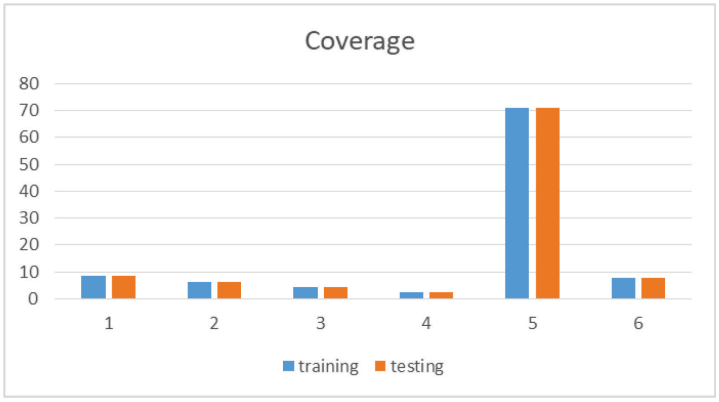


Figure 5.4.2. Coverage distribution on defect types

Then, we can calculate the distribution of Precision and Recall values of training and testing within different types of defects. Where, Recall metric represents the same meaning of classification accuracy.

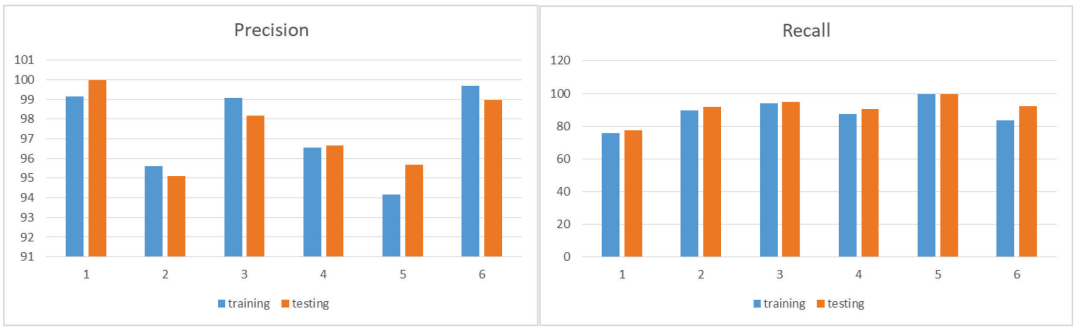


Figure 5.4.3. AI performance on each defect type

It is seen that in this training and testing division (evenness), metrics (e.g. Precision and recall) can also keep even in training and testing.

❖ Example 2

Moreover, taking the Casting Defect Data as an example, we can also take the most common description attributes of image as example, (image brightness and contrast), and get the distribution of these two attributes on both training and testing datasets.

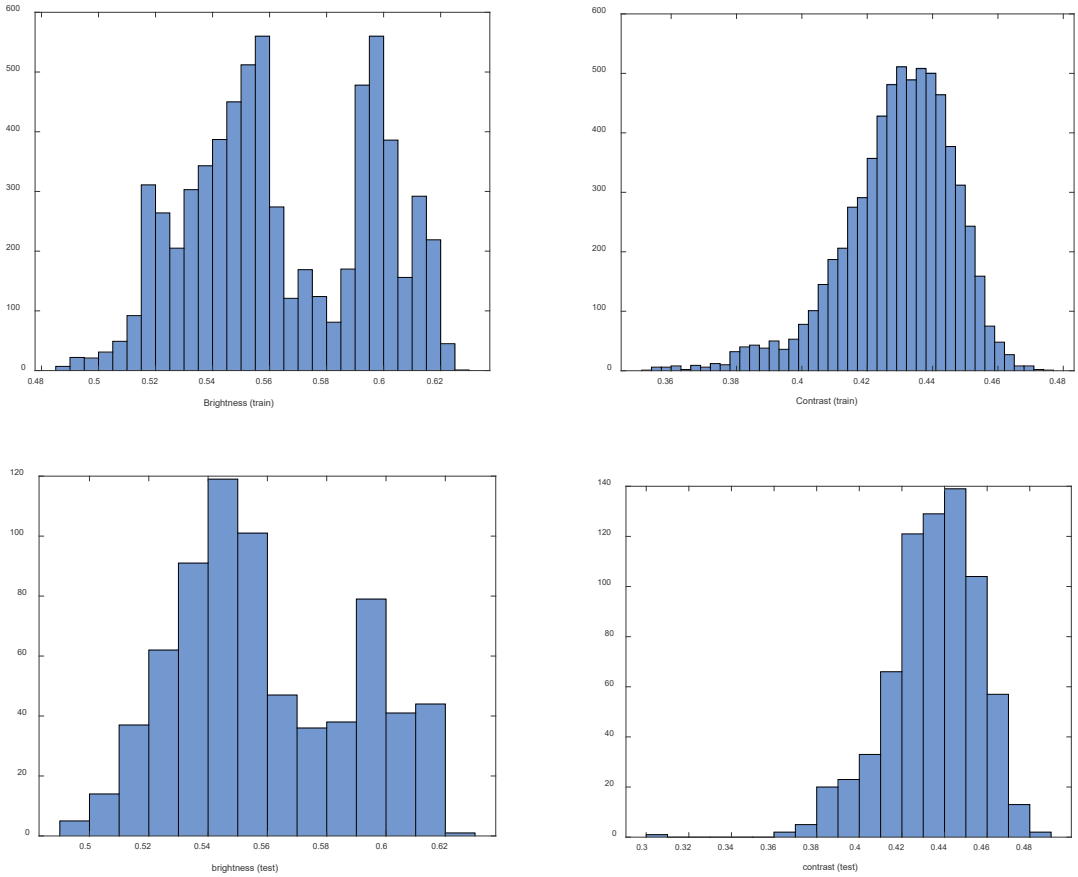


Figure. 5.4.4 Distribution of attributes in training and testing

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

5.4.3.3 Evenness between different cases

On the other hand, the uniformity of data requires to guarantee the sufficient amount of data for high-risk cases, which also means the evenness on different cases. For example, as we know, imbalance is always an important factor affecting the performance of classification applications. While, vision inspection problems usually involve classification, and the collected defect data from the real world is generally imbalanced, e.g. different defect types in the Magnetic Tile data. Moreover, as the division of cases representing different scenarios, the imbalance on different cases also happens commonly. Therefore, it is also necessary to guarantee AI quality via the evenness between different cases.

❖ Example

Taking the Magnetic Tile Defect data as the example, to realize the comparison study on uniform and non-uniform coverage in AIQM, here we can consider to sample training data, and the non-uniform and uniforming sampling for case division is implemented. First, taking the attribute of brightness as the studied object, the cases (1-weak, 2-medium, 3-strong) are shown below

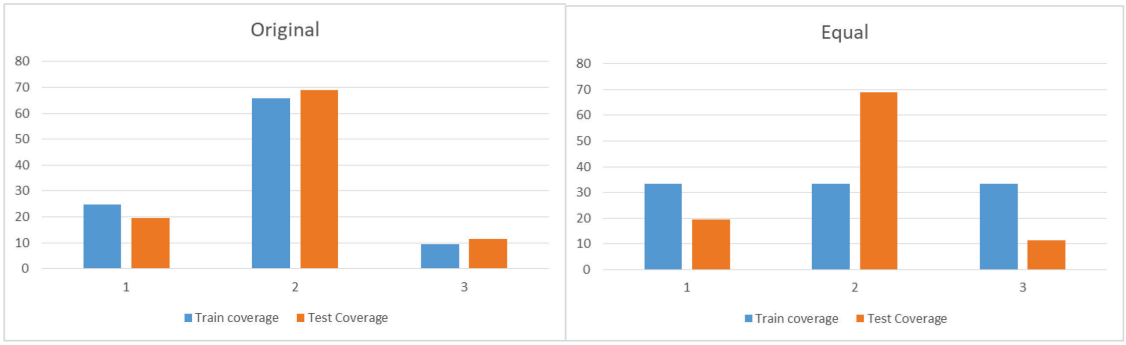


Figure 5.4.5. Uniform and non-uniform coverage design

For comparison analysis, we can compare these performance metrics on two different coverage of training data. Since the testing data keeps unchanged, images with medium brightness have the largest coverage. Therefore, training data and testing data are even under the original coverage (1st figure), uneven for the equal coverage of training data.

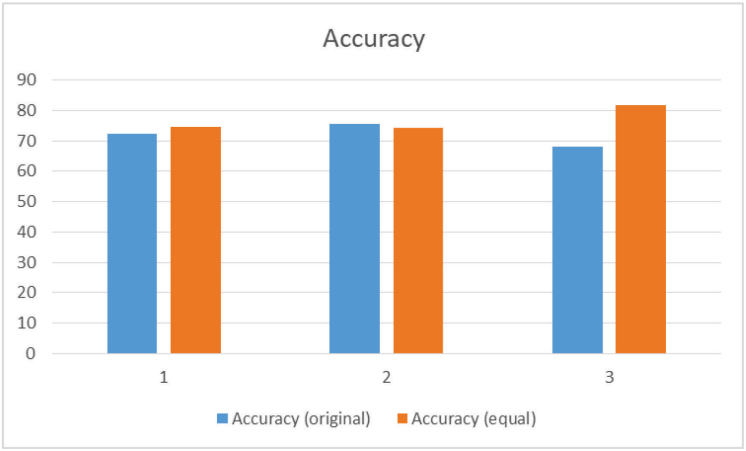


Figure 5.4.6. Classification accuracy in vision inspection

According to the comparison on accuracy, we can see equal coverage has a little superiority to the original coverage, not matter from the perspective of comprehensive testing accuracy (75% vs 74%), or from the perspective of case accuracy in the above figure.

Moreover, we can similarly do some comparison analysis on other description attributes, for example, designing uniform and non-uniform coverage in different defect types, as shown below

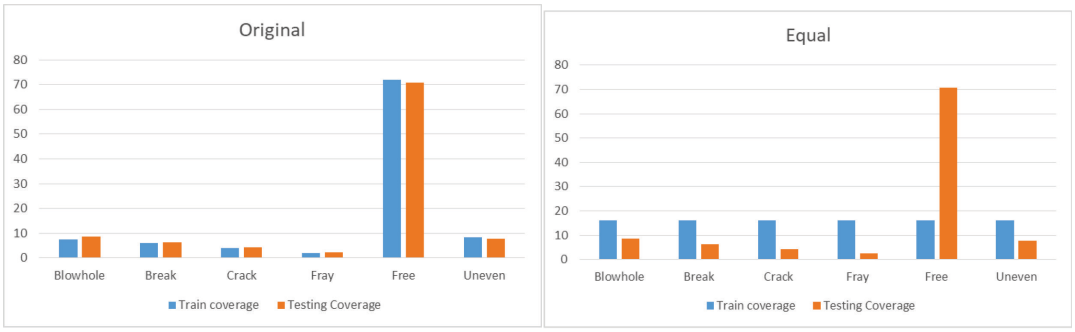


Figure 5.4.7. . Uniform and non-uniform coverage design on defect type

First, the coverage of training and testing data is compared in the above figures. It is seen that training and testing data are even in the first experiment which has original coverage. In the second experiment, all defect types has less coverage of testing than that of training data.

Second, according to the training results, the second experiment has a worse comprehensive accuracy (55%) than that of the first experiment (70%).

Third, we can further analyze AI models' performance based on results of precision and recall metrics, as below

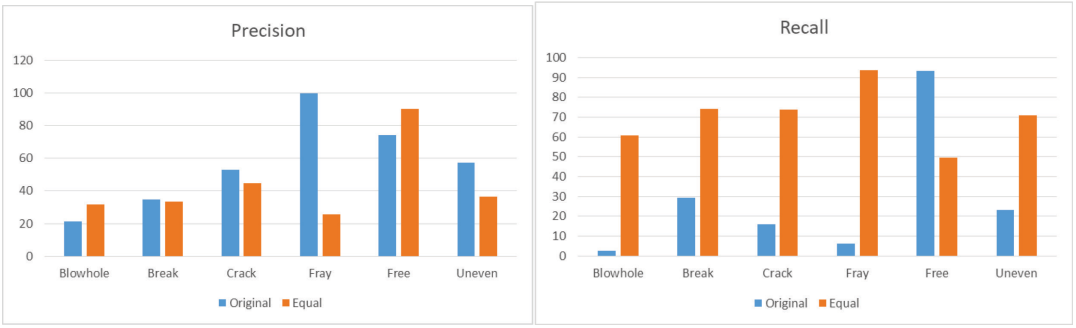


Figure 5.4.8. Performance of vision inspection

Comparing performance of precision, it is seen that original coverage and equal coverage have similar performance, except Fray defect where original coverage outperforms equal coverage obviously. On the performance of recall, equal coverage outperforms original coverage, except the free data. While, considering the comprehensive accuracy has tight relation with values of recall, so the second experiment having low accuracy (55%) is mainly influenced by the large coverage of free data in testing, as shown in the second figure.

5.4.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Regarding the amount of data for each case examined in L1 of the previous section, explicitly check if there is a sufficient amount of data for high-risk cases.
- In cases where data of rare cases is insufficient for training after comparing the amount of all sets of training data with a probability of occurrence of rare cases, prioritized learning of rare cases should be examined. However, in cases where Lv E2 is strongly required, the impact of reduced training of other cases on overall system quality should be examined.

5.5 Correctness of the trained model

5.5.1 General definition

- **What is meaning of this specific AI quality?**

The term "Correctness of the trained model" 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. In addition to numerical behaviors in the training stage such as convergence, this characteristic includes the non-existence of errors in data used for training (or the amount of erroneous data is so low that it does not cause any problem).

5.5.2 Contents in vision inspection

- **What should be done for the developer in the evaluation process**
  - Correctness metrics

According to the above definition, in machine learning based vision inspection systems, the correctness can be described by the correction detection of defects, as well as the correctness of defect recognition. In this way, the proposed vision inspection KPIs can be used for AI quality evaluation in the correctness analysis, e.g. Precision, Recall and Accuracy.

5.5.3 Possible approaches and experiments

5.5.3.1 General correctness metrics

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

The major type metric is based on confusion matrix, which aims at models whose output format is event or label, e.g. classification, clustering. The confusion matrix is defined as below

Table 5.5.1 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 as below.

C**lassification Rate/Accuracy:** classification rate or accuracy is given by the relation:

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

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.

$$Recall = \frac{TP}{TP + FN}$$

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

$$Precision = \frac{TP}{TP + FP}$$

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.

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

❖ Example 1

Taking the Magnetic Tile Defect data as an example, a general 5 layers CNN model is constructed for vision inspection on this dataset. After several epochs of training, the train accuracy of the model on magnetic tile defect images can reach 99.26 %, subsequently, we can also calculate the confusion matrix results, as below

Table 5.5.2 Confusion matrix on the training data

True \ Predicted	Blowhole	Break	Crack	Fray	Free	Uneven
Blowhole	52	0	0	0	2	0
Break	0	42	0	0	1	0
Crack	0	0	32	0	0	0
Fray	0	0	0	21	1	0
Free	0	0	0	0	469	0
Uneven	0	0	0	0	1	51

Moreover, based on the confusion matrix results, we can take the Precision and Recall as the correctness evaluation metrics, and calculate their values on the training data, as below.

Table 5.5.3. Results of Precision and Recall on training data

(%)	Blowhole	Break	Crack	Fray	Free	Uneven
Precision	100	100	100	100	98.95	100
Recall	96.3	97.67	100	95.45	100	98.08

❖ Example 2

Moreover, 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 requirements analysis", the vision inspection based on Casting Defect data is a standard binary classification problem, so here three classification models, such as CNN, VGG16 and ResNet34, are applied to construct the vision inspection systems. Their training parameters are presented in the following table.

Table 5.5.4 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 AI models are described by the classification accuracy, and only the training accuracy are presented. Furthermore, we can evaluate these trained models on the testing data, and study their performance on the correctness quality as the guideline requirements. Results of correctness analysis are shown below.

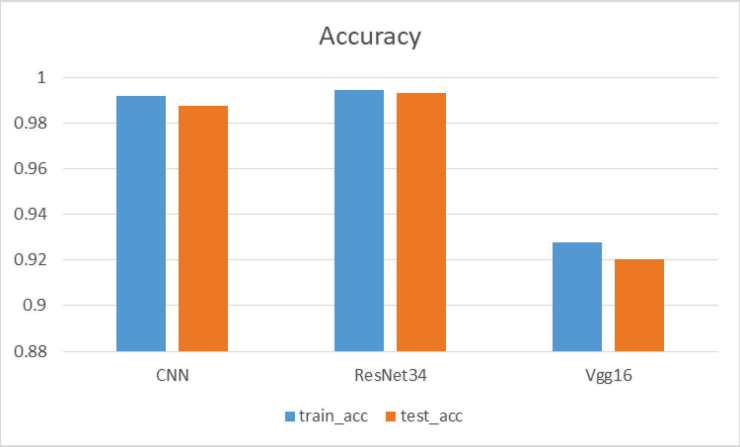


Figure 5.5.1 Accuracy of three models on training and testing process.

5.5.3.2 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 can propose a new metric that mainly focus on the evaluation of erroneous data which is namely the corner case data described in the "sufficiency of data design". 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

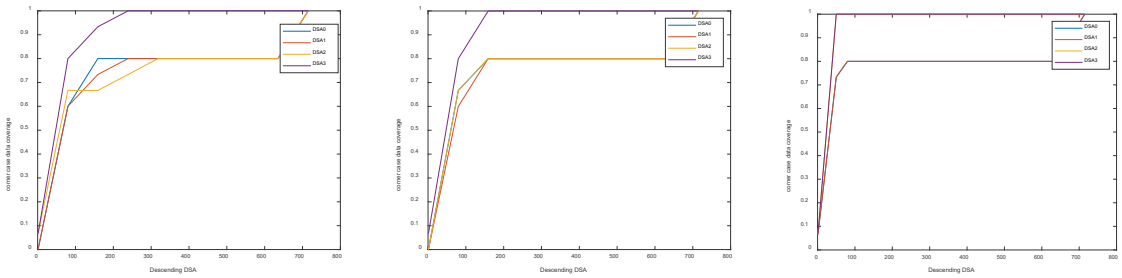
$$cov(v_{th}) = \frac{card(\{d|d \in CD,\ DSA(d) > v_{th}\})}{|CD|} \times 100\%$$

Where, v_{th} is a given threshold; CD represents the dataset of corner case data; |CD| is the cardinality; if we consider to use DSA for corner case detection here, $\{d\}$ represents the set consisting of all detected corner case data which have DSA values larger than the given threshold. While, considering not all of data in corner case are detectable, here we only take the detectable corner-case data into account, namely those data wrongly recognized by DL.

