Implementation and Validation of non-semantic Out-of-Distribution Detection on Image Data in Manufacturing

Meng Jin
Fraunhofer Institute for
Industrial Engineering IAO
meng.jin@iao.fraunhofer.de

Mehta Dharmil Rajesh
Fraunhofer Institute for
Industrial Engineering IAO
dharmil-rajesh.mehta@iao.fraunhofer.de

Wenting Xing
Audi AG
wenting.xing@audi.de

Johannes Schniertshauer
Audi AG
johannes.schniertshauer@audi.de

Jens Neuhüttler
Fraunhofer Institute for
Industrial Engineering IAO
jens.neuhuettler@iao.fraunhofer.de

Felix Zimmermann
Fraunhofer Institute for
Industrial Engineering IAO
felix.zimmermann
@iao.fraunhofer.de

Abstract

Small changes in the production environment can have a negative impact on the performance of machine learning models. This study investigates the feasibility of various methods for detecting non-semantic Out-of-Distribution (OOD) cases in input images, which could be caused by hardware-side malfunctions, such as a defective camera flash. For this purpose, we design four experiments based on a real-world computer vision use case to simulate hardware problems that may occur in manufacturing and verify the performance of the various methods for detecting OOD cases. Furthermore, we explore the optimal sample size of input data to ensure that OOD cases can be found efficiently and successfully. The experimental results show that the tested methods can effectively and correctly detect the presence of non-semantic OOD data. The next step is to focus on securing ML models to identify malignant OOD cases, which negatively affects the performance of deep learning models.

Keywords: manufacturing, computer vision, AI Security, Out-of-Distribution detection

1. Introduction

Deep learning (DL) techniques have been proven in recent years to handle complex, large, and high-dimensional image datasets and yield significant benefits at various computer vision tasks in manufacturing, such as process optimization, condition monitoring, quality control, etc. [6][18]. With the increasing use of DL solutions in manufacturing, it is vital to consider the safety, reliability and robustness of the entire AI (Artificial Intelligence) system. In 2019, the European Commission published an ethics guideline for Trustworthy AI [7], which sets out a framework for achieving Trustworthy AI on three levels: legitimacy, ethics, and technology. At the technical level, the factors that should be considered while developing an AI systems are accuracy, reliability, reproducibility, data privacy and data quality [7]. This work focuses on the quality assurance of data in DL, which is a widely utilized field in AI services.

Highly accurate and stable results of DL tasks rely heavily on the diversity of training data. Usually the test set which is used to test the performance of the trained model should be in the distribution of the data generated from real world scenarios, and the training data could be more diverse. In this paper, we assume that the training data, test data and production data all come from the same distribution. There are unpredictable scenarios when deploying DL models in manufacturing, such as camera hardware failures or noise in the transmitted images, which can generate out-of-distribution (OOD) data in input data and cause significant degradation in the performance of the DL model. This property is also reflected in practice. Hardware issues are a major

URI: https://hdl.handle.net/10125/103441
978-0-9981331-6-4

challenge in the operation of industrial computer vision solutions. Discussions with experts have shown that misclassification in daily operation are mainly due to hardware problems such as incorrect focus of cameras or lighting problems. Therefore, detecting the presence of OOD in datasets from real environments is crucial to secure the system, alert users and inform developers to retrain the model if needed [13].

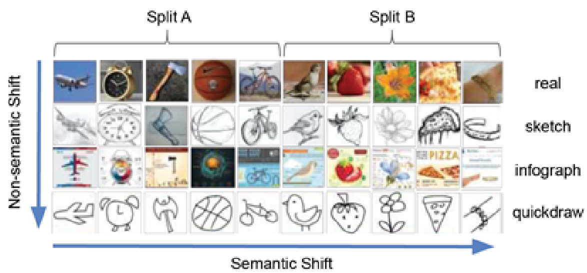

Figure 1. An example of semantic shift and non-semantic shift [13]

There are two categories of OODs, namely semantic and non-semantic OOD [13]. As can be seen from Figure 1, semantic OOD refers to two distributions from different semantics, such as Split A and Split B in Figure 1. Non-semantic OODs are two distributions from the same semantics but with varied manifestations. Especially for those DL tasks applied in manufacturing, they run in a strict and single hardware environment. Non-semantic OOD is more likely to be present once some changes in the hardware environment occur, such as changes in camera parameter settings, flash damage or sensor errors. Such data drifts are tiny and harder to detect than semantic OODs.

