TU Delft

Delft University of Technology

Sequential wafer map inspection via feedback loop with reinforcement learning

Dekhovich, Aleksandr; Soloviev, Oleg; Verhaegen, Michel

DOI
10.1016/j.eswa.2025.126996

Publication date
2025

Document Version
Final published version

Published in
Expert Systems with Applications

Citation (APA)
Dekhovich, A., Soloviev, O., & Verhaegen, M. (2025). Sequential wafer map inspection via feedback loop with reinforcement learning. *Expert Systems with Applications*, *275*, Article 126996. https://doi.org/10.1016/j.eswa.2025.126996

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

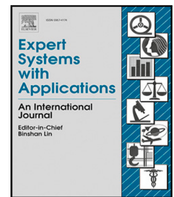
This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Expert Systems With Applications 275 (2025) 126996

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Sequential wafer map inspection via feedback loop with reinforcement learning

Aleksandr Dekhovich [a,*], Oleg Soloviev [a,b], Michel Verhaegen [a]

[a] Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, The Netherlands
[b] Flexible Optical B.V., Polakweg 10–11, Rijswijk, 2288 GG, The Netherlands

ARTICLE INFO

Dataset link: https://github.com/adekhovich/sequential_wafer_inspection

Keywords:
Reinforcement learning
Confidence estimation
Visual inspection
Wafer map
Industry 4.0

ABSTRACT

Wafer map defect recognition is a vital part of the semiconductor manufacturing process that requires a high level of precision. Measurement tools in such manufacturing systems can scan only a small region (patch) of the map at a time. However, this can be resource-intensive and lead to unnecessary additional costs if the full wafer map is measured. Instead, selective sparse measurements of the image save a considerable amount of resources (e.g. scanning time). Therefore, in this work, we propose a feedback loop approach for wafer map defect recognition. The algorithm aims to find sequentially the most informative regions in the image based on previously acquired ones and make a prediction of a defect type by having only these partial observations without scanning the full wafer map. To achieve our goal, we introduce a reinforcement learning-based measurement acquisition process and recurrent neural network-based classifier that takes the sequence of these measurements as an input. Additionally, we employ an ensemble technique to increase the accuracy of the prediction. As a result, we reduce the need for scanned patches by 38% having higher accuracy than the conventional convolutional neural network-based approach on a publicly available WM-811k dataset.

1. Introduction

Recent developments in artificial intelligence (AI), and especially deep learning (DL), attract much attention in many fields outside of Computer Science. One such area is the automation of manufacturing processes with intelligent digital technologies, which is also known as Industry 4.0 (Ghobakhloo, 2020). Deep learning applications are widely used in visual inspection problems including wafer defect recognition (Kim & Behdinan, 2023) which is crucial for high-quality semiconductor manufacturing. At the same time, deep neural networks (DNNs) have achieved tremendous progress in computer vision problems such as image classification (Russakovsky et al., 2015), object detection (Zhao, Zheng, Xu, & Wu, 2019) and image generation (Croitoru, Hondru, Ionescu, & Shah, 2023). Hence, a lot of modern vision-based monitoring industrial systems use deep neural networks for assessing the quality of the production (Tabernik, Šela, Skvarč, & Skočaj, 2020; Zhou, Zhang, & Konz, 2022), including wafer map analysis in semiconductor manufacturing (Yu, Xu, & Wang, 2019).

Convolutional neural networks (CNNs) have become a powerful tool in failure pattern recognition of the wafer maps (Bae & Kang, 2023; Chen, Zhang, Hou, Shang, & Yang, 2022; Ishida, Nitta, Fukuda, & Kanazawa, 2019; Wang, Chou, & Amogne, 2022). Recently, large vision-language models (LVLMs) (Wang, Li, & Li, 2024) has been also employed as pretrained backbone adapting it for industrial vision monitoring. The development of these approaches has also been accelerated due to the availability of open-access databases of real wafers (Wang, Xu, Yang, Zhang, & Li, 2020; Wu, Jang, & Chen, 2014) that contain enough images for training DNNs. In addition, CNNs require scanning the full image for the input which leads to the unwanted costs since the defect can be located in a small region, while the rest is not useful (see Fig. 1 (left)). However, CNN-based approaches show the state-of-the-art results in terms of prediction accuracy which makes them a powerful tool in solving the defect recognition problem. At the same time, CNNs are often overparameterized, which can make them difficult to optimize when data is insufficient. Thus, pretraining and transfer learning techniques have been utilized to facilitate the learning process (Yu, Shen, & Wang, 2021).

However, the goal of a good monitoring system for the semiconductor industry is increasing the throughput of the tool having the same precision to tolerance (P/T) ratio (Ukraintsev, 2003) by reducing the number of measurements required for making a decision. The state-of-the-art approaches still use the predefined sequence of measurements for making a prediction (Bergner, Lippert, & Mahendran, 2023; Doutt et al., 2023), therefore, making it individual for every example and

* Corresponding author.
E-mail addresses: A.Dekhovich@tudelft.nl (A. Dekhovich), O.A.Soloviev@tudelft.nl (O. Soloviev), M.Verhaegen@tudelft.nl (M. Verhaegen).

https://doi.org/10.1016/j.eswa.2025.126996
Received 28 October 2024; Received in revised form 13 February 2025; Accepted 19 February 2025
Available online 28 February 2025
0957-4174/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

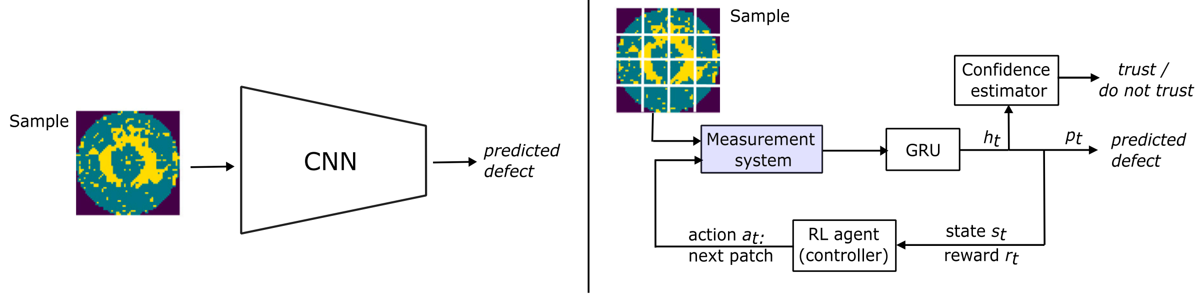


Fig. 1. Conventional CNN-based approach (left) and the proposed RL-based closed-loop with the GRU classifier (right). In the conventional approach, the whole wafer should be scanned by the measurement system; in the proposed approach, the wafer surface is analyzed by small subregions (patches); the location of the next patch to scan is defined by an RL agent based on the obtained information (feedback loop).

adding flexibility to the measuring tool is an open challenge. Adaptive measurements acquisition (Silvestri, Massoli, Orekondy, Abdi, & Behboodi, 2024; Yin et al., 2021) proved its importance in the fast MRI reconstruction (Zbontar et al., 2018). By taking fewer measurements during scanning and reconstructing the original MRI with the acquired information, the tool can significantly reduce the scanning time increasing the comfort of a patient. Thus, Bakker, van Hoof, and Welling (2020) proposed a simple RL-based greedy policy search for measurements acquisition in the frequency domain, also known as the k-space (Liang & Lauterbur, 2000). In contrast to these works, (Pineda, Basu, Romero, Calandra, & Drozdzal, 2020) adapted a Double Deep Q-Networks (Van Hasselt, Guez, & Silver, 2016) algorithm for k-space sampling. In this type of algorithm, a neural network predicts a value of each action in the current step instead of modeling the sampling policy directly.

In the context of the image classification problem, Yen et al. (2024) also employed the RL paradigm for pathology prediction from k-space accelerating the sampling process by 12 times. Also, Chu, Li, Chang, and Wang (2019) proposed an RL-based patch selector which finds the best input sequence for the Gated Recurrent Unit (GRU) network (Cho et al., 2014) in the few-shot image classification task. Noom, Thao, Soloviev, and Verhaegen (2020) proposed solving an automated visual inspection problem via the Closed-Loop Active Model Diagnosis (CLAMD) framework. The main novelty of CLAMD is the usage of the Bhattacharyya coefficients (Bhattacharyya, 1943, 1946) to minimize the model misdiagnosis in the vision inspection task. Also, the Bhattacharyya coefficients serve as a tool to stop the scanning process yielding the misdiagnosis error after each measurement. Following developments of CLAMD were employed to object the recognition problem with constrained illumination power. The algorithm aimed to minimize phototoxicity and decrease the number of measurements required for making a prediction (Noom, Soloviev, Smith, & Verhaegen, 2022). It should be noted that CLAMD focuses on minimizing the number of measurements needed, while other works in image classification (Chu et al., 2019; Yen et al., 2024) work with predefined number of measurements. However, the number of models in CLAMD increases with image size, which does not allow the method to be scaled.