In this way, the proposed corner case data coverage is actually to evaluate the percentage of erroneous behaviors of DL systems, namely the correctness of erroneous data detection.

❖ Example

Taking the Casting Defect data as an example, a 5-layers CNN model is constructed for this vision inspection problem. Then, using DSA as the tool for corner case data detection, and considering the correctness on two convolutional layers and the output layer respectively. The three layers are also named as Layer1, Layer2, Layer3, the corner case data coverage (correctness) analysis are presented below.



(a) (b) (c)

Figure 5.5.2. Corner case data coverage variance as the descending DSA. (a) Layer1; (b) Layer2; (c) Layer3;

In the above figure, when we consider DSAs for detecting the erroneous data in vision inspection, we can find as the DSA threshold decrease, the correctness of erroneous data (corner case data) will increase.

5.5.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Draw out the necessary amount of test data based on assumptions of PoC and past experiences and prepare it through the extraction process that satisfies the "coverage of data".
- Examine how to eliminate errors and outliers such as labels of test data and implement and record their results.
- The same principle applies to training dataset. However, another method can be adopted for how to take data distribution.

- In cases where a certain level of misjudgment is permitted in the test stage (including cases where a way of handling is changed based on false negative/false positive), reasonable standards of judgment should be determined and recorded in advance.
- In cases where fairness is required, establish means for comparison of fairness in advance. In cases where it is determined based on comparative test results, qualifying standards should be established in advanced.

5.6 Stability of the trained model

5.6.1 General definition

- **What is meaning of this specific AI quality?**

The term "stability of a trained model" represents that given an input that is not included in the learning dataset, a machine learning component behaves in a similar way to when given a nearest training data as input. When the trained model is not robust enough, it may function incorrectly upon first seen input, and may violate the safety/security of the system. Therefore, evaluating stability is important when the system is required to satisfy safety/security.

5.6.2 Contents in vision inspection

- **What should be done for the developer in the evaluation process**
  - Stability and robustness;
  - Robustness measurement and improvement;

Based on the above description, we can see that the stability of machine learning based vision inspection systems may involve the following issues. First, the adversarial attack will affect the stability of vision inspection systems. For example, a small amount of noise is added to a defect image which is input to the trained vision inspection system, but the system behave significantly differently, namely its stability is destroyed. These noise 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 the vision inspection systems. For example, when the machine learning based models is overfitting, the corresponding vision inspection systems can also function sensitively to data in the real world.

5.6.3 Possible approaches and experiments

As the definition of "stability of a trained model", the major approach in this quality assurance is to evaluate the machine learning based vision inspection systems' robustness. Therefore, we can introduce the robustness metrics and measurement methodology in vision inspection.

5.6.3.1 Mathematical definition

In software engineering terminology, the standard denotation of robustness 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). 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)):

$$r = E(S) - E(\delta(S))$$

Robustness thus measures the resilience of an ML system's correctness 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 and global adversarial robustness are introduced.

Definition 2 (Local Adversarial Robustness). 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'.

$$\forall x' : \left|\left|x - x'\right|\right|_p = \delta \rightarrow h(x) = h(x')$$

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

5.6.3.2 Robustness measurement methodology

According to the definition of robustness, robustness measurement is actually the minimum perturbation causing testing data to make a different decision. In the following Figure 5.6.1 (a), for a linear binary classifier, the minimal perturbation to change the DL model's decision-making on a given testing point x_0 is the minimum distance from x0 to the hyperplane F, described as the following form

$$\mathrm{r}_*(\mathrm{x}_0) = \operatorname{argmin} \|r\|_2$$
$$s.t. sign(f(x_0 + r)) \neq sign(f(x_0))$$

Similarly, for a multi-classes classifier which is assumed to consist of a set of binary classifiers, as shown in Figure 5.6.1 (b). The robustness measurement of DL model to x0 can also be calculated as the minimum distance from x0 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 part of classifier can be regarded as linear.

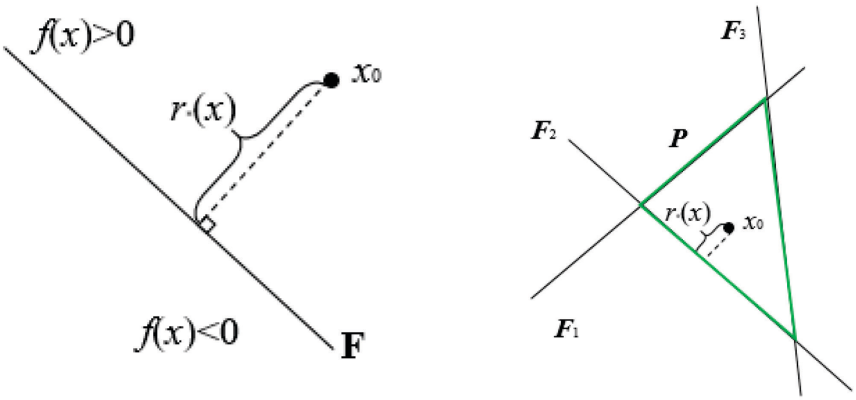


Figure 5.6.1 (a). linear binary classifier (b) multi-classes classifier consist of multiple binary classifiers

5.6.3.3 Robustness measurement with consideration of corner case data

Moreover, considering the stability quality requires to avoid overfitting, so one more useful application is to consider the corner case data's influence on the robustness analysis of AI model. Different with accuracy analysis, here we consider to detect the corner case data in training data, and deleting detected corner case data for model retraining, as described below.

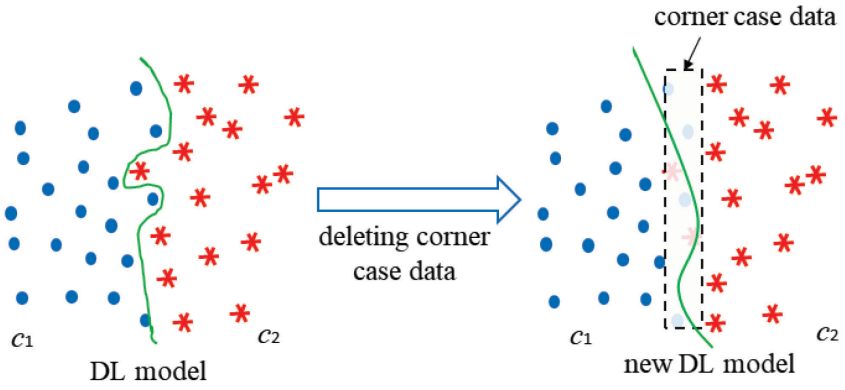


Figure 5.6.2 Diagram of robustness improvement

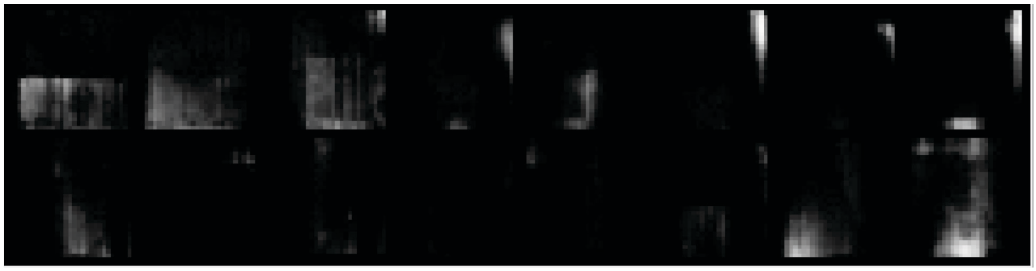
In this way, the trained model will be more robust, since it gets rid of the influence of boundary points (corner case data), and reduce the risk of model overfitting.

5.6.3.4 Experiment results

Based on the description on “stability of the trained model” and the provided techniques /approaches in vision inspection applications, we can design and implement some experiments to evaluate this quality of machine learning based vision inspection systems.

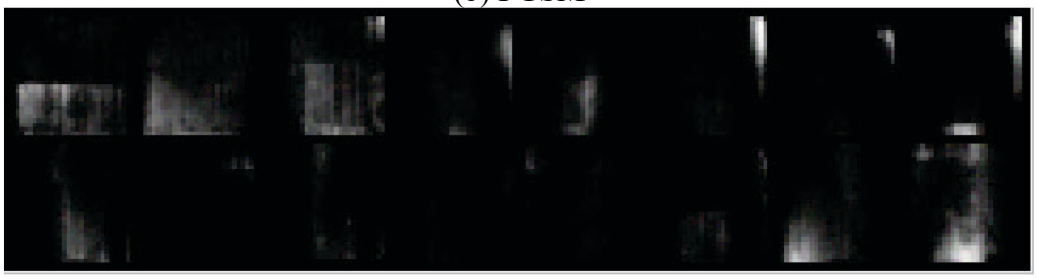
❖ Example 1

First, taking the Magnetic Tile Defect data as an example, we can study the influence of adversarial attack on the stability of vision inspection systems. Here, to generate adversarial input for the trained AI model, the commonly-used adversarial attack methods, like FGSM, CW, BIM, applied here. The newly generated defect images are drawn in the following figure.

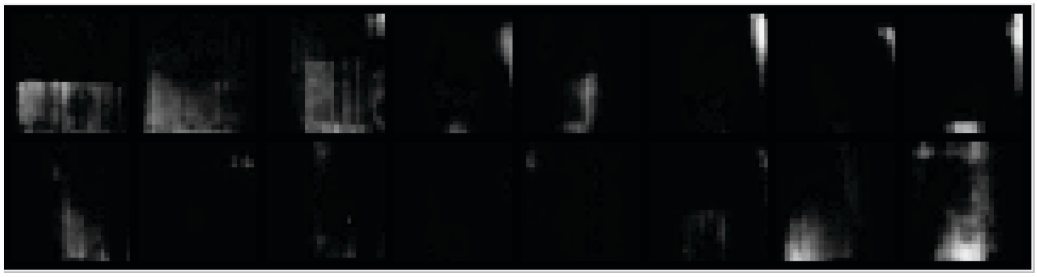


(a) Original (clean)

(b) FGSM



(c) BIM



(d) CW

Figure 5.6.3. Adversarial samples by different methods

Based on the generated adversarial samples as new testing sets, we can calculate its testing accuracy on defect type recognition. The following table shows that the classification accuracy of adversarial samples lower too much, and the FGSM adversarial samples work the worst, implying the trained machine learning based vision inspection system has lowest robustness to FGSM attacks.

Table 5.6.1. Accuracy of four adversarial testing sets

	Original	FGSM	BIM	CW
Accuracy (%)	97.5446 %	24.0327 %	61.2351 %	72.0238 %

❖ Example 2

Moreover, 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 can apply the mentioned methodology in section 5.6.2.2 in experiments. Results of robustness measurement are shown in the following table

Table 5.6.2. 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 in section 5.6.2.3 is also applied here. For example, we can delete top-k corner case data, then them in training data and retrain the vision inspection model. Subsequently, the robustness of the retrained model (Model2) is compared with that of the original model (Model1) in Table 5.6.2. 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 removing can improve models' robustness.

5.6.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Record technologies applied to improve stability
- At Lv1, it is recommended to apply such technologies as cross validation and regularization widely used to prevent over-fitting

5.7 Dependability of underlying software system

5.7.1 General definition

- **What is meaning of this specific AI quality?**

The term "dependability of underlying software system" 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 trained data given 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.

5.7.2 Contents in vision inspection

- **What should be done for the developer in the evaluation process**

In machine learning based vison inspection systems, this quality requires to guarantee the dependability of models, data and the execution environment. For example, for the given datasets, such as Magnetic Tile Defect data or Casting Defect data, there are some open-source codes available in the website. If the developers use them directly into the final vision inspection system development, they should be responsible for ensuring sufficient quality.

5.7.3 Possible approaches

While, 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 can be considered in this quality assurance.

First, the consistence between the developing environment and the actual operation environment. To develop a program for vision 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 vision inspection systems, can be developed on the basis of Python.

Second, the hardware for vision inspection is an important factor affecting the system's dependability. For example, in vision 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 requirement 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, a small one for testing. The dependability of the vision inspection systems cannot be guaranteed as well.

5.7.4 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

The requirements need to satisfy the Lv1 quality level, as follows

- Select reliable and proven software and record the background of selecting said software.
- Monitor the selected software during its operation to find any defect and take measures such as modifications where necessary.
- Examine in advance the impact of differences in an environment from learning to the test phase and an environment in the actual operation stage.

5.8 Maintainability of quality during operation

5.8.1 General definition

- **What is meaning of this specific AI quality?**

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.

5.8.2 Contents and possible approaches

According to the definition on this new AI quality, some possible researches can be studied in the machine learning based vision inspection systems.

The first one is how to carry out additional learning. For example, in vision inspection, if a new type of defects are found, the machine learning based system should be retrained to adapt this new kind of defects. Therefore, the additional learning is to make the vision 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.

The third one is how to monitor the changes in vision inspection systems’ performance. According to the guideline in section 7.8, there are several monitoring techniques, like accuracy monitoring, KPI monitoring, model output monitoring, and input data monitoring. Combining the following figure about types of change, we can describe some maintenance scenarios in vision inspection systems.

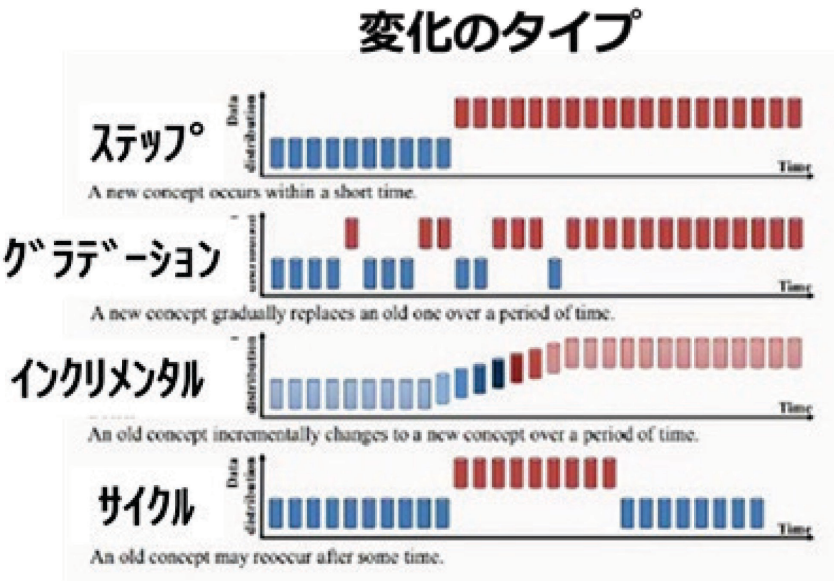


Figure 5.8.1 Types of changes in maintenance

For example, the accuracy monitoring in vision inspection is namely to monitor the classification accuracy on defect type recognition. Then, the red part in the figure can represent the low accuracy scenario. For example, for the first case, it describes the misclassification in vision inspection occurs in a short time, and defect recognition in the subsequent time all fail. This may be because a component in the trained AI model breaks down accidently. For the second case, it describes the process of a machine learning component breaks down gradually. For the third case, it describes the error accumulation process, memory overflow or operation environment change, which may cause the machine learning systems breaks down gradually. For the forth case, the vision inspection system has no fault since it operates incorrectly only for a specific period, it may be affected by input data. For example, a series of new type of defect images are input in the vision inspection system in a period, which are not included in the training dataset, so the output accuracy is low.

For KPI monitoring, as we discussed in the quality “sufficiency of requirement analysis”, two important KPIs in vision inspection applications are Precision and Recall. These two metrics are different with accuracy, they pay major attention on each defect type or specific case in evaluation, while accuracy aims at the global performance. According to the requirements at the PoC phase, Precision and Recall need to exceed 70%. For the metric Precision, its changes are caused by models’ overfitting. For the metric Recall, its changes may be caused by underfitting. Therefore, adaptive or online learning are required in the maintenance process. More maintenance problems can be analyzed based on the four types of changes in the above Figure5.8.1.

For model output monitoring, the changes can also described by the problems in the trained vision inspection systems. Assuming our machine learning based vision inspection model aims at defect recognition, namely a classification problem, the model output must be one of known defect types. For example, based on the Magnetic Tile data, the softmax function is usually adopted in the output layer. Then the output can be one of six defects (Blowhole, Crack, Break, Fray, Uneven, Free). One image cannot be recognized as more than two types of defect, or none of these six types. If this case occurs, we can categorize the type of change as the above Figure5.8.1, then analyze its reason for maintenance.

For input data monitoring, changes can be described by corner cases and risk cases. For example, the first case implies the trained vision inspection system operates in a new scenario which is regarded as unsound case in training process. To ensure the successful operation, new defect images requires to be collected, and the trained model requires to be retrained in the maintenance process. The second case describes the scenario of rare cases. The third case describe the inputs from risk case to corner cases. The forth case may describe the operation environment changes in a specific period, and the inputs in this environment are from a new defect type. Therefore, two measures can be taken in the maintenance process. One is to avoid this environment in the future operation, the second is to collect new defect data and retrain the current vision inspection model.

