Uncertainty-aware Metamorphic Testing for Robust Object Detection Models

Jianhong Wang[1], Yuta Ishimoto[1],
Masanari Kondo[1], Yasutaka Kamei[1], and Naoyasu Ubayashi[1]
[1] *Kyushu University, Fukuoka, Japan*
{wang, ishimoto}@posl.ait.kyushu-u.ac.jp, {kondo, kamei, ubayashi}@ait.kyushu-u.ac.jp

Abstract—Object detection models are widely used in safety-critical systems in industrial fields such as autonomous driving. It is essential to improve the robustness of object detection models capable of avoiding failures in realistic scenarios. This study proposes metamorphic testing on the object detection model robustness. The metamorphic testing generates metamorphic tested images that simulate the realistic perturbations in production environments. In order to evaluate the model robustness, Bayesian uncertainty is used to describe how the model maintains its reliability on the prediction. This paper shows how metamorphic testing performs the evaluation of the model robustness with Bayesian uncertainty, and how metamorphic testing can improve the model robustness by retraining the models.

Keywords–metamorphic testing, robustness, object detection model

1. Introduction

The object detection system is an important application system of computer vision in recent years. Object detection systems using models such as convolutional neural networks or transformers as the framework are widely used in various industrial fields.

However, object detection systems still face severe issues regarding reliability and safety. In real-world scenarios, it has been found that object detection systems are susceptible to small disturbances triggering failures [1, 2]. For instance, operational autopilot systems often overlook or misjudge sudden flying objects (birds) [3], resulting in temporary system failure. These failures are fatal and unacceptable. A robust object detection system needs to avoid failures as much as possible, and it can be achieved by improving the robustness of the model, which is pivotal in the object detection system.

Software testing can be used to verify the robustness of the object detection model. Wang and Su [4] applied metamorphic testing, a software testing technique, to object detectors. Metamorphic testing of object detection is proposed to test the performance of a model in a production environment, in which a synthetic image was synthesized from the original data image by inserting an object instance similar to the background, is transformed into metamorphic testing, which verifies whether the model maintains the predictability. However, existing studies [4, 5] have not discussed how metamorphic testing is used to improve the object detection model robustness.

This study aims to evaluate and improve the robustness of the object detection model using the metamorphic testing. The robustness of the model shows the model performance against realistic perturbations under the production environment [6]. In our research, we use *Bayesian uncertainty* [7] (how much confidence the model itself has in the prediction results) and *failure rate* (how often each image fails during metamorphic testing) to evaluate the model robustness with metamorphic testing. A robust model shall have lower uncertainty and failure rate.

Metamorphic testing evaluates the robustness by assessing the metamorphic relations. A metamorphic relation is established when the model still keeps the same prediction results before and after the tested images are transformed into their metamorphic images. In our experiments, the tested images and extracted instances are selected by their different characteristics from the whole dataset - COCO dataset [8], which is a large-scale object detection dataset with many features. During the metamorphic testing, models are tested by predicting both the original tested images and their metamorphic ones in order to see whether the model maintains the metamorphic relations or not. Metamorphic images that become unpredictable and break the metamorphic relations are called failed test cases. After the metamorphic testing, those failed test cases are used to create a new training dataset to retrain the model in order to improve its robustness. Based on this experiment, we aim to reconsider how to generate metamorphic tested images with more efficient test approaches in this study.

The rest of the paper is organized as follows. Section 2 briefly introduces the background knowledge of metamorphic testing for object detectors and Bayesian uncertainty for deep learning models. Section 3 describes the method workflow in detail. Experimental Setups are presented in Section 4. The results and analysis of experiments are explained in Section 5. Sections 6 and 7 discuss the remained questions and the validity of the research. Lastly, Section 8 comes to the conclusion.

2. Testing Object detection models

2.1. Object detection model

Current object detection models combine deep learning models with traditional image processing technology. An object detection model detects the location, contour, classes, and semantic information of various instances in an image. To evaluate the quality of an object detection model, it is necessary to test many aspects that are related to the prediction results. Our research mainly concentrates on the robustness of object detection models.

2.2. Testing robustness of deep learning models

Model robustness refers to how much model performance varies when using new data. Model robustness ensures the reliability and safety of deep learning models [9]. To ensure that a model is working as expected, it is critical to monitor and manage robustness. Therefore, almost all applications and

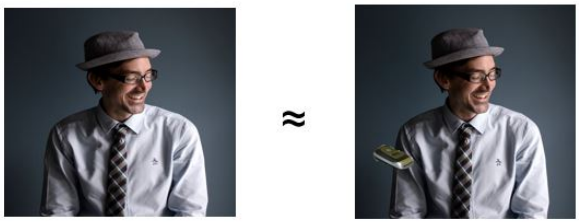

Figure 1: Synthesizing images by pasting an inserting instance. The left is the original image. The right with the cellphone pasted is the metamorphic image.

studies of deep learning models require *robustness testing*. Some robustness testing uses adversarial samples [10, 11]. Instead of this approach, we decided to adopt a metamorphic testing method for our research to test the object detection models' robustness in a realistic environment, which simulates a realistic attack in a production environment. This is because we aim to open a new research direction to assess the robustness of models.

