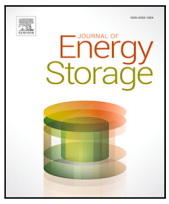
Journal of Energy Storage 105 (2025) 114378

Contents lists available at ScienceDirect

Journal of Energy Storage

journal homepage: www.elsevier.com/locate/est



Research papers

A comparison of transformer and CNN-based object detection models for surface defects on Li-Ion Battery Electrodes

Alexander Mattern [a,*], Henrik Gerdes [a], Dennis Grunert [a], Robert H. Schmitt [a,b]

[a] Fraunhofer Institute for Production Technology IPT, Department of Production Quality, Steinbachstrasse 17, Aachen, 52074, Germany
[b] Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, Chair of Production Metrology and Quality Management, Steinbachstrasse 17, Aachen, 52074, Germany

ARTICLE INFO

Keywords:
Electrode coating
Surface defects
Object detection
Transformer
Convolutional neural network

ABSTRACT

Deep learning-based defect detection approaches offer great potential for end-to-end surface defect detection. After the prevalent Convolutional Neural Network (CNN) models were state-of-the-art for almost a decade, transformer-based models recently surpassed the performance of CNN models on standard benchmark datasets. However, standard benchmarks such as the Common Objects in Context (COCO) dataset are not comparable to industrial use cases. To evaluate the applicability of transformer models in an industrial context, this paper applies a transformer-based object detection model for surface defect detection on Lithium-Ion Battery Electrodes **LIBE** and compares the results to a CNN-based object detection model. As a result, the transformer-based model outperforms the CNN model but is inferior in detection speed. In addition, the paper demonstrates the importance of a well-annotated dataset and shows the sensitivity of annotations for the model performance. Finally, this paper presents practical steps for an industrial application regarding backbone choice, inference speed, and metrics.

1. Introduction

Lithium-ion batteries (LIB) are an integral technology for various applications. Due to the transition to electric vehicles, the battery market is expected to rise from 250 GWh in 2020 to 4000 GWh in 2030, corresponding to an expected annual growth rate of 30% [1]. In an electric battery vehicle, LIB accounts for up to 40% of the costs, mainly due to the high material costs of the electrodes [2]. Within Lithium-Ion Battery production, the coating and drying process of the electrodes, namely the cathode and anode, are at the beginning of the production chain (see Fig. 1). First, the ingredients of an electrode are mixed into a liquefied slurry. Next, the liquefied electrode slurry is continuously coated on a carrier foil through a slot die. Industrial realizable coating speeds reach up to 80 m/min with an electrode width of up to 1500 mm [3]. The copper electrode foil measures 25 to 50 μm, and the coating applied by a slot die is 50 to 100 μm on the anode side and 40 to 80 μm on the cathode side. During the drying process step, the coating solvent dissipates in an oven with a well-defined temperature profile. After these two process steps, the electrode coating needs a high accuracy regarding the homogeneity and the absence of surface defects. However, the high complexity and coordination that is required for the coating and drying process leads to defects. Nelson et al. state that the yield of electrodes is only 92.2% [4]. Electrode defects are irreversible and cannot be fixed at a later process step [5], thus causing higher scrap rates, reducing efficiency, and increasing the overall costs of the battery [6]. Deviations or defects on the electrodes significantly impact the performance, safety, and longevity of the battery cell [7,8]. The severity of the defect depends on the type, location, and size of the defect [9]. Mohanty et al. were the first to introduce four types of defects on Lithium-Ion Battery Electrodes (LIBE): (1) blisters/ agglomerates, (2) divots/ pinholes, (3) metal particle contamination, and (4) exaggerated non-uniform coating [9]. Other authors defined similar or additional defects, such as bubbles and scratches [10], point defects and line defects [11], or micro-compressions, mud cracks, and coating cracks [12]. Currently, no common definition of defect types has been established in the research literature.

Since the coating and drying processes are at the beginning of the process chain, defects must be detected and sorted out at the next technically feasible process step. If defects are not detected and sorted out, a single defective electrode can cause a defective battery cell consisting of several electrodes and auxiliary materials. Therefore, not detecting and removing defective electrodes early in the process chain will increase costs and require additional recycling effort [13].

To monitor the quality of LIBE during manufacturing, defect detection systems are integrated. Traditionally, hand-crafted features are

* Corresponding author.
E-mail address: alexander.mattern@ipt.fraunhofer.de (A. Mattern).

https://doi.org/10.1016/j.est.2024.114378
Received 27 May 2024; Received in revised form 3 October 2024; Accepted 22 October 2024
Available online 20 November 2024

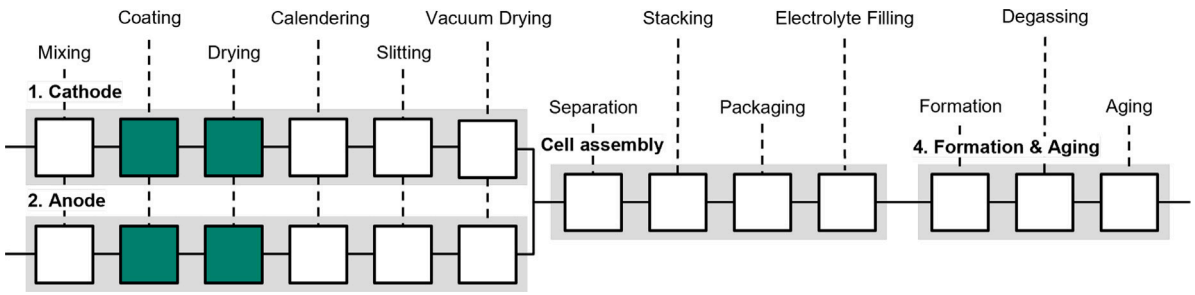


Fig. 1. Lithium-Ion manufacturing process. Defect detection is performed for defects that occur during coating and drying (in green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

used in camera-based machine vision systems to detect defects [14,15]. However, hand-crafted features have several drawbacks, e.g., the need for an expert to configure the features [16]. To overcome these limitations, Convolutional Neural Networks (CNN) as a sub-type of Deep Learning (DL) have been used as a promising approach for defect detection on LIBE, showing comparable or even superior performance to traditional machine vision approaches [12,17].

Inspired by the success in natural language processing, transformer-based DL architectures have been modified for machine vision tasks [18] and surpassed CNN on the standard benchmarks [19]. However, standard benchmark datasets such as the common objects in context (COCO) dataset are not comparable to LIBE defects. Therefore, this paper compares a CNN to a transformer-based machine vision approach.

Our contributions can be summarized in two main aspects. Firstly, we offer model-agnostic insights to guide industrial users, using LIBE as an example. We delve into labeling guidelines and explore two approaches for aligning process requirements with object detection parameters, such as backbone size and image resolution. Secondly, we conduct an in-depth comparison of various general-purpose backbones and a comparison between a state-of-the-art CNN-based object detection model and a transformer-based object detection model, specifically for the use case of LIBE.

2. Related works

In this section, the latest research advances are presented. To provide a basic understanding of the application use case, Section 2.1 presents an overview of defect detection approaches and lists related works of traditional defect detection for LIBE. Lastly, Section 2.2 provides an overview of DL-based defect detection models and their application for LIBE defect detection.

2.1. Defect detection on electrode surface

Defect detection on the electrode surface is of great importance due to the significant impact of the electrode on the performance of the lithium-ion battery cell [15] and the high impact of the electrodes on the total battery costs. Consequently, minimizing the scrap in electrode coating offers considerable cost-cutting potential on the one hand and leads to a more sustainable production due to lower consumption of raw materials on the other hand [5].

Surface defects are irregularities in the structure of the electrode lattice and can be detected by metrology. For example, pinholes, stripes, or agglomerates can be detected by measuring the surface thickness of the coating. Areas that are coated inhomogeneously show a lower coating weight per area, meaning monitoring these parameters can xeffectively detect defects [7]. In LIBE manufacturing, these parameters must be continuously measured at high speeds, which poses a significant challenge [15]. Consequently, the risk remains that critical surface defects are not detected, and electrode sections are incorrectly classified as defect-free [7]. The same applies to metrology, such as Raman spectroscopy or scanning electron microscopy (SEM). Although these measurement techniques allow an in-depth analysis, they are not well suited for large-area defects and in-line inspection [7]. Automated approaches eliminate subjectivity and can be utilized in industrial settings with web speeds of up to 80 m/min. One such method is infrared thermography [20,21]. This technology takes advantage of the heat the electrodes emit after drying. Areas with thinner coatings or exposed conductor foil store less heat, indicating pinholes, stripes, or scratches. Agglomerates or bubbles below the coating surface retain heat for longer, which can be detected by thermal imaging cameras. However, infrared thermography can only be used after drying and is, therefore, unsuitable for detecting defects after calendering or slitting. Furthermore, high hardware costs are associated with thermography [14]. Optical imaging through digital cameras and appropriate illumination compromises the drawbacks. The traversing electrode is continuously captured optically by an industrial camera and analyzed by image processing software. The hardware is comparatively cheap and can be used flexibly in electrode coating as well as in cell assembly. Merely calibrating and setting up the camera and illumination unit requires specialist expertise. In the industrial context, combining optical imaging with digital cameras and image processing software is also known as machine vision [7,12].