5.8.3 Quality level requirement

- "Safety": AISL 0.1
- "AI performance": AIPL 1
- "Fairness": AIFL 1

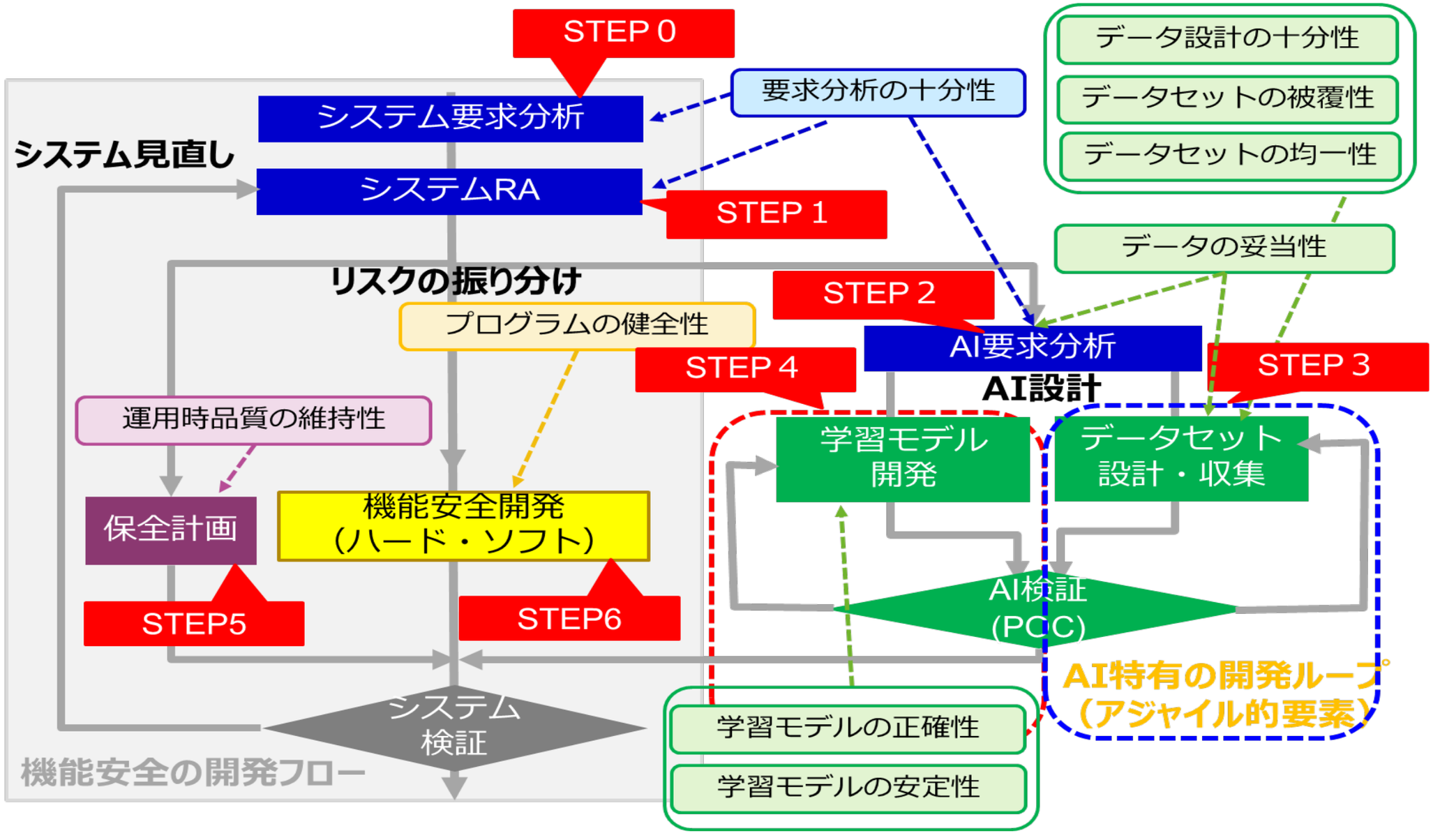
The requirements need to satisfy the Lv1 quality level, as follows

- Examine in advance how to respond to notable system quality deterioration caused by changes in external environment.
- In the case where on-line learning is given, examine in advance the impact of unexpected quality deterioration and take measures from the system side such as the limitation of operation range if necessary.
- In cases where additional learning is given off-line, quality management in line with the previous seven paragraphs should be introduced.

AI利用システム・品質アセスメントシート

STEP0	システム要求分析
STEP1	システム・リスクアセスメント
STEP2	AI要求分析
STEP3	データセット・アセスメント
STEP4	機械学習モデル・アセスメント
STEP5	保全計画アセスメント
STEP6	機能安全プロセス管理アセスメント

Ver.2.2


STEP 0
システム要求分析
要求分析の十分性
データ設計の十分性
データセットの被覆性
データセットの均一性
システム見直し
システムRA
STEP 1
データの妥当性
リスクの振り分け
STEP 2
プログラムの健全性
AI要求分析
STEP 4
STEP 3
AI設計
運用時品質の維持性
学習モデル
開発
データセット
設計・収集
保全計画
機能安全開発
（ハード・ソフト）
AI検証
(PCC)
STEP5
STEP6
AI特有の開発ループ
（アジャイル的要素）
システム
検証
学習モデルの正確性
学習モデルの安定性
機能安全の開発フロー

開発フローにおける入出力一覧

システム要求分析

入力	出力
目的，背景，サービス要求事項	システム機能、性能、環境条件の仕様

システムRA

入力	出力
システム機能、性能、環境条件の仕様	リスク，リスク低減方策，安全度要求（AI品質レベル:AISL）

保全計画

入力	出力
運用におけるリスク，リスク低減方策	保全計画（運用手順）

機能安全開発

入力	出力
ソフト・ハードによるリスク，リスク低減方策	機能安全開発ソフト・ハード

AI要求分析

入力	出力
AIのリスク，リスク低減方策（安全度要求）	AIシステム仕様，AIの機能仕様（データセット属性，モデルアーキへの要求），

学習モデルアセスメント

入力	出力
AIシステム仕様，AIの機能仕様（モデルアーキへの要求）	学習モデルのハイパーパラメータ，学習方法

データセットアセスメント

入力	出力
AIシステム仕様，AIの機能仕様（データセット属性への要求）	データセット属性・属性値とバランス（訓練用・妥当性確認用・テスト用）

学習モデル開発

入力	出力
学習モデルのハイパーパラメータ，学習方法	学習モデル（学習済モデル）

データセット収集

入力	出力
データセット属性・属性値とバランス（訓練用・妥当性確認用・テスト用）	訓練用・妥当性確認用・テスト用データセット（データ拡張・アノテーション含）

AI検証（PoC）

入力	出力
学習モデル（学習済モデル）、訓練用・妥当性確認用・テスト用データセット（データ拡張・アノテーション含）	学習済モデル，訓練・妥当性確認・検証結果，変更仕様

システム検証

入力	出力
学習済モデル．機能安全開発ソフト・ハード，保全計画（運用手順）	システム検証結果，変更仕様．安全度要求（AI品質レベル:AISL）

システム要求分析票

システム要件概要	製品名		システム想定構成	
	品　番			
	目　的			

No.	ユースケース	内容	入力	出力	条件

No	仕様分類	システム要求分析
1	機能	
2		
3		
4	性能	
5		
6		
7		
8	環境条件	
9		
10		

No	構成	
	分類	要素
1	ハードウェア	
2		
3		
4	ソフトウェア	
5		
6		

システム・リスクアセスメント票

システム要件概要	製品名		システム想定構成	
	品　番			
	目　的			

No	第1段階　リスクの抽出・見積り										第2段階　リスク低減策検討								
	使用ステップ	危険源（システムの部位）	危害モード	危害を受ける箇所	危害の発生状況（どのように危害が発生するか）	リスク推定				許容可否（○/×）	リスク低減方策（本質安全設計／防護安全設計／使用上の注意）	具体的な方策例	方策対象（AI／機能安全開発／保全）／（関連資料へのリンク）		リスク低減結果			許容可否（○/×）
						危害の重大性	危害の発生頻度	評価	危害の算出根拠（関連資料へのリンク等）						危害の重大性	危害の発生頻度	評価値	
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		

AI要求分析票

システム構成概要				AI品質の要求レベル		
製品名		想定構成		外部品質	リスク回避性	AISL**/Lv*
品　番					AI パフォーマンス	AIPL**/Lv*
目　的					公平性	AIFL**/Lv*

No	要求内容（外部仕様）				前提条件 （前処理・ 後処理）	教師あり／教師なし／強化学習	検討が必要な項目	属性 （主要属性一覧）	根拠・理由	データセットへの要求（Policy）				機械学習モデルへの要求								構成ハード・ソフトへの要求 （データセットと機械学習モデル以外）
	ユースケース （対象）	処理内容	入力	出力						過去の実績、POCでの知見	データセット 属性 （データの量， 分布）	データ条件 （データの質， 数，サイズ・時空間制約，種別，汚染対策、メタデータの精度・ルール）	データ妥当性の Policy	過去の実績、 POCでの知見	モデル精度 （正解率，適合率，再現性，F値， など）				入力特性 （空間的・ 時系列的）	出力特性 （多クラス分類，信頼度情報有無，閾値， など）	制約（学習時間，ハイパーパラメータ対象，必要なリソースなど）	

データセット・アセスメント票

＊「データ妥当性確認」シート参照

システム構成概要				AI品質の要求レベル		
製品名		想定構成		外部品質	リスク回避性	AISL**/Lv*
品　番					AI パフォーマンス	AIPL**/Lv*
目　的					公平性	AIFL**/Lv*

データ追加・拡張方法		アノテーション方法	
内容	ツール名	内容	ツール名

No	属性の抽出 データセット属性 中属性	小属性	属性値	対象	オリジナル・データセットの構成 訓練用データセットの構成 被覆性	分布 [%]	データ数または量 [件 or sec]	妥当性確認用データセットの構成 被覆性	分布 [%]	データ数または量 [件 or sec]	検証用データセットの構成 被覆性	分布 [%]	データ数量 [件 or sec]	データの妥当性確認 データ 妥当性確認No, *	ラベリング（メタデータ） 妥当性確認No, *

No	1回目データセット収集 データ追加・拡張の有無 データ数量 [件 or sec]（追加・削除・拡張）	拡張後の訓練用データセットの構成 被覆性	分布 [%]	データ数または量 [件 or sec]	拡張後の妥当性確認用データセットの構成 被覆性	分布 [%]	データ数または量 [件 or sec]	拡張後の検証用データセットの構成 被覆性	分布 [%]	データ数量 [件 or sec]	データの妥当性確認 データ 妥当性確認No, *	ラベリング（メタデータ） 妥当性確認No, *
	－											
	－											
	－											

No	1回目 確認結果 検証条件 利用するデータセットVer.	利用するMLモデルVer.	訓練用プログラムVer.	妥当性確認用プログラムVer.	訓検証用（テスト）プログラムVer.	検証結果 属性値ごとの結果 正解率 [%]	適合率 [%]	再現性 [%]	F値	その他	属性ごとの結果（精度）[%]	全体の結果（精度）[%]	ツールによる分析結果（Pair-wise Analysisなど）	評価

No	N回目 データセットの構成 訓練用データセットの構成 被覆性	分布 [%]	テータ数量 [件 or sec]	妥当性確認用データセットの構成 被覆性	分布 [%]	テータ数量 [件 or sec]	検証用データセットの構成 被覆性	分布 [%]	テータ数量 [件 or sec]	データの妥当性確認 データ 妥当性確認No, *	ラベリング（メタデータ） 妥当性確認No, *

No	N回目 データセット収集 データ追加・拡張の データ数量 [件 or sec]（追加・削除・拡張）	拡張後の訓練用データセットの構成 被覆性	分布 [%]	データ数または量 [件 or sec]	拡張後の妥当性確認用データセットの構成 被覆性	分布 [%]	データ数または量 [件 or sec]	拡張後の検証用データセットの構成 被覆性	分布 [%]	データ数量 [件 or sec]	データの妥当性確認 データ 妥当性確認No *,	ラベリング（メタデータ） 妥当性確認No, *	その他の改善ポイント（アノテーションの精度など）

No	N回目 確認結果 検証条件 利用するデータセットVer.	利用するMLモデルVer.	訓練用プログラムVer.	妥当性確認用プログラムVer.	訓検証用（テスト）プログラムVer.	検証結果 属性値ごとの結果 正解率 [%]	適合率 [%]	再現性 [%]	F値	その他	属性ごとの結果（精度）[%]	全体の結果（精度）[%]	ツールによる分析結果（Pair-wise Analysisなど）	評価

データ								ラベリング（メタデータ）								総合妥当性判定
妥当性確認No.	出所	時空間的妥当性（古すぎなど）	外れ値除去	汚染可能性	検査方法	ダブルチェック	使用可否判断	妥当性確認No.	出所	処理方法（ラベルポリシー準拠/バラツキ低減）	ラベルの揺らぎ範囲（分散・偏差など）	汚染可能性	検査方法	ダブルチェック	使用可否判断	

機械学習モデル・アセスメント票

AI要求分析における機械学習モデルへの要求								AI品質の要求レベル		
過去の実績、POCでの知見	モデル精度（正解率，適合率，再現性，F値，など）				入力特性（空間的・時系列的）	出力特性（多クラス分類，信頼度情報有無，閾値，など）	制約（学習時間，ハイパーパラメータ対象，必要なリソースなど）	外部品質	リスク回避性	AISL**/Lv*
									AI パフォーマンス	AIPL**/Lv*
									公平性	AIFL**/Lv*

MLモデル（ID）	出所（新規・改造・流用など）	ハイパーパラメータ	ハイパーパラメータ最適化方法
1			ブラックボックス最適化（ランダムサーチ，グリッドサーチ，ハイブリッドサーチ，ベイズ最適化法，Nelder-Mead法，遺伝的アルゴリズム方(GA)）
			グレーボックス最適化（データセット・サブサンプリング，学習の早期打ち切り，ウォームスタート(過去の経験ベース)）
			半教師あり学習，模倣学習，逆強化学習
			その他

学習方法（ID）	学習手順	学習終了基準（学習回数・時間・過学習防止方法など）	効率化方法（ツールの利用含む）
1			

No	機械学習モデルの設計		N回目（No.）機械学習																														仕様の制限（AI以外での処理などへの要求）
	構成（初期値）		学習条件					訓練時								妥当性確認時								検証時									
	MLモデル（ID）	学習方法（ID）	利用するデータセットVer.	利用するMLモデルVer.	訓練用プログラムVer.	妥当性確認用プログラムVer.	検証用（テスト）プログラムVer.	正確性（モデル精度）				安定性（ロバスト性）				正確性（モデル精度）				安定性（ロバスト性）				正確性（モデル精度）				安定性（ロバスト性）				評価値	
								正解率[%]	適合率[%]	再現性[%]	F値	大きな推論外れ	自然界のノイズ	敵対的データ	学習曲線の収束状況（学習時間・学習回数についての見解，学習不足・過学習の判断），ROC曲線/AUC	正解率[%]	適合率[%]	再現性[%]	F値	大きな推論外れ	自然界のノイズ	敵対的データ	学習曲線の収束状況（学習時間・学習回数についての見解，学習不足・過学習の判断），ROC曲線/AUC	正解率[%]	適合率[%]	再現性[%]	F値	大きな推論外れ	自然界のノイズ	敵対的データ	学習曲線の収束状況（学習時間・学習回数についての見解，学習不足・過学習の判断），ROC曲線/AUC		
1																																	
2																																	
3																																	
4																																	

保全計画アセスメント票

・「保全計画」と「保全実績」からなる

■保全計画

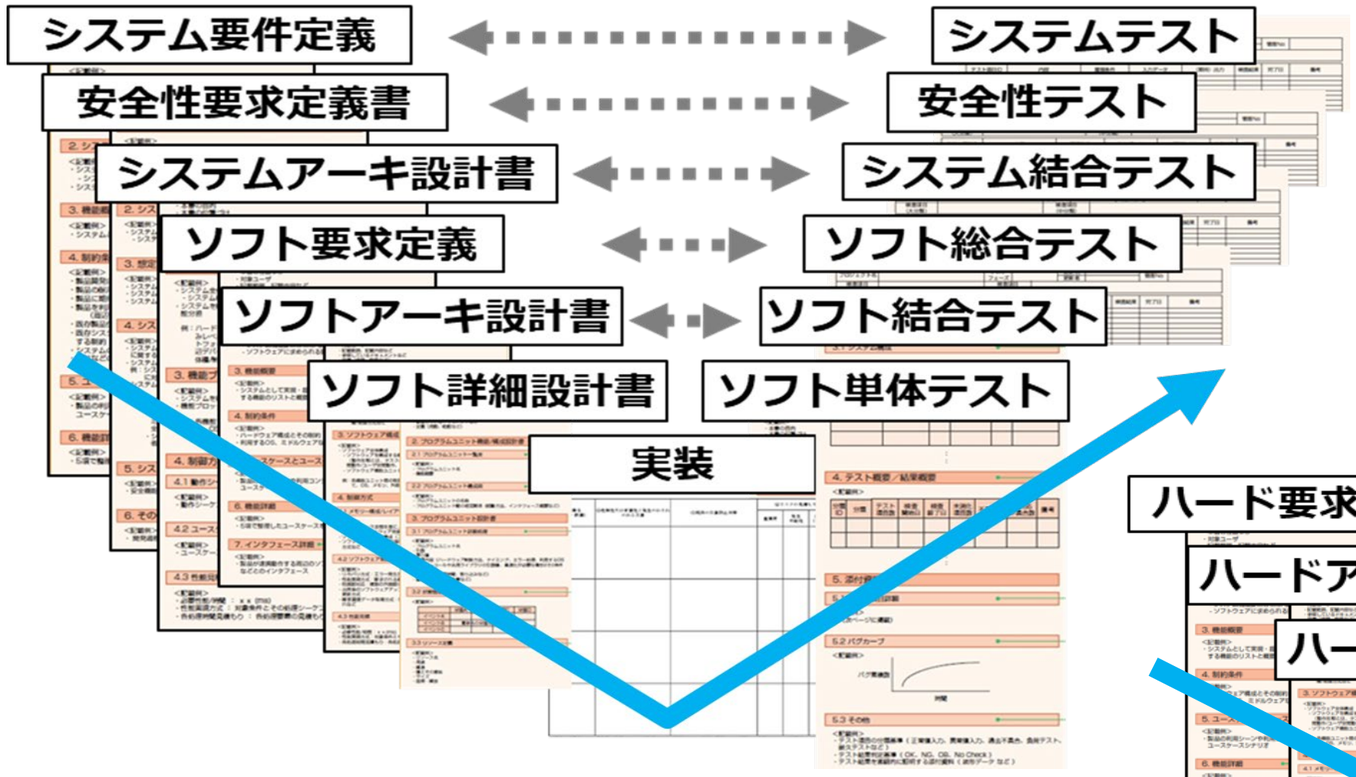
No.	変更対象（What/When）			変更方法(How)	確認方法（How）	変更内容（影響範囲）						
	保全対象	変更の目的 （環境変化への対応/仕様変更・追加・削除/不具合対応）	変更の条件 変化の検出方法（最大・最小・閾値/継続時間） 変更タイミング	変更手段・手順 （再学習，新規学習）	変更後のデグレ防止確認方法	システム要求分析	システムRA	AI要求分析	データセット設計・収集	MLモデル	保全計画	機能安全
1	機械学習モデル	環境変化への対応										
2		仕様変更 （変更・追加・削除）										
3	データセット	環境変化への対応										
4		仕様変更										
5	機能安全 （ソフト・ハード）	環境変化への対応										
6		仕様変更										
7	（人による） 運用方法	環境変化への対応										
8		仕様変更										