2.3. Metamorphic testing

Metamorphic testing is a property-based software testing technique [12, 13]. It describes and evaluates system functionality through metamorphic relations [14]. A metamorphic relation ensures that the output keeps *consistency* between the original and metamorphic inputs (i.e., whether the predictions for these inputs remain unchanged). We made such metamorphic inputs by inserting various objects into the original inputs and assessed the consistency. In our research, we discuss the consistency of predictions based on three aspects of object detection models as follows:

- Class consistency: The predicted class before and after the insertions should be the same.
- Localization consistency: The predicted bounding box (the predicted rectangle containing the instance) should remain similar in shape and coordinates before and after the insertions.
- Recognition consistency: The consistency of whether the model still retains the ability of successful prediction.

The above are the consistencies for all aspects of the prediction of an object detection model in this study. The metamorphic relation is kept if all three consistencies are kept; otherwise, it is broken. When the metamorphic relations of predictions are broken, the model robustness is degraded.

Metamorphic testing is used to evaluate the performance of object detectors [4]. They explored the technique of instance segmentation to create a data pool of instances after extracting high-quality object instances from the images, as shown in Figure 1[1]. These instances are inserted into the original images and combined with the original images to synthesize new images. Hence, metamorphic relations between the model predictions are the following: all model predictions for the *metamorphic images* (images synthesized by pasting instances onto the original image during metamorphic testing) and the original image (the raw images without the insertion) should remain equivalent. They [4] use delta-debugging styled insertion to find locations where the metamorphic relations are broken. Delta-debugging styled insertion is constantly bringing the instances closer to the base instance's bounding box boundaries to be detected.

The advantage of metamorphic testing is that it reveals the vulnerabilities of the object detector against interference under a production environment. This is more relevant in reality than the other method using adversarial attacks, such as the Fast Gradient Sign Method (FGSM) [15]. We further improved the delta debugging-styled insertion using Grad-CAM [16], considering that many instances have unusual shapes and are not suitable as reference points merely by their boundaries. Moreover, to evaluate the model robustness for a single image without including the accuracy of the entire dataset, we mainly used two metrics as the evaluation criteria: Bayesian uncertainty, which is introduced in Section 2.4, and the failure rate, which is introduced in Section 3.4.

2.4. Bayesian uncertainty

Bayesian uncertainty [7] is derived from Bayesian neural networks and is used to evaluate the reliability of model predictions. Bayesian neural networks convert the determined parameters and predicted outcomes into corresponding probability distributions, where the variance of these distributions can be called Bayesian uncertainty. To simplify the complexity of deriving Bayesian uncertainty, the MC Dropout [17] treats the variance in the probability distribution of the final prediction outcome as uncertainty. During the model training process, variance is similarly embedded in the loss function for training.

Bayesian uncertainty is applied to object detection models [7]. This study uses a probabilistic model of object detection to enhance the training effect, improves the overall accuracy of the bounding box prediction, and obtains fewer incorrect predictions of class. In our research, we use MC Dropout to rewrite the YOLO [18], a type of object detection model, into Bayesian YOLO to derive uncertainty and use uncertainty as one of the evaluation metrics.

3. Our approach

3.1. Overview

In this section, we introduce the methods and the mechanism of metamorphic testing to evaluate the robustness of object detection models.

As shown in Figure 2, metamorphic testing performs the prediction of a series of *data characteristics* (characteristics of data that can be explored after the prediction) on *test cases* and stores the prediction result into *prediction databases*, which record the data characteristics of both successful and failed test cases detected by metamorphic testing. In our research, through metamorphic testing of object detection models, we analyze what data characteristics influence the robustness of models by evaluating the difference in the prediction uncertainty and failure rate. Moreover, we retrain the failed test cases to improve the robustness of the object detection model.

[1]©Attribution-NonCommercial 2.0 Generic (CC BY-NC 2.0)

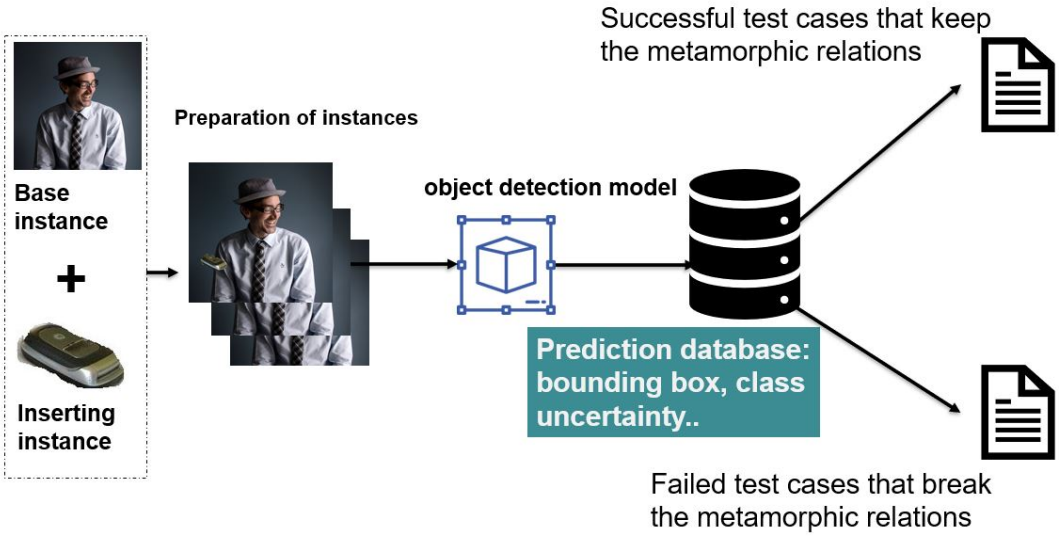

Figure 2: The metamorphic testing workflow: generation of databases of successful test cases and failed test cases.