We assume that the input (wafer map) is initially not fully observed and a measurement system can scan only a small patch at each time step. Therefore, for example, CNN-based algorithms should first scan all patches and then predict the defect type. We want to mitigate this challenge since this patch acquisition process has significant resource costs. Thus, in our work, we combine the feedback loop paradigm with the RL approach for the wafer defect recognition problem. The overall structure of the method and its difference from the conventional CNN-based approach are shown in Fig. 1. The proposed framework utilizes a GRU model for processing sequential measurements, while a controller, an RL algorithm, predicts the next best action — new patch (or measurement) based on already acquired ones. A confidence estimation model is trained to evaluate how certain the model is about its prediction after each measurement. This also allows the model to stop the scanning process if the confidence in the GRU output at the current time step is high enough. Current approaches in the literature aim for higher prediction accuracy while using the full wafer. Thus, to the best of our knowledge, our work is the first attempt at adaptive information acquisition for the wafer map recognition problem. We believe that our algorithm uses as many patches as necessary to make a correct prediction in contrast to the strategy where the number of scanned patches is fixed. We show the advantages of our feedback loop approach compared to the open-loop one with CNN both in terms of accuracy and patch acquisition costs. Our main contribution can be formulated as follows:

- We propose a feedback loop approach for wafer map defect recognition. The algorithm aims to minimize the number of measurements (in contrast to previous works on MRI measurements acquisition (Bakker et al., 2020; Pineda et al., 2020; Yen et al., 2024)) and scales well with the increase of image size or decrease of the scanned region (in contrast to CLAMD);
- We decrease the number of necessary patches scanned from the image by 38% on WM-811k dataset (Wu et al., 2014) of real wafers having the same prediction accuracy as CNN models that need access to the full image;
- By introducing the confidence estimator, we can determine the images that require additional inspection with more advanced models or human feedback. We send these images to a CNN-based model which leads to better accuracy without additional scanning costs.
- Additionally, we employ an ensemble procedure to increase the robustness of each component and to boost the accuracy, outperforming the CNN approach.

The structure of the paper is as follows: Section 2 gives the overview of the required theoretical background for the proposed algorithm and Section 3 explains the method and its core components in detail. Section 4 sets the experiments and provides the results on the open wafer map database (Wu et al., 2014). Finally, Section 5 concludes the article and shows future steps to explore. The code implementation of the work can be found here: https://github.com/adekhovich/sequential_wafer_inspection.

2. Related work

For the feedback loop formulation of this decision-making problem, we need to actively select the input that is the most informative at the current time step, feed it to the classifier and, select the next input or stop the process based on the feedback from the classifier. To address this challenge we consider a (Deep) Reinforcement learning (RL) approach (Sutton & Barto, 2018) which is well-suited for this type of problem. In addition, we want to evaluate the trustworthiness of the prediction after each measurement (scanned patch), and therefore, we also revisit some uncertainty quantification techniques for classification problems.

2.1. RL preliminaries

Reinforcement learning problems are formulated as Markov Decision Process (MDP) with a tuple $(\mathcal{S}, \mathcal{A}, p(s'|s,a), R(s,a), p_0(s), \gamma, T)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space and $a \in \mathcal{A}$ is an action, $p(s'|s,a)$ is a transition distribution, $R(s,a)$ is a reward function, $p_0(s)$ is an initial state distribution, $\gamma \in [0,1]$ is a discount factor, and T is a task horizon (Yen et al., 2024). An agent interacts with the environment according to its policy $\pi(a|s)$ that maps from $\mathcal{S}$ to $\mathcal{A}$. This policy can be found by maximizing the expected cumulative discounted sum of rewards $\mathbb{E}_\pi\left[\sum_{t=0}^{T-1}\gamma^t R(s_t,a_t)\right]$, where $s_0 \sim p_0(\cdot)$, $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim p(\cdot|s_t,a_t)$.

One way of maximizing the expected return and deriving the optimal policy π^* is by approximating it with some model π_θ with parameters θ (Sutton, McAllester, Singh, & Mansour, 1999) and optimizing the parameters of this model by gradient ascent. Williams (1992) proposed REINFORCE algorithm that computes the return $G_t = \sum_{k=t}^{T-1}\gamma^{k-t}R(s_k,a_k)$ and updates the parameters in the direction of $\nabla_\theta \ln \pi_\theta(a_t|s_t)G_t$, where $t = 0,1,\dots,T-1$. Additionally, a subtraction of a baseline model value $b_t(s_t)$ from the return G_t can reduce the variance in gradient estimates while leaving the expected value of the estimate unchanged. The update of θ occurs then in the direction of $\nabla_\theta \ln \pi_\theta(a_t|s_t)\big(G_t - b_t(s_t)\big)$.

Over the past decades, many further improvements have been made to this algorithm, which has come to be known as actor-critic algorithms (Sutton & Barto, 2018). In this type of approach, the critic evaluates the updates made by the actor to the policy, and the actor makes changes with respect to the critic's evaluation. Many variants of actor-critic algorithms also include DNNs as function approximates (Li, 2017). Among them are advantage actor-critic (A2C) and asynchronous advantage actor-critic (Mnih et al., 2016), proximal policy optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017).

2.2. Uncertainty estimation and ensembles

One of the most important challenges in deep learning is understanding how reliable the model's prediction is. During testing, we do not have ground truth labels, therefore, a learning system should have an estimation of the quality of its prediction. Moreover, deep neural networks tend to produce overconfident predictions even if they are wrong (Lakshminarayanan, Pritzel, & Blundell, 2017). For these reasons, several approaches have been developed over the last couple of decades to assess the certainty in prediction (Hüllermeier & Waegeman, 2021). We focus on the methods that can help us to evaluate the confidence in our GRU-based classifier, i.e. epistemic uncertainty. In this subsection, we briefly discuss some of them, namely Bayesian neural networks (BNNs), ensembles and confidence prediction.

In the Bayesian approach (Kendall & Gal, 2017; Maddox, Izmailov, Garipov, Vetrov, & Wilson, 2019), we aim to quantify the uncertainty of the prediction by modeling the distribution of the parameters instead of making a point-wise estimate. This means that during training Bayesian Neural Network (BNN) learns a posterior distribution $p(\theta|\mathcal{D})$ of its parameters θ on the observed data $\mathcal{D}$. At the inference stage, the following integral is computed by Bayesian model averaging (BMA) (Wilson & Izmailov, 2020): $p(y|\mathbf{X},\mathcal{D}) = \int_\theta p(y|\mathbf{X},\theta)p(\theta|\mathcal{D})d\theta$, where $p(y|\mathbf{X},\theta)$ is the output distribution for the set of parameters θ. However, in the case of neural networks, this integral cannot be computed analytically and should be approximated for example with Markov Chain Monte Carlo (MCMC) sampling (Izmailov, Vikram, Hoffman, & Wilson, 2021). This is an obvious flaw since MCMC sampling is computationally expensive. An alternative approach was proposed by Lakshminarayanan et al. (2017) that is called Deep Ensembles. In this algorithm, multiple neural networks with different parameters initialization are trained independently, and then for a given test point a prediction with each of them is made followed by the averaging of the outputs across all networks. The confidence boundaries can be computed from standard deviations of predictions. This method is much simpler in implementation and training than BNNs and yet is a good baseline for uncertainty quantification (Tan, Urata, Goldman, Dietschreit, & Gómez-Bombarelli, 2023). In addition, averaging of outputs often leads to a more accurate prediction (Lakshminarayanan et al., 2017).