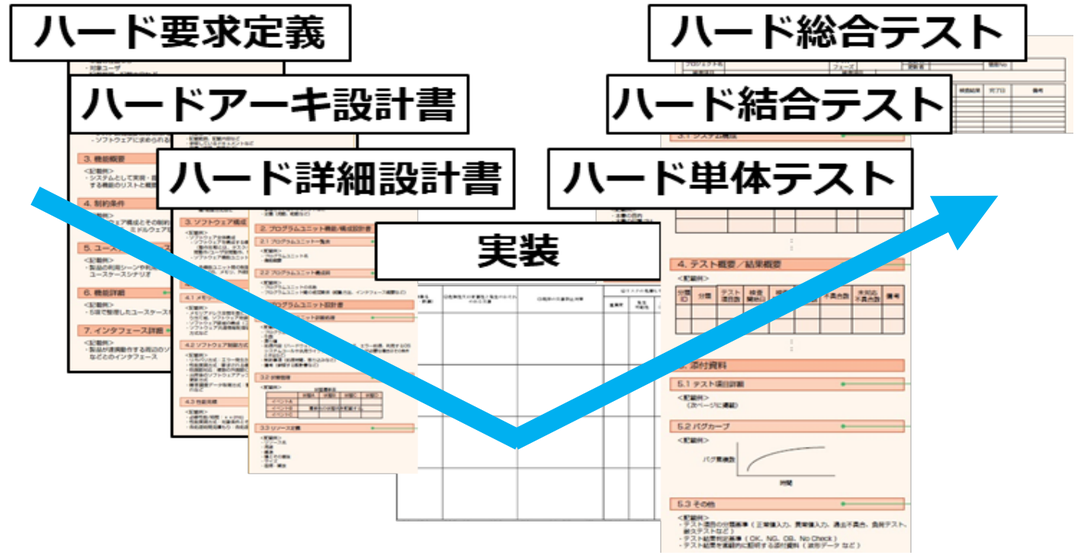
■保全実績

Ver.	変更要因	変更内容（影響範囲）							変更結果	
	仕様変更管理票/不具合管理表の登録番号	システム要求分析	システムRA	AI要求分析	データセット設計・収集	MLモデル	保全計画	機能安全	判定（OK/NG）	詳細（他のアセスメントシートへのリンク）
1.00	仕様変更管理表-***	ー	ー	ー	○	○	ー	ー		
	不具合管理票-***	ー	ー	ー	ー	○	ー	○		
1.01										

機能安全プロセス管理

■機能安全プロセス管理の手順と帳票を利用





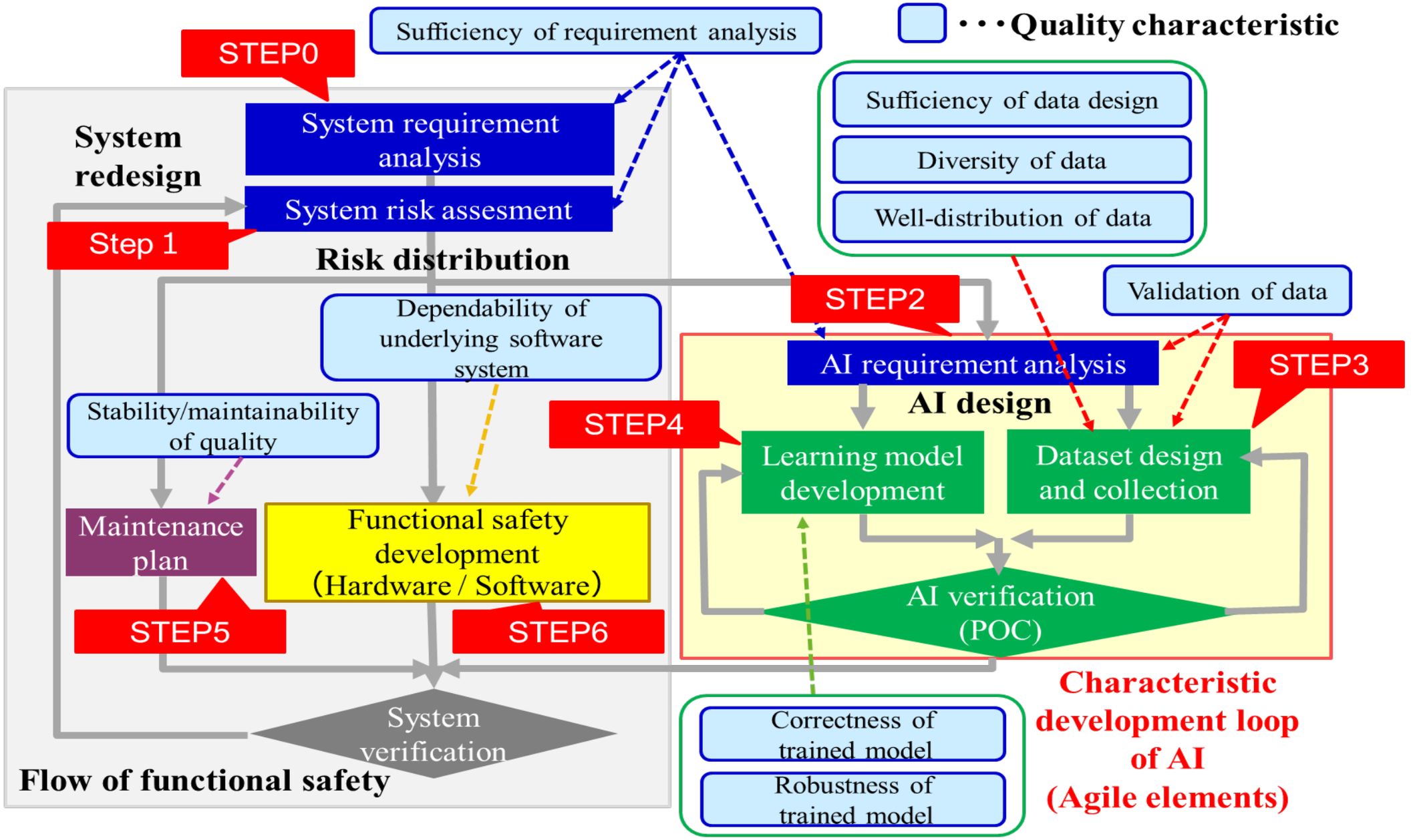
データセットや学習モデルなどAI要素以外

例）
・データ拡張ツール
・アノテーションツール
・学習モデルを訓練するソフト
・学習モデルをテストするソフト

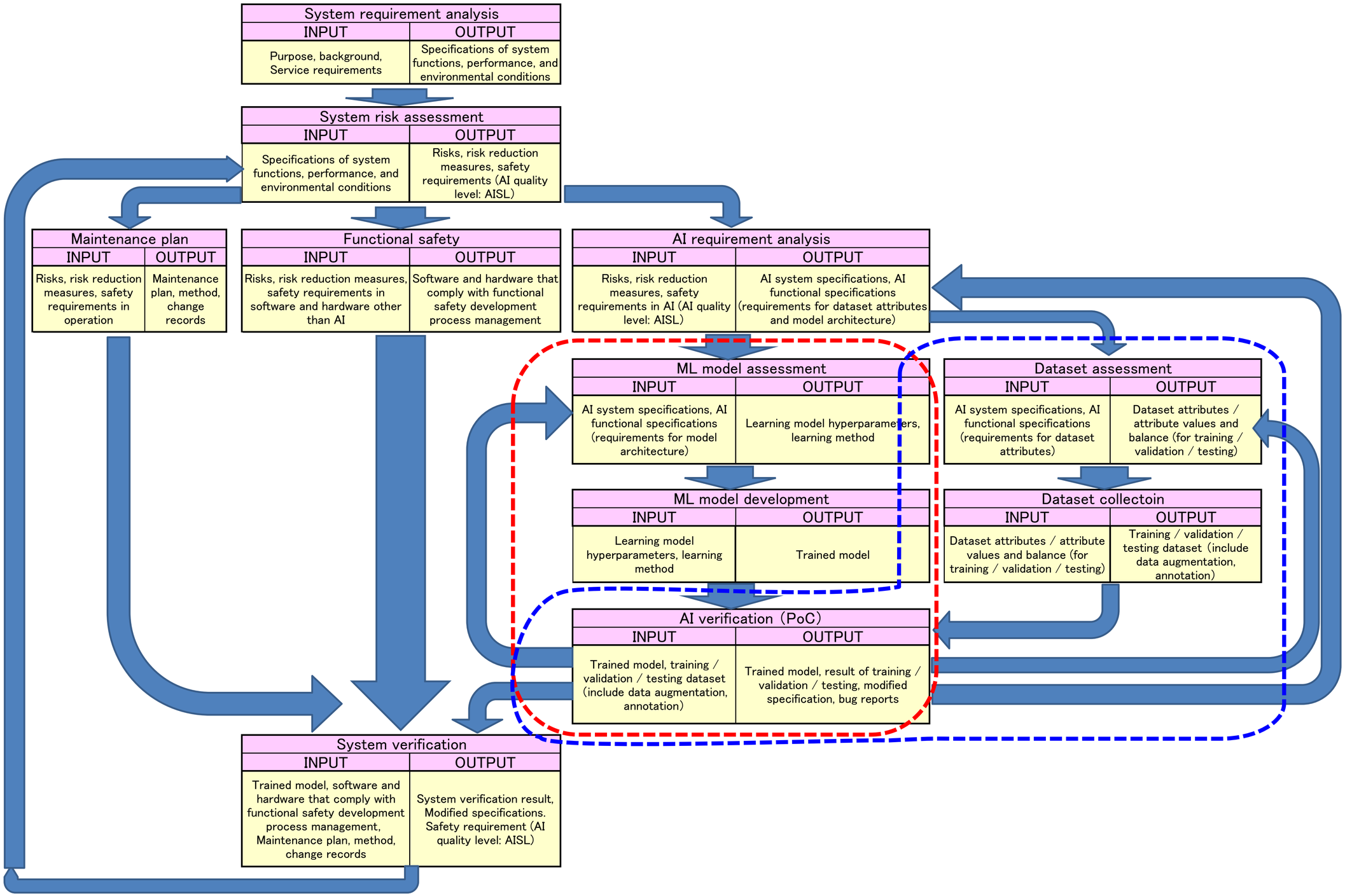
Quality assessment sheet for AI-based system

STEP 0	System requirement analysis
STEP 1	System risk assessment
STEP 2	AI requirement analysis
STEP 3	Dataset assessment
STEP 4	ML model assessment
STEP 5	Maintainance plan assessment
STEP 6	Functional safety process management assessment

Ver.2.2


Sufficiency of requirement analysis
・・・Quality characteristic
STEP0
System redesign
System requirement analysis
System risk assesment
Step 1
Risk distribution
Sufficiency of data design
Diversity of data
Well-distribution of data
Validation of data
STEP2
Dependability of underlying software system
AI requirement analysis
STEP3
AI design
Stability/maintainability of quality
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
Correctness of trained model
Robustness of trained model
Characteristic development loop of AI (Agile elements)

Input / output in the development flow



System requirements analysis sheet

Abstract of System requirement	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		

System risk assessment sheet

Abstract of System requirement	Product name		Assumed configuration of the system	
	Product number			
	Purpose			

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 (○/×)	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 (○/×)
						Severity of harm	Frequency of harm	Evaluation	Basis for calculating harm (Links to related materials, etc.)				Severity of harm	Frequency of harm	Evaluation	

AI要求分析票

System configuration overview				Requirement level of AI quality		
Product name		Assumed configuration of the system		External quality	Risk avoidance	AISL**/Lv*
Product number					AI performance	AIPL**/Lv*
Purpose					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	データセットへの要求 (Policy)				Request for ML model								Requirements for configuration hardware and software (Other than datasets and machine learning models)
	Use case (Target)	Content	Input	Output						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	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.)	

データセット・アセスメント票

*: Refer "Data validation" sheet.

System configuration overview				Requirement level of AI quality		
Product name		Assumed configuration of the system		External quality	Risk avoidance	AISL**/Lv*
Product number					AI performance	AIPL**/Lv*
Purpose					Fairness	AIFL**/Lv*

Add data, data augmentation		Annotation method	
Content	Tool name	Content	Tool name

No	Extraction of attributes				Original datasets configuration											
	Dataset attributes				Training dataset configuration			Validation dataset configuration			Verification dataset configration			Data validation	
														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]	ValidationNo, *	ValidationNo, *

No	1st dataset collection											
	With or without data augmentation	Training dataset configuration after augmentation			Validation dataset configuration after augmentaion			Testing dataset configration after augmentation			Data validation	
											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]	Validation No, *	Validation No, *
	–											
	–											
	–											

No	1st result													
	Testing conditons					Testing results								
						Results for each attribute value								
	Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Accuracy [%]	Precision [%]	Recall [%]	F-measure	others	Results (accuracy) for each attribut [%]	Results (accuracy) of total datasets[%]	Analysis result by tools(Pair-wise Analysis, etc.)	evaluation

No	Nth dataset configuration										
	Training dataset configuration			Validatoin dataset configuration			Testing dataset configuration			Data validation	
										Data	Labeling (Meta-data)
	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, *	ValidationNo, *

No	Nth dataset collection												
	データ追加・拡張の	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, etc.)
											Data	Labeling (Meta-data)	
	データ数量 [件 or sec] (追加・削除・拡張)	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]	ValidationNo, *	Validation No. *	

No	Nth results													
	Testing conditons					Testing results								
						Results for each attribute value								
	Dataset Ver.	ML model Ver.	Training program Ver.	Validation program Ver.	Testing Program Ver.	Accuracy [%]	Precision [%]	Recall [%]	F-measure	others	Results (accuracy) for each attribut [%]	Results (accuracy) of total datasets[%]	Analysis result by tools (Pair-wise Analysis, etc.)	evaluation

Data								Labeling(Meta-data)								Judgment of total validation
Validation No.	Source	Spatio-temporal validation (too old, etc.)	Outlier removal	Potential contaminati on	Inspection method	Double checked	Judgment of availability	Validation No.	Source	Processing method (label policy compliant / Reduction of variation)	Label fluctuation range (Variance and standard deviation, etc.)	Potential contaminati on	Inspection method	Double checked	Judgment of availability	

Machine Learning model assessment sheet

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*
									Fairness	AIFL**/Lv*

MLmodel (ID)	Source (New / Modified / Diverted)	Hyper-parameter	Hyper-parameter optimization method
1			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, etc.)	Efficiency method (Including the use of tools)
1			

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				Correctness				Robustness						
	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	Evaluato in value	
1																																
2																																
3																																
4																																

Changed to specifications that repeat in the vertical direction

Maintainance plan assessment

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

■Maintainance plan

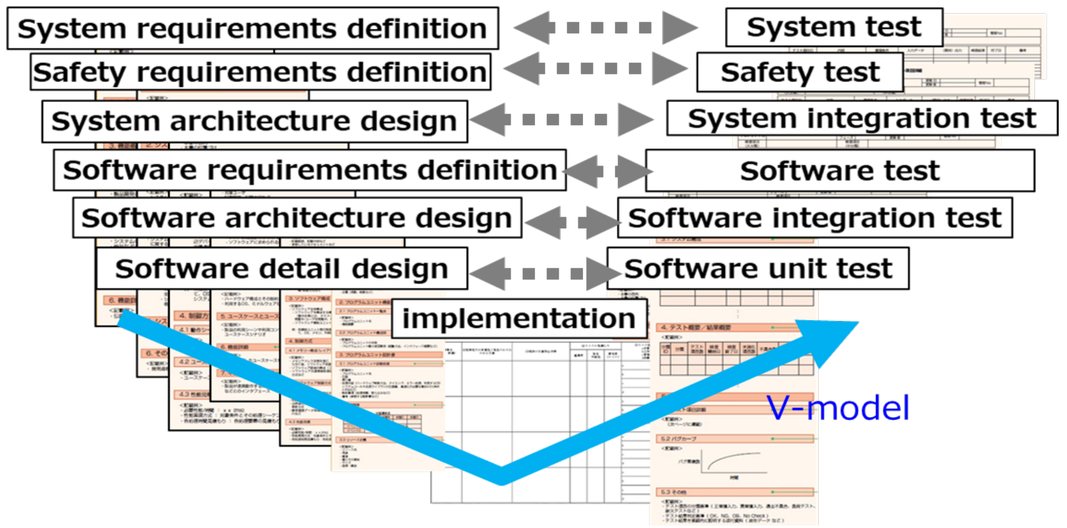
No.	Change target (What)			Change method (How)	Confirmation method (How)	Changes (Impact range)						
	Maintainance target	Purpose of change (Responding to changes in the environment / Specification change / addition / deletion /Bug handling)	Change conditions How to detect changes (Max / Minimum / Threshold / 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

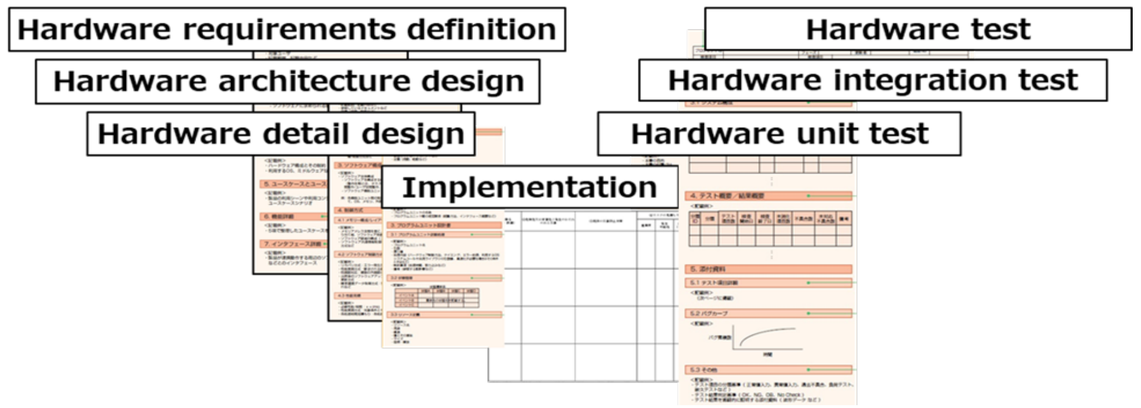
Ver.	Change factor (Why) Specification change management form /Defect management table registration number	Changes (Impact range)							Change result	
		System requirements analysis	System RA	AI requirements analysis	dataset Design and collection	Learning model	Maintainance plan	Functional safety	Judgment	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



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



:: House Price analysis ::

1 Purpose of the technical report:

Our goal is to create an example implementation to demonstrate how Machine Learning Quality Management (MLQM) guideline rule can used to assess interior properties of a House price dataset. Here we will discuss about house price problem it is 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.

2 Expected outcomes:

What a reader can expect from this technical report is the theme of this paragraph. The key takeaways from this technical report should be highlighted in a way so that readers can understand the advantages of applying MLQM guideline in a glance.

Example: We expect to achieve the following outcomes by applying MLQM guideline to an AI based product:

- Details analysis of 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.

3 Author's role/ perspective of authors:

MLQM guideline is created to benefit solution designers to maintain a standard for process design, system design and quality management. It also assists users of different stage to assess the quality, applicability, and performance of the final product. Hence to clarify which stages of the product development were kept in focus while writing this report, it should specify author's perspective explicitly.