3.2. Preparation of base and inserting instances

The data in our research are the instances in the image. In this study, we use two types of instances to synthesize metamorphic images: *base instances* and *inserting instances*. Base instances are instances to be tested. In all experiments, the model robustness is evaluated by observing the uncertainty and testing failure rate of the base instance, which are described later in the section. On the other hand, inserting instances are extracted from all images in the dataset by the instance segmentation model YOLACT [19] and pasted onto the background containing the base instance. Like in Figure 1, the man is the base instance, and the cellphone placed near the man is the inserting instance. For the inserting instances, the following criteria are adopted to extract high-quality inserting instances for each base instance used for metamorphic testing:

- Instance size: It is necessary to control the size of the inserting instance compared to the base instance.
- Uncertainty: We use YOLACT to derive the prediction uncertainty for all instances. Bayesian uncertainty can be approximately considered as the variance of the corresponding classes' probability distribution or bounding box localization's probability distribution in terms of *classification uncertainty* and *bounding box uncertainty*, respectively. For the following experiments, instances of different values of bounding box uncertainty and class uncertainty are used in metamorphic testing to evaluate the model.
- Scores of the instances (related to the ground truth): Scores describe the accuracy of the prediction. The scores of the instances also reflect the accuracy of the model's prediction on the instances w.r.t the ground truth. Similar to the uncertainty, the instances of different scores are to be used in metamorphic testing. It is calculated as:

$$Score = P(inst) * IoU * 100\% \quad (1)$$

$P(inst)$ is the predicted probability of the targeted instance' true category or bounding box coordinates. IoU represents the intersection over the union between the predicted bounding box and the ground truth bounding box.

- Semantic similarity: Maintaining a similarity between the image background and the inserting instance is necessary for simulating realistic scenarios. We used phash [20], an image hashing algorithm, to calculate the similarity.
- No occlusions: After recording the coordinates of the base instance and the inserting instance, the instances should be prevented from occluding with each other as much as possible.

For each base instance, the inserting instances are filtered and chosen by the above criteria; thus, each base instance corresponds to appropriate inserting instances. Besides, the uncertainty and scores of instances are considered as the data characteristics used for research questions, as described in Section 4.5.

3.3. Determine the location of inserting instances check

We extend a method of delta-debugging styled insertion [4] to determine the locations of inserting instances: gradually moving the inserting instances closer to the base instance from a distance. The existing method [4] determines the locations merely based on the bounding box, which is not always correct (e.g., triangular-shaped instances). In our research, it is noteworthy that CNN-based model prediction shows varying confidence for each pixel of the image, and the confidence from all pixels is aggregated to the image as a Grad-CAM heatmap [16]. Therefore, we use the Grad-CAM heatmap to better determine the location of insertion.

The delta-debugging styled insertion is used to gradually shift the instances closer to the contours. As shown in Figure 3, the warm-colored area of the heatmap has a high influence on predictions, while the cool-colored areas have no influence at all. We use the edges of the warm regions as contours and then sample a number of points on the contours. There is a

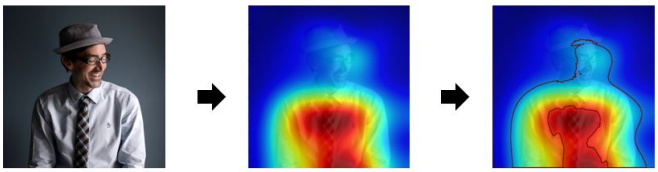

Figure 3: Generation of Grad-CAM heatmap's contours.

point among those sampled points with the closest distance to the inserting instance, which is the distance between the contour and the inserting instance. The delta-debugging styled insertion moves the inserting instances from a very remote place to the contour, with each time the distances halved (in a manner like 100%, 50%, 25%, ...).

3.4. Metamorphic testing workflow

In this section, the workflow of metamorphic testing is introduced.
(1) Prepare a data pool of instances including base and inserting instances after synthesizing the images.
(2) For each base instance to be tested, the metamorphic testing generates hundreds of metamorphic synthesized images, where the insertions are done in a delta-debugging styled way on different locations on the background of the images. Therefore, failure rate F_r is used to evaluate how frequently the metamorphic testing for one image fails, and it is one of the evaluation criteria of the model robustness:

$$F_r = \frac{N_F}{N}, \tag{2}$$

where N indicates the preset total number of metamorphic testing on one image and N_F indicates the number of failures of metamorphic testing on the image.
(3) Use the object detection model to predict all synthesized images and record the predictions (Bayesian uncertainty, scores) into the prediction databases. Then, compare the difference with the predicted result of the corresponding original tested image.
(4) The images that successfully predicted and maintained the metamorphic relations are classified as successful test cases, while the images that failed to be predicted or break the metamorphic relations are classified as failed test cases.
(5) By analyzing the variation of the uncertainty and the failure rate, metamorphic testing can be used to evaluate the robustness of the model according to data with different characteristics. Bayesian uncertainty describes whether the model has consistent confidence in predicting the same base instance. The failure rate shows the robustness of the object detection model against realistic attacks.