Predicting the model's confidence in the classification task is another way of determining the quality of its prediction. Early works (Hendrycks & Gimpel, 2016) used Maximum Class Probability (MCP) of the network output after applying Softmax to determine how a model is certain in its prediction: $\mathtt{MCP}(\mathbf{X}) = \max_{k\in\mathcal{Y}} \mathrm{P}(Y=k|\mathbf{X},\theta)$, where $\mathbf{X}$ is the input data point and $\mathcal{Y}$ is the set of labels. In this method, higher values of these probabilities correspond to greater certainty in the model. However, DNNs tend to overestimate these probabilities even if they produce wrong predictions, therefore, MCP is not reliable in many cases (Nguyen, Yosinski, & Clune, 2015). Thus, Corbière, Thome, Bar-Hen, Cord, and Pérez (2019) proposed the True Class Probability (TCP) measure, which is the probability that the neuron that corresponds to the correct label produces: $\mathtt{TCP}(\mathbf{X}) = \mathrm{P}(Y=y^*|\mathbf{X},\theta)$, where $\mathbf{X}$ is the input data point and y^* is its correct label. Then, a model called Confidnet is trained on pairs $\{(\mathbf{X}, \mathtt{TCP}(\mathbf{X})),\ \mathbf{X}\in\mathcal{D}_{\text{train}}\}$ which is possible since the correct label y^* is available on training set. At inference stage, Confidnet takes the test image $\mathbf{X}_{\text{test}}$ and predicts its confidence $c(\mathbf{X}_{\text{test}};\omega)$, where ω are the parameters of Confidnet.

3. Method

Let $\mathbf{X}\in\mathbb{R}^{N_1\times N_2}$ be an input wafer map given for an inspection and $y\in\{1,2,\dots,K\}$ is its defect type (label). Let us also denote with $\mathbf{y}$ the one-hot encoded label y. At each time step t, the measuring tool can scan only a small patch $\mathbf{x}_{i_t}\in\mathbb{R}^{n_1\times n_2}$, $n_1 \ll N_1$ and $n_2 \ll N_2$. We assume that these patches do not overlap and must cover the whole image $\mathbf{X}$, meaning that $n_1|N_1$, $n_2|N_2$ and for the final patch $\mathbf{x}_{i_T}$ holds: $T = \frac{N_1N_2}{n_1n_2}\in\mathbb{Z}$. Our goal is to collect measurements $\mathbf{x}_{i_1},\mathbf{x}_{i_2},\dots,\mathbf{x}_{i_{t^*}}$ and make a prediction $\hat{y}_{t^*} := \mathcal{F}_w(\mathbf{x}_{i_1},\mathbf{x}_{i_2},\dots,\mathbf{x}_{i_{t^*}})\in\{1,2,\dots,K\}$ based on these measurements using the classifier $\mathcal{F}_w$ with learnable parameters w, such that:

- For $\forall t<t^*$ $\hat{y}_t := \mathcal{F}_w(\mathbf{x}_{i_1},\mathbf{x}_{i_2},\dots,\mathbf{x}_{i_t}) \neq y$ and $\hat{y}_{t^*} = y$;
- t^* is the minimal time step over all possible patch permutations $\sigma = (i_1,i_2,\dots,i_T)$ of numbers $1,2,\dots,T$.

In other words, we aim to train an inspection model that makes a prediction based on the smallest possible number of scanned patches. We can define the ratio of non-scanned patches as $R = (1-\frac{t^*}{T})\cdot 100\%$. The higher this ratio, the better, as it means we can scan fewer patches. We also refer to the number of scanned patches as the number of measurements. The approach that we propose consists of three core models:

1. Classifier $\mathcal{F}_w$, that takes a sequence of patches as an input and is invariant to the order of the patches. For this purpose, we pretrain a GRU model permuting input sequence arbitrary for each image. We construct the loss for GRU in such a way that its intermediate outputs can also be used for prediction (see Section 3.1)
2. Confidence estimator c_ω that evaluates the trustworthiness of the prediction at the current time step and decides whether to continue measurements or stop them with the current prediction. This is also a GRU model that maps the hidden state of classifier $\mathbf{z}_t$ to a confidence $\hat{c} = c_\omega(\mathbf{z}_t)$ in classifiers' prediction (see Section 3.2).
3. Active patch selector π_θ – a sampler that generates an input sequence for the pretrained classifier $\mathcal{F}_w$ that minimizes the number of measurements needed to make a reliable prediction. Here we utilize a reinforcement learning approach for this task: next patch $a_t = \pi_\theta(\mathbf{z}_t,\mathbf{m}_t)$, where $\mathbf{z}_t$ is a feedback signal from the

GRU and $\mathbf{m}_t \in \{0,1\}^T$, $t = 0,1,\dots,T$, is a mask that indicates which patches are available at time step t, e.g. $\mathbf{m}_0 = (1,1,\dots,1)$ and $\mathbf{m}_T = (0,0,\dots,0)$ (see Section 3.3).

It is worth noting that these models are trained sequentially one after another and not all together. First, we train the classifier $\mathcal{F}_w$ to be invariant to different patch sequences, then we can train a confidence estimator c_ω based on classifier's response to evaluate the quality of its predictions at different time steps (from partial observations $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_t}$). At last, having the classifier and its confidence estimator, we train the input sequence generator π_θ that finds the shortest sequence of measurements needed for making a prediction for the given input $\mathbf{X}$.

In addition, we want to point out it is possible that we still need to scan the full image since confidence estimator c_ω may not be sure about the predictions of GRU classifier $\mathcal{F}_w$. In this case, we propose to pay additional attention to such images and evaluate them with a CNN-based model. It will to affect the total number of measurements since they all have been collected for GRU. We call such approach as *hybrid approach* since it includes both GRU and CNN-based classifiers.

3.1. GRU classifier

As mentioned before, we have selected GRU network (Cho et al., 2014) as a classifier, a variant of a recurrent network (Rumelhart, Hinton, & Williams, 1986), to process sequential data and make a prediction from partial observations. Also, for our problem the GRU need to be able to predict from an arbitrary order of patches, that is, it should be invariant to the patch order. To achieve this, we follow the Vision Transformer (ViT) strategy (Dosovitskiy et al., 2021) introducing a patch and its position together to the classifier. Each patch $\mathbf{x}_{i_t} \in \mathbb{R}^{n_1 \times n_2}$ at time step t is flattened to a vector $\mathbf{z}_t \in \mathbb{R}^{n_1 n_2}$ to which we apply linear projection using MLP and sum up with learnable embedding $\mathbf{E}_{i_t} = \texttt{Embedding}(i_t)$, the encoding of position i_t. The resulting vector is then fed to the GRU with L cells and a Layer normalization (Ba, Kiros, & Hinton, 2016) (LN) after each of them. Fig. 2 illustrates the architecture. Formally, the expressions for the model's architecture are as follows:

$$
\begin{aligned}
\mathbf{z}_t^{(0)} &= \texttt{MLP}(\texttt{Flattened}(\mathbf{x}_{i_t})) + \mathbf{E}_{i_t},\\
\mathbf{z}_t^{(l)} &= \texttt{LN}\left(\texttt{GRUCell}\left(\mathbf{z}_t^{(l-1)}\right)\right), \quad l = 1,2,\dots,L,\\
\hat{\mathbf{y}}_t &= \texttt{Softmax}\left(\texttt{MLP}\left(\mathbf{z}_t^{(L)}\right)\right),\\
\hat{y}_t &= \underset{k=1,2,\dots,K}{\arg\max}\{\hat{\mathbf{y}}_{t,k}\} \in \{1,2,\dots,K\}
\end{aligned}
\tag{1}
$$

Our objective is to make the classifier invariant to different sequences of patches, meaning that for any permutation $\sigma = (i_1, i_2, \dots, i_T)$ of numbers $1,2,\dots,T$ we want $\mathcal{F}_w(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T) = \mathcal{F}_w(\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_T})$. Then, it is unimportant to the classifier in which order we scan the image and it can predict correctly from any order. To achieve that, we apply some random permutation of patches to every image in the batch. This can be seen as part of the data augmentation process.

However, we require the classifier to make meaningful predictions at the last time step T and at intermediate steps $t < T$. Therefore, the loss function for GRU pretraining can have a form $\mathcal{L} = \sum_{t=1}^{T} \texttt{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}_t)$. To simplify the learning procedure, we propose to sample a small subset $\{i_1, i_2, \dots, i_\tau\} \subset \{1,2,\dots,T\}$ of output heads and optimize the loss $\mathcal{L} = \sum_{i \in \{i_1, i_2, \dots, i_\tau\}} \texttt{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}_i)$. This subset is sampled from the uniform distribution individually for each data point in a batch at every optimization step. As we show in Section 4.2, by introducing this loss we provide the model the ability to give reliable prediction much earlier than in the conventional training scenario.