Example: Here, we play as a 'service developer'. We consider the guideline as a 'technical starting point' for developing the service in question. The essential intention of this report is to introduce a total investigation of the issue and answer for it as indicated by the predefined AIQM appraisal standards. As a creator of this model issue, my objective will investigate each part of it and attempt

to arrange them on the characterized structure of the rule. It will help us to analysis the problem and finding some incite from the problem. We follow the guideline from early development stage to ensure quality standards are maintained throughout the process.

4 Problem definition:

This is a supervised learning problem. 'House Prices: Advanced Regression Techniques' is one of the most engaging Kaggle challenges that helps competitors developing their skills in solving problems using Machine-Learning algorithms. This challenge is all about predicting the sale-price of a house in Ames, Iowa based on the provided information about many key-factors that may have influence on the price of the houses. Therefore, it is a regression problem and the task here is to minimize the error of prediction. Several Machine-Learning algorithms including a DNN, SVM, Decision tree model has been developed and implemented in this kernel using Scikit-learn and TensorFlow. Stacking and ensembling of many algorithms have also been implemented to achieve better accuracy.

5 Product specification:

This paragraph should include requirements from development entrusted for the product. The requirements can be technical details, quality requirements etc.

Details that can be proposed for product specification can be stated as

5.1 Model specifications:

- Type of learning: Supervised
- Type of AI model: Regression
- Model architecture: Simple deep learning
- Task to perform: Price prediction

5.2 Data related specifications:

- Data related specifications: List of attributes to consider/ Attributes to ignore/ specified values for certain features

5.3 KPI specifications:

- Accuracy: LRMSE, MSE, RMSE etc.

6 Introduction of the datasets:

The report may include a very brief description of the dataset in hand. Common details of the dataset that can be easily found after initial investigation can be presented here.

Example: Here we have chosen 'House price' dataset which is a Regression problem for predicting the sale-price of a house. We will use Kaggle dataset for the analysis of the problem.

- Here is the link of Kaggle house price dataset
- (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data)
- We also can build our won dataset as per requirements.

7 Sample of input data:

'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. If we see the description of the dataset for 'GarageQual' we will see there have total number of counts data point is 1379 and the number of unique values is 5. There has different type of Garage Quality Gd (Good), TA (Typical/Average), Fa (Fair), Po (Poor), NA (No Garage). Typical/ average number of data is high that is 1311 others are nan-value.

Name: GarageQual, dtype: object (Description of a single attribute)

- count 1379
- unique 5
- top TA
- freq 1311

8 Quality assurance procedures using MLQM guideline:

After initial investigation of the current dataset, next the report should explore each of the eight characteristics axes of quality management in response to the achievement of two external qualities - risk avoidance and AI performance, as mentioned in the MLQM guideline.

A. Sufficiency of requirements analysis
B. Sufficiency of data design
C. Coverage of dataset
D. Uniformity of dataset
E. Accuracy of machine-learning models

F. Stability of machine-learning models
G. Soundness of components
H. Maintainability of quality during operation

8.1 Sufficiency of requirements analysis:

8.1.1 Definition:

The sufficiency of this requirement 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 in the stage between the preparation of training data to the test process is examined at this stage.

General Process or structure of the Analysis for Sufficiency of requirement analysis:

- **Defining the problem Domain and Check whether we have data for all ranges**
- **Identifying the corner cases in our problem domain**
- **Selecting Important feature by applying different types of feature selection method**
- **Setup Acceptance of variations for the selected features**

8.1.2 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.

8.1.3 Data for all possible price ranges:

We need to see if we have data for all possible price ranges. For example, in Minato Ward, Tokyo with an average apartment price of 2,133,000 Yen/sqm (1,855 USD/sq.ft). The most expensive apartment in this neighborhood was 3,063,000 Yen/sqm(29419.96 USD/sq.ft) and the cheapest was 1,093,000Yen/sqm(10498.21USD/sq.ft)(https://japanpropertycentral.com/tag/tokyo-apartment-prices/). So, our data for possible ranges should be in this range.

8.1.4 Selecting well-defined feature dimensions:

The attributes and their corresponding attribute values should cover any possible data specific scenario needs to be considered and listed for later analysis like coverage or sufficiency.

Here we have 79 features from there we need to analysis which feature we need to exclude and if there any option for Included any new feature we need to analysis on that also. For feature reduction we can apply different method PCA, Correlation matrix, backward elimination etc. For Include some new feature we need lot of human effort.

Example:

Problems like Kaggle: House Price has lots of features; both necessary and redundant. Decreasing this feature space will simplify the evaluation of dataset quality as per AIQM 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 and filled the missing values with relevant amount. Then we calculated the correlation matrix of the numeric dataset.

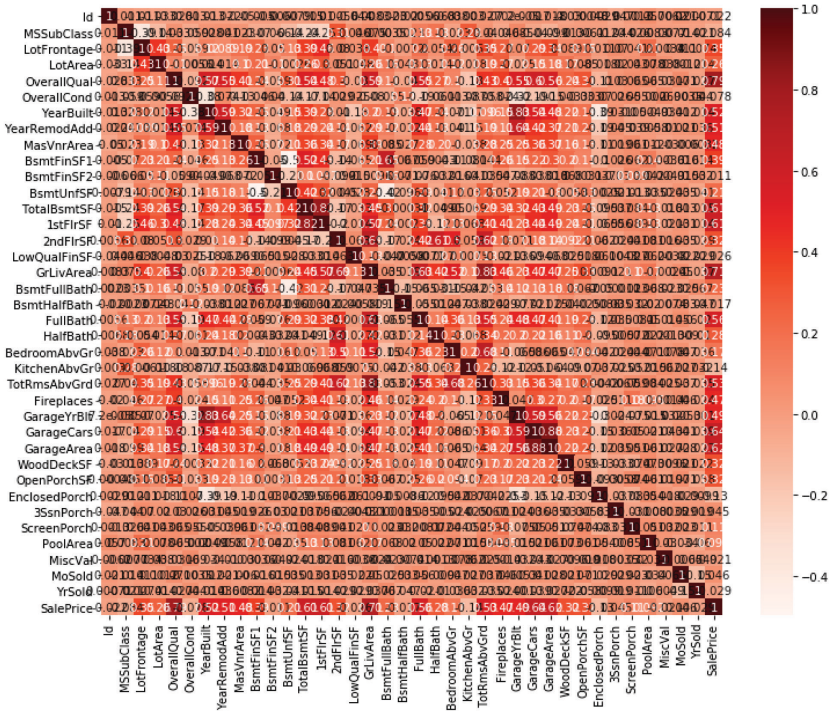


Figure: Correlation matrix for original numeric dataset (38 X 38).

Now, we have selected 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.

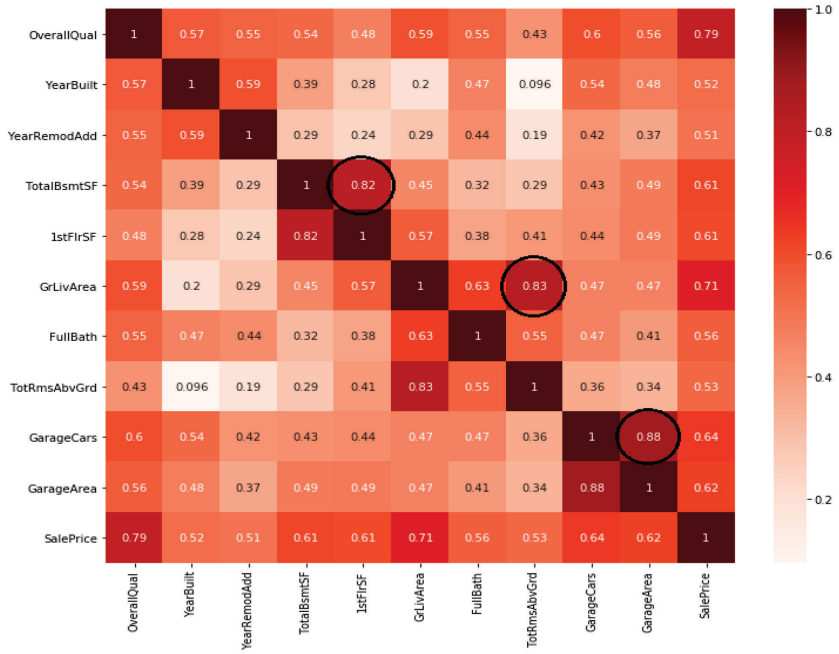


Figure: Correlation matrix for selected features (11 X 11)

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.

Example: It is not possible to take all example for our analysis for that we are using two features for selecting 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.

8.1.5 Selecting in-bound and out of bound areas:

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. Designer can define the range of values which we should include and exclude If we choose a feature called 'GrLivArea' Sale price is closely related to it. In Japan a single person living by himself in should have at a minimum 25-square meters (269-sqft) for one person and maximum recommended for four person is 125-square meters(1345.49-sqrt)(https://resources.realestate.co.jp/living/how-much-living-space-does-the-average-household-have-in-japan/). In Iowa state Houses in Iowa average 1,550 square feet, In Des Moines, the state capital, lowest range 1100 square feet and easier to build family homes upwards of 1800 square feet. (https://www.bobvila.com/slideshow/this-is-the-average-home-size-in-every-state-53461).New zoning laws make it harder to build small houses (less than 1,100 square feet) but we have old houses from 1930 which have houses in low ranges. So here I am choosing acceptance variation for our ground live area between 300 to 5000 square feet.

We have another feature called 'ExterQual' it is 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.

8.1.6 Identifying unsound cases:

Any Combination of attributes which looks like impossible for our analysis those case need to be excluded. For example, in some cases we have pool, but we don't have the area of the pool this is an unsound case, so we need to identify unsound cases.

8.1.7 Conclusion:

Finally, we have selected two feature 'GrLivArea' and 'ExterQual' for our analysis from 79 feature and these two features is well explain. Now the Problem domain analysis is complete, and our definition is covered all the area. Every possible real data can be fit on this dimension. We can say our requirement is fulfilled.

8.2 Sufficiency of data design/ Coverage for distinguished problem domain:

8.2.1 Definition of 'sufficiency of data design':

The term "sufficiency of requirements analysis" 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 the Analysis for sufficiency of data design:

- **Choosing the dataset for the define problem**
- **Apply a different type of augmentation or annotation rule to add new features if necessary.**
- **Checking if we have enough data for our potential ranges**

8.2.2 Data management in each feature dimension:

From the above defined problem domain, all possible number of combinations of attribute values needs to be calculated.

- Here, we need to design our dataset as our defined problem domain. As it is very difficult to make our own dataset so we can choose our dataset from most available data which is match with our problem domain for here we are choosing Kaggle house price analysis data which is very common in our case. First, we need to check the dataset is like our problem domain. For similarity check we can check coverage or distribution in the coverage of dataset part.
- If our dataset is expected that is fine. If we do not have available data, then we need to do some augmentation. If the solution designer wants to include new feature into the existing data set which have no relation with the existing attributes in that case, we need to give some human effort like annotation.
- Since House price is a Discreate dataset numerically augmentation is quite impossible.
- Now we need to check the distribution of the Sale price for checking whether we have data for all possible price ranges. The range of house price in Tokyo was between 10000 USD to 30000 USD which does not exist in our chosen dataset. So, we can estimate Iowa state house prices only. Here is our distribution for Sale Price for Iowa state.

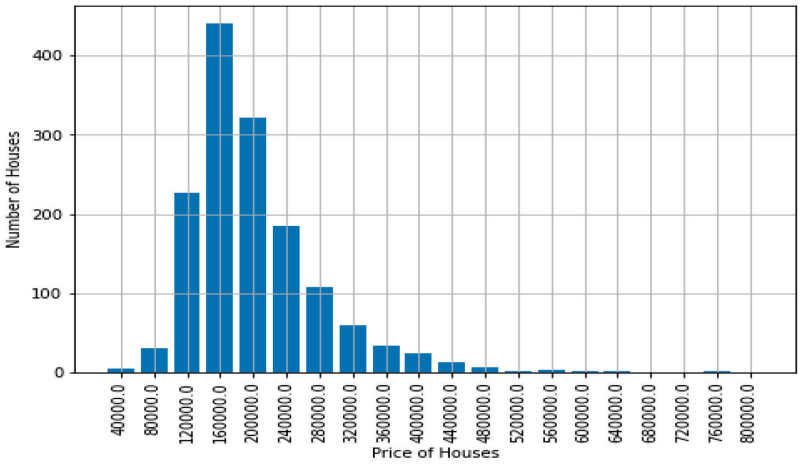


Figure: distribution for SalePrice

When we will design the data set, we will find huge combination 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/risky for the models can be considered.

8.2.3 Conclusion:

House price is a regression problem and its feature space are very vast. Adding new feature, Data Augmentation or feature deletion is not feasible. This data is not friendly for our data design if we can define or make our own dataset then the problem will be solved. But here we don't have enough manpower and time we will use this data set for the analysis.

8.3 Coverage of dataset:

The term "coverage of datasets" means that a sufficient amount of data is given to cases covered by establishing the standards 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 the Analysis for coverage of dataset:

- **Checking coverage for our selected combination**
- **Identifying rare or corner cases**
- **Try to Find out features or some ranges of values we can exclude from the problem domain**

8.3.1 Coverage for each combination:

For each described combination of feature to be considered, data coverage should be calculated and matched against previously set standards of coverage. This depicts the range of training dataset as well as scope of test dataset.

- We need to take full range of data in our defined problem domain and check coverage for each combination. So that we can see the complete distribution in all feature dimensions of our data set.
- A complete distribution of training data both feature dimensions has been presented in the following table.

	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 define range of 'GrlivArea' was 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 have full data coverage.

8.3.2 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/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.

8.3.3 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 ground leave area in the region of 5000 to 6000 we have only one data point so we don't have much coverage in that point we can delete or reduce the range but that will create limitation of the model.

8.3.4 Conclusion:

In this problem we have choose two features and those two features have not enough coverage in some certain region so our model will fail in those region If we can increase the data for that region that will be good for our model but if we cannot increase, then dataset is not capable for testing that region. So, we can say that coverage of data set is not enough for our selected problem. Here our selected data set cannot pass this internal quality test.

8.4 Uniformity of dataset:

8.4.1 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 the 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

8.4.2 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:

For training dataset, expected distribution for the attribute is uniform is shown for the selected attributes in the defined region of interest.

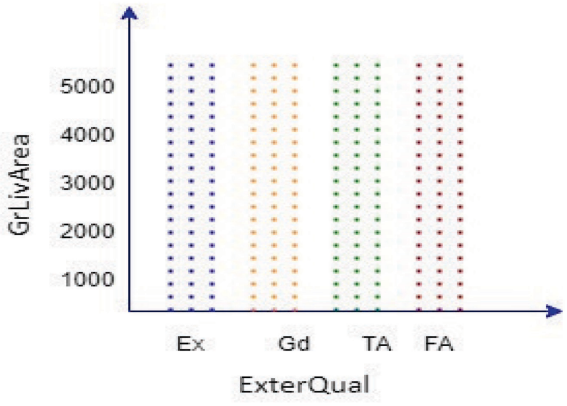


Figure: Expected distribution of houses for the Selected feature space

Actual distribution,

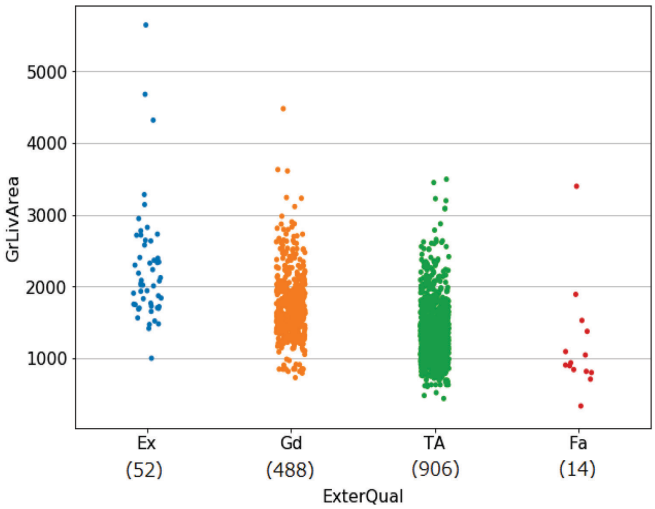


Figure: Distribution of training data in selected feature space.

In this distribution graph, we can see, there have no 'Ex' houses below 1000 sqft and number of houses between 4000 to 5000 sqft is very little. Similarly, number of 'Gd' house in range of 4000 to 5000 sqft also very less and for 'TA' house there have no house in that rages.

There are a few numbers of houses with 'Fair' external quality which are comparatively smaller houses in our defined range of problem domain. Also, the sizes of 'Ground Living Area' in houses are mostly from 1000 to 3000 sqft.

8.4.3 Conclusion:

Here Our expected distribution and actual distribution doesn't match with each other. So, we can 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.

8.5 Accuracy of machine learning models:

8.5.1 Definition of 'accuracy of machine learning models':

The term "accuracy of machine-learning 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 the Analysis for Accuracy of machine learning models:

- **We need to setup KPI for the performance checking**
- **Checking the performance of the model by using our selected KPI**
- **Try to find out the cases where model performance is failing**
- **Make the decision after all analysis**

8.5.2 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 setup KPI for the performance checking. There have 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 a better fit. By using Mean Square Error, we have found error result 3.211.

$$Mean\ Square\ Error = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

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 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

$$Root\ mean\ Squared\ Error = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

Mean Absolute Logarithmic Error: Result: It is the difference between the measured value and "true" value. Since Sales price distribution is scatter and we also need to take the distribution range in a small range for that we are using logarithmic error. By using Mean absolute logarithmic error, we have found error results 0.7653.

$$Mean\ Absolute\ Logarithmic\ Error = \frac{1}{n}\sum_{i=1}^{n}|log\ y_i - log\hat{y}_i|$$

From the analysis we have seen different measurement by using different types of KPI. Now we will check the **Performance comparison for different sized dataset** using Mean absolute Logarithmic error:

First, we trained and evaluated a fully connected **dense layer** with different region data from table in coverage of datasets. The results are listed in table.

	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.