In this paper, we pay attention to detecting the presence of the non-semantic OOD (short for "drift" later) of input data for DL models, which will help people quickly locate errors occurring in the image acquisition to avoid the model suddenly producing unusual outputs. For this purpose, an effective approach proposed in [17] is found to detect non-semantic OOD data in high-dimensional image datasets by combining different dimensionality reduction methods and hypothesis tests. The OOD detection is an unsupervised method and requires no prior labels. The main contribution of our work is to experimentally test its performance and feasibility in detecting the presence of non-semantic OOD data for hardware assurance and validate it on one deep-learning-based use case in the press shop of a German OEM. Furthermore, we explore the optimal sample size of input data to ensure that all OOD cases can be found efficiently and successfully in input images.

The remainder of this paper is organized as follows. In Section 2, we discuss related work in OOD detection and present the advantage of our used approach. Section 3 describes the tools and workflow of drift detection. Section 4 depicts the setup and results of four experiments based on a real-world use case. These experiments reflect the contingency of some real hardware problems that could happen in the press shop. The final section discusses the conclusion and future work.

2. Related Work

OOD detection has been studied for several decades. Some Statistical Process Control Methods, such as Drift Detection Methods (DDM) [9], Early Drift Detection Methods (EDDM) [2] and Page Hinckley (PH) [15] methods, were proposed in the early 2000s and applied to detect OOD in online learning models. The authors in [9] proposed a formula to calculate the changes in the distribution of data sequence according to the error rate, e.g., the probability of misclassification. In EDDM, the distance between two errors classification is used instead of the number of errors [2]. Page-Hinckley (PH) detects drift when the difference between the observed and the mean is more than the threshold [15]. These methods can classify data as standard, warning or error and are suitable for detecting gradual, abrupt, and incremental drifts. However, they are ideal for checking drifts in low-dimensional, unbounded data streams and require the labels of original class in advance. [4] introduced a method for finding drift based on time windows that are dynamically controlled. The windows shrink when a drift is detected and grow when no drift is present. The disadvantage of this approach is that the data needs to be stored in memory when there is no drift in the data.

In recent years, with the widespread application of deep learning techniques, some methods for detecting OOD in high-dimensional images have been investigated. In 2021, in a survey on OOD detection methods published in [22], the authors summarized various methods for detecting OOD, such as Post-hoc Detection [20][12], Confidence Enhancement [14], outlier exposure [8], etc., which are applicable to detecting semantic OOD in multidimensional images. Another approach by combining statistical hypothesis testing with feature extraction methods for detecting drift in high-dimensional image distributions was proposed in [17]. The performance of this method in recognizing semantic OODs was validated on an academic benchmark [22], using CIFAR-10 as In-Distribution (ID) in training and distinguishing CIFAR images from other datasets such as MNIST [17].

3. Methodology

This section describes the various methods for implementing a drift detector. Benefiting from statistical hypothesis testing, the method proposed in [17] for detecting OODs in high-dimensional image distributions can be applied to any type of ML model with unlabeled data. Consequently, we employ and validate this approach, i.e., by combining multiple methods of dimensionality reduction and statistical hypothesis testing to fulfill the two tasks: (1) identifying non-semantic OODs, which are more subtle and difficult to detect than semantic OODs (2) validating the performance of detecting non-semantic OOD detection in a real-world DL use case in manufacturing.