4. Experimental Setup

In this section, the experiment settings and two research questions are introduced.

4.1. Models in use

We use an instance segmentation model to extract the instances in the first step of metamorphic testing and another object detection model to conduct the metamorphic testing.

1) **Instance segmentation model**: YOLACT, the model used to extract the instances as described in Section 3.2.

2) **Object detection model**: Bayesian YOLO model, the model used to be trained, retrained, and derive the uncertainty and failure rate in metamorphic testing. This model is also to be retrained in RQ2.

4.2. Data in use

Data in our research refer to instances of two types: base and inserting instances. At the beginning of metamorphic testing, we prepare the base instance to be tested (test cases). Then, we use the criteria as described in Section 3.2 to choose inserting instances for them. All instances are from the COCO dataset, containing over 83 thousand train images and 41 thousand test images [8]. There are totally over 180 thousand instances that can be extracted from COCO test images and are available to be used as base instances in metamorphic testing.

Each research question uses different base instances (test cases) for different purposes, which are introduced in the later sections. Since the configurations in experiments for choosing the inserting instances are important, we adjust such configurations for each RQ.

The experiments are conducted on Google Colaboratory using PyTorch [21].

4.3. Prediction databases in metamorphic testing

After the metamorphic testing, the prediction databases, like in Figure 4, record the data characteristics such as categories, uncertainty, scores, and so forth. Prediction databases record the following information of data:
(1) Firstly, select the base instance whose scores are over the recognizable threshold (instances' scores greater than 0.7). Since the test cases need clear baselines to evaluate the model robustness in metamorphic testing, we only use the instances that are steadily detected by the model with scores over 0.70. Instances whose scores are below the recognizable threshold can not be steadily detected and provide the baselines for metamorphic testing. Hence, there are a total of 183,361 instances that can be regarded as tested images.
(2) Then, for each base instance, select its proper inserting instances from the instance pool by the criteria described in Section 3.2. For each base instance, there are variable numbers of available inserting instances for them to synthesize into a metamorphic image.
(3) Thirdly, conduct delta-debugging styled insertion, which determines the locations of different distances to the contour of base instances' Grad-CAM heatmap central region as described in Section 3.3. We set five levels of the inserting location to 0, 12.5%, 25%, 50%, and 100% distances at levels 1, 2, 3, 4, and 5, respectively. In short, these levels represent the closeness of insertion locations to the contour. Testing a base instance conducts 100 times of synthesizing new metamorphic images for each level, and there are totally hundreds of times of metamorphic testing on one single image. After that, the number of failure occurrences is recorded

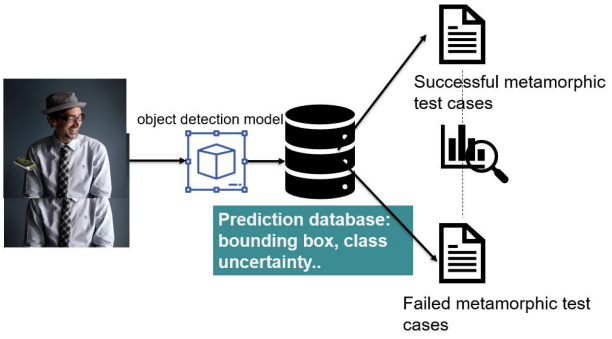

Figure 4: RQ1: experiment workflow.

and divided by the preset total number (e.g., 500 times) of metamorphic testing to calculate the failure rate F_r.

The prediction databases record the data characteristics of all these metamorphic synthesized images. They are used in both research questions.

4.4. Metrics in use

For most experiments, the model robustness is evaluated by Bayesian uncertainty Unc, which is introduced in Section 2.4, and failure rate F_r, which is introduced in Section 3.4.

4.5. Research questions

RQ1. What are the main data characteristics that influence the metamorphic tested results of the model?

The objective of this study question is to find whether certain characteristics in metamorphic testing, including uncertainty and scores, affect the test results of metamorphic testing. For testing instances with different data characteristics, the metamorphic relation of the model's predictions might be kept or broken. The model shows different robustness against data with different characteristics, and it deserves investigation of which data characteristics are more influential.

RQ2. How to improve the robustness of the model against realistic insertion attack after data augmentation with failed test cases found in metamorphic testing?

This study question aims to reorganize the training dataset by data augmentation with metamorphic images and retrain the model to improve its robustness in a production environment. Also, in this study question, we focus on the usage of the data characteristics in RQ1 to improve the effectiveness of retraining in terms of repairing existing failures and reducing the occurrence of new failures that do not occur in the metamorphic testing for the original prediction model, which is called *overfitting*.