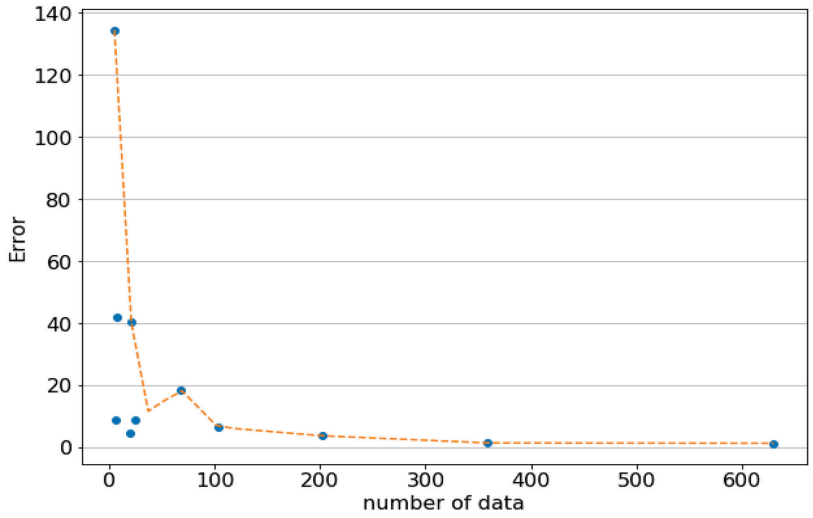


Figure: Error vs Number of training data interpolation.

The curve is like exponentially decaying which is expected. Error is inversely proportional to number of data.

$$\text{Error} \propto \frac{1}{number\ of\ data}$$

8.5.3 Performance comparison for different distribution of data:

In the above case, the categorical attribute was inactive during training; it was redundancy. 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.

	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 the distribution plot of this attribute below, we can see most of the data lies between 1001-2000 sqft. By taking only this range will minimalize its effect on training and evaluation.

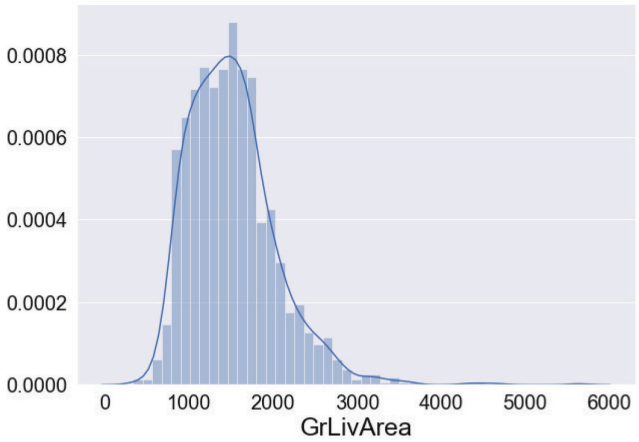


Figure: Distribution of data samples with respect to 'GrLivArea'.

So, we took the sub-group with best number of data samples; *'1001-2000 sqft GrLivArea'*. Distribution of selected sub-group for attribute 'ExterQual' is given below.

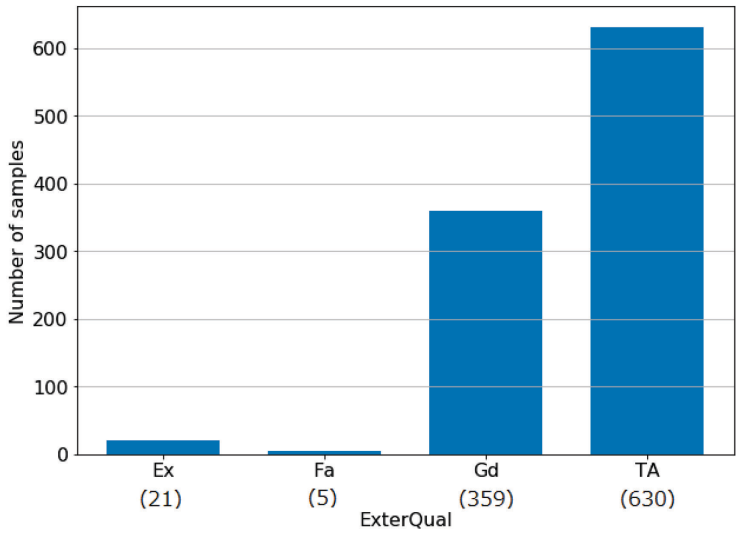


Figure: Distribution of sub-group of train data (GrLivArea: 1001-2000 sqft) for 'ExterQual'.

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 a table below.

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 = number of data available for the rarest category in a certain combination.

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 model can identify the corner case by measuring the performance of the model.

8.5.4 Finding corner cases:

Following the evaluation method, we can find erroneous behavior of the model. Thus, we will get the limitations of our design reform it.

8.5.5 Conclusion:

Prediction accuracy can be shown by KPI. Here we are using Mean absolute Logarithmic error for the KPI measurement. Our model becomes fail in some cases because we do not have enough coverage in some region. From the above analysis we can say, the more we have data coverage our error is decreasing. If the data is uniform the performance is also getting better.

8.6 Stability of machine-learning models:

8.6.1 Definition:

The term "stability of machine learning 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.

Measuring stability/ robustness of machine learning models:

Stability analysis enables us to determine how the input variations are going to impact the output of our system. Stability of a learning algorithm refers to the changes in the output of the system when we change the training dataset. A learning algorithm is said to be stable if the learned model doesn't change much when the training dataset is modified. A model changes when you change the training set. That's just how it is! But it shouldn't change more than a certain threshold regardless of what subset you choose for training. Corner cases and related detection are crucial in AI quality assurance for constructing safety- and security critical systems. The generic corner case researches involve two interesting topics.

- Enhance DL models' robustness to corner case data via the adjustment on parameters/structure
- Generate new corner cases for model retraining and improvement

Experiments:

According to guideline Evaluating stability is important in cases where safety is particularly required.

- Problem of making an inference far away from input other than training datasets: This problem may be caused by over-fitting on, for example, training datasets.

8.6.2 Specific handling:

Stability is strongly related to the following three phases in the machine learning lifecycle so that it needs to be evaluated and enhanced mainly in these phases in order to achieve the stability goals

- Iterative training phase: Avoid over-fitting of training datasets through separating training datasets and validation datasets, evaluating the impact of minimal changes in input on output and monitoring the training process.

Overfitting: Good performance on the training data, poor generalization to other data.

Underfitting: Poor performance on the training data and poor generalization to other data

For example, we have chosen a full connected dataset for 'Hose price analysis' problem. The model architecture is given below.

Architecture	FC(128)+ReLU FC(256)+ReLU FC(64)+ReLU FC(10)+ ReLU
Number of trainable parameters	88,449

The training loss: 0.0212 and validation loss: 0.0514 , epochs: 300, batch size = 32

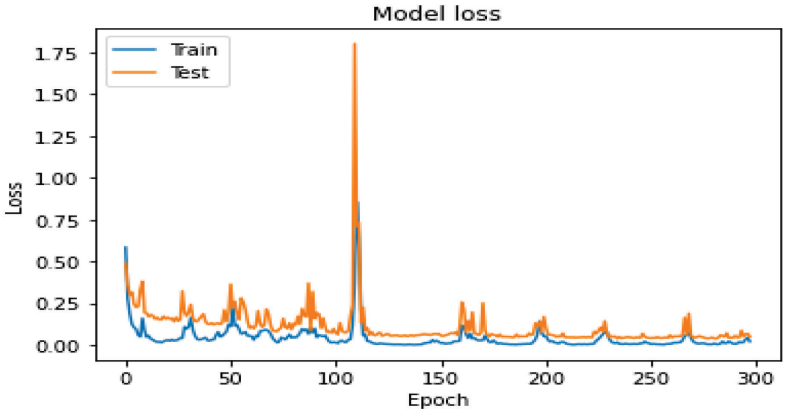


Figure 1: Loss Curve of training and testing

Here the training loss: 0.0451 and validation loss: 0.1237 when epochs: 100

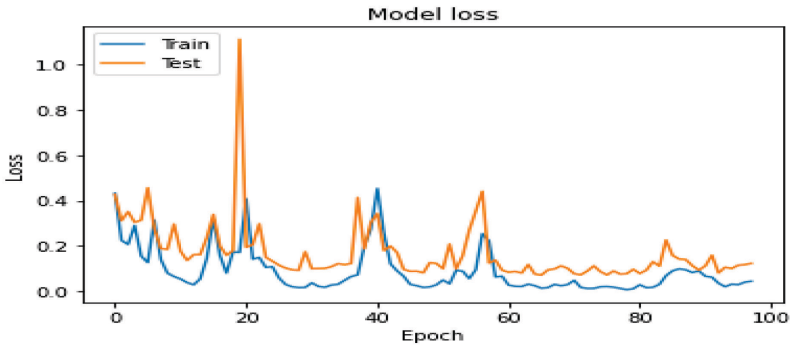


Figure 2: Loss Curve of training and testing

Model Architecture:

Layer (type)	Output Shape	Param #
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

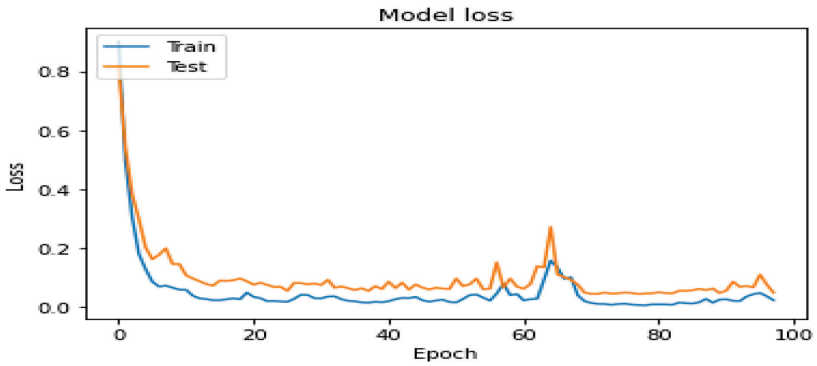


Figure 3: Loss Curve of training and testing

From the curve we are seeing that model is not making an inference far away from input other than training datasets that is model is not overfitted. Training loss and validation loss is corelated with each other, so overfitting problem is not happening for our problem.

- https://epub.wu.ac.at/5398/1/Report131.pdf
- https://www.jmlr.org/papers/volume2/bousquet02a/bousquet02a.pdf

8.6.3 Conclusion:

Stability of machine learning models help us to measure the model's generalization ability, evaluating a model's reaction to corner cases/rare cases, evaluating the model's performance on adversarial examples. Therefore, Robustness indicate model's performance for unknown inputs or under new environmental conditions.

8.7 Soundness of components/ Dependability of underlying software system:

8.7.1 Definition:

The term "soundness of components" 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 trained data given 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.

Listing necessary AI components and their description: The following AI components should be described in details and their quality need to be assured.

8.7.2 Language:

We can work with Python, R, Java, Julia, and Scala (briefly) 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.

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

8.7.3 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, Apache, AWS, 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.

8.7.4 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:** AI developers generally use Hierarchical Data Format (HDF) file (.h5) 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.
- **Codes/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 by programing languages are part of the workflow of the machine. These codes do 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, House price dataset takes about 449.88 KB space.

8.7.5 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 (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 is longer than binary is longer than 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.

8.7.6 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:

In some version of TensrFlow 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. In some version 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 the people uses GeForce instead of Quadro. On the other hand, Intel released the instruction set and driver software specialized for AI/ML was released. So, the provider of the framework (TesorFlow, 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.

8.7.7 Software security:

Security is a major concern when machine operates online. Here, we have described the examples of the theme that the solution designer should consider while building the application sets with AI/Machine learning functionality.

This document doesn't refer to the common aspect for the cyber security taken for the software that doesn't include the AI/Machine learning functionality also. (For the reference about the cyber security aspect doesn't include the AI/Machine learning, refer to the ISO/IEC 27000 series, NIST SP800 series, NIST Cyber Security Framework, ISO/IEC 15408 Common Criteria and so on).

Examples: The attack method is developed day by day. So, the following list is current examples. The solution designer should search the newest information periodically. The annual assessment and the measures are recommended.

- **Adversarial example:** A machine learning technique that attempts to fool models by supplying deceptive input with small and intentional perturbations.
  - "Explaining and Harnessing Adversarial Examples" Goodfellow et al.
  - https://arxiv.org/pdf/1412.6572.pdf
  - "Adversarial Examples in the Physical World" ICLR2017 Krakin et al.
  - https://arxiv.org/pdf/1607.02533.pdf

- **Membership inference:** Given the huge number of the input, obtaining whether a data point is from the target model's training set or not. "Membership Inference Attacks Against Machine Learning Models" Shokri et al.
  - https://arxiv.org/pdf/1610.05820.pdf

- **Poisoning:** Adversarial contamination of training data.

"Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning"

- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8418594

- **Model inversion:** Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures
  - https://dl.acm.org/doi/pdf/10.1145/2810103.2813677

- **Model extraction:** An attack in which an adversary utilizes a query access to the target model to obtain a new model whose performance is equivalent to the target model efficiently.
  "Stealing Machine Learning Models via Prediction APIs"
  - https://arxiv.org/pdf/1609.02943.pdf

- **Backdoor attack:** With a specific trigger by additionally training the malicious training data, including the specific trigger to the DNN model, the DNN correctly recognizes normal data without triggers, but the network misrecognizes data containing a specific trigger as a target class chosen by the attacker.
  "Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering"
  - https://arxiv.org/pdf/1811.03728.pdf

8.7.8 Conclusion:

It is 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.

8.8 Maintainability of quality during operation:

8.8.1 Definition:

The term “maintenance of quality” means that internal qualities 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.

8.8.2 Maintainability of quality during operation:

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

8.8.3 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.

8.8.4 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 of time.

8.8.5 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, respectively. The monitoring methods are divided into human monitoring in case of house price prediction data. For the house price problem concentrate on privacy of data for real environment use also need to check the preprocessing method of data additional training. We need to check how to handle cases where quality deteriorates.

8.8.6 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.

An AIQM Reference Report on Postal Code Analysis

1 Purpose of the technical report:

This report aims to describe a real-world problem and possible machine learning solution to it. The objective of this report is to express the idea verbally,

- How we can think about the problem
- How we can design the problem
- How we can manage to fulfill our requirements
- How we can execute our ideas
- How we can validate our work
- How we can ensure its performance

In this report, 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.

2 Expected outcome:

This report 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 AIQM guideline written and translated by AIST. This is not only analyzing postal code problem but also validating the self-sufficiency of the guideline. So, the reader of this report can expect to gather knowledge about,

- Details of 'postal code analysis' problem analyzing real cases
- Challenges and methods for data management
- Introducing AI to our problem
- Methods and ideas about validating data and AI

- Thoughts about future maintenance and handling

3 Author's role/ perspective of authors:

From author's point of view, the primary motive of this report is to present a complete analysis of the problem and solution to it according to the predefined AIQM assessment criteria.

This report will be a complete example of quality management of an AI. The example will help describing the completeness of AIQM guideline or give a new perspective to think upon which will eventually improve the guideline. The target will be exploring every aspect of it and try to organize them on the defined structure of the guideline.

This example is not a creation of a single person but congregation of different ideas and opinions of many researchers and engineers.

4 Problem definition:

This report will focus on 'postal code analysis' i.e., 'handwritten digit recognition'. In the field of machine learning, it is an image classification problem that identifies 10 numerical digits. The target behind this analysis is to evaluate an automated machine that can segregate posts according to their postal codes.

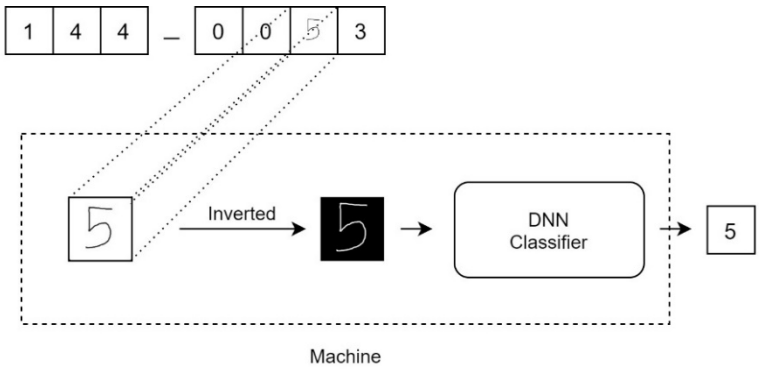


Figure 1: An overview of the workflow of 'Postal Code Analysis' machine.

Referred to fig. 1, 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.

5 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,

5.1 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'.

5.2 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).

5.3 KPI specifications:

- **Accuracy:** Accuracy, Recall, Precision, F-measure etc.

- **Stability/Robustness:** Mutational robustness, Distance-based Surprise Adequacy (DSA) etc.

6 Introduction of the datasets:

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
  [http://yann.lecun.com/exdb/mnist/]
  USPS dataset - Handwritten digits
  [https://www.kaggle.com/bistaumanga/usps-dataset]
  ARDIS – the Swedish dataset of historical handwritten digits
  [https://ardisdataset.github.io/ARDIS/]
- We can use multiple datasets combined.
- We also can build our won 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 the report.

MNIST dataset holds 70,000 image data where 60,000 is training data and the rest is for testing. These are gray scaled images having dimensions 28X28. It is written in the description of the dataset that there were 250 different hand writings 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 '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 AIQM guideline are universal. We know, MNIST is a US based handwritten dataset (Wikipedia/MNIST), so a classifier trained on MNIST will be suitable for the US people.

7 Sample of input data:

A sample of input data (image) for the model is given here. 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 AIQM assessment.

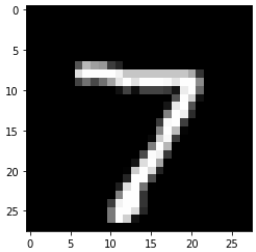


Figure 2: A sample input for 'Postal Code Analysis' machine from MNIST dataset.

8 Quality assurance procedures using AIQM guideline:

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

- Sufficiency of requirements analysis
- Sufficiency of data design
- Coverage of dataset
- Uniformity of dataset
- Accuracy of machine-learning models
- Stability of machine-learning models
- Soundness of components
- Maintainability of quality during operation

8.1 Sufficiency of requirements analysis:

8.1.1 Definition:

'Sufficiency of requirements 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 analyzed and defined impossible case scenarios if there is any.

Requirements is 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.

8.1.2 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.

8.1.3 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 3: Different positions of the digit.

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

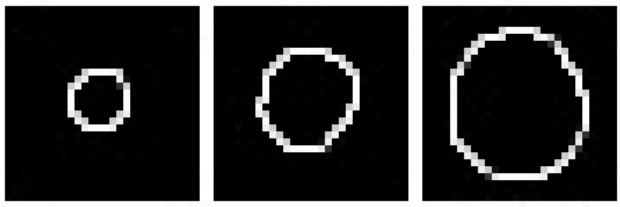


Figure 4: Different areas of the digit.

