Proceedings of the 8th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'22)
Prague, Czech Republic - July 28- 30, 2022
Paper No. MVML 104
DOI: 10.11159/mvml22.104

Supervising The Supervisor – Model Monitoring In Production Using Deep Feature Embeddings With Applications To Workpiece Inspection

Michael Banf, Gregor Steinhagen
Fabforce GmbH & Co. KG
57250 Nepthen, Germany
m.banf@fabforce.com; g.steinhagen@fabforce.com

Abstract – The automation of condition monitoring and workpiece inspection plays an essential role in maintaining high quality as well as high throughput of the manufacturing process. To this end, the recent rise of developments in machine learning has lead to vast improvements in the area of autonomous process supervision. However, the more complex and powerful these models become, the less transparent and explainable they generally are as well. One of the main challenges is the monitoring of live deployments of these machine learning systems and raising alerts when encountering events that might impact model performance. In particular, supervised classifiers are typically build under the assumption of stationarity in the underlying data distribution. For example, a visual inspection system trained on a set of material surface defects generally does not adapt or even recognize gradual changes in the data distribution - an issue known as "data drift" - such as the emergence of new types of surface defects. This, in turn, may lead to detrimental mispredictions, e.g. samples from new defect classes being classified as non-defective. To this end, it is desirable to provide real-time tracking of a classifier's performance to inform about the putative onset of additional error classes and the necessity for manual intervention with respect to classifier re-training. Here, we propose an unsupervised framework that acts on top of a supervised classification system, thereby harnessing its internal deep feature representations as a proxy to track changes in the data distribution during deployment and, hence, to anticipate classifier performance degradation.

Keywords**:** data drift detection, condition monitoring, model performance monitoring, transfer learning, deep feature learning, explainable artificial intelligence

1. Introduction

Today's increased level of automation in manufacturing also requires the automation of material quality inspection with as little as possible human intervention. To stay competitive while meeting industry standards, companies strive to achieve both quantity and quality in production without compromising one over the other. However, manual quality inspection or extensive equipment tests typically allow only for the analysis of individual samples from a given batch of products. With the emergence of vast improvements in the area of Artificial Intelligence, companies begin to employ such technologies during the production cycle to automate quality inspection, as well as monitor machine conditions, thereby minimizing human intervention, and optimizing factory capacities. Accordingly, a plethora of machine learning based condition monitoring and workpiece inspection methodologies and applications have been proposed, including gearbox [1] or rotary machinery analyses [2], as well as bearing [3, 4] or steel surface defect detection [5-10], just to name a few. For a more in-depth survey on machine learning for condition monitoring we refer the reader to [11, 12] or [13].

In contrast, model performance tracking of such classification systems during production is an area less explored. However, as discussed in [14], the life cycle of a machine learning system does extend beyond its deployment, and one of the big challenges is to design systems that are able to monitor live deployments and take appropriate actions on encountering model performance impacting events [15]. One such event may be a gradual or abrupt drift in the data distribution. In the case of a deployed machine learning model, this would be the change between real-time production and the baseline data set used for initial model training [16]. In general, supervised classification systems are trained under the assumption of stationarity in the underlying - in most cases - latent, i.e. non directly observable, data distribution. For example, a model that has been trained as a visual inspection classifier for a set of putative material surface defect types, generally will not adapt or even recognize gradual changes in the data distribution, such as the emergence of a new type of surface defect, which, as a consequence, may then be mispredicted as non-defective. These are critical weaknesses that

must be accounted for in deployed classifiers in order to avoid model performance degradation without the user even noticing. It has been proposed that, in the absence of labels for live data it is critical to monitor the statistics of input data and output predictions as these can serve as a proxy for model performance [17]. A monitoring system requires functionality to determine when significant changes to data and predictive distributions happen, also known as drift detection [2, 16, 18-24]. A related task is to identify incoming data points which fall outside the training data distribution, referred to as outlier detection [14]. Successful drift detection may be used to inform a user that the ongoing production data should be inspected and a deployed model may need to be retrained on an augmented dataset.

Here, we propose a conceptional approach to address the problem of monitoring model performance during production by autonomously detecting data drift in an unsupervised fashion (see figure 1). In particular, we want our approach to identify drift in the data that may cause false negative predictions or type II errors, that is newly introduced defect types being classified as "non-defect" by the original classifier. Hence, the framework harnesses the classifier's learned internal feature representations in order to: i) track and identify anomalies within the latent representation space as the most natural way to track the data sample space without imposing an implicit probability distribution; ii) save memory and training time by an implicit dimensionality reduction of the input feature space; and iii) allow for the usage of efficient, edge computing capable outlier detection algorithms for drift recognition. We perform experiments on two datasets to model different types of data drift as well as drift severity in a typical defect classification scenario, i.e. automated, visual quality inspection.