Initially, conventional vision-based systems employed hand-crafted features integrated into hand-crafted algorithms to detect defects within an image. With the advent of machine learning (ML), hand-crafted algorithms are frequently substituted by shallow ML models, such as support vector machines (SVM), simplifying the configuration of algorithms. However, hand-crafted features are still required and have several drawbacks, such as needing an expert to select and configure relevant features [16]. In recent years, DL has emerged as a subset of ML and is progressively employed for defect detection in vision-based applications. Instead of explicitly crafting features, a DL-based approach adapts its parameters (comparable to features) based on labeled training images. The DL model supersedes manually engineered features and operates as an end-to-end approach, utilizing image pixels as inputs to identify defects as outputs inherently. Fig. 2 provides an overview of the three operational paradigms of vision-based inspection systems.

In recent years, various authors have utilized various machine vision approaches. Chen et al. apply an Otsu threshold algorithm to extract defects and an SVM to classify the defects. Similarly, Frommknecht et al. use a traditional vision pipeline to detect defects on the electrode surface. After applying a homogenization filter, they used a binarization filter to detect pinholes and a texture analysis algorithm to detect agglomerates [15]. Lui et al. apply a combination of traditional vision algorithms. First, they apply a novel background compensation algorithm to compensate for uneven illumination. Next, they apply an adaptive thresholding approach [12,14]. Schoo et al. use an automated defect detection system that is based on brightness differences detected by two LED arrays. A categorization is performed by standard image parameters, such as aspect ratio, brightness, or size [12]. These approaches show the technical feasibility of traditional machine vision

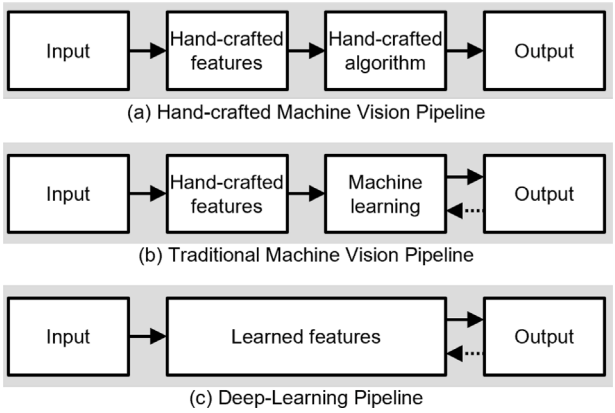


Fig. 2. Overview of three machine vision approaches [22].

approaches. However, all of these approaches require a machine vision expert to configure the system and its parameters. DL, with its end-to-end approach, is a promising approach to simplifying defect detection.

2.2. Deep learning models for defect detection

This section provides an overview of DL for LIBE defect detection. Section 2.2.1 provides an overview of DL models used for defect detection, with a focus on recent transformer-based models. Section 2.2.2 presents recent works for DL-based defect detection on LIBE.

2.2.1. Convolutional neural nets and transformers for object detection

CNNs are a well-established method to handle the complexity of image data, in addition to traditional machine learning. CNNs transpose an image into a feature representation of low and high-level features. Typically, CNNs find application as a general-purpose backbone in computer vision tasks. With each convolution layer, the resolution of a feature map decreases; however, it is enriched by larger, semantic, and contextual information [23]. Due to recent successes in language-prompting, the transformer encoder–decoder architecture is leveraged in the vision domain [24]. Transformer-based backbone models treat an image as a sequence of patches on which the attention mechanism is performed to interrelate between points and generate features [18,25]. Different research streams build attention-based general-purpose backbones with hierarchical feature maps (SWIN, FocalNet) or without multi-scale feature maps (ViT) [18,25,26]. A hierarchical structure introduces features across different scales to facilitate downstream tasks like object detection [27]. On pixel level, the convolution mechanism uses kernels to relate between pixels. Consequently, CNNs naturally possess an inductive bias towards neighboring pixels or regions (e.g., depending on the kernel size), whereas transformers must learn the relationships between pixels through data. Overall, the attention mechanism supports transformer models in building more global and long-range dependencies between regions of interest of the image [24]. Closing the gap between transformers and CNNs, the backbone ConvNeXt applies (modern) design strategies of hierarchical transformers to the ResNet architecture [27].

In CNN-based object detection, the neck serves as an intermediate processing stage and performs additional processing, such as feature fusion or feature pyramidalization, to refine features. [28–30] Carion et al. were the first to propose an end-to-end object detection model that uses a transformer encoder–decoder structure to produce predictions [31]. This detection transformer (DETR) uses a general-purpose backbone to extract features, which are fed to a non-autoregressive transformer encoder–decoder structure. Various works improved the performance of DETR-like models. DETR with improved denoising anchor boxes (DINO) is a state-of-the-art query-based object detector [32]. DINO is trained end-to-end as it is a modification of DETR. DINO fuses different techniques from DAB-DETR [33] and DN-DETR [34] and uses the deformable attention mechanism from deformable DETR. DINO adopts a similar architecture as DETR, where a SWIN-transformer backbone is utilized to extract multi-scale features alongside positional embeddings. A multi-layer transformer encoder and a multi-layer transformer decoder process the data features. Multiple prediction heads output the prediction. For the object detection task, this is a class and bounding box prediction. Unlike DETR, the enhanced features (the encoder output) of DINO undergo a novel method to mix query selections and initialize anchors in the decoder. Additionally, the decoder uses "contrastive denoising" to avoid box duplicates and "look forward twice" (an improvement of deformable DETR's look forward once) to enable the backpropagation of gradients for iterative box refinement and thus stabilize training [32].

2.2.2. Studies on deep learning model application

In recent years, numerous studies have been carried out to compare CNNs and transformers. Pinto et al. compare CNN-based (BiT, ConvNeXt) and transformer-based (SWIN-transformer, ViT) general-purpose backbones for image classification [35]. They compare CNNs and transformers in various experiments, including a simplicity bias experiment, an out-of-distribution detection task, a misclassification task, and a calibration task. However, they conclude that there is no superior architecture. Using transformer models in an industrial context, Wang et al. perform salient segmentation experiments to compare both architectures to defect defects on steel surfaces, rust on rails (NRSD-MN dataset), and errors in fabric [36]. Overall, Wang et al. find that transformer-based models show a slightly better performance than CNN-based models, especially when features are weak, such as for images with low contrast. However, only a small comparison of out-of-the-box detection models are discussed.

For defect detection on battery electrodes, Badmos et al. compare a traditional feature-based binary defect classification approach to a deep learning-based approach [16]. As a result, the deep-learning approach significantly outperforms the traditional machine vision approach. Similarly, Schoo et al. trained a deep learning ResNet-18 model to perform a multi-class classification based on extracted features, achieving a slightly improved result compared to a machine learning classifier [12]. Recently, Choudhary et al. trained a you-only-look-once (YOLO) deep learning model to detect, localize, and classify different electrode surface defects. As a result, the model achieved a promising mean average precision (mAP) of 88%.[1] However, the model suffers from drawbacks, such as poor precision for some defect classes, e.g., defective bubbles on the coating surface [10].

To the best of our knowledge, transformer-based models have not been used to detect surface defects on electrodes so far. Given their superior performance on common benchmarks, the use of transformer-based models seems promising to further refine the detection performance of electrode defects. Moreover, only little attention has been given to query-based object detection models for industrial use cases, although they offer end-to-end predictions without manually setting parameters, such as the non-maximum suppression (NMS) threshold. Thus, a better performing and easier to parameterize deep learning-based model could benefit the end-user to rapidly set up a defect detection system and improve the scrap detection even further.

2.2.3. Deployment of deep learning models in production

In recent years, various researchers have developed DL models for surface defect detection, as shown in a recently performed survey by Prunella et al. [37]. Applications span industrial applications across semiconductor wafer production [38], the steel industry [39], and PCB defects [40]. To date, the number of deployments of DL models for in-

[1] The IoU-threshold is set to .45.

dustrial surface detection tasks is relatively low compared to traditional machine vision approaches. Potential reasons are higher initial costs, data availability, or maintainability of DL models. O'Mohany et al. compare DL models to traditional machine vision applications [41] and point out various challenges of DL for an industrial application. Significant challenges are data availability, suitable hardware, and scalability of DL models. Currently, data is mainly collected for individual use cases and annotated manually [42]. This increases costs compared to traditional machine vision approaches. Similarly, the necessity of a powerful graphics processing unit (GPU) for training and inference increases costs. However, these limitations of DL are widely known, and researchers are focusing on overcoming them. For example, the annotation of defects is greatly simplified by the recently published models Segment Anything [43] and Segment Anything 2 [44]. Also, pretrained backbones and transfer learning have significantly reduced the required amount of labeled data. Small datasets of only a few hundred images are often sufficient to train models with sufficient results [45]. This also improves the adaptability and scalability of DL models since they can easily be retrained and adapted to changes in the production line or expanded to new production lines. In addition, approaches such as federated learning try to leverage available data from various sources to increase the availability of training data [46]. Thus, costs for annotation of training data are reduced by improving data availability and simplifying the annotation process.