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

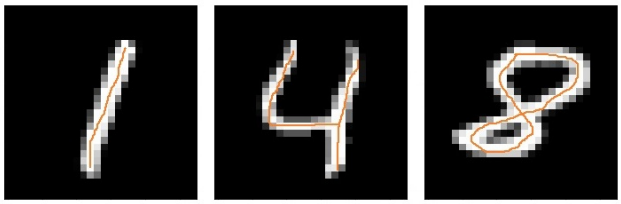


Figure 5: Different lengths of the digit.

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

Figure 6: Different contrasts.

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

Figure 7: Different tilts.

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

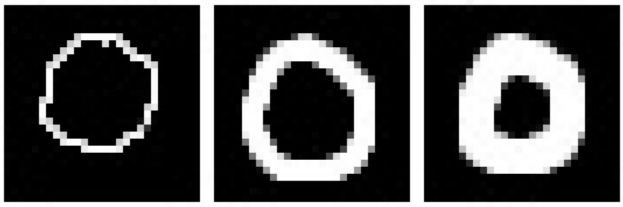


Figure 8: Different boldness.

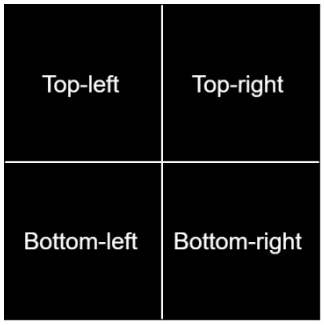
VII. **Handwriting:** There are a lot of hand writings proportional to the number of populations and this feature will vary among digits. So, a complete analysis will create several (10) digit specific problem domain. We are presenting some examples of different writing style of number '9'. In the next step, we will define some features and their values to describe these writing types.

Figure 8: Different types of writing '9'.

8.1.4 Selecting in-bound and out of bound areas:

In this section of analysis, we need to define the 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.

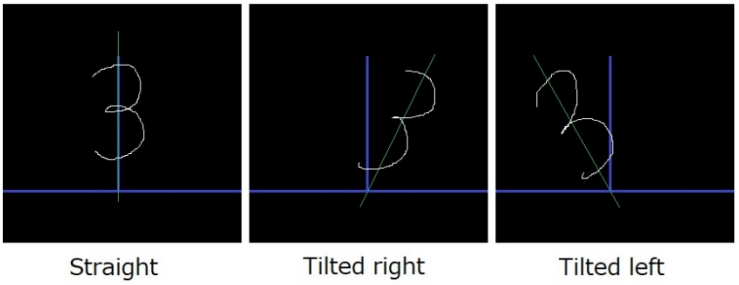
I. **Position of the digit:** The image frame can be divided into four basic regions like the figure below. This feature can be avoided if we can crop out the digit from any corner of the image.



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.

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.

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.



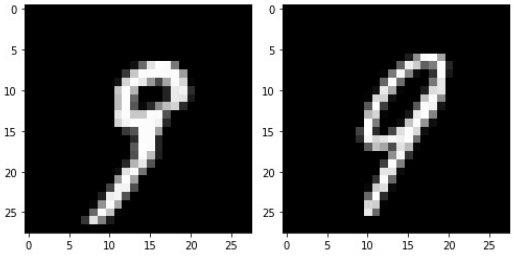
V. **Boldness:** This feature depends only on the ink width of the pen. Here, we will consider 3 levels of line thickness. Narrow, medium and wide.

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.

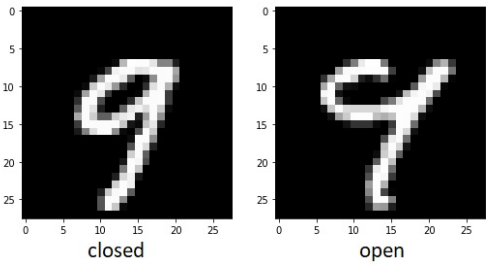
VI. **Handwriting:** This is not a single feature but a common identity for every digit specific problem domain. It holds as much as 10 separate domains varying in feature number as well as feature length. The detailed problem domain for digit '9' is described in this scope.

Based on different types of handwriting, we have come up with the following features that can occur while writing '9':

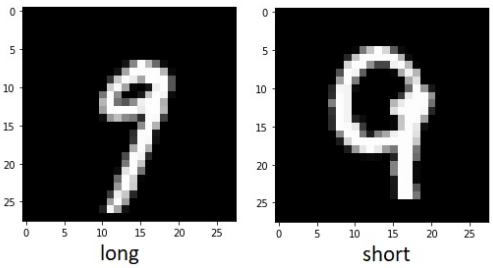
- **Loop size:**



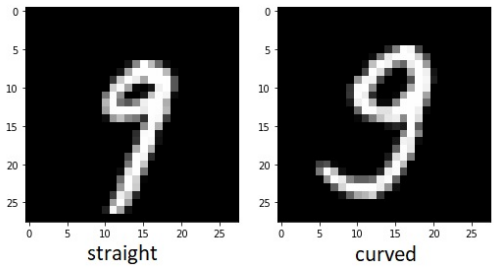
- **Loop ending:**



- **End line length:**



- **End line shape:**



8.1.5 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:

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 length: long, short (2) End line shape: straight, curved (2)

8.1.6 Identifying unsound (never to occur) cases:

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. 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,

$$Number\ of\ combinations = 4 \times 3 \times 2 \times 3 \times 3 = 216$$

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

8.1.7 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 build/design expected dataset for our defined problem domain. We will discuss data management, data augmentation etc.

8.2 Sufficiency of data design:

8.2.1 Definition:

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. The objective of this part of AIQM guideline, is to create or gather enough data points for every region of that feature space.

In order to do so, we can follow either of the following two.

- Build a new dataset
- Work with an existing dataset

8.2.2 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.

8.2.3 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.

8.2.4 Work with an existing dataset:

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

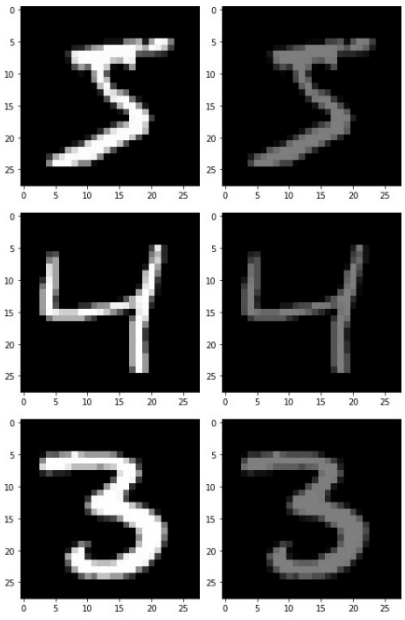
- We cannot expect data in all feature dimensions. We need to check the coverage in later section.
- We also cannot ensure uniform distribution across the feature space.
- Later, from 'Coverage' and 'Uniformity' analysis, we can find the actual distribution of the dataset and identify the cases where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

8.2.5 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 '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. Few sample examples are shown here.

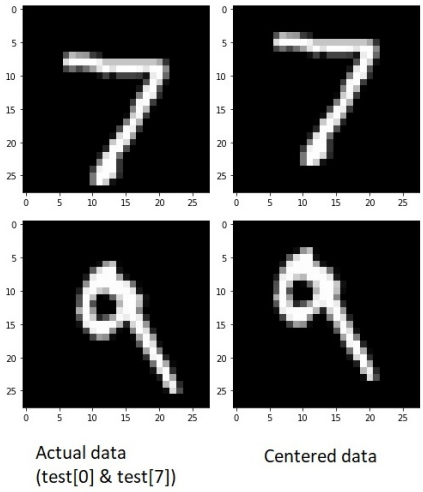


Data from MNIST | Modified low contrast data

- **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.



8.2.6 Conclusion:

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.

8.3 Coverage of dataset:

8.3.1 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.

8.3.2 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 28X28. According to our defined problem domain, 'Area' feature contains three levels of categories.

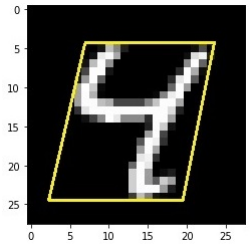


Figure 9: A sample bounding box; defines the measurement of 'Area' feature.

- 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.

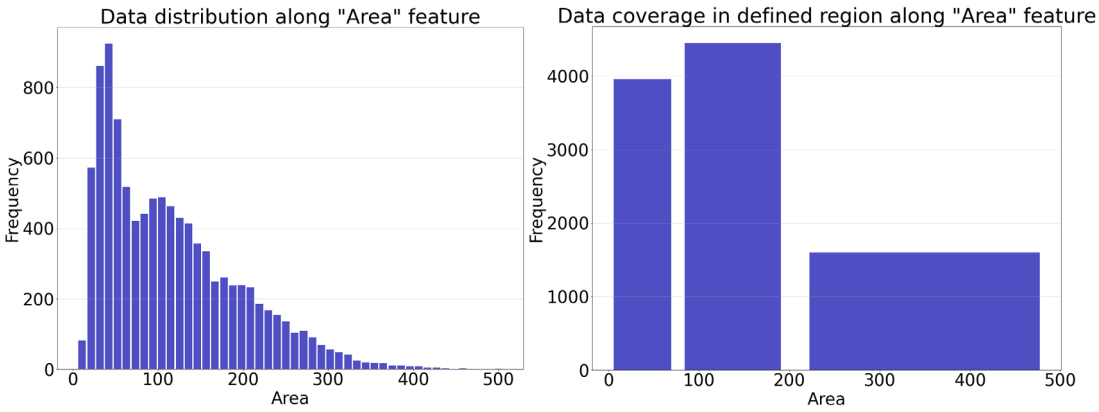


Figure 10: **(a)** Distribution along feature dimension and **(b)** Data coverage using defined boundary values; for 'Area' feature.

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 supporting documents.

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 $x \in dataset, for\ n^{th} feature,$

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

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

Coverage of test dataset for 'Area' feature has been calculated above. This coverage analysis is described as conventional coverage. Earlier researches 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 supporting documents.

8.3.3 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). [Reference: Guiding Deep Learning System Testing Using Surprise Adequacy; https://arxiv.org/pdf/1808.08444.pdf]

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.

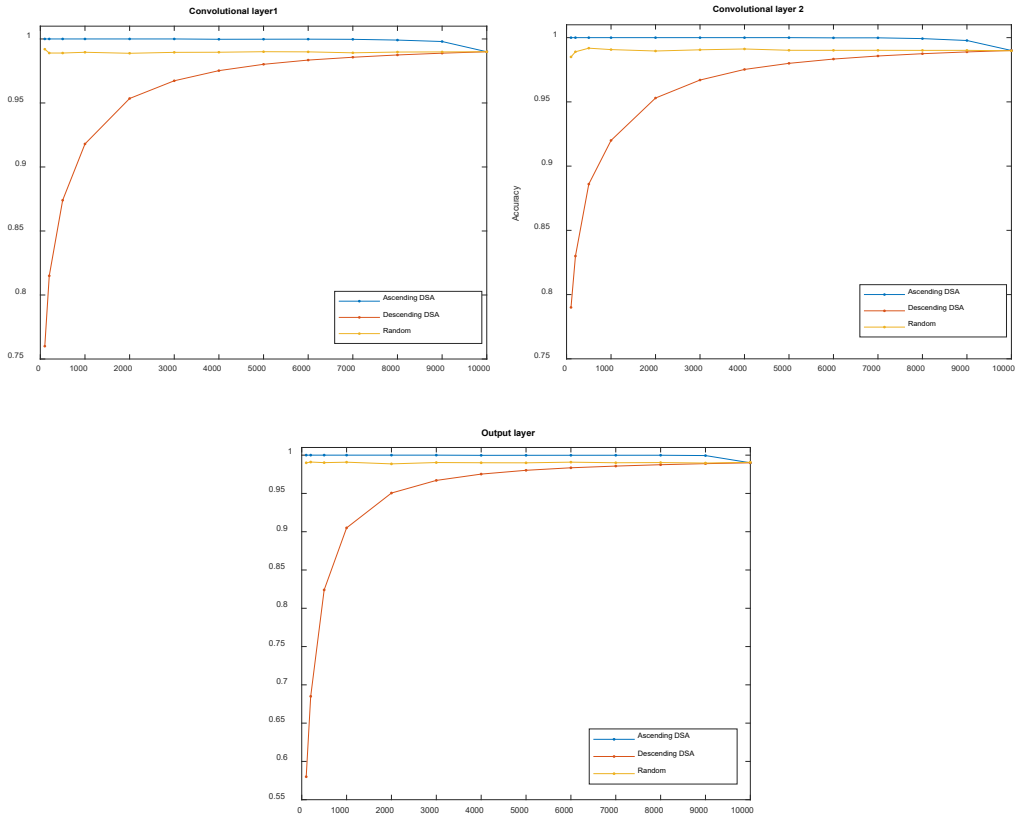


Figure 11: Changes in accuracy with the changes in DSA.

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.

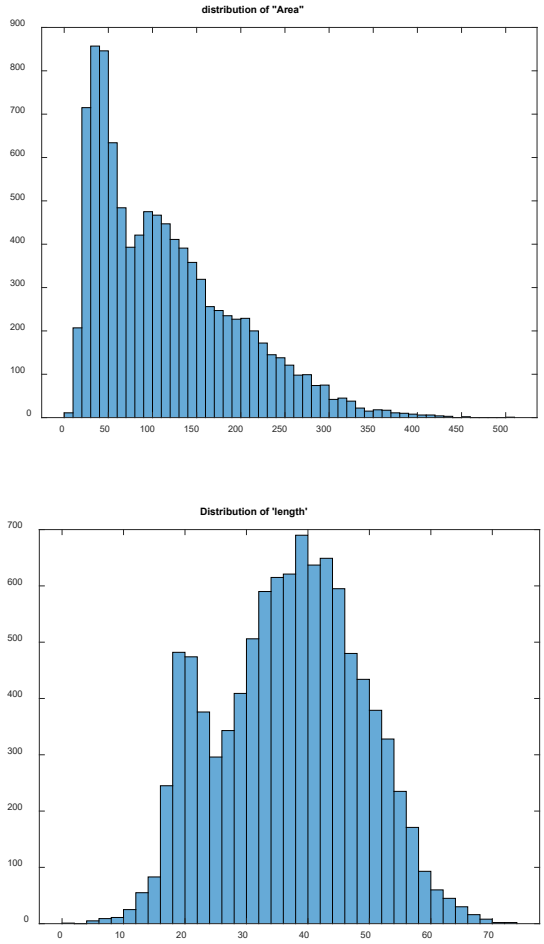


Figure 12: Distribution of MNIST (test dataset) along the feature 'Area' & 'Length'.

Table: Divisions of 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: Divisions of 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

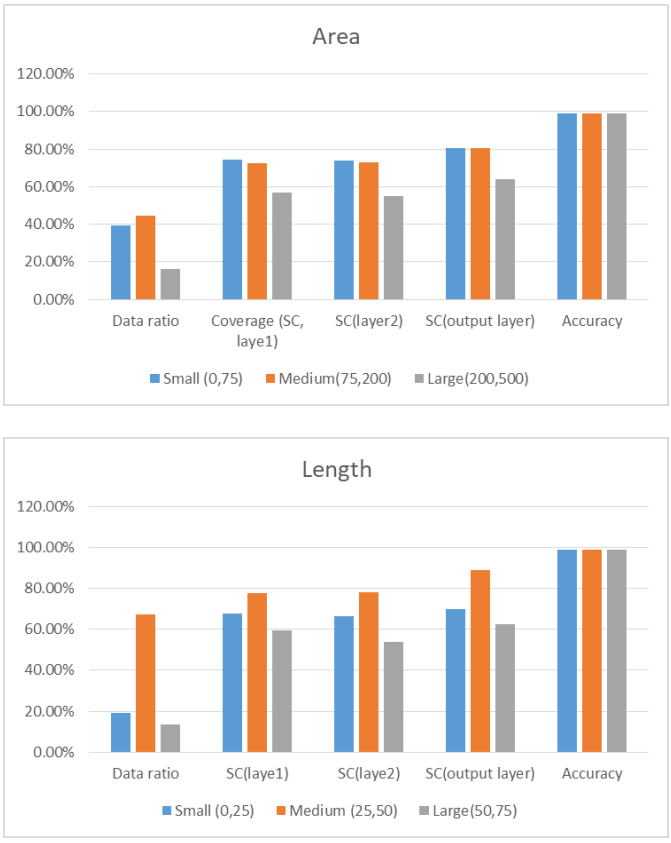


Figure 13: Graphical representation of SC and accuracy for each divisions of the features; 'Area' & 'Length'.

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

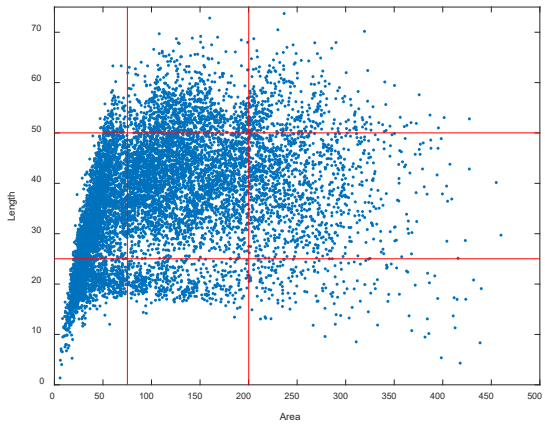


Figure 14: Data (MNIST; test data) distribution across 2-dimensional ('Area', 'Length') feature space.

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

Table: 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

8.3.4 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.

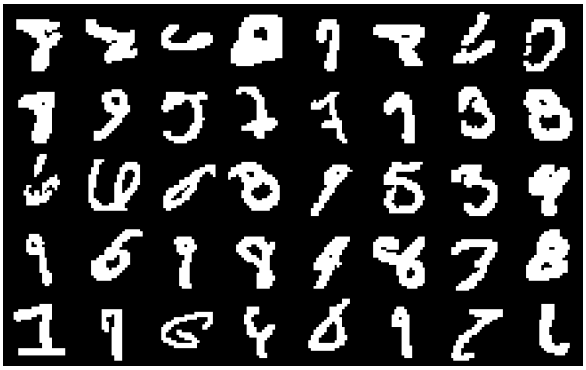
For example, from the above coverage calculation, we have found, there is no large-scale digit image present in the training dataset where there is a probability of getting such images in real time operation. So, large-scale digit images are corner cases.

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.

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

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.