3.2. Confidence estimator

Once the classifier is trained, we need to have the ability to make a reliable prediction based on some intermediate time step $t < T$. The existence of such a time step is provided by the loss we constructed, however, the optimal stopping point depends on the patch ordering $\sigma = (i_1, i_2, \dots, i_T)$ and input $\mathbf{X}$. To address this task, we train a separate model on top of the classifier that estimates the confidence c_t in classifier prediction $\hat{y}_t$ at time step t. Following the work (Corbière et al., 2019), we build a confidence network c_ω which aims to predict the True Class Probability (TPC) $\mathrm{P}(Y = y|\mathbf{w}, \mathbf{X})$ for the image $\mathbf{X}$ with label y. In contrast to the examples in Corbière et al. (2019), we use the GRU model for confidence estimation since our signal is time-dependent.

Fig. 3 shows how confidGRU cooperates with the classifier $\mathcal{F}_w$: at time step t we first compute the feature vector $\mathbf{z}_t^{(L)}$ which is then fed to the confidGRU c_ω as input. Based on the estimated confidence $\hat{c}_t = c_\omega(\mathbf{z}_t^{(L)})$, we make a decision on whether to continue to take new patches (if $\hat{c}_t < 0.5$), or stop scanning (if $\hat{c}_t \geq 0.5$) and our final prediction is $\hat{y} := \arg\max_{k=1,2,\dots,K}\{\hat{\mathbf{y}}_{t,k}\}$. The reason why 0.5 is set as a threshold is explained in the original article (Corbière et al., 2019). The main idea here is that the classifier correctly predicts a defect y if $\mathrm{P}(Y = y|\mathbf{w}, \mathbf{X}) \geq 0.5$. Therefore, we guarantee a correct prediction as long as c_ω approximates $\mathrm{P}(Y = y|\mathbf{w}, \mathbf{X})$ well and $\hat{c}_t \geq 0.5$. As a result, if for some input wafer image $\mathbf{X}$ there exists t^* such that $c_t^* \geq 0.5$, we do not need to process the remaining patches. Otherwise, we measure the full image $\mathbf{X}$.

As indicated in Section 2.2, there are many techniques for uncertainty/confidence estimation (Corbière et al., 2019; Gal & Ghahramani, 2016). We justify the choice of confidGRU by the simple interpretability of its output for our task. In contrast to Bayesian deep learning techniques (Maddox et al., 2019) that estimate the distribution of the prediction, confidGRU produces the binary rule that we can easily interpret as *trust* or *do not trust* the prediction at the given time step.

3.3. Active patch selector

In the previous subsection, we introduced the confidence-based stopping criterion for taking the measurements from an image. However, we considered an arbitrary patch ordering σ. Thus, our next goal is generating the optimal ordering σ^* such that $t^* = t(\mathbf{X}, \sigma^*)$ was the minimal possible time step for which $\hat{c}_{t^*} \geq 0.5$. In other words, we want to find a patch ordering σ^* such that $\hat{c}_{t^*} \geq 0.5$, $\hat{c}_t < 0.5\ \forall t < t^*$ and for any other patch ordering σ its optimal stopping time step $t(\mathbf{X}, \sigma) > t^*$.

A natural choice for this task is formulating it as a reinforcement learning problem and finding an optimal policy π_θ^* that generates the optimal ordering σ^*. At every time step t, the policy maps the current state s_t to the next best action a_t, $\pi_\theta^* : s_t \mapsto a_t$, where we define the current state as $s_t = [\mathbf{z}_t^{(L)}, \mathbf{m}_t]$ – a vector that consists of the normalized last hidden state $\mathbf{z}_t^{(L)}$ (before MLP layer) of the GRU network and a mask $\mathbf{m}_t \in \{0,1\}^T$ of available actions (patches) at time t. The total return is defined as $G = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$, where $\gamma \in [0,1]$ and $r_{t+1} = R(s_t, a_t)$ is a reward obtained by taking action a_t from state s_t at time t. We define this reward as:

$$
r_t = \begin{cases} 1, & \text{if } \hat{c}_t \geq 0.5, \\ 0, & \text{otherwise.} \end{cases}
\tag{2}
$$

Thus, the reward function defined this way promotes actions that lead to more confident predictions of the classifier and stop the scanning process earlier. To train the policy model π_θ, we follow the policy-gradient type RL algorithm (Sutton et al., 1999). We observed that a simpler REINFORCE algorithm (Williams, 1992) works better in our case than more advanced actor-critic-based ones such as PPO (Schulman et al., 2017). We explain this with the difficulty of training a more complex model with the limited amount of data we have (see Section 4.1 for details).

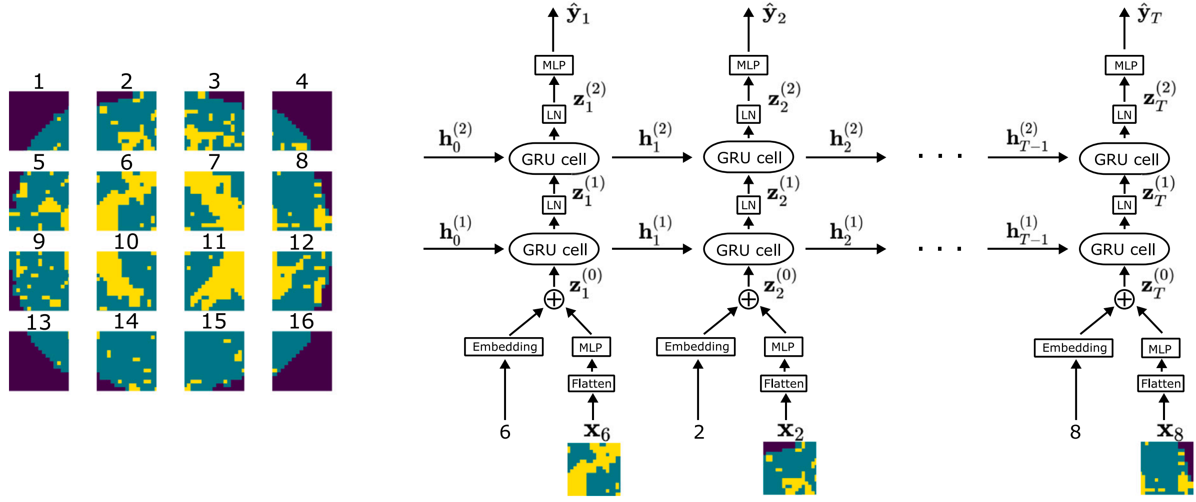


Fig. 2. Proposed multi-output GRU (MO-GRU) model that is invariant to the random patch ordering: at each time t, MO-GRU takes the patch and its position in some predefined order (e.g. $1 \to 2 \to \ldots \to 16$).

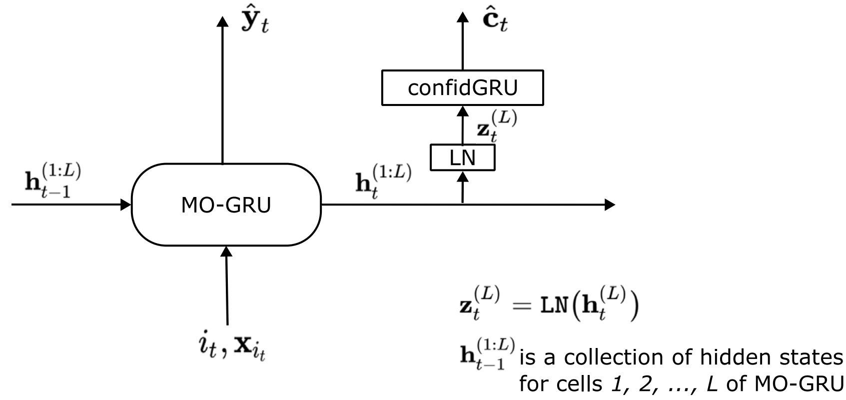


Fig. 3. Schematic of confidGRU that evaluates the confidence in the current prediction $\hat{y}_t$. It takes the hidden state from the last GRU cell as input and outputs $\hat{c}_t$ which approximates the TPC.

During inference, we take measurements according to the learned policy π_θ and stop the process when $\hat{c}_t \geq 0.5$. The pseudocode is described in Algorithm 1. Also, we want to notice that if all $\hat{c}_t < 0.5$ that means that we scanned the full image and the GRU model is still not sure about its prediction. In this case, we can classify the image with a convolutional neural network, e.g. ResNet (He, Zhang, Ren, & Sun, 2016), which typically works better for fully observed images. Algorithm 1 also illustrates this idea.