5. Experimental Results

In this section, the approaches and results of the experiments for each research question are described.

5.1. RQ1. What are the main data characteristics that influence the metamorphic tested results of the model?

In this section, the influence of different data characteristics including uncertainty and score is investigated. Uncertainty is the property of the model that describes the model's confidence in the prediction. The score represents the accuracy of the model's prediction compared with the ground truth values. Through experiments, it deserves investigation on which is more influential to model robustness by counting the occurrence of failures in metamorphic testing.

TABLE I: Average instances' uncertainty and score of each preset level. *The intervals* show the equivalent numerical intervals divided from the numerical interval from the highest value and the lowest values. *Collected number* are the numbers of instances for each level of uncertainty or score.

The intervals	Average uncertainty	Collected number
all instances	2.214	183,361
0% - 20%	1.435	66,872
20% - 40%	1.973	51,634
40% - 60%	2.305	32,308
60% - 80%	2.456	27,860
80% - 100%	2.720	4,687

The intervals	Average scores	Collected number
all instances	0.9075	183,361
0% - 20%	0.7623	9,901
20% - 40%	0.8328	23,984
40% - 60%	0.8990	44,355
60% - 80%	0.9251	59,720
80% - 100%	0.9632	45,401

Approach: We analyze the relationship between the failure rate and two data characteristics (uncertainty and scores) of two types of instances (base instances and inserting instances) separately. For each experiment, metamorphic images (each image contains one base instance and several inserting instances) are synthesized by controlling the base instances of different data characteristics. Then, metamorphic testing on those different metamorphic images starts. The workflow of RQ1 is almost the same as the workflow introduced in Section 3, as shown in Figure 4.

1) Preparation of instances: Before the metamorphic testing, we prepare the instance pool for base instances and their available inserting instances. For instance, if performing the experiments on low uncertain base instances, we only synthesize metamorphic images with the low uncertain base instances. Then, we perform delta-debugging styled insertion to determine the insertion location. Finally, we synthesize the metamorphic images from the chosen base instances and insert instances to form a new dataset.
2) Prediction and testing: We feed the new dataset, which consists of the chosen metamorphic images and the original tested images, to the object detection model. In the meantime, the predicted uncertainty, score, and other information are recorded in the prediction database.
3) Comparative analysis: After obtaining the prediction databases of metamorphic images and the original tested images, we can analyze the influence of the data characteristics on the metamorphic relations of prediction. To evaluate the robustness, we use the failure rate F_r as the criteria and compare the difference between the successful metamorphic testing cases and failed ones

Lowest scores → Highest scores

Lowest uncertainty ↓ Highest uncertainty

Uncertainty\Score	0-20%	20%-40%	40-60%	60%-80%	80%-100%
0-20%	0.3675	0.2712	0.2478	0.1854	0.1446
20%-40%	0.2581	0.2336	0.2045	0.1832	0.125
40%-60%	0.225	0.2653	0.1729	0.1564	0.05
60%-80%	0.1935	0.2431	0.2167	0.1	0
80%-100%	0.2	0.33	0.1415	0(no data)	0(no data)

Figure 5: RQ1 results of 25 combinations of base instances and inserting instances. The numbers inside the table are the failure rate during the metamorphic testing. Each row represents the failure rate of instances of different numerical intervals of uncertainty. Each column represents different numerical intervals of scores. In fact, the amount of the high-uncertain data and low-scored data is small so there is no data with both high uncertainty and high scores, and it is acceptable to see some unexpected failure rate for those parts of the data.

in terms of their data characteristics. The workflow is demonstrated in Figure 4.

Results and analysis: For all base instances to be tested, we measure the uncertainty (both bounding box uncertainty and classification uncertainty) and scores. We computed the bounding box and classification uncertainties for each instance and found the bounding box uncertainty is a dominant uncertainty compared to the classification uncertainty after the realistic insertions. Hence, two types of uncertainty are deemed as one uncertainty by summing up these two uncertainties instead of considering them separately. Five levels of uncertainty or scores are set to separate all instances by their values of uncertainty or scores, which represent the numerical intervals of their values. Specifically, we divided the range of uncertainty and scores into five regions from the lowest to the highest value. For instance, as Table I shows, 0% - 20% of uncertainty represents the instances that have the lowest uncertainty (highest confident predictions from the model). 0% - 20% of scores represent the instances that have the lowest accuracy compared to the ground truth values.

As shown in Figure 5, there are 25 combinations of levels to classify instances. We experiment with all 25 combinations of uncertainty and scores of the base instances and collect the results of their failure rate F_r during metamorphic testing. Observation of the failure rate change can reveal which characteristics are more influential to model robustness.

Figure 5 shows that data with lower uncertainty has a higher failure rate when comparing each row in the same column. Also, data with high scores have a lower failure rate when comparing each column in the same row.

About RQ1: There is an obvious relation between the failure rate, uncertainty, and scores of the base instances. The instances with low uncertainty, in the metamorphic testing, have a higher failure rate compared to instances with high uncertainty. The score shows a negative correlation to the failure rate. High-scored instances are usually behaving well, whereas low-scored instances have a higher failure rate. For high-scored instances, the failure rate tends to be zero, especially when its instances' uncertainty is not low.