2. Methods

2.1. Definition of a latent representation space using deep feature embeddings

The main rationale of our proposed framework is to define a latent representation space by extracting - per training sample - the deep feature embeddings provided by the original supervised classifier, and, subsequently, comparing the feature representation of any new production sample with the set of training representations to identify putative outliers that might indicate a shift in the underlying data distribution, e.g. the emergence of a new defect type for which the original classifier has not been trained and may not correctly classify. This way, we may be able to track changes within the representation space, although - given the discriminative nature of classifier systems - we may not a priori assume that the learned feature representation space constitutes any type of continuous probability distribution. We extract the sample-based feature representation of the penultimate layer, assuming the classifier to be a typical feed forward type deep neural network (see figure 2a). We demonstrate our approach for a classification model trained to distinguish between two classes, i.e. "defect" and "non-defect". However, the same rationale can be extended to arbitrary multi-class setups.

2.2. Measuring data drift using Isolation Forests

In principle, given our latent space representation, a variety of anomaly detection algorithms may be used to detect outliers in latent space, ranging from statistical based methods to more recent developments in the area of Deep Learning, such as (Variational) Autoencoders [26]. Here, we choose Isolation Forests [27-29], an unsupervised, scalable and non-parametric outlier detection algorithm with comparatively low computational and memory requirements, making it ideal for on-edge computing applications [30]. In addition, one of the major steps in recent outlier detection algorithms such as autoencoders is the implicit creation of a reduced feature representation space [24], a putative computationally and training data expensive procedure that we avoid by directly applying Isolation Forests on the feature representations already learned and provided by the classifier as discussed above. The main rationale of the Isolation Forest framework is to randomly generate binary tree structures from subsets of the training data and identify anomalies in each subset as instances with the shortest average path length in a given tree, i.e. instances that can be separated from the remaining training set with the least number of splits (see figure 2b for an illustration). A single tree is created by: i) sampling a subset from the training data; ii) randomly choosing a splitting feature per node; and iii) randomly selecting a splitting value from a uniform distribution, spanning from the minimum to the maximum value of the feature selected in the step ii).