3.4. Ensembling (inference)

In the previous subsections, we described the core blocks of the feedback loop model, namely the GRU network $\mathcal{F}_w$ that plays the role of a classifier, confidGRU c_ω that estimates the confidence in classifier prediction at every time t, and a patch sequence generator that aims to facilitate inference by giving the classifier the most informative patches to speed up the decision. It is worth noting that training of these three models involves a lot of randomnesses: first, we train GRU on random sequences of patches, then confidGRU is trained on top of this pretrained GRU, so its performance depends on how well the GRU was trained. At last, patch selector π_θ uses both GRU and confidGRU in its training loop, making it sensitive to the GRU and confidGRU training stages. To increase the robustness of our approach, we propose using ensembles of the networks at each stage: classification, confidence estimation and next patch prediction. However, we do not use ensembles for estimating the standard deviations of the outputs of these models.

In ensembling, one has to decide how to aggregate the predictions from several models. The simplest way is averaging the outputs of the networks. For example, if we have M GRU models $\mathcal{F}_w^{(1)}, \mathcal{F}_w^{(2)}, \ldots, \mathcal{F}_w^{(M)}$ and the corresponding output probabilities are $\hat{\mathbf{y}}_t^{(1)}, \hat{\mathbf{y}}_t^{(2)}, \ldots, \hat{\mathbf{y}}_t^{(M)}$, the final output at time t can be found as $\hat{\mathbf{y}}_t = \frac{1}{M}\sum_{i=1}^M \hat{\mathbf{y}}_t^{(i)}$. Similarly, we put $\hat{c}_t = \frac{1}{M}\sum_{i=1}^M \hat{c}_t^{(i)}$ for confidence estimation. For aggregation of policy outputs, we can consider two different strategies:

- `averaging`: similarly to the previous cases, $p_t^{(i)} = \pi_\theta^{(i)}(s_t)$ and $p_t = \frac{1}{M}\sum_{i=1}^M p_t^{(i)}$, then the next patch number a_t is a mode of categorical distribution with K categories and probabilities $p_t = (p_{t1}, p_{t2}, \ldots, p_{tK})$.
- `min-entropy selection`: in contrast to averaging, we can select the policy π_θ that is most certain at current time t. We propose to evaluate the certainty of the policy models with the entropy of their prediction: $\mathrm{H}_t^{(i)} = -\sum_{k=1}^K p_{tk}^{(i)} \log p_{tk}^{(i)}$, $i = 1, 2, \ldots, K$. Therefore, at time t, we select the policy $\pi_\theta \in \{\pi_{\theta_1}^{(1)}, \pi_{\theta_2}^{(2)}, \ldots, \pi_{\theta_M}^{(M)}\}$ that gives the smallest value among $\mathrm{H}_t^{(1)}, \mathrm{H}_t^{(2)}, \ldots, \mathrm{H}_t^{(M)}$.

The pseudocode for ensemble strategy during inference is also shown in Algorithm 2. Analogously with previous cases, if we needed to scan the full image, we can use another model (e.g. CNN-based) to

Algorithm 1 Active patch selection with RL

```
1: function INFERENCE(GRU classifer F_w, confidGRU c_ω, policy network π_θ, auxiliary CNN, patches {x_1, x_2, ..., x_T}, hybrid = True or False)
2:     ŷ ← ∅
3:     m_0 ← (1, 1, ..., 1)                                   ▷ mask initialization
4:     z_0^{(L)(i)} ← (0, 0, ..., 0)
5:     for t = 0, 1, ..., T − 1 do
6:         s_t ← [z_t^{(L)}, m_t]                               ▷ state initialization
7:         a_t ← π_θ(s_t)                                       ▷ next patch prediction
8:         m_{t+1} ← m_t;  m_{t+1}[a_t] ← 0                     ▷ mask update
9:         ŷ_{t+1}, z_{t+1}^{(L)} ← F_w(a_t, x_{a_t})
10:        ĉ_{t+1} ← c_ω(z_{t+1}^{(L)})                         ▷ confidence estimation
11:        if ĉ_{t+1} ≥ 0.5 then ŷ ← ŷ_{t+1} and stop loop      ▷ not full scan case
12:        end if
13:    end for
14:    if ŷ = ∅ then                                            ▷ full scan case
15:        if hybrid then  ŷ ← CNN(x)
16:        else ŷ ← ŷ_T
17:        end if
18:    end if
19:    return ŷ
20: end function
```

Algorithm 2 Ensemble strategy during inference

```
1: function ENSEMBLE INFERENCE(GRU classifers {F_{w_i}^{(i)}}_{i=1}^M, confidGRUs {c_{ω_i}^{(i)}}_{i=1}^M, policy networks {π_{θ_i}^i}_{i=1}^M, auxiliary CNN, patches {x_1, x_2, ..., x_T},
   aggregation_type = averaging or min-entropy, hybrid = True or False)
2:     ŷ ← ∅
3:     m_0 ← (1, 1, ..., 1)                                   ▷ mask initialization
4:     z_0^{(L)(i)} ← (0, 0, ..., 0),    i = 1, 2, ..., M
5:     for t = 0, 1, ..., T − 1 do
6:         for i = 1, 2, ..., M do
7:             s_t^{(i)} ← [z_t^{(L)(i)}, m_t]                  ▷ state initialization
8:         end for
9:         π_θ, s_t ← Aggregate(π_{θ_1}^{(1)}, π_{θ_2}^{(2)}, ..., π_{θ_M}^{(M)}, s_t^{(1)}, s_t^{(2)}, ..., s_t^{(M)}; aggregation_type)   ▷ policies ensembles
10:        a_t ← π_θ(s_t)                                       ▷ next patch prediction
11:        m_{t+1} ← m_t;  m_{t+1}[a_t] ← 0                     ▷ mask update
12:        for i = 1, 2, ..., M do
13:            ŷ_{t+1}^{(i)}, z_{t+1}^{(L)(i)} ← F_w^{(i)}(a_t, x_{a_t})
14:            ĉ_{t+1}^{(i)} ← c_ω^{(i)}(z_{t+1}^{(L)(i)})      ▷ confidence estimation
15:        end for
16:        ŷ_{t+1} ← (1/M) Σ_{i=1}^M ŷ_{t+1}^{(i)}              ▷ prediction ensemble
17:        ĉ_{t+1} ← (1/M) Σ_{i=1}^M ĉ_{t+1}^{(i)}              ▷ confidence ensemble
18:        if ĉ_{t+1} ≥ 0.5 then ŷ ← ŷ_{t+1} and stop loop      ▷ not full scan case
19:        end if
20:    end for
21:    if ŷ = ∅ then                                            ▷ full scan case
22:        if hybrid then  ŷ ← CNN(x)
23:        else ŷ ← ŷ_T
24:        end if
25:    end if
26:    return ŷ
27: end function
```

make the prediction (see line 21 of the algorithm), i.e. use the hybrid approach.

4. Experiments

4.1. Setup

We use the WM-811 dataset (Wu et al., 2014) to validate our algorithm. The dataset contains 811k wafer map images of 9 categories, but only around 173k of them are annotated. Among the annotated images, 147k do not have a pattern of the defect (class *None*) and therefore we leave it out of the consideration remaining with $K = 8$ defect patterns. By default, these subsets of the dataset are split into 17 625 train and 7894 test images. As we discussed in Sections 3.2 and 3.3, we need different images for confidence estimator c_ω and next patch selector π_θ training than for GRU classifier $\mathcal{F}_w$. Therefore, we split the test subset into two. Table 1 summarizes the subsets of datasets and for which purpose they were used.

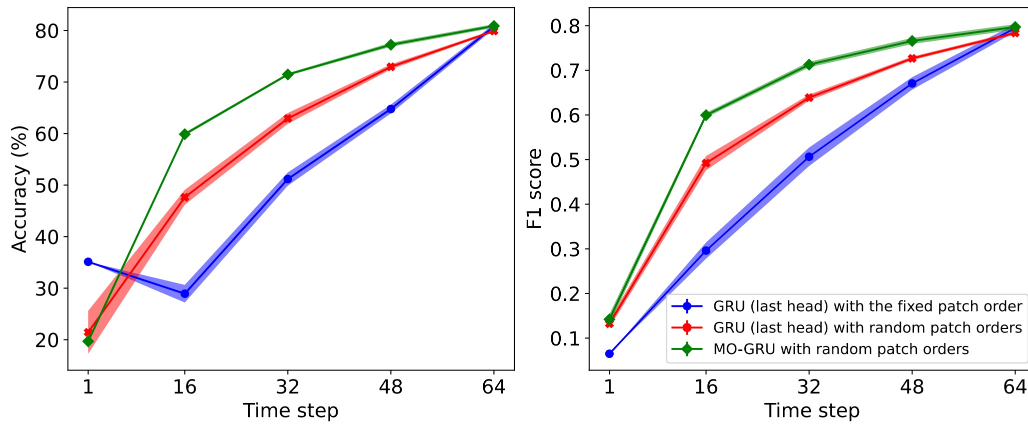


Fig. 4. Comparison between GRU that is trained with the conventional cross-entropy loss on random (red) and fixed (blue) patch orders, and MO-GRU trained with the loss proposed in Section 3.1 on random patch orders. MO-GRU (green) outperforms at every time step the GRU variants trained to predict the defect only at the last time T. The shaded areas represent the standard deviation of different runs.