3. Methods

First, Section 3.1 describes the methodology which is applied to conduct the experiments. Second, the dataset creation and the dataset statistics are provided in Section 3.2. Lastly, the CNN and transformer-based models are described in Section 3.3.

3.1. Methodology

We have conducted a total of 18 experiments (E) that were assigned to five goals. The first three goals establish a baseline model for a CNN and a transformed-based model. The last two goals compare the performance of the baseline models on different metrics. Fig. 3 shows an overview of all experiments.

In previously conducted experiments, we evaluate the effect of the labeling quality on the CNN baseline model. For this purpose, we use two guidelines to label the dataset (see Section 3.2). Applying these guidelines creates an optimistically labeled dataset for CNN baseline experiment one (E1) and a conservatively labeled dataset for CNN baseline experiment two (E2). The dataset with the better-performing baseline CNN model is used for all remaining CNN-based experiments.

The web speed of the electrode coating application defines the image acquisition frequency and, thus, requires a sufficient inference speed for the image analysis. Based on Eq. (1) and web speeds of up to 80 m/min, we define a minimum FPS rate of 10 to ensure real-time applicability. For this reason, we analyze the impact of the backbone on the detection inference time. We evaluate four backbone sizes with regard to their inference speed and detection performance (E3–E6) to analyze goal number two.

From a production perspective, a lower image resolution is favorable for the inference speed. However, a lower image resolution also decreases the number of pixels that cover a defect. Especially small defects might not be covered by enough pixels to ensure reliable detection. For this reason, the third goal evaluates the performance and inference speed for different resolutions and tries to identify the best trade-off between performance and inference speed (E7–E12).

Lastly, we evaluate the performance of the CNN and transformer-based models. Experiments E13 and E14 evaluate the detection performance on standard metrics that are commonly used for object detection tasks independent of the domain. However, for defect detection on electrodes, the standard metrics are not ideal (see Section 4.3.2). Therefore, we also evaluate the two models on a metric that is adapted for electrode defect detection (E15–E18).

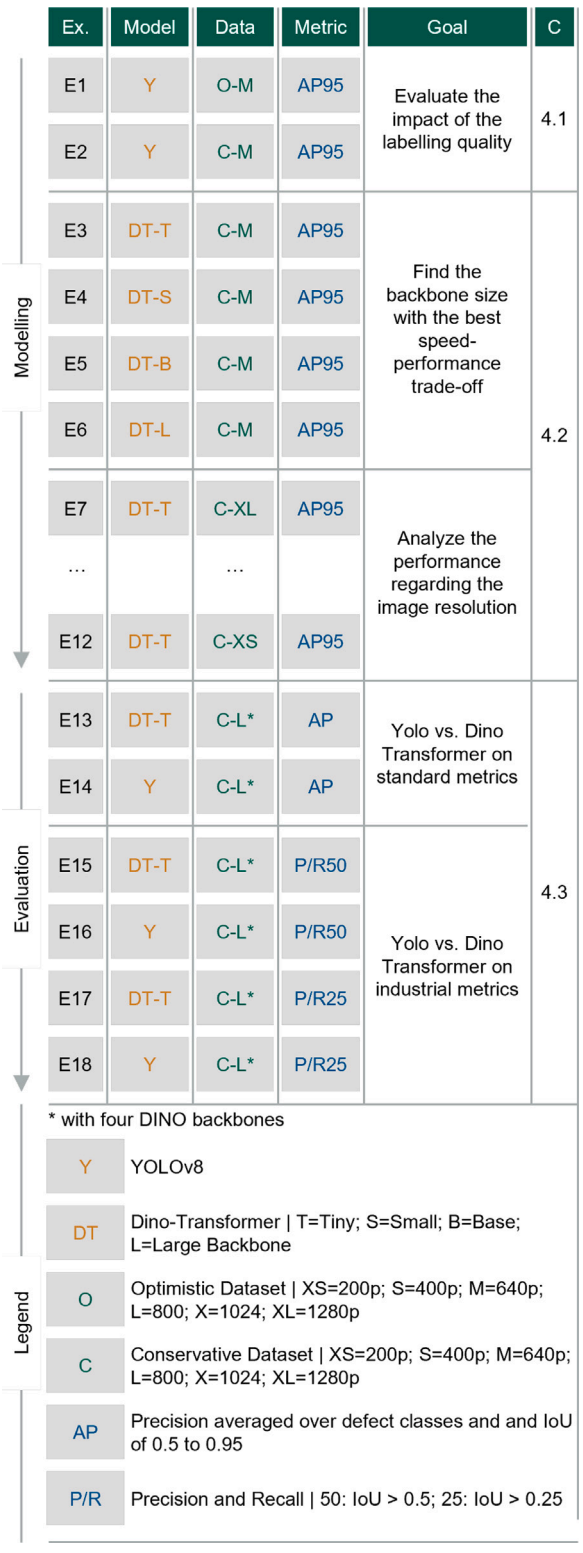


Fig. 3. Overview of the conducted experiments.

3.2. Data description

To benchmark the CNN and transformer models on an industrial dataset, we labeled images that were taken after the drying process of the manufacturing process of LIB cells (see Fig. 1). Data labeling is crucial when training supervised deep learning models and significantly

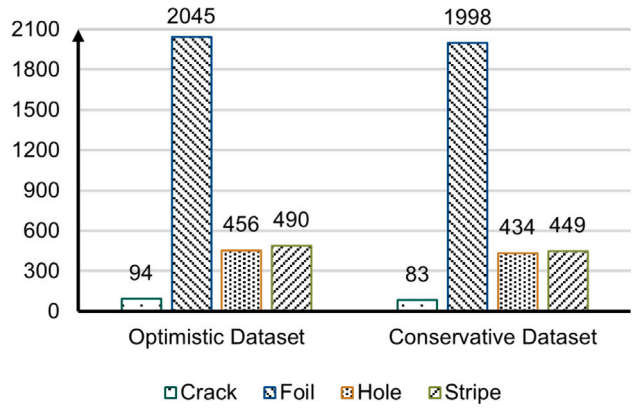


Fig. 4. Number of images with a defect in the conservative and optimistic dataset.

impacts performance [47]. The dataset that was used in this paper was created during a design of experiment parameterization run to test parameters on the electrode coater and dryer machines. In total, 4086 images with a resolution of 2046 × 2448 pixels in a grayscale format are included in the dataset. In a manual cleaning process, images that do not contain coating material and therefore only show the blank conductor foil and duplicated images are removed. This decreases the size of the dataset to 2808 images. Next, the images were labeled. In general, labeling can be performed with a different level of granularity. While classification requires an image-level annotation, object detection, and segmentation require an object (defect) level annotation within an image. An object-level annotation enables the localization of defects. Since LIBE are slit into electrode sheets in a subsequent process step (see Fig. 1), defects must be broadly localized to sort out defective sheets. For this reason, classification was not selected as a suitable approach for LIBE defect detection. Since pixel-wise localization is not required, we selected an object detection approach as drawing rectangular bounding boxes around defects requires less annotation effort for the expert compared to pixel-wise segmentation. To not inflict a human bias, a labeling guideline ensures high accuracy, completeness, consistency, validity, and uniqueness throughout the annotation process [48]. However, in an industrial dataset, the exact positions of ground truth labels can be ambiguous, even for process experts. For this reason, two labeling guidelines were created. The first guideline enforces bounding box labels on all defects, even if the defect class is ambiguous or determining a defect's exact start or end positioning is challenging. The second guideline disregards all images with ambiguous defects in the training dataset. This results in two datasets: The optimistic dataset follows the first labeling guideline and contains 2045 images, while the conservative dataset contains 1998 images and follows the second labeling guideline. We distinguish between three defect types (crack, hole, stripe) and the conductor foil, which is visible on the sides of every image. Fig. A.9 contains examples of the defect images and a partly coated conductor foil image. While the optimistically labeled dataset contains a total of 11 422 object annotations, this number amounts to 8868 in the conservative dataset. Fig. 4 describes the defect distribution of the two datasets.

3.3. Model description

To compare CNN to transformer-based models, state-of-the-art (SOTA) models are selected for each architecture. Both models were pre-trained on COCO and their weights must be publicly available. As SOTA CNN model, we select Ultralytic's YOLOv8. YOLOv8 leverages features from a convolutional neural network as a backbone. As model neck, Glen et al. use top-down and bottom-up pathways that result in three feature maps. The head uses the feature maps as in input to compute a bounding box and class prediction. Unlike previous models of the YOLO family, YOLOv8 is anchor-free. Still, it relies on NMS to remove duplicate boxes [49].