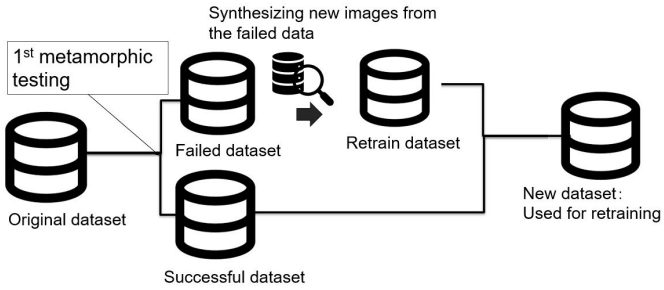

Figure 6: RQ2: generation of new training dataset

5.2. RQ2. How to improve the robustness of the model against realistic insertion attack after data augmentation with failed test cases found in metamorphic testing?

In this section, we aim to improve the robustness of the object detection model by retraining the model with failed data (failed test cases). As shown in Figure 6, we make new training datasets by adding the failed data with some changes to avoid failed test cases. Through all experiments, the goal is to find an appropriate retraining dataset using failed test cases found in metamorphic testing. We assume that retraining the model with synthesized images whose insertions are localized onto the *failure locations* (location on the background where failures in metamorphic testing occur) improves the robustness rather than retraining with images that are synthesized randomly. But this is an assumption; hence, it is necessary to prove that the retraining dataset is better at improving the robustness when its synthesized images are from the failed test cases but randomly synthesized images.

Approach: *Repairing the failures*: We aim to find an effective solution to making training datasets to improve the model robustness through retraining the model. Figure 6 illustrates the generation of new training datasets in this study question. The instances from the images are separated into the successful dataset and failed dataset, whose images pass or fail the metamorphic testing in the first time. Then, we synthesize new training data from the original images of the failed test cases and their prediction databases. The prediction databases for the metamorphic testing record the uncertainty, scores of the metamorphic images' instances, and the locations where the failures are triggered. Next, the new training dataset is fed into the prediction model to retrain. The retraining is done in a few epochs, with 1,000 to 3,000 epochs by YOLO. Ultimately, we put the retrained model into the second time metamorphic testing and see whether the robustness is improved.

Reduction of the overfitting: New failures are detected during the second time metamorphic testing. It is worthy of notice that retraining can not always reduce model prediction failures for all instances. As Table II shows, in all experiments, there are over 20% (Pr_3) of instances whose failure rate is increased after the retraining. There are probably unremovable new failures for each tested image. These are called overfitting for retraining. In the following experiments, we aim to reduce overfitting by two methods:

1) *Clusters* are defined as the locations where some failure

TABLE II: Retrain dataset with synthesized images of the different number of preset failure locations. Each column stands for the following: Unc: average uncertainty. F_r: average failure rate. Pr_1: proportions of data with almost no failures. Pr_2: proportions of data with fewer failures than the original prediction, excluding Pr_1. Pr_3: proportions of data with more failures than original prediction. The sum of these proportions equals 1. f and r represent the images synthesized by how many numbers of inserting instances from the failure locations (the lower table) or random locations (the upper table). The baseline row indicates the uncertainty and failure rate after the first time metamorphic testing before retraining.

	Unc	F_r	Pr_1	Pr_2	Pr_3
$0f$ + $1r$	2.351	0.136	0.286	0.322	0.393
$0f$ + $2r$	2.463	0.131	0.260	0.339	0.402
$0f$ + $3r$	2.438	0.141	0.250	0.378	0.383
$0f$ + $4r$	2.890	0.150	0.267	0.401	0.333
$0f$ + $5r$	2.692	0.165	0.308	0.328	0.364
baseline	2.314	0.142	-	-	-
	Unc	F_r	Pr_1	Pr_2	Pr_3
$1f$ + $0r$	2.130	0.133	0.395	0.284	0.325
$2f$ + $0r$	1.875	0.125	0.351	0.343	0.307
$3f$ + $0r$	2.027	0.103	0.331	0.405	0.262
$4f$ + $0r$	2.257	0.105	0.233	0.493	0.274
$5f$ + $0r$	2.325	0.114	0.221	0.468	0.324
baseline	2.314	0.142	-	-	-

locations in the test images are located closely. Other failure locations that do not cluster in test images are scattered randomly on test images. It is checked in the experiments to see if retraining with images that are inserted on the clusters of failure locations can further erase the new failures.

2) Because we use the data characteristics from prediction databases (uncertainty and scores) when synthesizing the image data, we intend to find the instances having specific combinations of data characteristics that reduce overfitting.

Results and analysis: As Table II shows, we investigate the variation of the uncertainty, failure rate, and repaired failures based on the proportions of instances with fewer failures. We compare the difference of these values between the random insertions and insertions on failure locations. It is obviously shown that retraining with synthesized images whose insertions on the failure locations performs better than the random locations. In the table, columns Unc and F_r show data characteristics, uncertainty, and failure rate. Columns Pr_1, Pr_2, and Pr_3 separately show the proportions of data about how the retraining reduces the original failures in metamorphic testing. Column Pr_1 is the best and the most effective part of the data after retraining the model, whereas Column Pr_3 is the part of data, in which retraining triggers more failures unexpectedly. Since inserting too many instances is likely to overlap with the base instance and the other inserting instances, in order to avoid this issue, a maximum of five times of insertions onto an image is set for each new training dataset.