Table 1
Number of wafer maps in every subdataset that are used for training and evaluation.

Dataset	Purpose	# Of images
$\mathcal{D}_1$	Train $\mathcal{F}_w$	17 625
$\mathcal{D}_2$	Train c_ω and π_θ	3947
$\mathcal{D}_3$	Test $\mathcal{F}_w, c_\omega, \pi_\theta$	3947

Thus, subdatasets $\mathcal{D}_1$ and $\mathcal{D}_2$ are used only for training separate models, while $\mathcal{D}_3$ for testing of all models and overall approach as well. The images have different sizes, therefore, we resize them all to the size of 64×64. Table 2 summarizes the variables for setting up the experiments. We also apply rotation by the random angle in $[-\frac{\pi}{2}, \frac{\pi}{2}]$, vertical and horizontal flips with probability 0.5 during training to increase the variability of the example.

For evaluation of the proposed approach, we use the standard accuracy metric, as well as F_1-score to take into account the data imbalance issue:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FP}} \tag{3}$$

$$F_1 = 2 \cdot \frac{\mathrm{Recall} \times \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}} \tag{4}$$

where $\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$, $\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$, and TP is the true positive, FP is the true negative, FN is the false negative and FP is the false positive predictions.

4.2. Main result

In this subsection, we want to validate our approach using the components described in Section 3. Every image is split into $T = 64$ patches of size 8×8. We train MO-GRU and confidGRU using Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-3} and weight decay 10^{-5} for 240 and 300 epochs respectively. The patch selector π_θ consists of a feed-forward neural network with 2 hidden layers of size 400, which is trained with learning rate 10^{-3}. The discounting factor $\gamma = 0.99$ for this case. We train our model with 5 different initializations for collecting statistics and constructing ensembles. We also experimented with different numbers of layers and hidden layer sizes for π_θ but did not notice any significant difference.

First, we justify the need for the training procedure described in 3.1. Fig. 4 shows that MO-GRU is significantly better at intermediate steps $t < T$ than GRU trained on random patch orders but classifies defects only at time T. For completeness, we also present the numbers for conventional GRU trained with the predefined fixed patch order and only the last output contributes to the loss. MO-GRU significantly outperforms GRU in intermediate time steps, since it was specifically trained to recognize defect patterns not only at the final time moment. Moreover, MO-GRU is more robust to parameters initialization since it has lower variance in results as can be seen from the shaded areas in the figure. We explain this by having a training procedure for intermediate time steps, which makes the model less dependent on initialization at times $t = 16, 32, 48$. At time $t = 64$, the models converge to the same accuracy since all of them were optimized for that time step. At time $t = 1$, we observe a higher accuracy and lower F_1-score for GRU with fixed patch order which is explained by the GRU bias at $t = 1$ towards two specific classes *Loc* and *Scratch*, which together comprise 50% of the dataset. However, as more patches are acquired, this phenomenon is eliminated. Also, we want to emphasize the importance of positional embedding in pretraining for an arbitrary patch sequence. MO-GRU without patch position as an input yields average classification accuracy around 44%.

From Table 3, we see that the RL approach significantly helps in optimizing the order of input patch sequence for both single-model case (80.23% w/o RL against 81.04% w/RL) and ensemble case (82.22% w/o RL against 83.33% w/RL). Moreover, the approach with patch selector π_θ outperforms the scenario where the patch order is random in every run. Both policy aggregation strategies improve the accuracy, however, `min-entropy` is slightly better than `averaging` one yielding the advantage of 1.22% over the random patch order. Also, the average number of scanned patches decreases in every case where RL was applied to optimize the input patch order. From the results, we observe that we need 39–40 patches out of 64 which corresponds to 61–62.5% of the full image. That means that we save up to $R = 39\%$ of scanning time.

We can see an interesting natural separation of the defect classes. Fig. 5 displays how the number of measurements is distributed (left), the average number of required patches per class, and the corresponding accuracy for `min-entropy` ensemble strategy (right). From the figure, we see that we still have a considerable part of the images that require full scanning. At the same time, we observe that the patterns that require more measurements have lower accuracy and vice versa. That hints us that such patterns as *Scratch*, *Donut* and *Loc* are the main source of both error and thelarger number of required patches, while for *Near-full*, *Edge-Ring* and *Center* we could decrease the number of measurements twice having ~90% of accuracy. We can also conclude that patterns that required fewer measurements were detected more accurately. That means that the algorithm achieves a sufficient confidence level for *Near-full*, *Edge-Ring* and *Center* faster than for other types. At the same time, recognizing the other types is more difficult. Therefore, the algorithm is less confident in its prediction, leading to more scanning steps and lower accuracy.

For further investigation, we split all wafer maps into two categories: those that require all patches scanning and those for which the

Table 2
Variables from Section 3 and their values that were used in the numerical experiments.

Variables	Image size ($N_1 \times N_2$)	Patch size ($n_1 \times n_2$)	Number of patches (T)	Number of defects (K)
Values	64 × 64	8 × 8	64	8

Table 3
Accuracy (%), F_1-score and average stopping time t^* over five different runs.The last column represents averaged results for the Single MO-GRU model and Ensemble results for the ensemble of models. The best indicators for each scenario are in bold.

	Metrics	Run 1	Run 2	Run 3	Run 4	Run 5	
Single MO-GRU + confidGRU							Average
	acc	80.01	80.62	80.03	80.29	80.18	80.23
+ Random patch order	f1	0.787	0.793	0.785	0.788	0.785	0.788
	t^*	42.3	42.6	41.4	41.5	43	42.1/64
	acc	**80.92**	**81.76**	**80.75**	**81.35**	**80.44**	**81.04**
+ RL	f1	0.806	0.808	0.807	0.801	0.784	**0.801**
	t^*	38.5	40	37.3	39.1	41.3	**39.2**/64
Ensemble MO-GRU + confidGRU							Ensemble
	acc	–	–	–	–	–	82.11
+ Random patch order	f1	–	–	–	–	–	0.804
	t^*						42.5/64
	acc	–	–	–	–	–	82.62
+ `Averaging` ensemble for RL	f1	–	–	–	–	–	0.816
	t^*	–	–	–	–	–	40.7/64
+ `Min-entropy` ensemble for RL	acc	–	–	–	–	–	**83.33**
	f1	–	–	–	–	–	**0.818**
	t^*	–	–	–	–	–	**39.7**/64

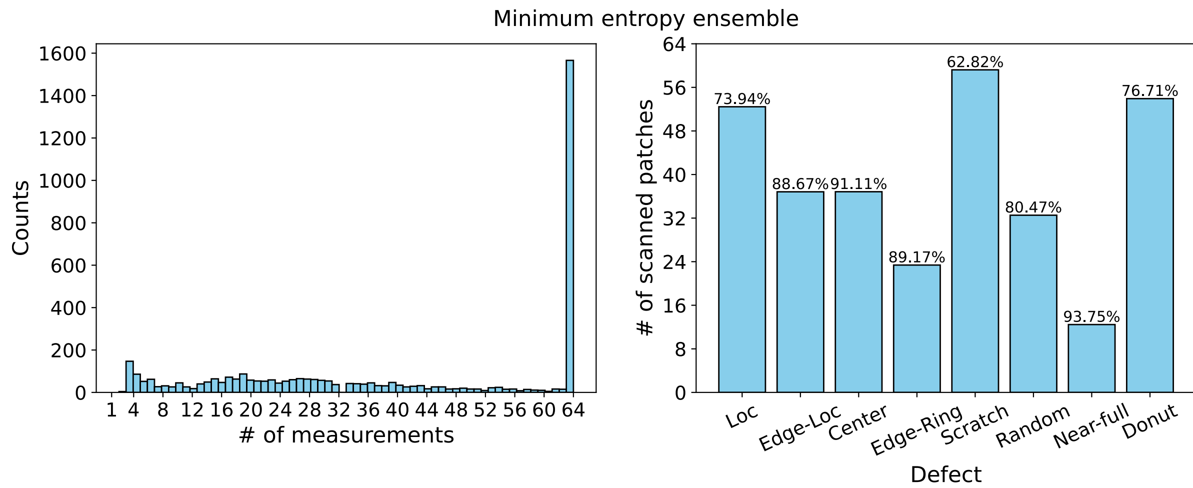


Fig. 5. Histogram of the number of measurements (left) and an average number of measurements by defect (right) with the classification accuracy on top of each bar.