For the transformer-based model, we select the DETR-like object detection model DINO. Unlike YOLO, DINO applies object detection based on a set-based prediction. The DINO model uses a SWIN-Transformer [25] as a backbone to compute hierarchical features, which are fed into a transformer encoder–decoder structure that predicts the class and bounding box of a set of objects. DINO solves the object detection problem in an end-to-end manner and, therefore, does not need NMS by design [31]. We use DINO within the framework Detrex [50], a toolbox for DETR-like models. Detrex is based on Detectron2, a framework for detection and segmentation algorithms [51].

Detrex and Ultralytics provide evaluators for DINO and YOLO. However, we evaluate both model predictions in fifty-one [52] using the coco-style evaluator to ensure a fair comparison. Therefore, the model performance is measured in mean average precision (mAP). This standard metric is used for evaluation in experiments E1 to E14. Additionally, we provide class-specific average precision (AP) for each defect class. In order to evaluate the model's real-time capabilities, we also indicate the model speed in frames per second (FPS). For this reason, the inference time is measured and computed using Eq. (1). Experiments show that the data loading time depends on source and target resolution. The closer the source and target resolution are, the lower the data loading time. In a production environment, the camera adjusts the source resolution manually. Therefore, this gap usually tends to be small. For instance, for a target resolution of 1280p, we measure a data loading time of 0.0017 s. Thus, we conservatively estimate the time in a production environment to be 0.002 s. For smaller target and source resolutions, the time would decrease further. The inference time measures the processing time of a single image on a single GPU (NVIDIA Tesla V100-SXM2-16 GB).

$$FPS = \frac{1}{t_{inference} + t_{dataloading}} = \frac{1}{t_{inference} + 0.002\ \text{s}} \tag{1}$$

4. Experiments and results

In the following, we introduce the baseline model (Section 4.1), evaluate suitable parameters for the transformer-based object detection model (Section 4.2), and perform a twofold comparison (Section 4.3).

4.1. Establish a YOLO-model as baseline

First, we establish a CNN baseline model. For this purpose, we conduct two baseline experiments: In E1, we train the YOLOv8 model on the optimistic dataset, and in E2, on the conservative dataset. For all YOLO experiments, we select the extra large YOLOv8 version. Even the extra large version has comparably few parameters and for this reason, it is capable of achieving inference times of only a few milliseconds making it sufficiently fast for our LIBE defect detection use case [49].

The YOLOv8 model achieved a performance of 55.5% mAP for E1 and 60.2% mAP for E2, resulting in a 4, 7% performance increase by selecting the conservative dataset over the optimistic dataset. In comparison, mAP@0.5 improves by only 2.8%, indicating that the localization accuracy at higher intersection over union (IoU) thresholds increased for E2, leading to more precisely predicted bounding boxes for the conservative dataset. The results of E1 and E2 demonstrate the negative performance impact of labeling uncertainty in a dataset. Based on the results of E1 and E2 we select the conservative dataset for all further experiments. Lastly, we conclude the baseline YOLOv8 experiments by performing a hyperparameter tuning on the conservative dataset. The hyperparameter tuning further improves the performance

from 60.2% mAP to 61.4% mAP[2] on the evaluation dataset. We use the best model configuration for our baseline experiments (E13–E14).

4.2. Configuration of the DINO model; experiments E3–E12

For the experiments E3 to E6, we select a DINO model that is pre-trained on COCO and only fine-tune the weights on the conservative dataset. To save computational capacities, we freeze the backbone weights. Freezing the backbone reduces the training duration from six to two hours. The training of the DINO model is performed on two NVIDIA Tesla V100-SXM2-16 GB.

The COCO object detection benchmark focuses solely on detection performance without considering the models' detection speed. In order to choose a model configuration that meets the FPS and performance requirements of LIBE manufacturing, we analyze FPS speed and the detection performance in E3–E6. To analyze the impact of the model size of DINO on the FPS, we select four DINO sizes. Since the size of a DINO model strongly correlates with the size of the backbone, we select four different SWIN-transformer backbone sizes to approximate the DINO size: Tiny (T), Small (S), Base (B), and Large (L). As a rule of thumb, more complex tasks require larger backbone sizes that are computationally more expensive. Thus, larger backbone sizes require higher inference time and result in lower FPS (see B.10 for more details). As depicted in Fig. 5(a), the FPS significantly decreases with larger backbone sizes. At a resolution of 800 × 960 pixels (800p), the FPS of the tiny model (T) is more than two times higher than the FPS of the large model (L) (Fig. 5(a) - I). The performance of the four models is at a similar level regardless of their backbone size (Fig. 5(a) - II).

Since only SWIN-T and SWIN-S meet the process requirement of 10 FPS, SWIN-B and SWIN-L are not suitable for our industrial application. Because the SWIN-T backbone achieves the highest FPS rate at a comparable performance level to the other backbones we use SWIN-T as a backbone for further experiments.

Next, in E7 to E12 we analyze the impact of the image resolution on the performance. Investigating the influence of image compression on the performance of object detection models on COCO, Gandor et al. observe that the model performance drops gradually with increased compression until a certain threshold causes a rapid performance drop [53]. Therefore, experiments E7 to E12 investigate how the class-specific and overall model performance behave with image resizing to different resolutions. On the one hand, the performance is evaluated when the model is trained with a fixed resolution in a range between 220 and 800p. On the other hand, training with multiple resolutions is evaluated. For instance, for a resolution of 800p, at training time, the image resolution is varied between 640 and 800p.

Fig. 5(b) shows the results of experiments E7 to E12. As a general observation, decreasing the resolution gradually hurts the model's performance until it drops rapidly at a resolution of about 300p. These findings are consistent with the findings of Gandor et al.. The performance drop after 300p is significantly higher for small objects like holes. Based on this observation, we explain the performance drop with the decreasing pixel coverage of small defects that result from decreasing the image resolution. In context, a hole defect at a resolution size of 2046p is covered by roughly 20 pixels, corresponding to ≈ 2.46 mm. If the models' capabilities allow resizing until a resolution of 300p, we conjecture that a minimum detectable object size lies at roughly 3 pixels, which corresponds to 0.36mm (see C.4 for more details). For this reason, we argue that the image resolution must be chosen based on the smallest defects in a dataset. For the industrial user, we suggest, as a rule of thumb, covering the smallest defect class with a minimum of roughly 5 to 10 pixels to achieve a good off-trade between performance and speed.

[2] The metrics are based on the validation dataset and calculated with the evaluator of YOLO. For this reason, the metrics are not comparable to the values in Table 2.

Moreover, we discover that multi-resolution training has a beneficial effect on the detection performance. Especially small objects benefit from multi-resolution training (see Fig. 5(b): compare III and IV). We speculated that this sensitizes the model for the loss of specific features during compression: When the model revisits defects at a smaller resolution, this makes the detection more robust to compression and, thus, lowers the possible detection size. In addition, it reduces training duration with the same number of epochs since images partially need less inference time due to their lower resolutions.

Lastly, we observe that increasing the resolution to more than 800p does not increase performance. This finding holds true regardless of the object size. As higher resolutions do not improve the detection performance, we conclude that the performance has surpassed the minimum object size threshold that is required to recognize the smallest defects at 800p. It should be noted that altering resolution above 800p has a relatively high impact on the inference speed with little to no performance improvements. Regarding the FPS, a DINO with a SWIN-T backbone achieves 12 FPS at a resolution of 800p. The minimum FPS depends on the web speed v_{web} of the coating machine and the image height l_{frame}. Eq. (2) estimates the minimum FPS requirements. Using this equation, a web speed of 80 m/min, requires a minimum FPS rate of 5.33. This is easily exceeded with 12 FPS, allowing a deployment at a maximum industrial web speed.

$$\text{FPS}_{\text{min}} = \frac{v_{\text{web}}}{l_{\text{frame}}} \tag{2}$$

Based on these findings, we use the SWIN-T DINO with a mixed-resolution training strategy and evaluate its performance at 800 × 960 pixels. For an in-depth study of different CNN- and transformer-based backbones, we apply this training strategy to InternImage-T (CNN) [54], ConvNeXt-T (CNN redesigned following transformer architectures) [27], and ResNet-50 (CNN, similar to YOLO's backbone) [55].

4.3. Experimental comparison; experiments E13–E18

This section compares the previously configured YOLO model and DINO models on different metrics. Section 4.3.1 evaluates both models in the metric *AP* that is widely used for model benchmarking. Section 4.3.2 shifts its attention towards conducting a comprehensive comparison on an adapted industrial metric, exemplified by the application on LIBE.

4.3.1. Comparison in average precision; experiments E13–E14

The comparison evaluates both models in the metric average precision (AP). All models run inference at 800 × 960 pixels (800p) on a single NVIDIA V100.