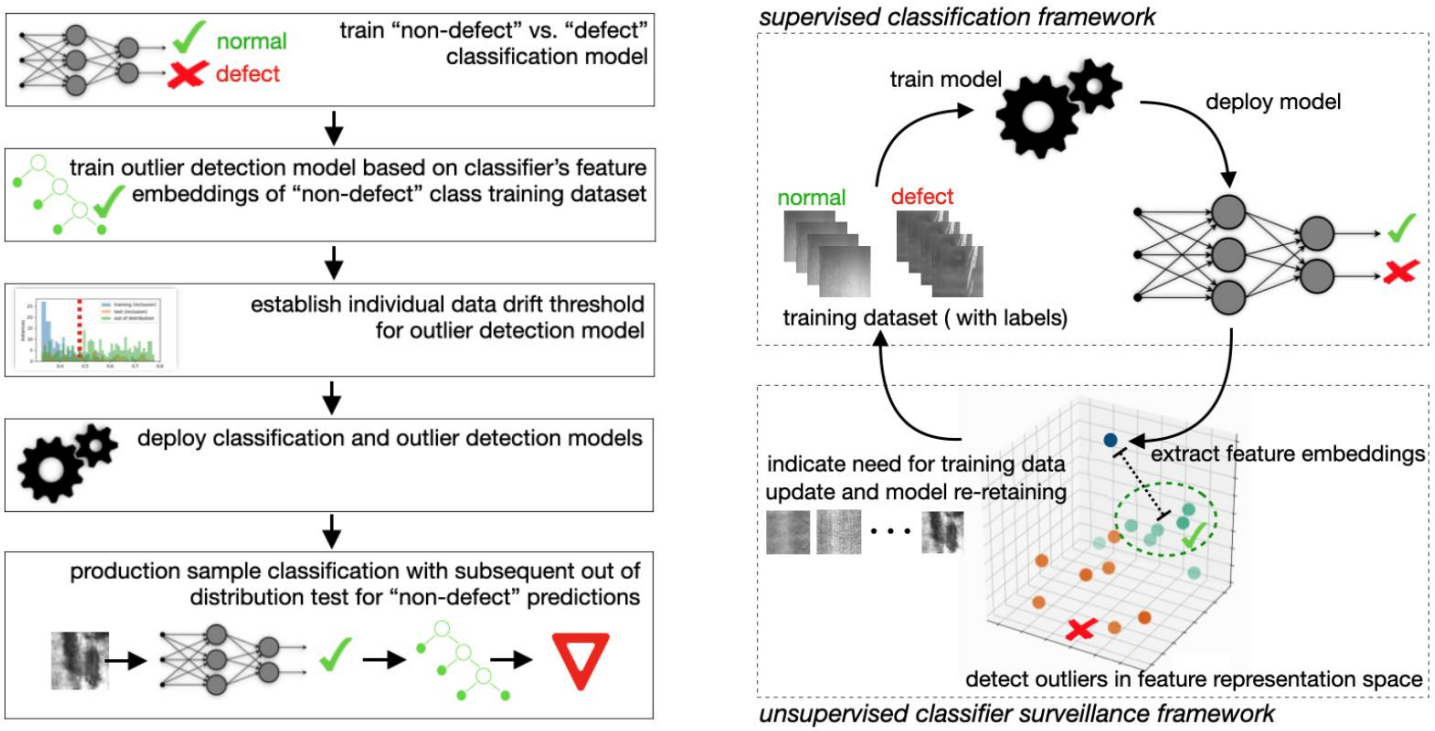


Fig. 1: Overview of our framework (left) designed to complete the machine learning system's lifecycle (right). Our framework monitors model performance via latent space representation tracking during production and informs about the need for training dataset augmentation and re-training.

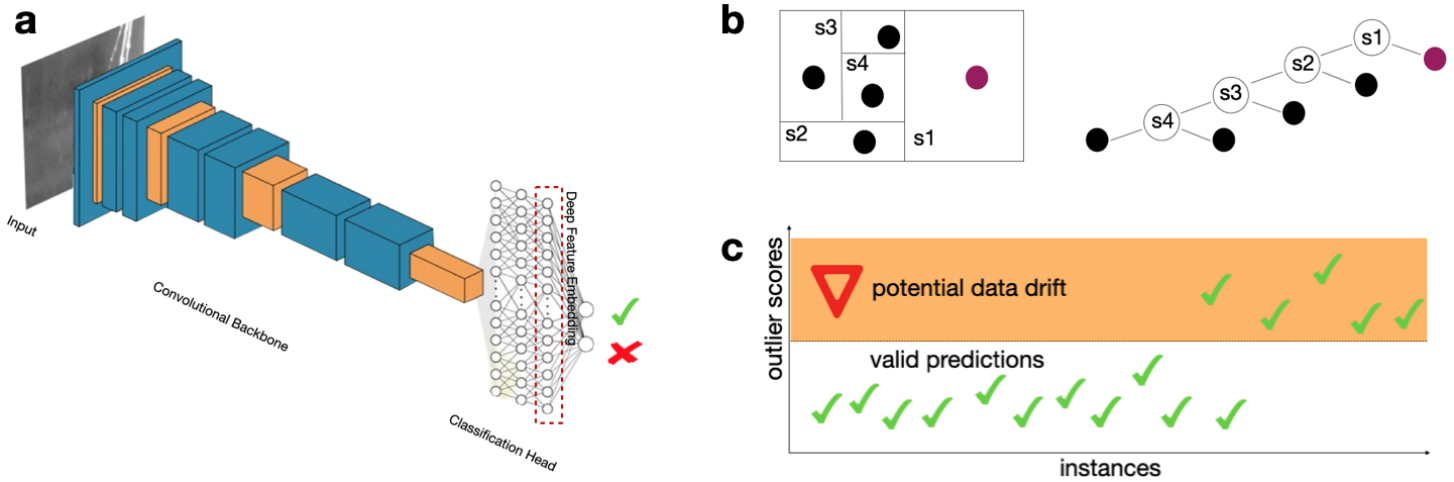
Steps ii) and iii) are then repeated recursively, in theory, until all instances from the subset are "isolated" in individual leaf nodes. In practice, however, a height limit is generally applied, based on the assumption that outliers may be easier to isolate in leaf nodes and thus, on average, do require fewer random splits, resulting in a shorter path length from the root node to the leaf node. If, on building the tree, a height limit is applied, some leaf nodes will end up with more training instances than others. Therefore, an ensemble of tree models is trained, with outlier scores being averaged across their individual outputs to reduce the variance of the model. The outlier score S for a particular instance x is then computed as a function of the average path length from the root to the leaf node compared to the total number m of training instances across all constructed trees, i.e.:

$$S(x,m) = 2^{-\frac{E(h(x))}{c(m)}} \tag{1}$$

with $E(h(x))$ denoting the average search height for x across all trees, and $c(m)$ being a normalization constant for the training data subset of size m, defined as the average depth in an unsuccessful search in a Binary Search Tree, i.e. $c(m) = 2H(m-1) - 2(m-1)/m$. Here, H is referred to as the harmonic number, which can be estimated by $H(i) = ln(i) + \gamma$ with γ denoting the Euler's constant [28]. As a consequence, if $E(h(x)) \ll c(m)$, then $S(x,m) = 1$, that is x will most certainly be an outlier compared to the remaining m instances. In contrast, if $E(h(x)) \approx c(m)$, then $S(x,m) \approx 0.5$, i.e. x may safely be assumed to be a normal instance. In an unsupervised setting, the number of hyper-parameters for the Isolation Forest model reduces to selecting the number of trees as well as the training subset sampling size. Since we are primarily interested in the detection of type II errors, that is newly introduced defect types being classified as "non-defect" by the original classifier, we use the latent space feature representations of the "non-defect" training dataset to train an Isolation Forest as data drift detection model for "non-defect" predictions. Hence, during deployment, if a production sample has been classified as "non-defective", our framework performs an outlier detection based on the sample's feature representation using the "non-defect" Isolation Forest model. Depending on an estimated threshold, the distance of the

sample's outlier score with respect to the training data may then be used as an indicator of the validity of the classifier's prediction. Predictions, so identified as potentially invalid, may then be utilized as indicators of a classifier's conceptional inability to recognize novel types of defects, and, hence, potential drift in the data (see figure 2c).