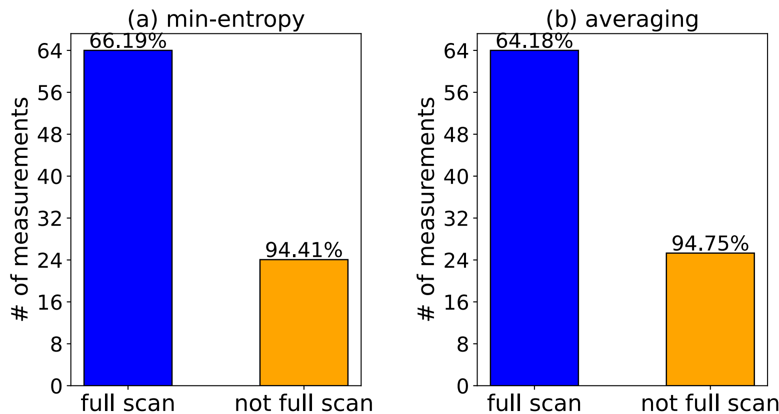


Fig. 6. Accuracy (on top of the bars) on images that require (blue) and do not require (orange) full scan for (a) `min-entropy` and (b) `averaging` ensemble approaches.

Table 4
Accuracy (%), F_1-score and stopping time t^* for the hybrid approach over five different runs. The best indicators are in bold and the second best are underlined.

	Metrics	Run 1	Run 2	Run 3	Run 4	Run 5	Average
ResNet-34	acc	84.98	85.33	85.25	85.56	85.30	85.28
	f1	0.841	0.836	0.844	0.841	0.845	0.841
	t^*	64	64	64	64	64	64/64
ResNet-34 + Single MO-GRU + confidGRU							
+ Random patch order	acc	84.56	84.91	84.53	85.01	85.04	84.81
	f1	0.838	0.833	0.832	0.834	0.836	0.835
	t^*	42.3	42.6	41.4	41.5	43	42.1/64
+ RL	acc	84.55	84.90	84.32	84.77	85.10	84.73
	f1	0.845	0.836	0.841	0.831	0.839	0.838
	t^*	38.5	40	37.3	39.1	41.3	**39.2**/64
ResNet-34 + Ensemble MO-GRU + confidGRU							
+ Random patch order	acc	84.65	85.18	84.8	85.31	85.13	85.01
	f1	0.834	0.838	0.839	0.840	0.841	0.838
	t^*	42.5	42.5	42.5	42.5	42.5	42.5/64
+ `Averaging` ensemble for RL	acc	85.08	85.43	85.28	85.79	85.33	**85.38**
	f1	0.844	0.843	0.845	0.844	0.846	**0.844**
	t^*	40.7	40.7	40.7	40.7	40.7	40.7/64
+ `Min-entropy` ensemble for RL	acc	85.03	85.33	85.10	85.74	85.56	<u>85.35</u>
	f1	0.840	0.840	0.838	0.846	0.846	<u>0.842</u>
	t^*	39.7	39.7	39.7	39.7	39.7	<u>39.7</u>/64

prediction is made faster. Fig. 6 displays the findings, the numbers on the top of the bars correspond to the accuracy for each category, meaning that the images that do not require full scanning have more than 94% accuracy while using only about 24 patches on average for both ensemble types. At the same time, images that require all patches are predicted with the accuracy ∼64–66%.

4.3. Hybrid approach

We want to make our approach comparable with CNN-based models, e.g. ResNet architecture. Therefore, we train ResNet-34 for the comparison receiving 85.27% average accuracy. However, ResNet requires scanning of the full image, which gives it an advantage over our approach. From Fig. 5, we see that a significant part of the images still require all patches for scanning. Thus, we can combine these two models (see also line 21 of Algorithm 2). Table 4 shows the results for this hybrid approach where the wafer map images that require full scanning are sent to the ResNet model. Here we have applied one ResNet for every run without ensembling them. Therefore, the columns for the Ensemble case are not empty and correspond to different ResNets, which we then average. Overall, we consider three scenarios: (1) only ResNet-34; (2) MO-GRU + confidGRU with random and RL-optimized patch orders, and ResNet-34 for the full scan case (see line 14 of Algorithm 1); and (3) an Ensemble of MO-GRUs + confidGRUs with random patch orders and two aggregation strategies for RL-optimized patch sequence, and ResNet-34 for the full scan case (see line 21 of Algorithm 2).

As we can see from the results, we could significantly increase accuracy when suspicious images are classified with ResNet instead of MO-GRU. We want to emphasize that the number of scanned patches does not change when we use ResNet as an auxiliary model since we have already measured all patches for MO-GRU. Therefore, this hybrid approach also saves $R =$ ∼38% of scanning time in terms of the number of scanned patches while having slightly higher accuracy than ResNet which requires full scanning. We can also observe that random patch order leads to lower accuracy and higher measured patches.

Our approach can be used with any CNN-based solution. Instead of conventional ResNet-34, one can use a state-of-the-art approach that uses the full image for classification. For illustration, we consider an approach in which a CNN model is trained with supervised contrastive loss (SCL) proposed by Bae and Kang (2023) in addition to cross-entropy loss. Table 5 shows this comparison, SCL ResNet-34 refers to ResNet trained with a new loss. We can see from the table that by

Table 5
Proposed hybrid approach with SCL ResNet-34 in comparison with pure SCL ResNet-34. The results are averaged over five different runs with different initialization for ResNet-34 and SCL ResNet-34.

Method	Acc	f1	t^*
ResNet-34	85.28	0.841	64/64
SCL ResNet-34 (Bae & Kang, 2023)	85.52	0.848	64/64
SCL ResNet-34 + Ensemble MO-GRU + confidGRU			
+ `Averaging` ensemble for RL	85.41	**0.849**	40.7/64
+ `Min-entropy` ensemble for RL	**85.57**	0.847	**39.7**/64

replacing ResNet-34 with SCL ResNet-34, our hybrid approach with ensemble outperforms the pure SCL ResNet-34 strategy while keeping the property that only ∼62% of patches should be scanned. `Min-entropy` ensemble is a bit better in terms of accuracy while `averaging` one provides greater F_1 score.

The purpose of this example is to demonstrate that our approach can be used in combination with other approaches to images that require full scanning. In this way, we improve the classification accuracy while saving scanning time.

4.4. Cost of computing resources

Additionally, we want to outline that computational complexity increases with the number of models in the ensemble. However, computations in the ensemble can be performed independently, and, therefore, this effect can be mitigated by parallelizing the computations in the code implementation of the ensemble. In addition, our main objective is to reduce the scanning steps due to the time associated with this process and the financial costs.

Concerning memory costs, we need 1.2 MB for the classifier, 905 KB for confidGRU and 1.2 MB for the policy network, which results in 4.6 MB for one model. In the case of an ensemble of 5 models, we need 5 × 4.6 MB = 23 MB. For comparison, one ResNet-34 allocates 81.3 MB. Thus, in the non-hybrid case, we allocate even 3.5 times than for ResNet-34 while saving 38% of measurement steps but sacrificing around 2% of accuracy (see Table 3). In the hybrid approach, the ensemble adds 28% of memory costs achieving the best accuracy, however, the single-model case produces slightly lower accuracy compared to ResNet-34, while adding only 5% of memory costs and decreasing the number of scanning steps by 39%.

5. Conclusion

Automatic visual inspection of wafer maps is an important part of semiconductor manufacturing. However, this process is expensive and requires high confidence in the prediction. In this work, we have proposed an algorithm for the adaptive sequential measurement of the wafer map for defect pattern classification that aims to reduce costs without sacrificing accuracy. In contrast to the conventional CNN approach, the proposed feedback loop leverages already acquired patches in deciding which patch should be taken next. This allows the algorithm to exclude information that is not useful for prediction.