Before comparing YOLO and DINO directly, we evaluated the performance of both models with different backbones. Table 1 shows the performance of the backbones. Of these four backbones, ResNet-50 and InternImage-T represent classical CNN-based general-purpose backbones, SWIN a hierarchical transformer backbone, and ConvNeXt-T a purely convolutional backbone that was redesigned following best practices from modern transformer architectures such as SWIN. Overall, the performance is comparable. No backbone significantly outperforms the others.

In general, convolutional layers possess strong local receptive fields that the attention mechanism must learn through sufficient data [24]. Surprisingly, we observe that the SWIN-T backbone performs weaker for large shallow cracks. However, it achieves the best performance on holes (Observation 1).

Both models achieve similar performances (see Table 2). With 56.8 mAP, DINO (SWIN-T) achieves a +2.7 AP compared to the YOLO model, while DINO achieves even a +3.3 AP. Concerning individual classes, DINO tends to perform better for large objects like cracks or stripes by +5.0 AP for cracks and +8.6 AP for stripes (Observation 2). A possible explanation for this observation could lie in the attention mechanism

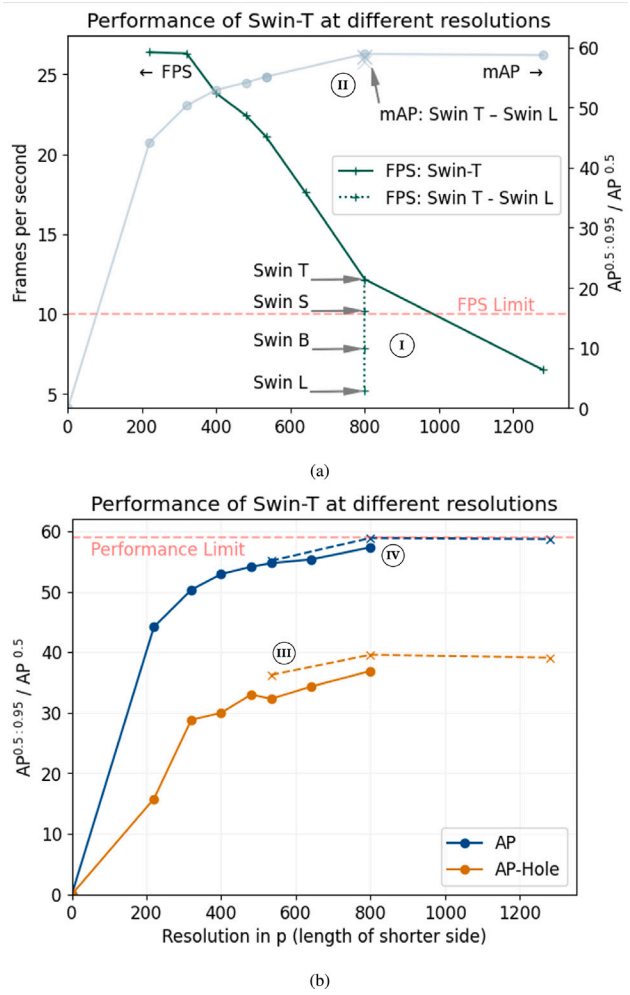


Fig. 5. Offtrade between model performance and speed. Figure (a) show the FPS (I) and the performance (II) for different SWIN backbone sizes over the image resolution. FPS were calculated based on a single NVIDIA V100 GPU. Figure (b) shows the overall AP performance (III) and the detection AP for the hole defect class (IV) of SWIN-T backbone over the image resolution. Dashed lines indicate a multi-resolution training strategy.

of the DINO object detection model, as it is more capable of connecting far-apart features. Unintuitively, observations 1 and 2 do not rule themselves out. It becomes more evident that DINO could simply be better at connecting local features than YOLO.

An additional explanation of the lower performance of crack detection compared to foil and stripe defects is the small number of training examples. Crack defects are the rarest defect type (see Fig. 4) in the dataset and thus provide the smallest supervision during model training. In this case, the performance of DL models is tied to the performance of rarely occurring defects. Notably, holes do achieve low AP scores. However, this is most likely caused by the annotated bounding boxes. Because the AP metric consists of evaluations at high IoU thresholds, inaccurately labeled defects in the training and test datasets worsen the model's score significantly for small defects (see Section 4.3.2 for a detailed discussion).

Unlike DETR, DINO uses the deformable attention module to mitigate drawbacks that its predecessor has. Namely, modern object detection models leverage high-resolution feature maps. However, due to the quadratic complexity of the attention module, this poses a problem as the transformer attends to all possible spatial points. Consequently, closely related to deformable convolution, DINO's deformable attention mechanism selects a small set of sampling points around a reference point [56]. To further gain insights into its attention mechanism and

Table 1
Comparison of different backbone architectures in DINO at 800 × 960 pixels.

Backbone	mAP	AP_{50}	Crack[a]	Foil[a]	Hole[a]	Stripe[a]
ResNet-50	55.6	86.2	48.5	98.9	30.4	79.1
InternImage	**57.4**	90.8	**49.6**	98.9	30.9	78.0
SWIN	56.8	**91.5**	46.3	98.9	**31.4**	**79.4**
ConvNeXT	57.0	88.0	48.8	98.9	29.8	78.3

[a] Average Precision.

Table 2
Comparison of YOLO and DINO-SWIN-T at 800 × 960 pixels.

Model	mAP	AP_{50}	Crack[a]	Foil[a]	Hole[a]	Stripe[a]
YOLO	54.1	90.3	41.3	98.8	30.0	70.8
DINO	**56.8**	**91.5**	**46.3**	**98.9**	**31.4**	**79.4**

[a] Average Precision.

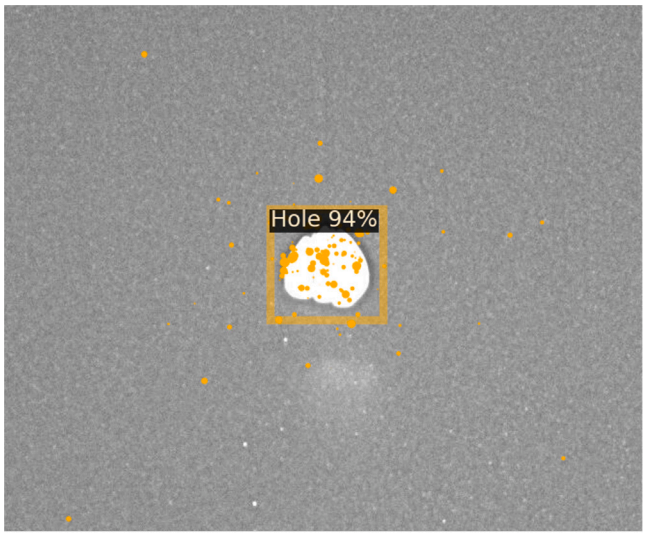
how it explains the model's performance, Fig. 6 offers insight into the last decoder layer. In Fig. 6-a, the sampling locations cover the hole densely. The attention module explores the conductor foil at the edge of the coating as they also include locations further away from the hole. As a consequence, the model assigns a high confidence score. DINO approaches cracks differently (Fig. 6-b). The sampling locations are evenly spread across the defect. At the bottom of the crack, the object detection modules further explore weak features. However, as they are too ambiguous, they do not fall inside the bounding box. Analogues of the detection performance for cracks, the certainty of the crack classification reflects the difficulty posed by ambiguous shallow defects.

4.3.2. Comparison in an industrial setting; experiments E15–E18

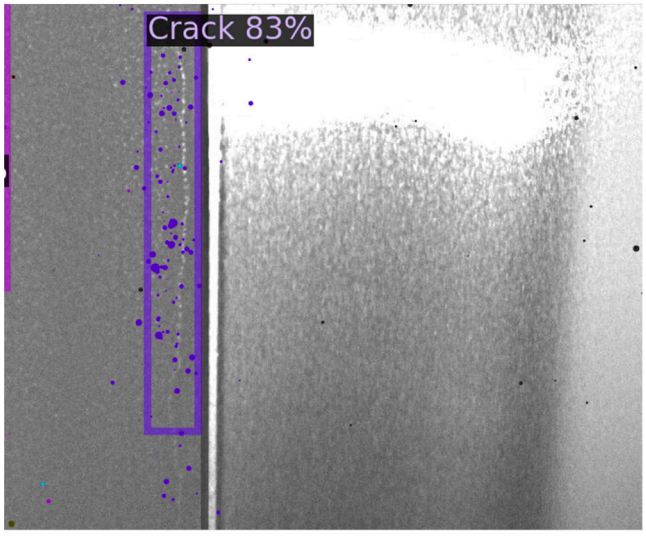
From an industrial perspective, detecting defects with a broad localization is more important than locating defects accurately with a lower detection performance. In the specific context of LIBE coating, the presence of a single defective electrode can result in the rejection of an entire battery, which may contain numerous electrodes. This undermines the criticality of false negatives, where a defect is present but not detected, compared to a false positive detection. To quantify false positives and negatives, this comparison evaluates both models in precision and recall metrics. The previously used mAP metric uses an averaged score that is calculated based on ten different IoU thresholds that range from 0.5 to 0.95 [57]. Unlike mAP, precision and recall use a single IoU threshold to decide whether an object was correctly located. We argue that the commonly used IoU threshold of 0.5 is not ideal for an industrial defect dataset that contains small defects. Fig. 7 shows the ground truth annotation of a small point defect (a) and three bounding boxes with IoU thresholds ranging from 0.25 to 0.95 (b–d). Depending on the industrial requirement and the labeling quality, an IoU threshold of 0.25 is sufficient to detect and locate defects. In addition, a focus should be put on annotating small defects accurately to avoid misguiding the model due to labeling inconsistencies in the bounding box sizes.