Fig. 2: a) Example of a typical feed forward convolutional neural network with several interspersed convolutional (blue) and pooling



(orange) layers, followed by multiple dense layers and a two class softmax layer to close. Extraction of the deep feature embedding from the classification head's penultimate layer (image adapted from [25]). b) Toy example of identifying an anomalous (purple) in 2D space by the number of tree splits needed to isolate it from the population (blue). c) Outlier scores may be used to flag a classifiers's "non-defect" predictions as potentially invalid and to identify data drift.

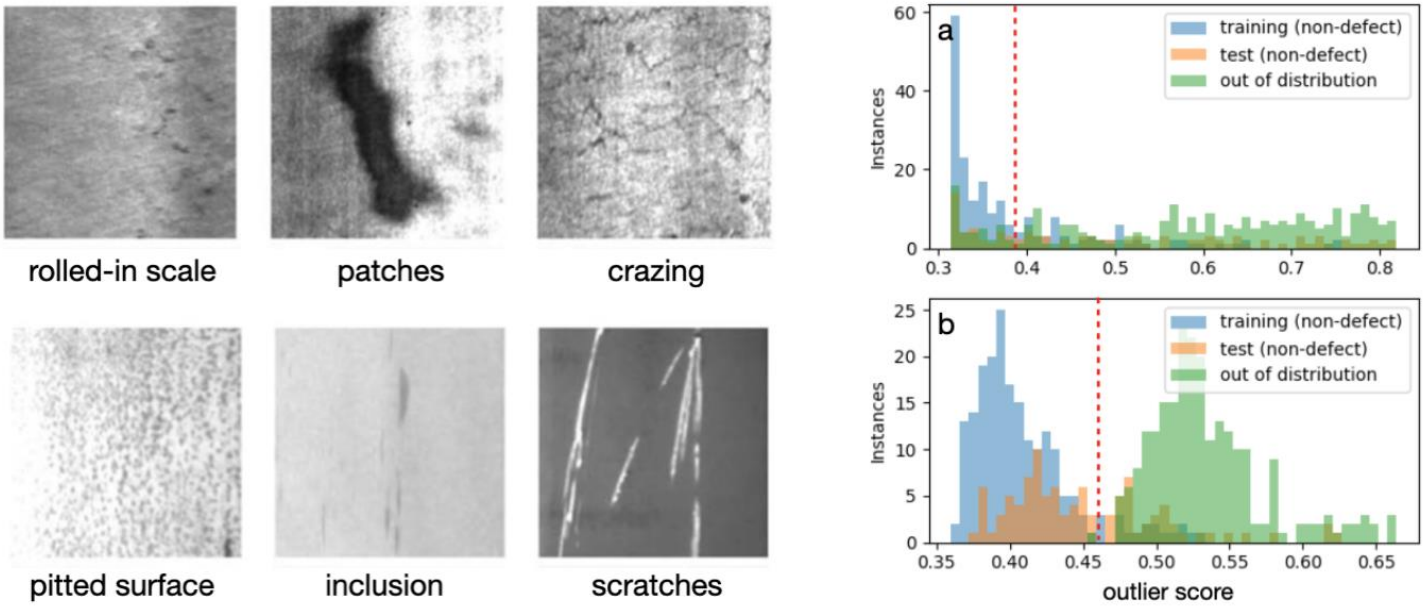
2.3. Median absolute deviation data drift thresholding and classifier re-training

Given a trained data drift detector, we need to define an individual threshold on when to flag a production sample as a putative outlier for manual inspection. Here we do not impose a prior probability on the class-specific distribution of outlier scores from the training set, but propose a common heuristic, that is to calculate the median absolute deviation (MAD) [31, 32]. In brief, for a univariate data set $X_1, X_2, \ldots, X_n$, the MAD is defined as $median\left(\left\|X_i - \tilde{X}\right\|\right)$ i.e. the median of the absolute deviations from the data's median $\tilde{X}$. Hence, the MAD is a robust measure of statistical dispersion and commonly used thresholds for outlier and, hence, drift identification in a distribution are 3.5 MADs. Each so identified outlier may be flagged for further inspection. Given that a growing number of instances - so classified to constitute outliers - would be observed, these instances should be manually inspected to examine the emergence of data drift. If confirmed, all samples containing a new type of error class should be combined with the original training data in order to retrain the classifier, either to augment the "defect" class in the original binary classification model, or to define additional defect classes and change the classification model into a multi-class setup.