As shown in Table II, the baseline of the failed tested data is of 2.314 uncertainty and 14.24% failure rate. It is apparent that the retraining images with randomly localized inserting instances cause higher uncertainty to the model predictions. With increasing numbers of random locations, the uncertainty roars up, which points out the increasing prediction variance. The proportions of data with more failures increase to 40% (Column Pr_3: $0f$ + $2r$ in the first table), which is more than the counterpart in the failure localized insertions (30% around).

Retraining datasets with synthesized data whose insertions are localized on the failure locations can improve the model robustness by lowering the uncertainty from 2.314 to 2.027 and failure rate from 0.142 to 0.103 ($3f$ + $0r$ in the second table). The failure rates, as seen in the second table of Table II are lower than the baseline, which suggests that the model after the retraining reduces the possible failures than the original model. Moreover, only increasing the number of failures localized inserting instances does not have a positive influence, as seen from the last rows of each table in Table II, on the improvement of model robustness. To improve the retraining, it is also crucial to control the number of instances inserted.

As seen in the results, it is inevitable that new failures occur due to the retraining of the model, which is seen as overfitting. In the next section, we introduce the results with the methods to reduce overfitting as described in the approach in this section.

1. Reduction of overfitting through the choices of failure locations.

In this method, we focus on the choice of failure locations of failed test cases. By observing the failed test cases (i.e., images), it is found that there are some specific locations on which the prediction of base instances is more likely to be affected during the metamorphic testing. Hence, we call a set of failure locations, which are located near each other, the *cluster* of the failure locations. Some of the images with the clusters are called *clusterable data*, whereas the others without the clusters are called the *non-clusterable data*. In the experiments, for clusterable data, we place inserting instances onto the failure locations' cluster rather than randomly choosing the failure locations, and synthesize the metamorphic images for retraining. For non-clusterable data, we randomly insert the inserting instances onto the failure locations. In the preliminary experiments, it is found that there are 36.4% clusterable data with 11.4% F_r, and 63.6% non-clusterable data with 18.9% F_r. It is investigated throughout all instances in the dataset about how many new failures occur due to retraining.

As shown in Table III, the overfitting (new failures due to the retraining) can be alleviated for the clusterable data, with new failures reduced from 2.66 to 1.84 (new failures in the second time metamorphic testing for C_c and C_r). Additionally, there are 35.4% of the clusterable data retrained, almost without new failures in the second time metamorphic testing. On the other hand, the failures of around 15% non-clusterable data are reduced no matter how the insertions are located on the images. In contrast, retraining clusterable data can reduce more failures than retraining non-clusterable data.

2. Reduction of overfitting of data with different characteristics of instances.

As seen from Table IV, low uncertain base instances show more repaired failures (3.451 per hundred), and fewer new failures (0.036 per hundred). This tendency is regardless of

TABLE III: Repairing the failures of the clusterable data and non-clusterable data for all instances of the category *animal*. The table demonstrates the changes in uncertainty and the number of failures in the second time metamorphic testing. The first row is the specified methods of inserting instances. C_c is the insertions into the clusters of clusterable data. C_r is the insertions into the random failure locations of clusterable data. Nc_n is the insertions into regions near the contours of non-clusterable data. Nc_f is the insertions into regions far from the contours of non-clusterable data. The baseline represents the average values of all data in the first time of metamorphic testing.

Insertion locations	C_c	C_r	Nc_n	Nc_f	Baseline
1. uncertainty	2.617	2.936	2.072	1.996	2.568
2. total number of failures (among 100)	7.52	11.41	8.15	7.43	12.39
3. repaired failures compared to the first time metamorphic testing	3.31	2.57	1.46	1.29	-
4. new failures in the second time metamorphic testing	1.84	2.66	1.62	1.37	-
5. proportions of data whose failures are mostly removed	23.61%	20.12%	25.61%	22.34%	-
6. proportions of data with almost no new failures	35.40%	19.50%	14.72%	16.88%	-

TABLE IV: New failures and repaired old failures for different instances used in second time retraining. Each cell shows the number of new failures and the parenthesis shows the number of total repaired failures. These numbers are the average numbers in 100 synthesized test cases.

new failures(repaired old failures)	low uncertain inserting instance	high uncertain inserting instance	low score inserting instance	high score inserting instance
low uncertain base instance	0.036 (3.451)	0.052 (3.447)	0.842 (3.325)	0.151 (2.632)
high uncertain base instance	1.825 (0.875)	1.469 (0.924)	1.536 (0.843)	1.057 (0.617)
low score base instance	1.215 (1.774)	1.226 (1.554)	2.536 (1.197)	2.057 (0.383)
high score base instance	1.657 (2.148)	1.832 (1.843)	1.317 (1.486)	3.057 (2.57)

the uncertainty of inserting instances. Comparatively, high uncertain base instances generate more new failures (1.825 per hundred) when inserting low uncertain inserting instances. Also, the number of new failures is larger than repaired failures.