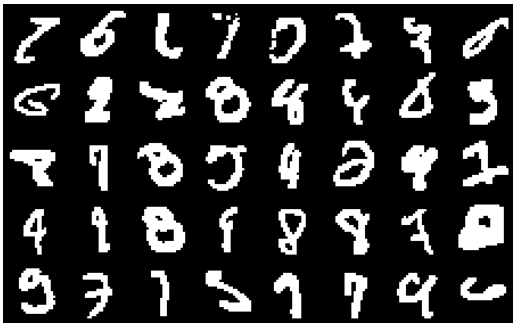
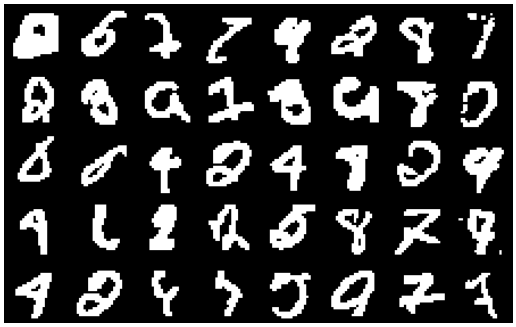
We have used the CNN described in the upper section. The testing accuracy is 99%, and there are total 100 images that are incorrectly classified. Referring to the accuracy vs DSA graph in Fig 11, it is seen when the values of DSA are high, the accuracy locates in [0.6-0.8], implying not all of outliers have high values of 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.

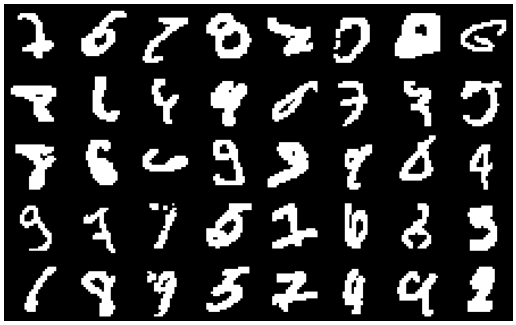
Images	Labels
	Actual label [8 2 6 2 7 8 6 0 8 9 5 7 7 7 3 8 6 6 0 8 9 5 3 4 9 5 9 8 4 8 3 8 1 7 6 9 0 9 7 6] Predicted Label (13) [7 7 4 0 9 7 6 7 7 9 7 2 3 9 5 8 6 6 8 0 9 5 3 9 1 6 9 9 4 8 7 8 1 9 5 4 6 1 8 1]

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 [Reference: Corner Case Data Description and Detection; https://arxiv.org/pdf/2101.02494.pdf]. 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.

	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 mostly corner cases.

8.3.5 Feature deletion:

Based on the coverage results, we need 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.

8.3.6 Conclusion:

This AIQM 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 need to rely on the previous section, 'data design'. In the following section, we will analyze the distribution of data i.e., uniformity/ evenness of dataset.

8.4 Uniformity of dataset:

8.3.1 Definition:

This AIQM 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.

8.3.2 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.

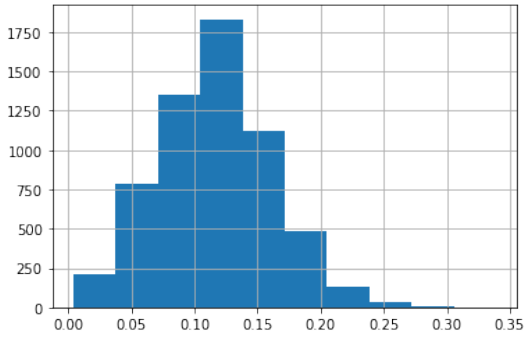


Figure 15: Data distribution based on digit boldness.

In the above distribution graph, x-axis holds % 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'.

8.3.3 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, 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.

$$TPCov[x(n)] = \frac{|\{S|density(S) = p\%\}|}{high\_n - low\_n}$$

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

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

If the value of EI_{err} is low, implying the data is evenly distributed, as shown in the following figure.

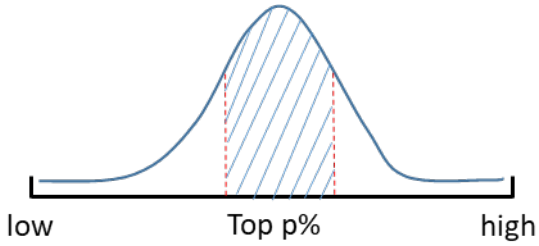


Figure 16: An approximate data distribution to aid visualization of evenness computation.

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

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

8.3.4 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.

8.3.5 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.

8.5 Correctness of machine-learning models:

8.5.1 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.

8.5.2 Different Accuracy measures/ Key performance indicator (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.

Confusion matrix

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

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,

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

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

$$Recall = \frac{TP}{TP + FN}$$

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

$$Precision = \frac{TP}{TP + FP}$$

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

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

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

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.

	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'.

8.5.3 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.

"Input Prioritization for Testing Neural Networks"

https://ieeexplore.ieee.org/abstract/document/8718224

8.5.4 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.

8.6 Stability of machine-learning models:

8.6.1 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. "The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions", the IEEE definition of robustness.

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)):

$$r = E(S) - E(\delta(S))$$

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.

8.6.2 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',

  $$\forall x' : \left|\left|x - x'\right|\right|_p = \delta \rightarrow h(x) = h(x')$$

  $\left|\left|*\right|\right|_p$ 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',

  $$\forall x, x' : \left|\left|x - x'\right|\right|_p = \delta \rightarrow h(x) - h(x') \leq \epsilon$$

  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,

$$r_1 = \frac{\delta_1}{0.5 \times d}, \quad ||x - x'||_1 = \delta_1$$

$$r_2 = \frac{\delta_2}{\sqrt{0.5 \times d}}, \quad ||x - x'||_2 = \delta_2$$

$$r_\infty = \frac{\delta_\infty}{0.5}, \quad ||x - x'||_\infty = \delta_\infty$$

There are several studies on robustness (δ) measurement. For example, 'CNN-Cert: AN Efficient Framework for Certifying Robustness of Conventional Neural Networks' (https://arxiv.org/abs/1811.12395)

'Towards Fast Computation of Certified Robustness for ReLU Networks' [Fast-Lin] (https://arxiv.org/abs/1804.09699)

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.

		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 5 layers 5 filters 10680 hidden nodes	$L_\square$ L_2 L_1	0.0305 0.1262 0.2482	0.0248 0.1007 0.2013	6.10 0.64 6.33	4.96 0.51 5.14

8.6.3 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',

$$\forall w' : \left|\left|w - w'\right|\right|_p = \delta_w \rightarrow h(x) = h'(x)$$

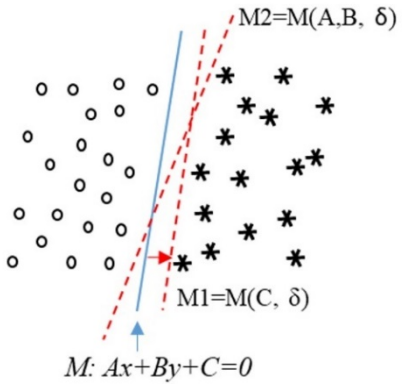


Figure 17: Changes in a linear classifier due to parameter perturbation.

Fig. 17 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 δ_{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.

- **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 $\mathcal{P}$), a set of mutation operators M (where each $m \in$ M is a function mapping $\mathcal{P} \rightarrow \mathcal{P}$), and a test suite T: $\mathcal{P} \rightarrow$ {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), $\forall m \in$ M, which both compile and pass T,

$$MutRB(P,T,M) = \frac{|\{P' \mid m \in M.P' = m(P) \cap T(P') = true\}|}{|\{P' | m \in M.P' = m(P)\}|}$$

This measurement can be transferred in AIQM. For example, if we take the results of DeepMutation as an example, where three studied DL models A,B and C are tested on MNIST dataset.
[DeepMutation: Mutation Testing of Deep Learning Systems; https://ieeexplore.ieee.org/abstract/document/8539073]

	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.

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

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 https://dl.acm.org/doi/abs/10.1145/3132747.3132785
- Guiding Deep Learning System Testing Using Surprise Adequacy https://ieeexplore.ieee.org/abstract/document/8812069
- Tensorfuzz: Debugging Neural Networks with Coverage-Guided Fuzzing http://proceedings.mlr.press/v97/odena19a.html

8.6.4 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 AIQM 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.

8.7 Soundness of components:

8.7.1 Definition:

According to AIQM guideline, the term 'soundness of components' 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. No AI solution is built from scratch. It 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.

8.7.2 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.

Programing language	Version
Python	3.6.12
Package	Version
NumPy	1.18.5
TensorFlow	2.3.1
Pillow (PIL fork)	8.0.1

8.7.3 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 regular 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.

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.

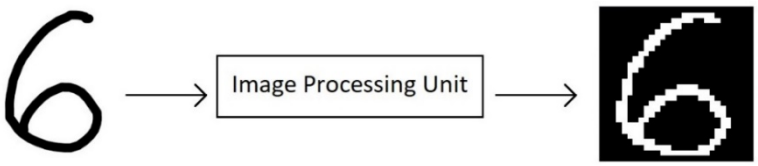


Figure 18: Modification on image by 'Image processing unit'.

Inside data preprocessing, python programing language has been used and open source package 'Pillow' has been used as image processing tool.

8.7.4 Unit for external methods:

In this part of the device, we define algorithm for external method to omit certain problem domain characteristics described in 'sufficiency of data design' section. 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.

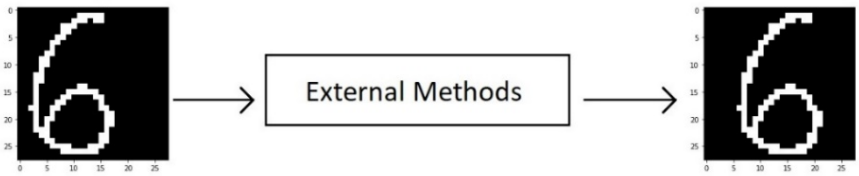


Figure 19: Modification on image by 'External methods unit'.

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.

8.7.5 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:** AI developers generally use Hierarchical Data Format (HDF) file (.h5) to store trained AI networks and their weights. In 'accuracy of machine-learning models', 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.

8.7.6 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.

8.7.7 Software security:

Security is a major concern when machine operates online. Here, we have described the examples of the theme that the solution designer should consider while building the application sets with AI/Machine learning functionality.

This document doesn't refer to the common aspect for the cyber security taken for the software that doesn't include the AI/Machine learning functionality also. (For the reference about the cyber security aspect doesn't include the AI/Machine learning, refer to the ISO/IEC 27000 series, NIST SP800 series, NIST Cyber Security Framework, ISO/IEC 15408 Common Criteria and so on).

Examples: The attack method is developed day by day. So, the following list is current examples. The solution designer should search the newest information

periodically. The annual assessment and the measures are recommended.

- **Adversarial example:** A machine learning technique that attempts to fool models by supplying deceptive input with small and intentional perturbations.
  "Explaining and Harnessing Adversarial Examples" Goodfellow et al.
  https://arxiv.org/pdf/1412.6572.pdf
  "Adversarial Examples in the Physical World" ICLR2017 Krakin et al.
  https://arxiv.org/pdf/1607.02533.pdf

- **Membership inference:** Given the huge number of the input, obtaining whether a data point is from the target model's training set or not.
  "Membership Inference Attacks Against Machine Learning Models" Shokri et al.
  https://arxiv.org/pdf/1610.05820.pdf

- **Poisoning:** Adversarial contamination of training data.
  "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning"
  https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8418594

- **Model inversion:** Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures
  https://dl.acm.org/doi/pdf/10.1145/2810103.2813677

- **Model extraction:** An attack in which an adversary utilizes a query access to the target model to obtain a new model whose performance is equivalent to the target model efficiently.
  "Stealing Machine Learning Models via Prediction APIs"
  https://arxiv.org/pdf/1609.02943.pdf

- **Backdoor attack:** With a specific trigger by additionally training the malicious training data, including the specific trigger to the DNN model, the DNN correctly recognizes normal data without triggers, but the network misrecognizes data containing a specific trigger as a target class chosen by the attacker.
  "Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering"
  https://arxiv.org/pdf/1811.03728.pdf

8.7.8 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.

8.7.9 Conclusion:

Above program components should be defined and assured for quality and security before the solution goes into operational phase.

8.8 Maintainability of quality during operation:

8.8.1 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 AIQM guideline, 'this section describes technologies to maintain internal qualities satisfied at the commencement of operation throughout the operation period'.

8.8.2 Task flow during maintenance:

Here, we will discuss various 'in operation' machine fails 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 Fig. 20, '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').

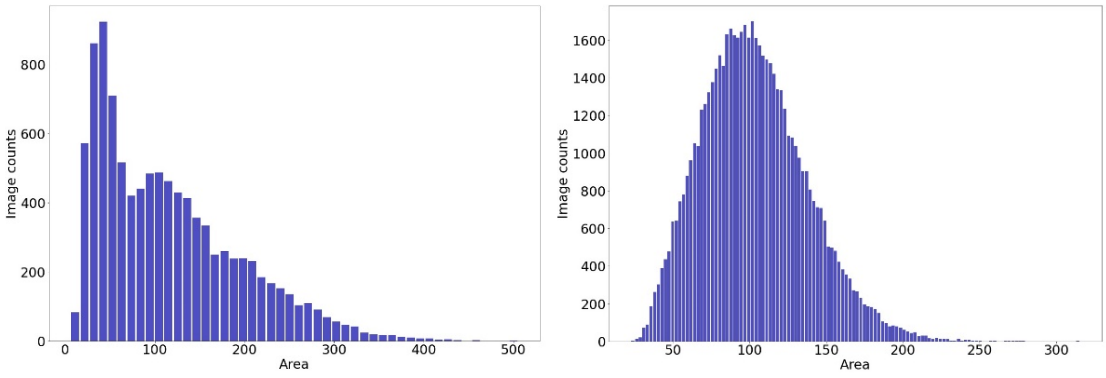


Figure 20: Data distribution based on 'Area' feature of train and test set respectively.

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.

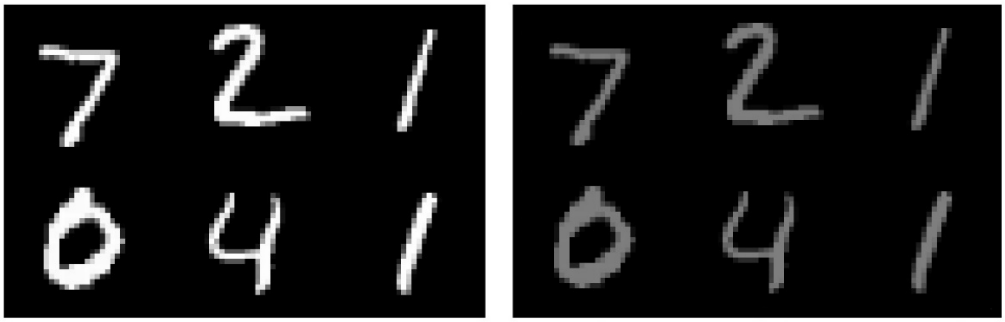


Figure 21: Example inputs from test_in2 and test_out2 respectively.

8.8.2.1 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 (Fig. 22).

Secondly, test_in2 is the complete test dataset having even distribution of digits and high contrast images. The following table shows the results of accuracy monitoring.

Test set	No. of images	Accuracy
test_in1	9,450	99.21%
test_in2	10,000	99.20%

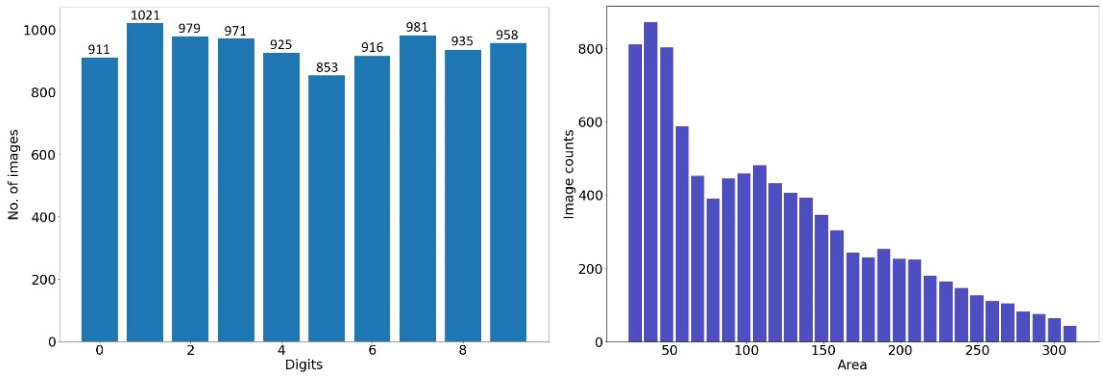


Figure 22: Class distribution and data distribution of test_in1 set.

8.8.2.2 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.

8.8.2.3 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 the following table, we can understand the effect of novel data on our trained model.

Test set	No. of images	Accuracy
test_out1	550	99.09%
test_out2	10,000	97.66%

8.8.2.4 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.

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%

8.8.2.5 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 (Reference: 'TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing' https://arxiv.org/pdf/1807.10875.pdf), a tool is described to find input that results Nan value.

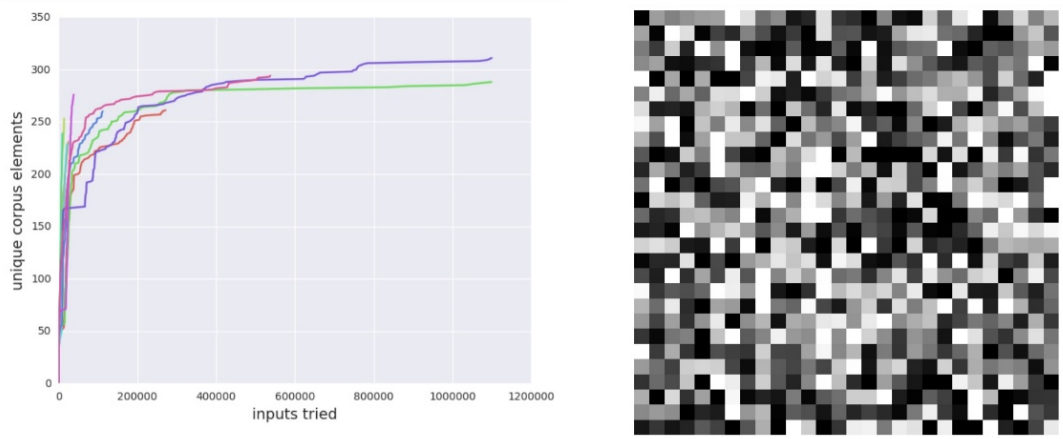


Figure 23: Left: the accumulated corpus size of the fuzzer while running, for 10 runs. Right: an example satisfying image found by the fuzzer.

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.
In Fig. 23, the image looks completely noise, but it did satisfy the condition of Nan output by the model. This sort of example is relevant if the AI solution runs on time scale; when there is no input, it will take noise. But if the solution is input dependent, this test is unnecessary.

8.8.2.6 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.

8.8.3 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.