3. Results and Discussion

We evaluate our approach for two typical in production quality inspection tasks.

3.1. Visual inspection of steel surface defects on the NEU benchmark dataset



As a first experiment, we train a classifier for a visual inspection task of steel surface defects. Hence, we resort to a typically used benchmark dataset, i.e. the surface defect database from the North-Eastern University (NEU), China[6]. The dataset consists of six kinds of typical surface defects of the hot-rolled steel strip, i.e., rolled-in scale, patches, crazing, pitted surface, inclusion and scratches (see figure 3 left), including in total 1800 grayscale images of 200 x 200 dimensionality with 300 samples per class. We separate the set of samples per class into 200 images for training and 100 for testing, respectively. Fig. 3: Overview of the six kinds of typical surface defects of the hot-rolled steel strip in the NEU benchmark dataset [6] (left). Outlier scores based on isolation forest (right) trained on samples of inclusion ("non-defect") using per sample latent representation of classifier trained on inclusion ("non-defect") vs. scratches ("defect"). Latent representation spaces (penultimate layer) sizes were 8 (a) and 1024 neurons (b). Dashed red vertical lines highlight data drift thresholds, defined as 3.5 MADs from the median with respect to the "non-defect" training data based outlier score distribution. Compared to the 8 neuron model, the drift detector based on 1024 neurons allows for a clear separation between valid and invalid "non-defect" predictions and, thereby, the identification of data drift.

As classifier, we use a feed forward convolutional neural network based on the "MobileNet" architecture [33], given it's suitability as a go-to model for app and edge deployment[34]. Further, given the relative small size of our training data per class, i.e. 200 samples each, we apply "transfer learning" [35], that is, the original model had been pre-trained on the large ImageNet database [36], and, subsequently, we customize and fine-tune the model for our prediction task by freezing the weights of the convolutional backbone, and replacing the classification head with a customized one (see figure 2a for a general illustration) with a two class output, i.e. "non-defect" vs "defect". We choose the penultimate layer - which later serves as the latent representation layer for feature extraction (see figure 2a) - to be a dense layer containing 1024 neurons. Model training is performed with a validation split of 20 percent on the training dataset. Given that the original benchmark dataset only consists of six defect classes, missing a general "non-defect" class, we need to define such a non-defect class for our experiments. In order to evaluate the robustness of our approach, that is to investigate the putative dependencies of performance on the specific distributions of selected class pairs, we use the six defect classes to create five representative class pairs labelled as "non-defect" and "defect" classes (see 1) and perform individual experiments on each pair. The test sets, i.e. 100 images each, of the respective remaining four classes then serve as additional defect types, denoted as out-of-distribution (OOD) data, to be introduced during our simulated production experiments. Table 1 shows the results of our five experiments. Note that the performances of the original binary classifiers, each trained for a given pair of classes, are $\geq$ 99 percent on the training and validation datasets across experiments.

Since we are particularly interested in the proportion of type II errors, that is newly introduced defect types being classified as "non-defect" by the classifier, we classify instances from the test sets of the four remaining classes.

Throughout the five experiments, we observe type II error rates ranging between 36 and 82 percent (see table 1, column 2). This confirms the necessity for a classifier supervision framework to at least inform a user about such critical events of covert non-defect mispredictions. Subsequently, we learn our Isolation Forest based drift detection framework using the per sample latent representations extracted from the classifier's penultimate layer, thereby setting the number of trees to be 100. Using a non-parametric and distribution-free two-sample Kolomogorov-Smirnov test[37], we then examine the differences in the resulting distributions of outlier scores between the training and test datasets for the respective "non-defect" classes, as well as the OOD based distributions (see table 1; Note that a smaller p-value is a higher indicator that the Null-hypothesis, i.e. the two samples have been produced by the same distribution, may be rejected). Table 1 highlights that in all five experiments, the "non-defect" based outlier distributions of training and test data is more similar - with respect to shape and location - than the OOD distribution of "non-defect" predictions. As mentioned before, a data drift threshold is defined as 3.5 MADs from the median with respect to the "non-defect" training data based outlier score distribution, allowing for detection rate of invalid "non-defect" predictions (i.e. type II errors) of on average 95 percent (see table 1, column 6) and, thereby, the identification of data drift (see figure 3).

Finally, we assume that different penultimate layer sizes may have separate effects on the classifier and the supervision model performances. Hence, we further perform a series of qualitative experiments in which the classifier as well as the data drift detectors are trained with varying latent representation sizes, i.e. different penultimate layer sizes, ranging from dimensionalities of 8 to 1024 neurons. We observe that in all five experiments, while classifier performance remain relatively stable, the data drift detection on the introduced OOD datasets typically performs poorly for very small penultimate layers (see figure 3a). Hence, for practical deployment of a supervision framework, considerations on how to dimension the size of the feature representation space have to be taken into account.