> About RQ2: Retraining the model with the synthesized images inserted instances from the failure location rather than from the random location improves the robustness of the model. However, it is difficult to avoid overfitting. The low uncertain base instances are more easily repaired than the high uncertain ones in terms of the reduction of the old failures and the addition of new failures. Moreover, to avoid overfitting, it is necessary to make choices of failure locations and inserting instances.

6. Discussions

6.1. About RQ1: The relation of failure rate and uncertainty reflects the latent problems of the models.

The uncertainty can be explained by the prediction of variance derived from the object detection model itself. Thus, it is unstable for an object detection model to evaluate robustness merely by uncertainty. Given our results, the low-uncertainty instances highly trusted by the model are not reliable. The instability of the model when predicting low-uncertainty instances during metamorphic testing shows that a self-confident model requires further training to improve its robustness.

In the COCO dataset, the highly uncertain instances were mostly from images with sophisticated semantic contexts (with many instances located at every corner). They were not vulnerable to realistic attacks during metamorphic testing. Thus, when retraining highly uncertain instances, the failure rates for some metamorphic images should be lower than those of the original tested images. The object detection model shows unstable predictions for highly uncertain instances.

6.2. About RQ2: Repairing through retraining is effective for only part of the data.

Model retraining is a commonly used method to improve it. However, the retraining requires practical solutions. Currently, the questions discussed in RQ2 relate to the data characteristics of a realistic attack, such as those in the production environment. The detection of instances in an image is complex. Some data improved after retraining, whereas others retained their failures in metamorphic testing. From the results of RQ2, we found that the low-uncertainty data, which are highly trusted by the model, deserve further retraining to improve model robustness.

7. Threats and Validity

7.1. Construct validity

The metamorphic testing requires instances that can be detected with low uncertainty. Most of the instances to be tested can be easily detected by the object detection model. For the undetectable instances with low scores, metamorphic testing is not effective in improving the robustness of the object detection model. There remains the issue of how to utilize the undetectable instances to improve the current metamorphic testing.

7.2. External validity

Instance insertion in our current metamorphic testing can not be applied to the dynamic object detection scene, such as object tracking. The influence of the inserting instance might have a great difference between static images and dynamic images (videos). To ensure the validity of metamorphic testing for object detection, it is necessary to validate our metamorphic testing in a dynamic scene.

8. Conclusion

In this paper, metamorphic testing to evaluate and improve the robustness of object detection models is discussed. Metamorphic testing asserts the metamorphic relations of

object detection model prediction to evaluate the robustness. Our results show that the metamorphic testing reveals the differences in the robustness of the models due to the input data characteristics, including the uncertainty and scores. Retraining the model with data augmentation of failed test cases improves the model robustness by repairing the failures and preventing new failures.

In future research, we will find more effective methods to augment the training dataset to improve the model robustness. The data characteristics are limited to describing all realistic scenarios since it is essential to improve the model robustness for highly uncertain instances.

ACKNOWLEDGMENT

This study is partially supported by JSPS KAKENHI Japan (Grant Numbers: JP20H04167).

REFERENCES

[1] D. Li, J. Zhang, and K. Huang, "Universal adversarial perturbations against object detection," *Pattern Recognition*, vol. 110, p. 107584, 2021.

[2] S. Guo, S. Wang, Z. Yang, L. Wang, H. Zhang, P. Guo, Y. Gao, and J. Guo, "A review of deep learning-based visual multi-object tracking algorithms for autonomous driving," *Applied Sciences*, vol. 12, no. 21, p. 10741, 2022.

[3] U. Nepal and H. Eslamiat, "Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs," *Sensors*, vol. 22, no. 2, p. 464, 2022.

[4] S. Wang and Z. Su, "Metamorphic object insertion for testing object detection systems," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 1053–1065.

[5] Z. Zhang, P. Wang, H. Guo, Z. Wang, Y. Zhou, and Z. Huang, "Deepbackground: Metamorphic testing for deep-learning-driven image recognition systems accompanied by background-relevance," *Information and Software Technology*, vol. 140, p. 106701, 2021.

[6] H. Zhang and J. Wang, "Towards adversarially robust object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 421–430.

[7] A. Harakeh, M. Smart, and S. L. Waslander, "Bayesod: A bayesian approach for uncertainty estimation in deep object detectors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 87–93.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. Ieee, 2017, pp. 39–57.

[10] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Towards proving the adversarial robustness of deep neural networks," *arXiv preprint arXiv:1709.02802*, 2017.

[11] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig *et al.*, "Adversarial robustness toolbox v1. 0.0," *arXiv preprint arXiv:1807.01069*, 2018.

[12] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 1–27, 2018.

[13] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," *arXiv preprint arXiv:2002.12543*, 2020.

[14] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[17] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[19] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9157–9166.

[20] X.-m. Niu and Y.-h. Jiao, "An overview of perceptual hashing," *ACTA ELECTONICA SINICA*, vol. 36, no. 7, p. 1405, 2008.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch. nips-w 2017."