Figure 2 depicts the workflow of drift detection based on Auto-Encoder method, where the left phase converts high-dimensional input images to low-dimensional data, while extracting and retaining the image's essential features. The right-side stage utilizes the low-dimensional data to detect drift in test data. Here, statistical two-sample hypothesis testing is utilized to examine whether the distributions of test data and training data are the same. The null hypothesis is that the distributions of the two datasets are identical. The conventional significance level 5% is chosen in our research. The null hypothesis can be rejected if the calculated p-value is less than 5%, which indicates that there is a high probability that the distribution of the test data does not match the distribution of the training data, i.e., there is a drift in the test data. Next, we describe the specific methods used for the dimensionality reduction and statistical hypothesis testing.

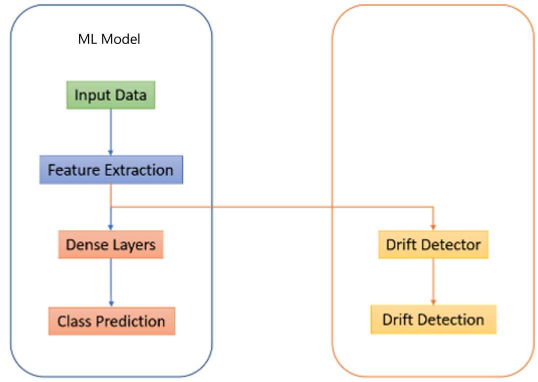

Figure 2. Workflow of drift detection based on Auto-Encoder method

3.1. Feature Extraction

The input data of the computer vision algorithms are images and videos. Best practices to compare the distribution of high-dimensional data have not yet been established. Therefore, we first use feature extraction methods to reduce the high-dimensional image to low-dimensional data. Feature extraction is essentially a dimension reduction process. The following dimensionality reduction techniques are employed.

3.1.1. Untrained and Trained Auto-Encoder (UAE and TAE) An Auto-Encoder is a non-linear dimensional reduction method consisting of an encoding and a decoding part. The encoding part converts high-dimensional raw input data to low-dimensional encoded data. The decoding part transforms the low-dimensional encoded data to reconstruct the high-dimensional data [21]. In drift detection, we only implement the encoding function of the Auto-Encoder and perform statistical hypothesis tests on the low-dimensional encoded data.

3.1.2. Black box shift detection (BBSDs and BBSDh) In black box shift detection, the dimensionality reduction step is achieved by utilizing the outputs of a DL model. There are two possible options. The first one is to use the softmax outputs from DL model (Black Box Shift Detection BBSDs) and the other option is to use the hard-thresholded predictions (BBSDh). [16].

3.2. Statistical Hypothesis Testing

We implement multi-variate Maximum Mean Discrepancy (MMD) Test [5] and multiple univariate Kolmogorov-Smirnov (KS) Test [3] plus Bonferroni Correction in hypothesis testing.

3.2.1. Multi-variate Maximum Mean Discrepancy (MMD) Test MMD is a kernel-based statistical test to estimate whether two distributions are equal. It calculates the distance between feature means. A feature map Φ maps X to kernel Hilbert space τ [5].

$$k(X,Y) = \langle \phi(X), \phi(y) \rangle_\tau \tag{1}$$

Given a probability P on X, the feature means μ_P maps $\Phi(X)$ to the mean of every coordinate of $\Phi(X)$.

$$\mu_p(\Phi(X))) = \left[E\left[\Phi(X_1)\right], ..., E\left[\Phi(X_m)\right]\right]^T \tag{2}$$

Inner product of feature means of X and Y can also be written as a kernel function such that:

$$\begin{aligned} &\langle \mu_p(\Phi(X)), \mu_q(\Phi(X)) \rangle_\tau \\ &= E_{p,q}\left[\langle \Phi(X), \Phi(Y) \rangle_\tau\right] \\ &\quad = E_{p,q}\left[k(X,Y)\right] \end{aligned} \tag{3}$$