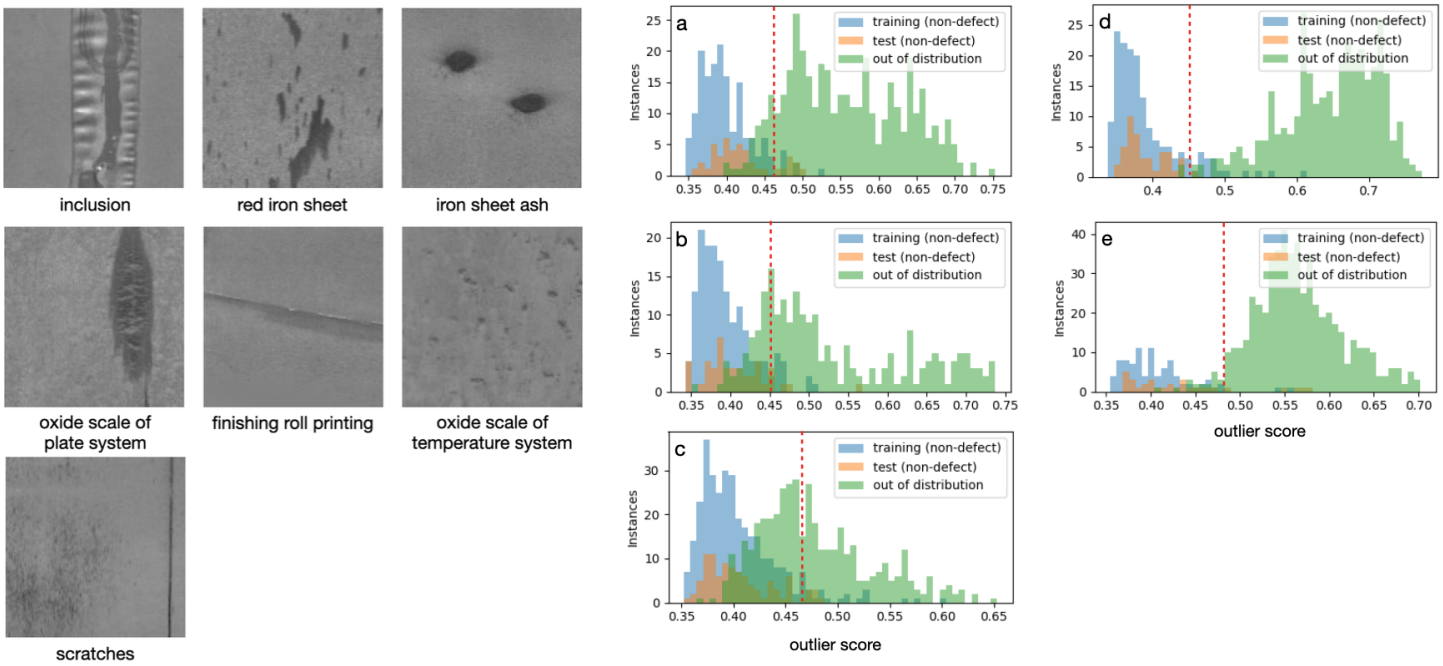
Table 1: Experiments with models trained on selected class pairs for the NEU benchmark dataset [6].

Training classes for classifier Non-defect : Defect	Type II errors (false neg.) on OOD	Outlier score distribution similarity test (using "non-defect" distributions) Training vs test	 Training vs OOD	Outlier scorethreshold (3.5 MAD)	Detection rate of type II errors
Inclusion : Scratches	58 %	−12 p < 2.5	−124 p < 1.9	0.46	99 %
Patches : Crazing	62 %	−5 p < 1.5	−16 p < 5.5	0.48	77 %
Inclusion : Crazing	36 %	−8 p < 4.1	−16 p < 4.4	0.44	99 %
Inclusion : Patches	62 %	−8 p < 2.5	p = 0	0.46	100 %
Rolled-in scale : Crazing	82 %	−12 p < 2.6	−260 p < 3.5	0.47	100 %

3.2. Visual inspection of steel surface defects on the Xsteel benchmark dataset

As a second experiment, we resort to another, very recent, benchmark dataset for steel surface defects, the Xsteel surface defect dataset [38]. The dataset also consists of typical surface defects of the hot-rolled steel strip, including in total 1360 grayscale images of 128 x 128 dimensionality, including 238 (slag) inclusions, 397 red iron sheet, 122 iron sheet ash, 134 (surface) scratches, 63 oxide scale of plate system, 203 finishing roll printing and 203 oxide scale of temperature system (see figure 4 left). We separate the set of samples per class into approximately 80 percent of the images for training and 20 percent for testing, respectively. As classifier, we use the same, pre-trained, feed forward convolutional neural network based on the "MobileNet" architecture [33], as in our first experiment, except that reduced the input dimensions from 200 by 200 to 128 by 128. Model training is performed with a validation split of 20 percent on the training dataset. As in our first experiment, in order to evaluate the robustness of our approach, that is to investigate the putative dependencies of performance on the specific distributions of selected class pairs, we use the seven defect classes to create