Experiments E15 to E18 compare the precision and recall (P/R) values for each defect class at IoU thresholds of 0.25 and 0.5. We aim to address the models' ability to detect different defect classes, as each class has its criticality. For the LIBE defects, we define a target area of a precision value of at least 90% and a recall value of at least 95%, as we initially found that the recall is more important.

Table 3 shows the results of experiments E15 to E18. Large common defects (stripes and foil) score high P/R values at both IoU thresholds. Crack and hole defects have lower P/R values but improve greatly when the IoU threshold is decreased from 0.5 to 0.25 (see Table 3 and Fig. 8). Similar to findings in previous experiments, the stripes and foil are better detected by the DINO models (see Section 4.3.1). DINO achieves a significantly higher precision for stripe defects compared to the YOLO model (0.99 vs. 0.92). The precision difference can be explained by two

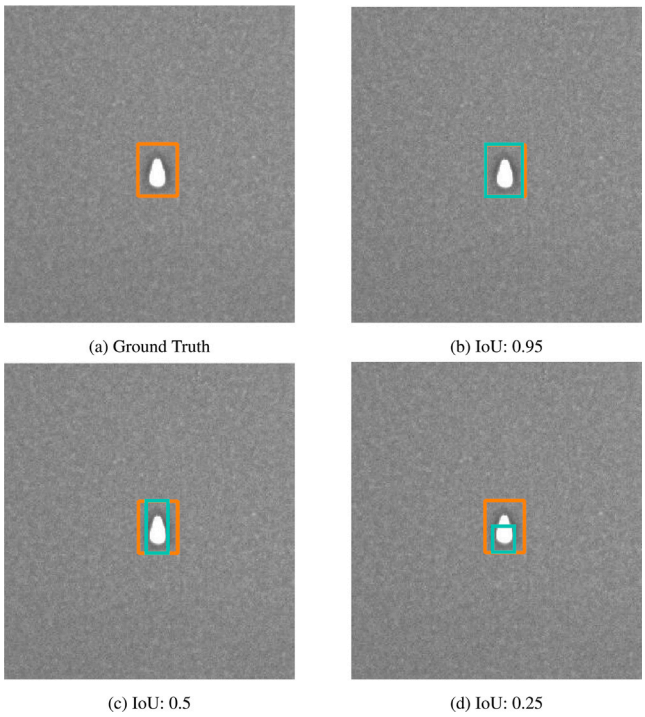


a) Attention mechanism for hole defect



b) Attention mechanism for crack defect

Fig. 6. DINO's attention mechanism for a hole and crack defect.



(a) Ground Truth (b) IoU: 0.95

(c) IoU: 0.5 (d) IoU: 0.25

Fig. 7. Subplot (a) shows a small point defect with a ground truth annotation (orange). Subplots (b) to (d) additionally show a bounding box (green) with an IoU threshold, ranging from 0.25 to 0.95. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

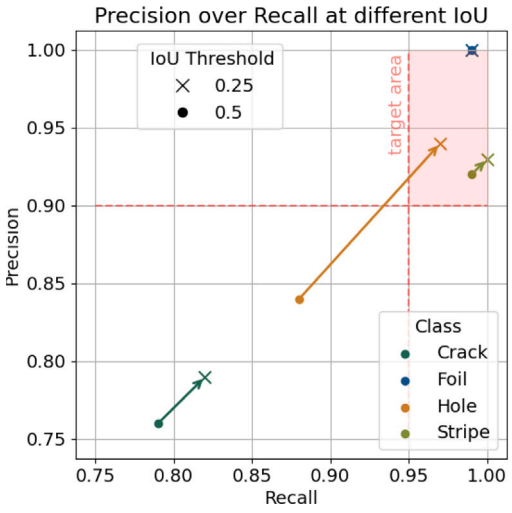


Fig. 8. Precision and recall improve towards target area when lowering the IoU threshold. Here an exemplary visualization for YOLO.

reasons. Either the YOLO model is inferior in detecting large objects compared to DINO and misclassifies large objects as stripes. Or else, this is also possible due to a low NMS score, resulting in duplicate predictions. Conversely, recall suffers from an NMS threshold that is too high and is not affected by a setting that is too low. For crack defects, DINO achieves higher P/R values. This observation aligns with the previously stated hypothesis that global attention improves the prediction of large objects. In the case of small defects, the YOLO model and DINO with CNN backbones show better precision but slightly lower recall. The DINO-SWIN model is slightly more sensitive to detect surface impurities or tiny and unlabeled pinholes. Moreover, it provides circumstantial evidence that object detection models with CNN-based backbones like YOLO perform better for small defects compared to transformers that use the attention mechanism to extract features.

In summary, YOLO and DINO have been selected to compare CNN-based models to transformer-based models. DINO performs better for larger defects (w.r.t. higher IoU thresholds such as in mAP). However, for LIBE surface defect detection, there is no performance superiority of one of the two models. Independent of the model, better performance and localization test results are achieved by lowering the IoU threshold of the test metric. Lowering the threshold from 0.5 to 0.25 improves precision and recall for holes and cracks significantly, resulting in recall values for the dataset's most common defects (foil, hole, stripes) of more than 96% and, thus, demonstrating applicability for industrial applications (see Fig. 8).

5. Conclusion

Convolutional neural networks (CNN) are state-of-the-art for deep learning-based object detection. In recent years, transformer-based deep learning models have surpassed CNN on standard benchmark datasets. However, standard benchmark datasets and their respective metrics are not comparable to industrial datasets for defect detection. Also, only performance metrics are considered to compare deep

Table 3
Performance comparison of various detectors and backbones at different IoU thresholds.

Detector	Backbone	IoU	Conf	Crack		Foil		Hole		Stripe	
				Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Yolo v8	CSPDarknet53	0.5	0.25	0.76	0.79	1.00	0.99	0.84	0.88	0.92	0.99
DINO	ResNet50	0.5	0.25	0.77	0.86	1.00	0.99	0.86	0.88	0.99	1.00
DINO	InternImage-T	0.5	0.25	0.66	0.89	1.00	0.99	0.83	0.90	0.98	1.00
DINO	SWIN-T	0.5	0.25	0.79	0.82	1.00	1.00	0.81	0.89	0.99	1.00
DINO	ConvNeXt-T	0.5	0.25	0.85	0.82	1.00	1.00	0.82	0.87	0.98	0.99
Yolo v8	CSPDarknet53	0.25	0.25	0.79	0.82	1.00	0.99	0.94	0.97	0.95	1.00
DINO	ResNet50	0.25	0.25	0.81	0.89	1.00	0.99	0.96	0.98	0.99	1.00
DINO	InternImage-T	0.25	0.25	0.71	0.96	1.00	0.99	0.92	0.99	0.98	1.00
DINO	SWIN-T	0.25	0.25	0.86	0.89	1.00	1.00	0.89	0.98	0.99	1.00
DINO	ConvNeXt-T	0.25	0.25	0.89	0.86	1.00	1.00	0.93	0.98	0.98	0.99

learning models on standard benchmarks. In this work, we compare a CNN-based YOLOv8 model and a transformer-based DINO model regarding the inference speed and performance. Additionally, we offer insights on key considerations for applying deep learning-based defect detection in industrial applications. These methods are use case agnostic and can be adapted for other processes and defects. In this work, the detection of four defect classes on lithium-ion battery electrodes is selected as an industrial use case. In our findings, we have identified that the labeling quality strongly correlates with the model's final performance, demonstrating the sensitivity of the label quality to the performance of a deep learning model. From a company's point of view, this finding requires capturing a significant number of defects across all defect classes and assigning process experts to label the defects accurately. A model with small training data will fail for a defect class with few images. This is problematic for rarely occurring defects where collecting images with various defects is challenging. To overcome these limitations, unsupervised learning approaches, such as unsupervised anomaly detection, should be considered for future research. Further, the experiments show that in contrast to standard benchmarks, larger backbones and higher image resolutions do not necessarily improve the model performance but significantly hurt detection speed, as demonstrated in the example of DINO. Given our use case, decreasing the input resolution to 800×960 pixels provides a maximal performance with significantly improved inference speed. Decreasing the resolution further gradually hurts the model's performance. Thus, we conclude that the smallest defects limit the suitable image resolution because they need to be covered by enough pixels to be detected reliably. Regardless, YOLOv8 has a higher inference speed compared to DINO. For this reason, YOLOv8 better suits the high inference speed requirements of industrial applications. Subsequent, we have compared YOLOv8 and DINO on standard performance metrics resulting in a slight performance advantage of the DINO model. However, small defects are highly sensitive to labeling inaccuracies and are easily misclassified when the intersection over union threshold is set to the commonly used value of 0.5. For these reasons, we have adopted a precision–recall metric with a comparably low intersection over union value of 0.25. Lowering the intersection over union threshold represents a simple way to adapt a standard metric to the requirements of an industrial use case. We suggest adapting the metrics in other industrial fields as well, such as the semiconductor, automotive, or aerospace industry. Based on the adapted metric, the YOLOv8 and DINO models achieve both precision and recall values of over 90% and, thus, are well-suited for surface defect detection on the surface of lithium-ion battery electrodes. Without additional substantiation, we conclude that this is a straightforward and efficient method for validating the model within a realistic environment for any given use case where standard benchmarks do not match production-related requirements.