MMD is the distance between feature means of X, Y that is of μ_p, μ_q:

$$MMD^2(p,q) = \|u_p - u_q\| \tag{4}$$

By using the norm of the inner product, the Equation 3 becomes:

$$\begin{aligned} MMD^2(p,q) &= \langle u_p - u_q, u_p - u_q \rangle \\ &= \langle u_p - u_p \rangle - 2 \langle u_p, u_q \rangle \\ &+ \langle u_q, u_q \rangle \end{aligned} \tag{5}$$

We calculate unbiased estimate of the squared MMD using samples from both distributions as follows:

$$\begin{aligned} MMD^2(X,Y) = \frac{1}{m(m-1)} \sum_i \sum_{j \neq i} \kappa(x_i, x_j) - \\ \frac{1}{mm} \sum_i \sum_j \kappa(x_i, y_j) \\ + \frac{1}{m(m-1)} \sum_i \sum_{j \neq i} \kappa(y_i, y_j) \end{aligned} \tag{6}$$

Using squared exponential kernel:

$$\kappa(x,y) = e^{-\frac{1}{\sigma}||x-y||^2} \tag{7}$$

Then p-value is obtained by performing permutation test on the resulting kernel matrix.

3.2.2. Kolmogorov-Smirnov (KS) Test plus Bonferroni Correction KS Test is a non-parametric test that compares K dimensions independently [3]. Z is the maximum absolute difference between two empirical cumulative distribution functions $F_n(x)$ and $G_m(x)$. We perform the KS Test K times and calculate p-values for each test. It is difficult to decide whether there is a drift in overall data because we don't know which p-value to select. Hence Bonferroni correction is required. It rejects the null hypothesis when minimum p-value is less than α/K.

$$Z = sup\left|F_p(z) - F_q(z)\right| \tag{8}$$

4. Experiments and Results

In this section, we examine and validate whether a combination of the various techniques discussed previously can detect non-semantic OOD cases in input images. If it can recognize drift due to hardware issues, such as camera hardware failure or environmental changes, it can be incorporated into the AI service as an alarm system, improving the stability of the AI model's operation. To validate the performance of this approach, we choose a DL-based classification task from a German OEM press shop as a case study and construct four experiments to imitate various forms of non-semantic OOD cases in input data.

We use our own built cloud platform for our experiments, each user can use 8 Inter CPU cores and a NIVIDIA Quadro RTX 5000 GPU. All experiments below are done in sequence on this cloud platform by using Jupyter Notebook and Pytorch.

Our experiments are carried out on the specific dataset from one case study in the press shop. Three types of cameras were used to take a total of 6131 photos, which were divided into two categories based on production quality: normal parts (3,345 photos) and faulty parts (2,786 photos). The ratio of these two classes is artificially set for the purpose of training and testing the AI model and does not reflect the actual press shop condition. A common practice for developing deep learning models is to divide the dataset into two parts: training, testing. The training dataset is used to fit the model, while the testing dataset is used to evaluate the performance of the model. In the following experiments, the training data are always treated as In-Distribution data, while artificial noise is added to the testing data to randomly generate some non-semantic OOD cases.

The first step of our approach is to convert high-dimensional images into low-dimensional feature data using a network previously trained for this classification task with an F1 score of up to 98%. Before feeding input images to the neural network, they must be preprocessed for standardization and reproductivity, since they come from three different cameras and have varying shapes. Each image is converted to an RGB image with a channel size of 3, then cropped and resized to 1024 by 1024 pixels. After being processed by numerous convolutional layers, batch normalization, padding etc., the initial input of the form (1024, 1024, 3) with 1024 x 1024 x 3 = 3,145,728 dimensions is reduced to 2048 dimensions. The exact structure of our used classification model is not disclosed here. In the process of reducing the dimensionality of the input data, the Auto-Encoder extracts the dense layer output as input for the next step, whereas the BBSDs method extracts the softmax layer as input. Notable is the fact that an untrained Resnet [11] is used in the UAE method, which is the recommended and adopted model in [17].