Our approach consists of three core blocks: 1) a recurrent network to process measurements sequentially; 2) an RL-based sampling procedure to facilitate and speed up classification by selecting only the most informative regions to be scanned; and 3) a confidence network which evaluates the confidence in the classifier's prediction after every new measurement and indicates when scanning procedure can be stopped. Our framework allows us to reduce the scanning area by 38%. Moreover, a confidence score is provided along with the prediction, which signals suspicious predictions that require additional consideration, or tells us that a prediction is trustworthy. We provide an ensemble strategy to increase the performance of these models and outperform the conventional CNN-based approach. In addition, we have illustrated that our approach could be combined with other CNN-based methods in the hybrid case, improving their performance while requiring fewer measurements.

Further steps might be considered to improve the core blocks proposed in our approach: classifier, confidence estimator, or patch sequence generator. Also, another direction is addressing a wider range of problems under different real-life restrictions such as wafers with multiple defect types, few-shot learning, and learning from noisy data. However, we advocate the advantages of the feedback loop approach and believe that it opens new opportunities to improve inspection and measuring systems.

CRediT authorship contribution statement

Aleksandr Dekhovich: Conceptualization, Methodology, Writing – original draft preparation, Software. **Oleg Soloviev:** Conceptualization, Writing – review & editing, Supervision. **Michel Verhaegen:** Conceptualization, Writing – review & editing, 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

The project is supported by the Chips Joint Undertaking and its members, including the top-up funding by RVO (The Netherlands Enterprise Agency).

Data availability

The code implementation of the work can be found here: https://github.com/adekhovich/sequential_wafer_inspection.

References

Ba, J., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. ArXiv Preprint abs/1607.06450.

Bae, Y., & Kang, S. (2023). Supervised contrastive learning for wafer map pattern classification. *Engineering Applications of Artificial Intelligence*, *126*, Article 107154.

Bakker, T., van Hoof, H., & Welling, M. (2020). Experimental design for MRI by greedy policy search. *Advances in Neural Information Processing Systems*, *33*, 18954–18966.

Bergner, B., Lippert, C., & Mahendran, A. (2023). Iterative patch selection for high-resolution image recognition. In *The eleventh international conference on learning representations*.

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distribution. *Bulletin of the Calcutta Mathematical Society*, *35*, 99–110.

Bhattacharyya, A. (1946). On a measure of divergence between two multinomial populations. *SankhyÃ: The Indian Journal of Statistics*, 401–406.

Chen, S., Zhang, Y., Hou, X., Shang, Y., & Yang, P. (2022). Wafer map failure pattern recognition based on deep convolutional neural network. *Expert Systems with Applications*, *209*, Article 118254.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on empirical methods in natural language processing*.

Chu, W.-H., Li, Y.-J., Chang, J.-C., & Wang, Y.-C. F. (2019). Spot and learn: A maximum-entropy patch sampler for few-shot image classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6251–6260).

Corbière, C., Thome, N., Bar-Hen, A., Cord, M., & Pérez, P. (2019). Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, *32*.

Croitoru, F.-A., Hondru, V., Ionescu, R. T., & Shah, M. (2023). Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(9), 10850–10869.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International conference on learning representations*.

Doutt, D., Chen, P.-j., Ravoori, B., Tran, T. K., Rothstein, E., Kampel, N., et al. (2023). Full wafer OCD metrology: increasing sampling rate without the cost of ownership penalty. *vol. 12496*, In *Metrology, inspection, and process control XXXVII* (pp. 47–56). SPIE.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning* (pp. 1050–1059). PMLR.

Ghobakhloo, M. (2020). Industry 4.0, digitization, and opportunities for sustainability. *Journal of Cleaner Production*, *252*, Article 119869.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136.

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, *110*(3), 457–506.

Ishida, T., Nitta, I., Fukuda, D., & Kanazawa, Y. (2019). Deep learning-based wafer-map failure pattern recognition framework. In *20th international symposium on quality electronic design* (pp. 291–297). IEEE.

Izmailov, P., Vikram, S., Hoffman, M. D., & Wilson, A. G. G. (2021). What are Bayesian neural network posteriors really like? In *International conference on machine learning* (pp. 4629–4640). PMLR.

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, *30*.

Kim, T., & Behdinan, K. (2023). Advances in machine learning and deep learning applications towards wafer map defect recognition and classification: a review. *Journal of Intelligent Manufacturing*, *34*(8), 3215–3247.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations, ICLR*.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, *30*.

Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

Liang, Z.-P., & Lauterbur, P. C. (2000). *Principles of magnetic resonance imaging*. SPIE Optical Engineering Press Bellingham.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., & Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, *32*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., et al. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937). PMLR.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 427–436).

Noom, J., Soloviev, O., Smith, C., & Verhaegen, M. (2022). Closed-loop active object recognition with constrained illumination power. *vol. 12102*, In *Real-time image processing and deep learning 2022* (pp. 9–14). SPIE.

Noom, J., Thao, N. H., Soloviev, O., & Verhaegen, M. (2020). Closed-loop active model diagnosis using bhattacharyya coefficient: Application to automated visual inspection. In *International conference on intelligent systems design and applications* (pp. 657–667). Springer.

Pineda, L., Basu, S., Romero, A., Calandra, R., & Drozdzal, M. (2020). Active MR k-space sampling with reinforcement learning. In *Medical image computing and computer assisted intervention–mICCAI 2020: 23rd international conference, lima, peru, October 4–8, 2020, proceedings, part II 23* (pp. 23–33). Springer.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*, 211–252.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Silvestri, G., Massoli, F. V., Orekondy, T., Abdi, A., & Behboodi, A. (2024). Reinforcement learning of adaptive acquisition policies for inverse problems. arXiv preprint arXiv:2407.07794.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, *12*.

Tabernik, D., Šela, S., Skvarč, J., & Skočaj, D. (2020). Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*, *31*(3), 759–776.

Tan, A. R., Urata, S., Goldman, S., Dietschreit, J. C., & Gómez-Bombarelli, R. (2023). Single-model uncertainty quantification in neural network potentials does not consistently outperform model ensembles. *Npj Computational Materials*, *9*(1), 225.

Ukraintsev, V. A. (2003). Effect of bias variation on total uncertainty of CD measurements. *vol. 5038*, In *Metrology, inspection, and process control for microlithography XVII* (pp. 644–650). SPIE.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *vol. 30*, In *Proceedings of the AAAI conference on artificial intelligence*.

Wang, F.-K., Chou, J.-H., & Amogne, Z. E. (2022). A deep convolutional neural network with residual blocks for wafer map defect pattern recognition. *Quality and Reliability Engineering International*, *38*(1), 343–357.

Wang, H., Li, C., & Li, Y.-F. (2024). Large-scale visual language model boosted by contrast domain adaptation for intelligent industrial visual monitoring. *IEEE Transactions on Industrial Informatics*.

Wang, J., Xu, C., Yang, Z., Zhang, J., & Li, X. (2020). Deformable convolutional networks for efficient mixed-type wafer defect pattern recognition. *IEEE Transactions on Semiconductor Manufacturing*, *33*(4), 587–596.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Wilson, A. G., & Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. *Advances in Neural Information Processing Systems*, *33*, 4697–4708.

Wu, M.-J., Jang, J.-S. R., & Chen, J.-L. (2014). Wafer map failure pattern recognition and similarity ranking for large-scale data sets. *IEEE Transactions on Semiconductor Manufacturing*, *28*(1), 1–12.

Yen, C.-Y., Singhal, R., Sharma, U., Ranganath, R., Chopra, S., & Pinto, L. (2024). Adaptive sampling of k-space in magnetic resonance for rapid pathology prediction. arXiv preprint arXiv:2406.04318.

Yin, T., Wu, Z., Sun, H., Dalca, A. V., Yue, Y., & Bouman, K. L. (2021). End-to-end sequential sampling and reconstruction for MRI. arXiv preprint arXiv:2105.06460.

Yu, J., Shen, Z., & Wang, S. (2021). Wafer map defect recognition based on deep transfer learning-based densely connected convolutional network and deep forest. *Engineering Applications of Artificial Intelligence*, *105*, Article 104387.

Yu, N., Xu, Q., & Wang, H. (2019). Wafer defect pattern recognition and analysis based on convolutional neural network. *IEEE Transactions on Semiconductor Manufacturing*, *32*(4), 566–573.

Zbontar, J., Knoll, F., Sriram, A., Murrell, T., Huang, Z., Muckley, M. J., et al. (2018). fastMRI: An open dataset and benchmarks for accelerated MRI. arXiv preprint arXiv:1811.08839.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232.

Zhou, L., Zhang, L., & Konz, N. (2022). Computer vision techniques in manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *53*(1), 105–117.