CRediT authorship contribution statement

Alexander Mattern: Writing – review & editing, Writing – original draft, Supervision, Conceptualization. **Henrik Gerdes:** Writing – review & editing, Writing – original draft, Conceptualization. **Dennis Grunert:** Supervision. **Robert H. Schmitt:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Funding reference: The project "FoFeBat – Research Fab Battery Cells" is funded by the German Federal Ministry of Education and Research. Grant number: 03XP0256.

Appendix A. Defect images

Defects commonly occur during LIBE production. Common defects include agglomerates, holes, contaminations, stripes, cracks, and edge

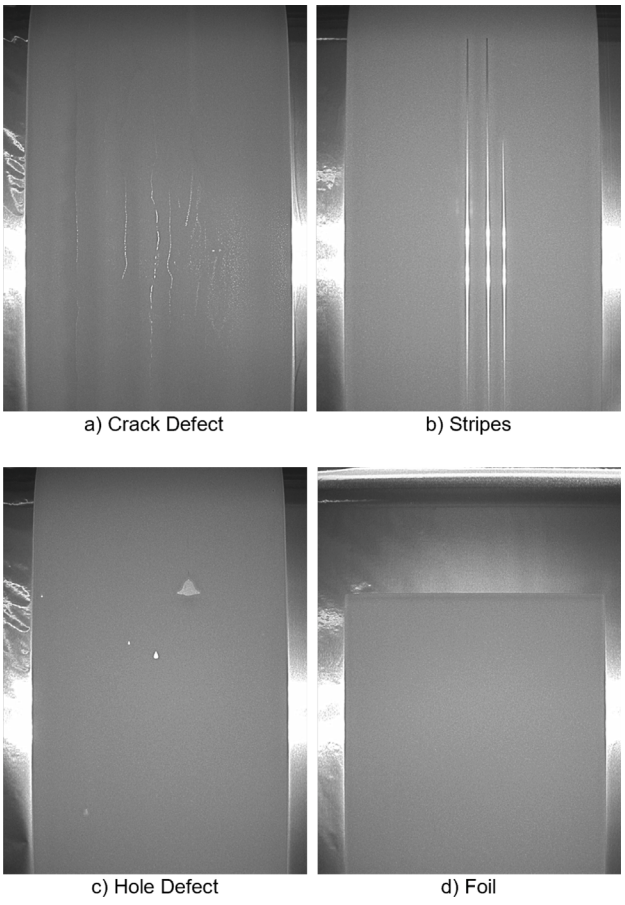
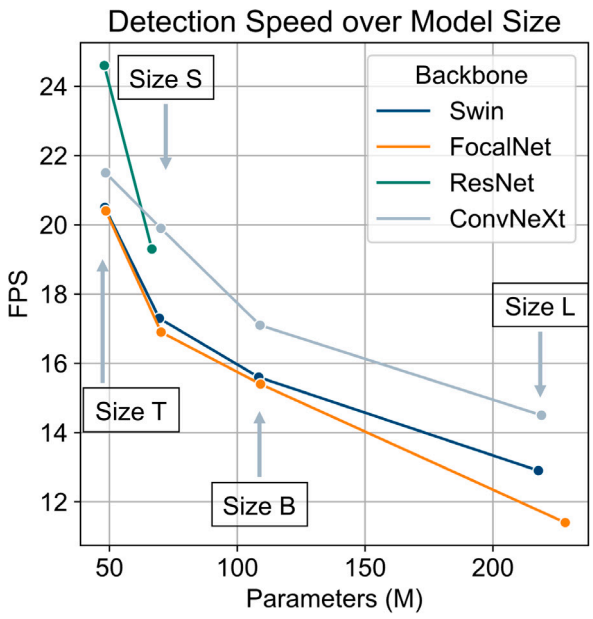
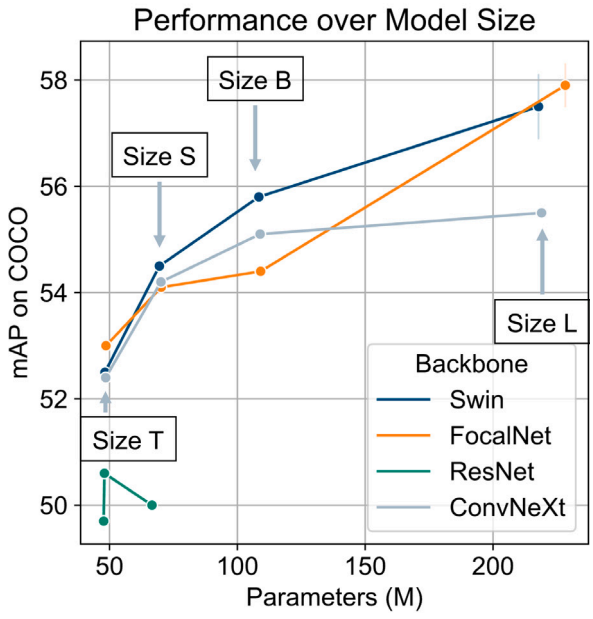


Fig. A.9. Defect example of cracks (a), stripes (b), holes (c), and foil (d).



(a)



(b)

Fig. B.10. Detection speed (a) and performance (b) of DINO with different backbones. All models were benchmarked on a single NVIDIA A100 [50].

defects [11,12]. The dataset used for the experiments in this paper contains cracks, holes, and stripe defects. The model was also trained to recognize the foil. This is not necessarily a defect and needs further evaluation. Fig. A.9 shows examples of these defects.

Appendix B. Detection speed and performance on COCO

As a reference, Ren et al. benchmarked (without whistles and bells) DINO with different backbones and sizes on COCO using a single NVIDIA A100. In opposition to the LIB electrode dataset, an increase in parameters leads to superior detection performance. While FocalNet outscales SWIN performance-wise, SWIN and ConvNeXt remain competitive for small backbone sizes [50].

Table C.4
Relation between units: pixel and mm for the electrode dataset.

Pixel fullsize	Pixel 800p	Defect size (mm)
2	0.782	0.247
4	1.564	0.493
5	1.955	0.616
10	3.910	1.233
15	5.865	1.849
20	7.820	2.465
25	9.775	3.082
5.115	2	0.631
10.23	4	1.261
12.7875	5	1.576
25.575	10	3.153
38.3625	15	4.729
51.15	20	6.305
63.9375	25	7.881

Appendix C. Relation between defect and pixel size

For the comparison, a resolution of 800 × 960 pixels is chosen. The following tables relates objects sizes from pixels to millimeters.

Data availability

Data will be made available on request.