4.1. Experiment 1: Drift detector on unseen data from the same distribution

The first experiment evaluates the performance of the drift detector in the absence of data drifts, i.e., where the test data have the same distribution as the training data and contains no OOD cases. The purpose of this experiment is designed to test the general functionality of the drift detector and to check whether it produces false positive results. To quantify the drift detector's performance, we randomly split the data into a training and test set (80% and 20%, respectively), both datasets contain almost identical distributions of classes and camera types, which are shown in Figure 3 and Figure 4.

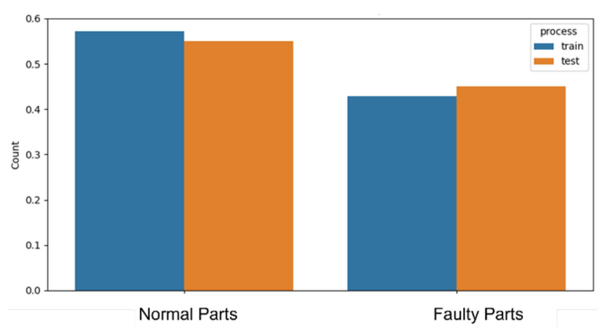

Figure 3. Class distribution of training and test datasets

Table 1 summarizes the computed p-values for all possible combinations of dimensionality reduction techniques and hypothesis testing

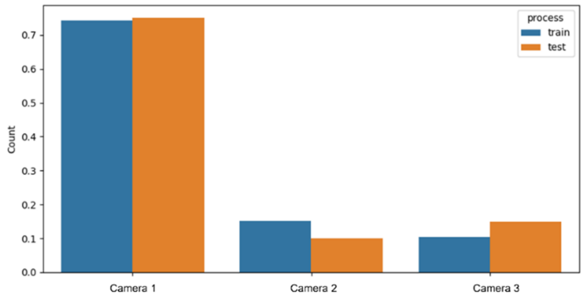

Figure 4. Distribution of camera types for train and test datasets

methods, where bold green entries indicate the best combination for identifying data drift, and red italics indicate the worst. Since there is no OOD in this experiment, the p-value should be greater than 0.05 and expected to be a value close to 1. All combinations provide correct results. UAE performs the worst since Resnet was not trained on this dataset previously. The KS technique surpasses the MMD method, because the combination of the KS method plus the other three dimensionality reduction methods yields the best p-value of 1.

Table 1. p-value on unseen data from same distribution

p-value	MMD	KS
UAE	*0.12*	*0.10*
TAE	0.77	**1.00**
BBSDs	0.50	**1.00**
BBSDh	0.67	**1.00**

4.2. Experiment 2: Drift detector on simulated camera faults

This experiment examines whether the drift detector is capable of detecting drifts caused by simulated hardware setup issues. For this experiment we used 80% data as training dataset and 20% data as test dataset. The main distinction is that the test dataset is distorted randomly with various simulated noises, representing hardware defects that may occur in the press shop. If the drift detector can identify these cases of non-semantic OOD, we would like to study the sample size of the input data to determine the smallest sample size that can detect all sorts of OODs. Thus, drift detection can be implemented in AI applications more frequently and effectively.

Table 2. p-value under different camera faults

p-value	UAE	TAE	BBSDs	BBSDh
			MMD	
Salt-and-pepper noise	0	0	0	0
Gaussian blur	0	0	0	0
Motion blur	0	0	0	0
Random noise	0	0	0	0
underexpose	0	0	0,01	0
overexpose	0	0	0.40	0.01
			KS	
Salt-and-pepper noise	0	0	0	0
Gaussian blur	0	0	0	0
Motion blur	0	0	0	0
Random noise	0	0	0	0
Underexpose	0	0	0.03	0.03
Overexpose	0	0	0	0

4.2.1. Simulated problems in hardware setup

Six applied simulation methods are described below in detail. The professional assessment of experts from the manufacturing area were considered when simulating hardware issues. The simulation was optimized so that real faults can not be distinguished from the simulated ones. We add noise using the OpenCV library. An example photo with simulated noises is shown in Figure 5.