five representative class pairs labelled as "non-defect" and "defect" classes (see table 2) and perform individual experiments on each pair. The test sets of the respective remaining five classes then served as additional defect types, denoted as out-of-distribution (OOD) data, to be introduced during our simulated production experiments. Table 2 shows the results of our five experiments. Note that the performances of the original binary classifiers, each trained for a given pair of classes, are ≥ 98 percent on the training and validation datasets across experiments. Throughout all experiments,



type II error rates are ranging between 31 and 65 percent (see table 2, column 2) and, as shown in table 2 and figure 4 (right), the "non-defect" based outlier distributions of training and test data are, again, found to be more similar than the OOD distribution of "non-defect" predictions. As before, a data drift threshold for the Isolation Forest based drift detection framework is defined as 3.5 MADs from the median with respect to the "non-defect" training data based outlier score distribution, resulting in a detection rate of false negative predictions of on average 82 percent (see table 2, column 6).

Fig. 4: Overview of the seven kinds of surface defects of the hot-rolled steel strip in the XSteel benchmark dataset [38] (left). Outlier scores based on isolation forest (right) trained on samples of the respective "non-defect" classes: a) and b) finishing roll printing, c) red iron sheet, d) oxide scale of temp. system, e) iron sheet ash) using per sample latent representation of the respective trained classifier. Dashed red vertical line highlights data drift threshold, defined as 3.5 MADs from the median with respect to the "non-defect" training data based outlier score distribution.

Table 2: Experiments with models trained on selected class pairs for the Xsteel benchmark dataset [38].

Training classes for classifier Non-defect : Defect	Type II errors (false neg.) on OOD	Outlier score distribution similarity test (using “non-defect” distributions) Training vs test	 Training vs OOD	Outlier scorethreshold (3.5 MAD)	Detection rate of type II errors
Finished roll printing : Iron sheet ash	48 %	$p < 0.0033$	-102 $p = 4.4$	0.47	89 %
Finished roll printing : Inclusion	31 %	$p < 0.04$	-61 $p < 1.1$	0.45	76 %
Red iron sheet : Scratches	65 %	$p < 0.81$	$p = 0$	0.47	49 %

Oxide scale of temp. system : Oxide scale of plate system	41 %	$p < 0.07$	-137 $p < 2.9$	0.44	99 %
Iron sheet ash : Oxide scale of plate system	58 %	$p < 0.031$	-16 $p < 3.3$	0.48	97 %

4. Conclusion

One of the main challenges in the automation of condition monitoring and workpiece inspection for high quality, high throughput manufacturing is the monitoring of live deployments of assistive machine learning systems to track model performance. Here, we addressed this less explored field of production model performance monitoring and proposed an unsupervised framework that acts on top of an existing, supervised classification system, thereby harnessing its internal deep feature representations as a proxy to track changes in the data distribution during deployment and, hence, to anticipate classifier performance degradation. Further, our approach harnesses the classifier's implicit dimensionality reduction of the input feature space and, hence, allows for the use of highly efficient, edge computing capable outlier detection algorithms for data drift recognition, resulting in detection rates of false negative predictions of, on average, 82 to 95 percent in our experiments. These experiments were performed on two datasets related to automated, visual quality inspection.

In future work, we would like to extend experimental testing to a wider variety of data types and classifier architectures. Further, we plan to analyze the value of model supervision for other relevant tasks in automated quality inspection, that is defect localisation and segmentation. We see our approach as a generalizable tool to assist in the manufacturing process, avoiding covert mis-predictions and reducing time and costs wasted by allowing for an early response to classifier degradation e.g. due to machine malfunctioning, sensor misreadings, environmental effects or workpiece quality related emergence of drift in the data distribution. In the words of chess grandmaster Garry Kasparov who famously lost to IBM's 'Deep Blue 'computer in 1997: "Human plus machine means finding a better way to combine better interfaces and better processes." [39]

References

[1] L. Jing, M. Zhao, P. Li, and X. Xu, "A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox," Measurement, vol. 111, pp. 1–10, 2017.

[2] K. Chen, Y. S. Koh, and P. J. Riddle, "Tracking drift severity in data streams," in Australasian Conference on Artificial Intelligence, 2015.

[3] S. ke Wen, Z. Chen, and C. P. Li, "Vision-based surface inspection system for bearing rollers using convolutional neural networks," Applied Sciences, 2018.

[4] M. Lu and C. ling Chen, "Detection and classification of bearing surface defects based on machine vision," Applied Sciences, vol. 11, p. 1825, 2021.

[5] A. M. Deshpande, A. Minai, and M. Kumar, "One-shot recognition of manufacturing defects in steel surfaces," Procedia Manufacturing, vol. 48, pp. 1064–1071, 2020.