References

[1] H.H. Heimes, W. Bernhart, A. Kampker, B. Dorn, C. Offermanns, G. Bockey, S. Wennemar, K. Sasse, J. Vienenköter, N. Lackner, J.-P. Hasenberg, T. Hotz, D. Gallus, K. Knoche, I. Demir, Haisch, Battery monitor 2022. The value chain in the field of tension between economy and ecology, 2022.
[2] A. König, L. Nicoletti, D. Schröder, S. Wolff, A. Waclaw, M. Lienkamp, An overview of parameter and cost for battery electric vehicles, World Electr. Veh. J. 12 (1) (2021) 21.
[3] H.H. Heimes, A. Kampker, C. Lienemann, M. Locke, C. Offermanns, S. Michaelis, E. Rahimzei, Lithium-Ion Battery Cell Production Process, PEM der RWTH Aachen University and DVMA, Aachen and Frankfurt am Main, December 2018.
[4] P. Nelson, S. Ahmed, K. Gallagher, D. Dees, Modeling the performance and cost of lithium-ion batteries for electric-drive vehicles, third ed., 2019.
[5] K.-H. Pettinger, Prüfverfahren in der fertigung, in: R. Korthauer (Ed.), Handbuch Lithium-Ionen-Batterien, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 259–267.
[6] M.F. Niri, K. Liu, G. Apachitei, L.A. Román-Ramírez, M. Lain, D. Widanage, J. Marco, Quantifying key factors for optimised manufacturing of li-ion battery anode and cathode via artificial intelligence, Energy AI 7 (2022) 100129.
[7] C.D. Reynolds, P.R. Slater, S.D. Hare, M.J. Simmons, E. Kendrick, A review of metrology in lithium-ion electrode coating processes, Mater. Des. 209 (2021) 109971.
[8] A. Turetskyy, J. Wessel, C. Herrmann, S. Thiede, Battery production design using multi-output machine learning models, Energy Storage Mater. 38 (2021) 93–112.
[9] D. Mohanty, E. Hockaday, J. Li, D.K. Hensley, C. Daniel, D.L. Wood, Effect of electrode manufacturing defects on electrochemical performance of lithium-ion batteries: Cognizance of the battery failure sources, J. Power Sources 312 (2016) 70–79.
[10] N. Choudhary, H. Clever, R. Ludwigs, M. Rath, A. Gannouni, A. Schmetz, T. Hülsmann, J. Sawodny, L. Fischer, A. Kampker, J. Fleischer, H.S. Stein, Autonomous visual detection of defects from battery electrode manufacturing, Adv. Intell. Syst. 4 (12) (2022) 2200142.
[11] A. Du Baret de Limé, T. Lein, S. Maletti, K. Schmal, S. Reuber, C. Heubner, A. Michaelis, Impact of electrode defects on battery cell performance: A review, Batter. Supercaps 5 (10) (2022).
[12] A. Schoo, R. Moschner, J. Hülsmann, A. Kwade, Coating defects of lithium-ion battery electrodes and their inline detection and tracking, Batteries 9 (2) (2023) 111.
[13] A. Kwade, W. Haselrieder, R. Leithoff, A. Modlinger, F. Dietrich, K. Droeder, Current status and challenges for automotive battery production technologies, Nature Energy 3 (4) (2018) 290–300.
[14] Y. Liu, Y. Chen, J. Xu, An automatic defects detection scheme for lithium-ion battery electrode surface, in: 2020 International Symposium on Autonomous Systems, ISAS, IEEE, 2020, pp. 94–99.

[15] A. Frommknecht, M. Schmauder, L. Boonen, C. Glanz, Automated inline visual inspection and 3D measuring in electrode manufacturing, in: P. Lehmann, W. Osten, A. Albertazzi Gonçalves (Eds.), Optical Measurement Systems for Industrial Inspection XI, SPIE, p. 66, 24.06.2019 - 27.06.2019.

[16] O. Badmos, A. Kopp, T. Bernthaler, G. Schneider, Image-based defect detection in lithium-ion battery electrode using convolutional neural networks, J. Intell. Manuf. 31 (4) (2020) 885–897.

[17] N. Choudhary, H. Clever, R. Ludwigs, M. Rath, A. Gannouni, A. Schmetz, T. Hülsmann, J. Sawodny, L. Fischer, A. Kampker, J. Fleischer, H.S. Stein, Autonomous visual detection of defects from battery electrode manufacturing, Adv. Intell. Syst. 4 (12) (2022).

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, 2021, arXiv:2010.11929 [cs].

[19] A. Khan, Z. Rauf, A. Sohail, A.R. Khan, H. Asif, A. Asif, U. Farooq, A survey of the vision transformers and their CNN-transformer based variants, Artif. Intell. Rev. 56 (S3) (2023) 2917–2970.

[20] N. Sharp, P. O'Regan, D. Adams, J. Caruthers, A. David, M. Suchomel, Lithium-ion battery electrode inspection using pulse thermography, NDT E Int. 64 (2014) 41–51.

[21] P. Just, J. Rost, T. Echelmeyer, L. Ebert, M.A. Roscher, A method to quantify coating thickness and porosity of electrodes for lithium-ion-batteries, Measurement 89 (2016) 312–315.

[22] R. Szeliski, Computer Vision, Springer International Publishing, Cham, 2022.

[23] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2117–2125.

[24] J. Wang, G. Xu, F. Yan, J. Wang, Z. Wang, Defect transformer: An efficient hybrid transformer architecture for surface defect detection, 2022, arXiv:2207.08319 [cs].

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, 2021, arXiv:2103.14030 [cs].

[26] J. Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, J. Gao, Focal self-attention for local-global interactions in vision transformers, 2021, arXiv preprint arXiv:2107.00641.

[27] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A convnet for the 2020s, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11976–11986.

[28] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, 2020, arXiv preprint arXiv:2004.10934.

[29] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8759–8768.

[30] M. Tan, R. Pang, Q.V. Le, Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020.

[31] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, 2020, arXiv:2005.12872 [cs].

[32] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L.M. Ni, H.-Y. Shum, Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022, arXiv preprint arXiv:2203.03605.

[33] S. Liu, F. Li, H. Zhang, X. Yang, X. Qi, H. Su, J. Zhu, L. Zhang, Dab-detr: Dynamic anchor boxes are better queries for detr, 2022, arXiv preprint arXiv:2201.12329.

[34] F. Li, H. Zhang, S. Liu, J. Guo, L.M. Ni, L. Zhang, Dn-detr: Accelerate detr training by introducing query denoising, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 13619–13627.

[35] F. Pinto, P.H. Torr, P. K. Dokania, An impartial take to the cnn vs transformer robustness contest, in: European Conference on Computer Vision, Springer, 2022, pp. 466–480.

[36] C.-Y. Wang, A. Bochkovskiy, H.-Y.M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[37] M. Prunella, R.M. Scardigno, D. Buongiorno, A. Brunetti, N. Longo, R. Carli, M. Dotoli, V. Bevilacqua, Deep learning for automatic vision-based recognition of industrial surface defects: A survey, IEEE Access 11 (2023) 43370–43423.

[38] F.L. de la Rosa, J.L. Gómez-Sirvent, R. Morales, R. Sánchez-Reolid, A. Fernández-Caballero, Defect detection and classification on semiconductor wafers using two-stage geometric transformation-based data augmentation and SqueezeNet lightweight convolutional neural network, Comput. Ind. Eng. 183 (2023) 109549.

[39] Z. Li, X. Wei, M. Hassaballah, Y. Li, X. Jiang, A deep learning model for steel surface defect detection, Complex Intell. Syst. 10 (1) (2024) 885–897.

[40] A. Bhattacharya, S.G. Cloutier, End-to-end deep learning framework for printed circuit board manufacturing defect classification, Sci. Rep. 12 (1) (2022) 12559.

[41] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G.V. Hernandez, L. Krpalkova, D. Riordan, J. Walsh, Deep learning vs. Traditional computer vision, in: K. Arai, S. Kapoor (Eds.), Advances in Computer Vision, Springer International Publishing, Cham, 2020, pp. 128–144.

[42] K. Imoto, T. Nakai, T. Ike, K. Haruki, Y. Sato, A CNN-based transfer learning method for defect classification in semiconductor manufacturing, IEEE Trans. Semicond. Manuf. 32 (4) (2019) 455–459.

[43] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.-Y. Lo, P. Dollar, R. Girshick, Segment anything, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2023, pp. 4015–4026.

[44] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K.V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, C. Feichtenhofer, SAM 2: Segment anything in images and videos, 2024, arXiv preprint arXiv:2408.00714.

[45] S. Kim, Y.-K. Noh, F.C. Park, Efficient neural network compression via transfer learning for machine vision inspection, Neurocomputing 413 (2020) 294–304.

[46] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Comput. Ind. Eng. 149 (2020) 106854.

[47] G. Algan, İ. Ulusoy, Label noise types and their effects on deep learning, 2020.

[48] C. Sager, C. Janiesch, P. Zschech, A survey of image labelling for computer vision applications, J. Bus. Anal. 4 (2) (2021) 91–110.

[49] G. Jocher, A. Chaurasia, J. Qiu, Ultralytics YOLOv8, 2023.

[50] T. Ren, S. Liu, F. Li, H. Zhang, A. Zeng, J. Yang, X. Liao, D. Jia, H. Li, H. Cao, et al., Detrex: Benchmarking detection transformers, 2023, arXiv preprint arXiv:2306.07265.

[51] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2, 2019, https://github.com/facebookresearch/detectron2.

[52] B.E. Moore, J.J. Corso, Fiftyone, 2020, GitHub. Note: https://github.com/voxel51/fiftyone.

[53] T. Gandor, J. Nalepa, First gradually, then suddenly: Understanding the impact of image compression on object detection using deep learning, Sensors 22 (3) (2022) 1104.

[54] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, et al., Internimage: Exploring large-scale vision foundation models with deformable convolutions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 14408–14419.

[55] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[56] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, J. Dai, Deformable detr: Deformable transformers for end-to-end object detection, 2020, arXiv preprint arXiv:2010.04159.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, 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.