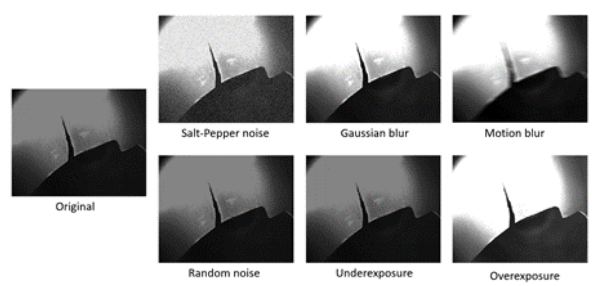

Figure 5. An example of original and simulated images

- **Salt-and-pepper noise** is one of the noises that simulates a camera calibration issue and can negatively impact image quality. This type of noise can be brought on by abrupt and powerful disturbances in the image signal. It appears as sparse black and white pixels [1]. In our experiment, pictures are contaminated with salt-and-pepper noise with a predefined probability of 50%.

- **Gaussian blur** approach smooths image and eliminates image edges using a Gaussian function [10]. Gaussian noise is a type of noise characterized by a probability density function with a Gaussian distribution. Possible causes of this type of noise include the image sensor is not bright enough or the brightness is not uniform when the image is captured, the intrinsic noise of the circuit components and mutual influence, the image sensor is working for an extended period of time and at an excessively high temperature.
- **Motion blur** is a simulation of the blurred images produced by a camera while capturing moving objects [19]. This frequently occurs on the shop floor while the camera is shooting numerous components being moved on the conveyor belt.
- **Random noise** adds noise at random to an image. The pixel values are modified within the range of -20 to +20. Random noise is a variation in brightness or color information that occurs at random in images. It can be generated by a scanner or digital camera's image sensor and circuitry.
- **Underexposed** lighting conditions diminish the brightness of the image, simulating a change in the production environment, such as when the camera's lighting fails.
- **Overexposed** lighting conditions boost the image's brightness, simulating changes in the press shop environment, such as a sudden increase in light intensity.

The six simulation approaches listed above are examples of faults that may occur in practice, although in a real-world scenario, unexpected hardware failures may occur infrequently. However, once it happens, it may lead to unusual output of the model. Therefore, if the drift detector is capable of recognizing these rare and small OOD events, it can alert the workers to fix possible hardware problems. In this way we can realise the AI hardware assurance.

In these six simulated scenarios, all combinations of dimensionality reduction technologies and hypothesis testing methodologies are evaluated, allowing us to discover which strategy works best with our data. As stated previously, the p-value determines whether the null hypothesis (drift exists) may be rejected. Table 2 provides the p-values for all simulation strategies to facilitate comparison of findings. Only the MMD and BBSDs methods provided incorrect responses (highlighted in red in Table 2), whereas KS as a hypothesis test method combined with the other four dimension reduction methods may accurately and effectively identify the drifts.

Next, we conduct further experiments with various sample sizes of test data to find an optimal sample size. To continue the research, we select the combination of BBSDs and KS, which was advised by the authors in [17]. Table 3 shows the results for various sample sizes of test data, with the wrong results highlighted in red. It may be inferred that it is more difficult to detect drift in the "Overexposure" scene, with one possible explanation being that the change in lighting conditions is significantly lower than in other simulated problematic scenes. In conclusion, if the sample size is less than 600, it may produce inaccurate results in some simulation scenarios. Above a sample size of 600, all results are accurate and have p-values less than 0.05. If this method is to be incorporated into the AI service, the minimum dataset required to detect all scene-specific drifts is 600 images per round.