[6] X. Lv, F. jie Duan, J. Jiang, X. Fu, and L. Gan, "Deep metallic surface defect detection: The new benchmark and detection network," Sensors (Basel, Switzerland), vol. 20, 2020.

[7] Y. Huang, C. Qiu, and K. Yuan, "Surface defect saliency of magnetic tile," The Visual Computer, vol. 36, pp. 85–96, 2018.

[8] X. Tao, D. Zhang, W. Ma, X. Liu, and D. Xu, "Automatic metallic surface defect detection and recognition with convolutional neural networks," Applied Sciences, 2018.

[9] I. Konovalenko, P. Maruschak, J. Brezinova, J. Vin, and J. Brezina, "Steel surface defect classification using deep residual neural network," 2020.

[10] M. Ferguson, R. Ak, Y. T. Lee, and K. H. Law, "Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning," Smart and sustainable manufacturing systems, vol. 2, 2018.

[11] J. Yang, S. Li, Z. ping Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges," Materials, vol. 13, 2020.

[12] X. Fang, Q. Luo, B. Zhou, C. Li, and L. Tian, "Research progress of automated visual surface defect detection for industrial metal planar materials," Sensors (Basel, Switzerland), vol. 20, 2020.

[13] S. Qi, J. Yang, and Z. Zhong, "A review on industrial surface defect detection based on deep learning technology," 2020 The 3rd International Conference on Machine Learning and Machine Intelligence, 2020.
[14] J. Klaise, A. V. Looveren, C. Cox, G. Vacanti, and A. Coca, "Monitoring and explainability of models in production," ArXiv, vol. abs/2007.06299, 2020.
[15] T. Diethe, T. Borchert, E. Thereska, B. Balle, and N. Lawrence, "Continual learning in practice," ArXiv, vol. abs/1903.05202, 2019.
[16] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," 2016.
[17] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ml testscore: A rubric for ml production readiness and technical debt reduction," 2017 IEEE International Conference on Big Data (Big Data), pp. 1123–1132, 2017.
[18] A. Dries and U. Rückert, "Adaptive concept drift detection," Stat. Anal. Data Min., vol. 2, pp. 311–327, 2009.
[19] S. H. Bach and M. A. Maloof, "A bayesian approach to concept drift," in NIPS, 2010.
[20] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACMComputing Surveys (CSUR), vol. 46, pp. 1 – 37, 2014.
[21] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," IEEE Transactions on Knowledge and Data Engineering, vol. 22, pp. 730–742, 2010.
[22] G. I. Webb, R. Hyde, H. Cao, H.-L. Nguyen, and F. Petitjean, "Characterizing concept drift," Data Mining and Knowledge Discovery, vol. 30, pp. 964–994, 2015.
[23] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," ArXiv,vol. Abs/1802.03916, 2018.
[24] S. Rabanser, S. Günnemann, and Z. C. Lipton, "Failing loudly: An empirical study of methods for detecting datasetshift," in NeurIPS, 2019.
[25] T. Hoeser and C. Kuenzer, "Object detection and image segmentation with deep learning on earth observation data: Areview-part i: Evolution and recent trends," Remote. Sens., vol. 12, p. 1667, 2020.
[26] S. Thudumu, P. Branch, J. Jin, and J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," Journal of Big Data, vol. 7, pp. 1–30, 2020.
[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422, 2008.
[28] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," ACM Trans. Knowl. Discov. Data, vol. 6, pp. 3:1–3:39, 2012.
[29] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," IEEE Transactions on Knowledge and Data Engineering, vol. 33, pp. 1479–1489, 2021.
[30] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," Pattern Recognit., vol. 74, pp. 406–421, 2018.
[31] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," Journal of Experimental Social Psychology, vol. 49, pp. 764–766, 2013.
[32] P. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," Journal of the American Statistical Association, vol. 88, pp. 1273–1283, 1993.
[33] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520, 2018.
[34] S. Ahmed and M. Bons, "Edge computed nilm: a phone-based implementation using mobilenet compressed by tensorflow lite," Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring, 2020.
[35] M. Abu, A. Amir, Y. Lean, N. A. H. Zahri, and S. A. Azemi, "The performance analysis of transfer learning for steel defect detection by using deep learning," Journal of Physics: Conference Series, vol. 1755, 2021.
[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, pp. 211–252, 2015.
[37] J. A. Peacock, "Two-dimensional goodness-of-fit testing in astronomy," Monthly Notices of the Royal Astronomical Society, vol. 202, pp. 615–627, 1983.
[38] X. Feng, X. Gao, and L. Luo, "X-sdd: A new benchmark for hot rolled steel strip surface defects detection," Symmetry, vol. 13, p. 706, 2021.

[39] G. Kasparov, “Keynote at imagine by automation anywhere,” InterContinental Hotel 02 London 19-21st March 2019, 2019.