4.3. Experiment 3: Drift detector on abnormal class distribution

This experiment aims to determine whether our method can detect the distribution of abnormal classes in input data. The majority of data comes from the category "normal parts" in practice. We assume the following situation is plausible: an aberrant piece of equipment on the factory floor simultaneously creates a high number of defective parts. If our approach is able to recognize such

Table 3. p-value with different sample sizes

p-value	noise simulation method					
sample size	salt pepper noise	Gaussian blur	Motion blur	random noise	under-expose	over-expose
20	0.26	0.26	0.50	0.15	0.74	0.98
50	0.30	0	0.03	0	0.32	1.00
100	0.04	0	0.06	0	0.09	1.00
200	0	0	0	0	0.57	0.74
300	0	0	0	0	0.06	0.48
400	0	0	0	0	0.05	0.54
500	0	0	0	0	0.01	0.55
600	0	0	0	0	0.01	0.02
800	0	0	0	0	0.01	0.01
1000	0	0	0	0	0.02	0
1200	0	0	0	0	0.01	0
1500	0	0	0	0	0.03	0

defects and deliver warnings, the faulty equipment can be repaired in time to minimize costs.

In this experiment, the same distribution of training and test data is used as in Experiment 1, all photos of normal parts are eliminated from the test dataset to imitate the circumstances described above. Thus, the test data only contains parts with quality issues. All combinations yielded a p-value of 0 and correctly detected the drift, as shown in the Table 4. In conclusion, the drift detector may detect such data drifts when there is an issue in the press shop and a large number of defective parts are being manufactured.

Table 4. p-value on abnormal class distribution

p-value	MMD	KS
UAE	0	0
TAE	0	0
BBSDs	0	0
BBSDh	0	0

4.4. Experiment 4: Drift detector on different camera types

The goal of this experiment is to determine whether the drift detector can detect camera-induced drifts. When upgrading or switching camera types in a real-world setting or deploying an ML system in a new place, it is essential to determine if the images acquired by the new camera types are biased. Experiment 4 employs the exact same data distribution as experiment 1. The main difference is that only photos from cameras 1 and 2 are kept in the training data, while images from camera 3 are included in the test data. In this experiment, all proposed combinations are utilized, and all yield findings with p-values of 0. This indicates that the drift can be detected with these techniques. In other words, it is possible to identify the differences between the test data and the training data. The drift detector can differentiate between the distributions of various cameras.

5. Conclusion and Future Work

The robust operation of deep learning-based software in an AI manufacturing solution is significantly reliant on dependable hardware. The capacity to detect such non-semantics OOD as changes in lighting conditions or camera setup during computer vision is critical in ensuring the robust execution of the AI software. The recurrence of such issues can lead to a decline in the accuracy of AI models' predictions. In this work, we study the performance of a drift detector on non-semantic OOD and validate it in an industrial scenario: an image classification job based on deep learning in the press shop of a German OEM. This method can also be applied to various industrial applications of image-based artificial intelligence for drift detection.

After introducing the theory of the drift detector, we simulate non-semantic OOD cases that may occur in the press shop on the hardware to evaluate the performance of the drift detector. Experimentation demonstrates that the drift detector may identify non-semantic drifts, such as irregular changes in lighting circumstances (underexposure or overexposure owing to a failed flash), distortion in image capture, noise added to image transmission, etc. In addition, the usage of a new camera type and abnormalities in the distribution of image categories (e.g., the press shop producing a large number of defective auto parts) can be accurately recognized. Since drift detection is accomplished via statistical hypothesis testing, a large test set sample size is typically necessary. Experiment 2 consists of testing the performance of the drift detector on various sample sizes to determine the smallest batch size that can detect drift in a short amount of time for all circumstances. In this instance, we've discovered that the drift detector can function effectively with at least 600 samples, allowing us to identify drift at an optimal speed during manufacturing. A sample size greater than one thousand yields more precise results.

This capability to detect data drift caused by hardware device flaws may aid in preserving the resilience of AI models in real-world settings. As soon as the drift detector identifies changes in the data, the responsible workers in the press shop may correct hardware-related faults rapidly. A notable feature of this method is that it can be utilized unsupervisedly, i.e., without the requirement to label the data. This reduces monitoring time and expense for personnel.

The next step is to test the performance and generalization of the drift detector in different deep learning-based use cases. This work verifies the drift detector's capacity to detect non-semantic OODs. However, drifts can be categorized as either good or harmful based on whether the performance of the AI model is negatively impacted. In the following stage, we intend to do in-depth research on how to detect harmful drift so that AI models can be iteratively optimized on these datasets. By comparing the results of the (malicious) drift detection, we can distinguish whether the anomaly is caused by the hardware device or the AI model itself, hence safeguarding the performance of the AI model in the operation.

6. References

References

[1] Jamil Al Azzeh, Bilal Zahran, and Ziad Alqadi. "Salt and Pepper Noise: Effects and Removal". In: *JOIV : International Journal on Informatics Visualization* 2 (July 2018). DOI: `10.30630/joiv.2.4.151`.

[2] Manuel Baena-García et al. "Early Drift Detection Method". In: (Jan. 2006).

[3] Vance Berger and YanYan Zhou. "Kolmogorov–Smirnov Test: Overview". In: Sept. 2014. ISBN: 9781118445112. DOI: `10.1002/9781118445112.stat06558`.

[4] Albert Bifet and Ricard Gavaldà. "Learning from Time-Changing Data with Adaptive Windowing". In: vol. 7. Apr. 2007. DOI: `10.1137/1.9781611972771.42`.

[5] Francois-Xavier Briol et al. *Statistical Inference for Generative Models with Maximum Mean Discrepancy*. June 2019.

[6] Zeki Cinar et al. "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0". In: *Sustainability* 12 (Oct. 2020), p. 8211. DOI: `10.3390/su12198211`.

[7] European Commission. "High-Level Expert Group on AI. Ethics guidelines for trustworthy AI." In: (2019). URL: `https://digital-strategy.ec.europa.eu/de/node/73`.

[8] Akshay Dhamija, Manuel Günther, and Terrance Boult. *Reducing Network Agnostophobia*. Nov. 2018.

[9] João Gama et al. "Learning with Drift Detection". In: vol. 8. Sept. 2004, pp. 286–295. ISBN: 978-3-540-23237-7. DOI: `10.1007/978-3-540-28645-5_29`.

[10] Estevao Gedraite and M. Hadad. "Investigation on the effect of a Gaussian Blur in image filtering and segmentation". In: Jan. 2011, pp. 393–396. ISBN: 978-1-61284-949-2.

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: June 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[12] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: (Oct. 2016).

[13] Yen-Chang Hsu et al. "Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data". In: June 2020, pp. 10948–10957. DOI: 10.1109/CVPR42600.2020.01096.

[14] Yen-Chang Hsu et al. "Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data". In: June 2020, pp. 10948–10957. DOI: 10.1109/CVPR42600.2020.01096.

[15] Hayet Mouss et al. "Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system". In: Aug. 2004, 815–818 Vol.2. ISBN: 0-7803-8873-9. DOI: 10.1109/ASCC.2004.184970.

[16] Chigozie Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. Dec. 2020.

[17] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. *Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift*. Oct. 2018.

[18] Georg Siedel, Stefan Voß, and Silvia Vock. "An Overview of the Research Landscape in the Field of Safe Machine Learning". In: Nov. 2021. DOI: 10.1115/IMECE2021-69390.

[19] Shamik Tiwari et al. "Review of Motion Blur Estimation Techniques". In: *Journal of Image and Graphics* 1 (Jan. 2014), pp. 176–184. DOI: 10.12720/joig.1.4.176-184.

[20] Haoran Wang et al. *Can multi-label classification networks know what they don't know?* Sept. 2021.

[21] Yasi Wang, Hongxun Yao, and Sicheng Zhao. "Auto-Encoder Based Dimensionality Reduction". In: *Neurocomputing* 184 (Nov. 2015). DOI: 10.1016/j.neucom.2015.08.104.

[22] Jingkang Yang et al. *Generalized Out-of-Distribution Detection: A Survey*. Oct. 2021.